Вінницький державний педагогічний університет імені Михайла Коцюбинського

Факультет математики, фізики і комп'ютерних наук

Кафедра математики та інформатики

Г.М. КОВТОНЮК

Основи програмування мовою Python
Лабораторний практикум зі шкільного курсу інформатики

для студентів

предметної спеціальності

014.04 Середня освіта (Математика)

Вінницький державний педагогічний університет
імені Михайла Коцюбинського
Факультет математики, фізики і комп'ютерних наук

Кафедра математики та інформатики

Г. М. Ковтонюк

ОСНОВИ ПРОГРАМУВАННЯ МОВОЮ PYTHON. ЛАБОРАТОРНИЙ ПРАКТИКУМ ЗІ ШКІЛЬНОГО КУРСУ ІНФОРМАТИКИ

для студентів предметної спеціальності
014.04 Середня освіта (Математика)

Вінниця 2023

УДК 004.42
(075.8)
К56

Рекомендовано до видання Вченою радою
факультету математики, фізики і комп'ютерних наук
Протокол № 9 від «31» травня 2023 року

Рецензенти: Бак С. М., доктор фізико-математичних наук, професор, професор кафедри математики та інформатики

Крупський Я. В., кандидат педагогічних наук, доцент, доцент кафедри математики та інформатики

Ковтонюк Г. М. *Основи програмування мовою Python. Лабораторний практикум зі шкільного курсу інформатики (для студентів предметної спеціальності 014.04 Середня освіта (Математика)).* Вінниця: ФОП Рогальська І. О., 2023. 172 с.

Лабораторний практикум «Основи програмування мовою Python» укладено відповідно до діючої програми навчальної дисципліни «Шкільний курс інформатики», затвердженої Вченою радою Вінницького державного педагогічного університету імені Михайла Коцюбинського, та освітньо-професійної програми «Середня освіта. Математика, інформатика» підготовки бакалаврів за предметною спеціальністю 014.04 Середня освіта (Математика). Практикум містить 18 лабораторних робіт з основ програмування мовою Python.

Передмова

Програмування сьогодні є одним із найперспективніших видів діяльності, адже потреби людства у автоматизації різних процесів невпинно зростають. Крім того, програмування розвиває творчий підхід, логічне мислення та вміння розв'язувати проблеми.

Python – це високорівнева, інтерпретована мова програмування. Має простий синтаксис, що дозволяє легко і швидко писати код, що робить її популярною серед початківців та професіоналів. Мова Python є популярною серед програмістів і використовується в багатьох галузях, таких як веб-розробка, наукові обчислення, штучний інтелект, автоматизація тестування та інше. Python підтримує багато парадигм програмування, таких як процедурне, об'єктно-орієнтоване та функціональне програмування. Підтримує багато сторонніх бібліотек і модулів, які розширюють її функціональність і дозволяють виконувати різноманітні задачі. Також Python є кросплатформною мовою програмування, а отже, програми, написані цією мовою, можуть працювати на різних операційних системах, зокрема на таких, як Linux і Unix, Microsoft Windows, Mac OS та інші.

Лабораторний практикум «Основи програмування мовою Python» укладено відповідно до діючої програми навчальної дисципліни «Шкільний курс інформатики», затвердженої Вченою радою Вінницького державного педагогічного університету імені Михайла Коцюбинського, та освітньо-професійної програми «Середня освіта. Математика, інформатика» підготовки бакалаврів за предметною спеціальністю 014.04 Середня освіта (Математика). Практикум містить 18 лабораторних робіт з основ програмування мовою Python, яке відповідно до робочої програми дисципліни вивчається студентами вказаної спеціальності на факультеті математики, фізики і комп'ютерних наук в другому семестрі. Кожна робота містить теоретичні відомості, коди програм, контрольні запитання та індивідуальні завдання для самостійного виконання. Індивідуальні завдання спрямовано на формування у здобувачів освіти практичних навичок програмування мовою Python, готовності використовувати різноманітні прийоми, методи і засоби, необхідні в процесі розробки програм, а також сформувати готовність застосовувати отримані знання і вміння в майбутній професійній діяльності.

Даний лабораторний практикум призначений для використання студентами предметної спеціальності 014.04 Середня освіта (Математика) ВНЗ, а також може бути корисним для учнів ЗЗСО, вчителів та викладачів інформатики, студентів інших спеціальностей, які вивчають програмування мовою Python.

Зміст

Лабораторна робота № 1

ТЕМА: Інтегроване середовище проектування. Введення-виведення даних.

МЕТА: *ознайомитися з інтерфейсом інтегрованого середовища, та засвоїти основні команди для створення та редагування програм*

ТЕОРЕТИЧНІ ВІДОМОСТІ

Для написання програм використовують **текстові редактори** або **інтегровані середовища розробки**, які включають в себе різні інструменти для роботи з кодом: засіб для написання коду (текстовий редактор), інтерактивний інтерпретатор, відлагоджувач тощо.

Текстовий редактор коду - це програмне забезпечення, призначене для написання та редагування текстових файлів з програмним кодом. Ці редактори зазвичай мають спеціальний набір функцій для полегшення роботи з кодом, таких як виділення синтаксису, автодоповнення, перевірка помилок, можливість перегляду документації та інші.

Інтегроване середовище розробки (ICP**)** або **інтегроване середовище проектування (**IDE – Integrated Design Environment**)** – це комп'ютерна програма, що допомагає програмістові розробляти нове програмне забезпечення чи модифікувати (удосконалювати) вже існуюче.

Основні компоненти **IDE** можуть включати:

Транслятор – програма-перекладач, яка перетворює програму, написану на одній з мов високого рівня, на програму, що складається з машинних команд.

Розрізняють такі види трансляторів: ***компілятори*** та ***інтерпретатори***;

- ✓ ***компілятор*** – читає всю програму цілком, робить її переклад і створює варіант програми машинною мовою, що потім виконується;
- ✓ ***інтерпретатор*** – програма, що аналізує кожен рядок програми і потім виконує зазначену в ній команду крок за кроком; програма, яку обробляє інтерпретатор, на

відміну від компільованої, має заново перекладатися на машинну мову при черговому запуску програми. Як правило, програми компілятори та інтерпретатори називаються так само, як і мови, для перекладу з яких вони призначені. Слова Бейсік, Сі можна сприймати і як назви мов, і як назви відповідних програм-трансляторів;

- ***засоби створення і редагування текстів програм (текстовий редактор)***;
- ***засоби автоматизації компонування;***
- ***бібліотеки стандартних програм і функцій;***
- ***програми налагодження*** – допомагають знаходити й усувати помилки в програмі. Це так звані *відлагоджувачі* або *дебаггери* (англ. Debugger) – комп'ютерні програми, які використовуються для тестування і налагодження інших програм;
- потужні ***графічні бібліотеки*** та ***утиліти*** для роботи з ними;
- *вбудований* ***асемблер***;
- *вбудована* ***довідкова служба.***

Іноді сюди також входять системи контролю версій, засоби для профілювання, а також різноманітні засоби та утиліти для спрощення розробки графічного інтерфейсу користувача. Багато сучасних ІСР також включають оглядач класів, інспектор об'єктів та діаграм ієрархії класів для використання об'єктно-орієнтованого підходу у розробці програмного забезпечення.

Текстові редактори та інтегровані середовища програмування для Python

IDLE – стандартний редактор **Python**. Встановлюється разом з **Python** для користувачів **Windows**, окремим пакетом для користувачів **Linux**.

Notepad++ – безкоштовний текстовий редактор вихідного коду, який підтримує велику кількість мов, в тому числі і **Python**. Лише для користувачів **Windows**.

PyScripter – інтегроване середовище розробки для мови програмування **Python**, працює під **Windows**. Поширюється безкоштовно.

PyCharm — інтегроване середовище розробки для мови програмування **Python**. Підтримує веб-розробку на **Django**. **PyCharm** є власницьким програмним забезпеченням. Присутні безкоштовна версія **Community** з усіченим набором можливостей і безкоштовна версія **Edu** для навчальних закладів.

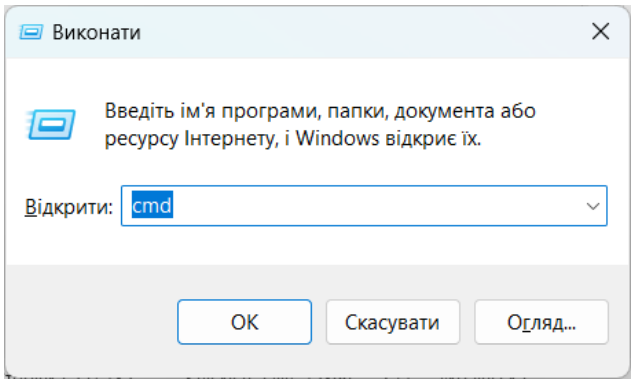
Існує велика кількість онлайнових інтерпретаторів, на таких як: https://www.python.org/, https://replit.com, https://live.sympy.org/, та інших, в яких можна реалізувати код програми мовою **Python**.

Режими виконання програмного коду в середовищі IDLE

У середовищі **IDLE** програмний код мовою **Python** можна виконувати в інтерактивному режимі та режимі виконання файлів програм.

*Розглянемо способи запуску інтерактивного режиму роботи інтерпретатора **IDLE**.*

У режимі інтерактивного інтерпретатора команди вводяться у термінальному вікні одна за одною і по натисненні клавіші **Enter** відразу виконуються з відображенням результату виконання. Для переходу в цей режим: натисніть сполучення клавіш **Win+R** на клавіатурі, введіть команду **cmd**, натисніть **OK** .



У термінальному вікні, що з'явилося, введіть команду: ***python.*** Якщо на екрані з'явиться запрошення **>>>,** значить система виявила встановлену версію **Python**:

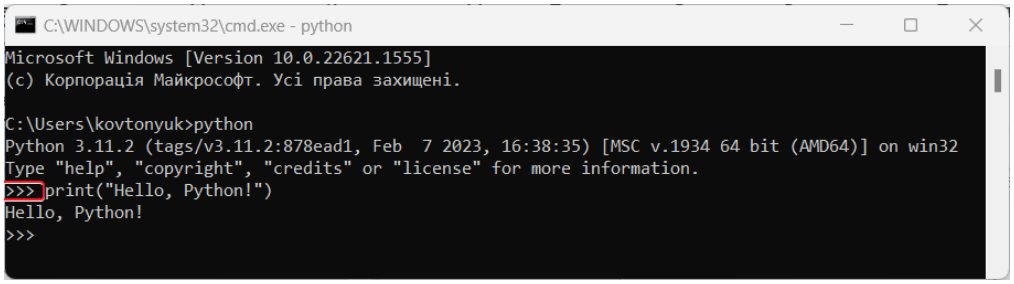


Рис. 1. *Режим інтерактивного інтерпретатора* ***Python*** *у термінальному вікні* ***Windows****: запрошення до введення команд*

Будь-яка програма стає більш зрозумілою, якщо її рядки короткі. Рекомендована (але не обов'язкова) максимальна довжина рядка дорівнює 80 символам. Якщо ви не можете висловити свою думку в рамках 80 символів, скористайтеся символом **продовження рядка** (\). Просто помістіть \ в кінець рядка, а **Python** буде діяти так, ніби це все той самий рядок. *Наприклад*:

```
>>> 'Hello'+\
... 'world'+\
... '!'
'Helloworld!'
>>>
```

```
>>> 1+23+\
... 20
44
>>>
```

Коментарі надзвичайно корисні в будь-якій мові програмування. У мові **Python** ознакою коментаря є символ #. Інтерпретатор **Python** ігнорує всі символи в коді після # до кінця рядка. *Наприклад*:

```
>>> #Мова Python
... print('Hello!')
Hello!
>>>
```

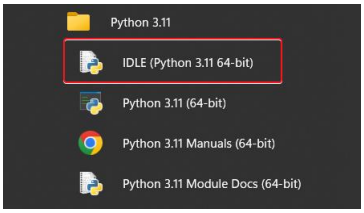
Щоб завершити роботу інтерактивного режиму, необхідно натиснути сполучення клавіш **Ctrl+z.**

Програми, написані на мові **Python**, зберігають у вигляді текстових файлів з розширенням ***.py***.

Запуск **IDLE:**

1. З робочого стола. Якщо ярлик програми **Python** розміщено на робочому столі, то слід клацнути цей ярлик.

2. З головного вікна середовища програмування **IDLE**. Для цього слід виконати команди**: Пуск → Усі програми → Python 3.11 → IDLE (Python 3.11.64-bit).**



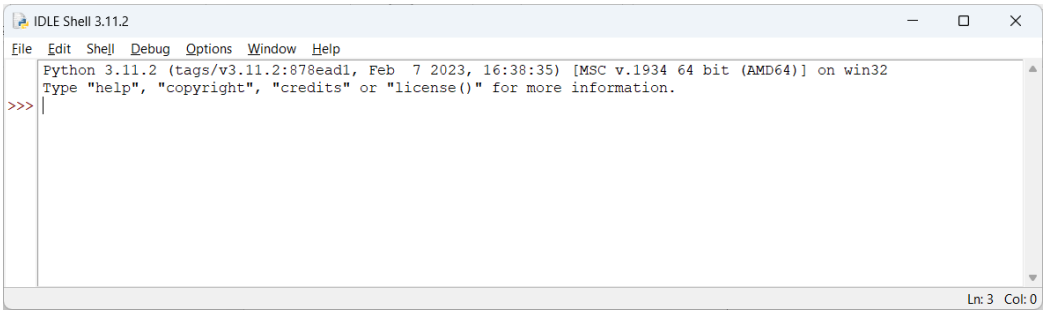
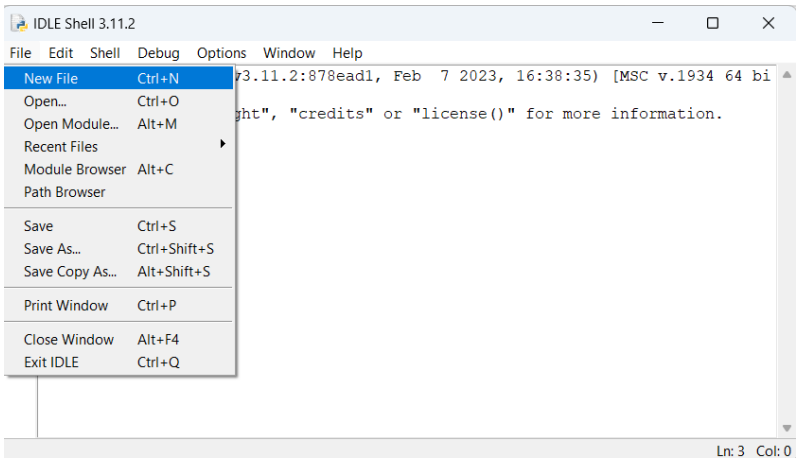


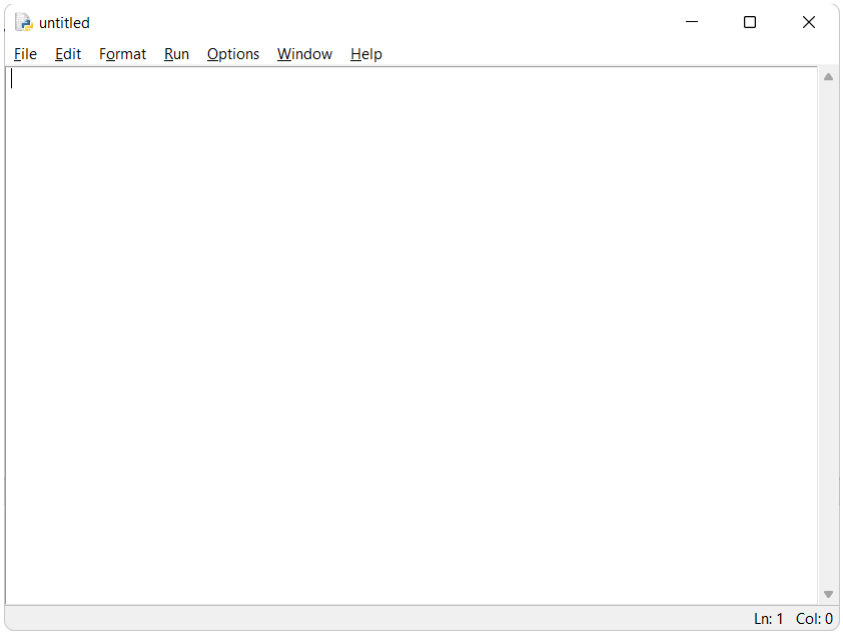
Рис. 2. *Вікно інтерпретатора, яке відкрито у середовищі* ***IDLE***

Після знаку запрошення (>>>) не повинно бути пробілів, інакше видаватиметься повідомлення про синтаксичну помилку. Команди відокремлюються крапкою з комою (;), якщо в одному рядку уводяться дві або більше команд.

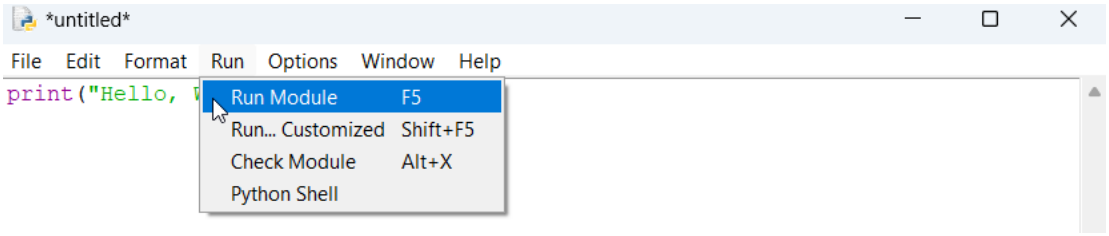
Створити файл програми можна за допомогою все тієї ж оболонки середовища розробки. Якщо клацнути меню **File**, відкриється список команд і підменю, серед яких є й команда **New File**



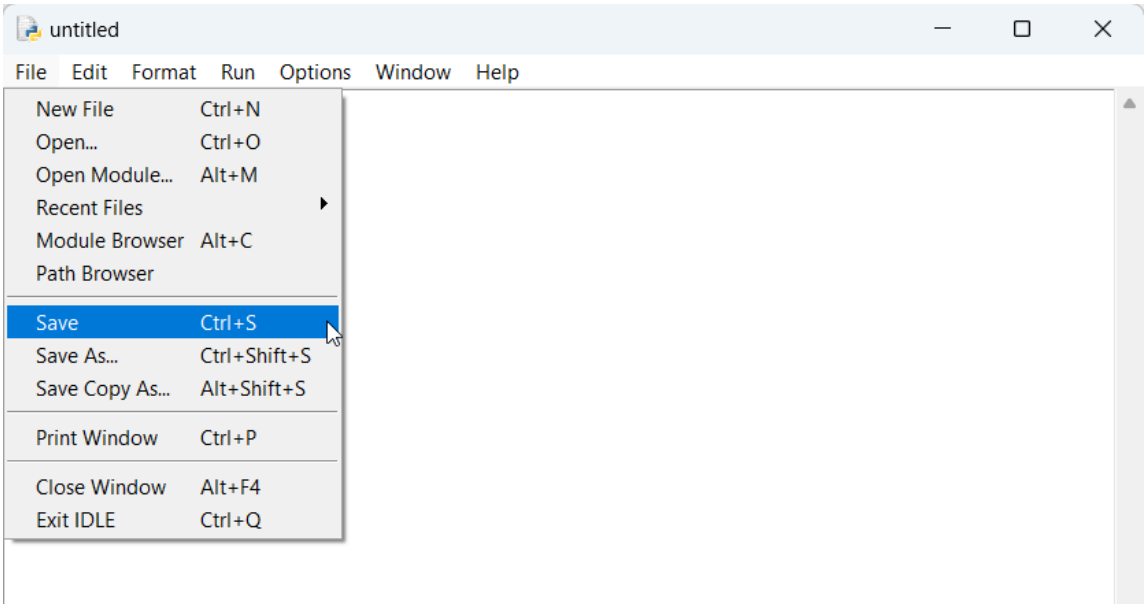
Одразу після вибору цієї команди відкривається редактор кодів.



Запуск програми на виконання **Run – Run Module** або **F5**.



Збереження програми **File – Save.**

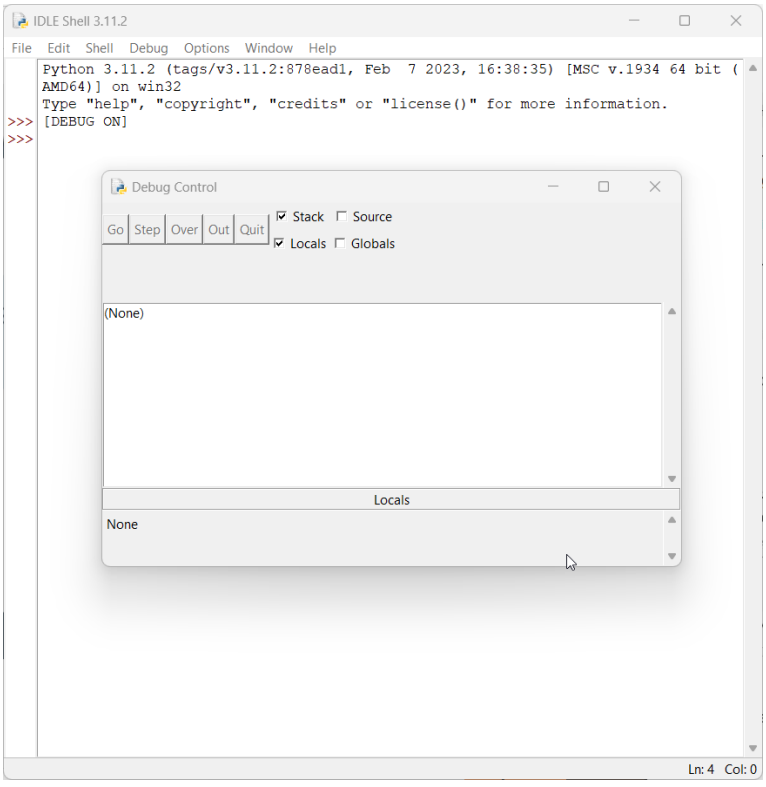


IDLE забезпечує підсвічування синтаксису програмного коду, який вводиться як в головному вікні, так і у всіх вікнах текстового редактора – ключові слова виділяються одним кольором, літерали іншим кольором і т. д.

IDLE Colour Coding

Colour	Use for	Examples
Black	Data & variables	`23.6 area`
Green	Strings	`"Hello World"`
Purple	Functions	`len() print()`
Orange	Commands	`if for else`
Blue	User functions	`get_area()`
Dark red	Comments	`#Remember VAT`
Light red	Error messages	`SyntaxError:`

Крім основних функцій редагування і запуску середовище **IDLE** надає цілий ряд додаткових можливостей, включаючи налагоджувач і інспектор об’єктів. Налагоджувач **IDLE** активується за допомогою меню **Debug** (Налагодження), а інспектор об’єктів – за допомогою меню **File** (Файл). Інспектор об’єктів дозволяє переходити, переміщаючись по шляху пошуку модулів, до файлів і об’єктів в файлах – клацання на файлі або об’єкті призводить до відкриття відповідного вихідного тексту в вікні редагування.



Результати виконання програми і повідомлення про помилки у програмному коді відображаються у вікні консолі.

Введення-виведення даних. Функція **print**(), **input ()**

Функція **print**() забезпечує виведення на екран повідомлення, що міститься у круглих дужках у подвійних лапках.

Введення даних з клавіатури в програму здійснюється за допомогою функції **input ()**. Коли дана функція виконується, то потік виконання програми зупиняється в очікуванні даних, які користувач повинен ввести за допомогою клавіатури. Після введення даних і натискання **Enter**, функція **input ()** завершує своє виконання і повертає результат, який представляє собою рядок символів, введених користувачем.

```
>>> input()
1234
'1234'
>>> input()
Це Python!
'Це Python!'
>>>
```

Коли виконувана програма пропонує користувачеві що-небудь ввести, то користувач може не зрозуміти, що від нього хочуть. Треба якось повідомити, введення яких даних очікує програма. З цією метою функція **input()** може приймати необов'язковий *аргумент-запрошення* рядкового типу; при виконанні функції повідомлення буде з'являтися на екрані та інформувати людину про запитувані дані.

Приклад 1.

```
>>> input('Введіть номер картки:')
Введіть номер картки:12345656
'12345656'
>>>
>>> input('Введіть прізвище:')
Введіть прізвище:Костюк
'Костюк'
```

Приклад 2. Програма повинна містити повідомлення про мову програмування **Python**.

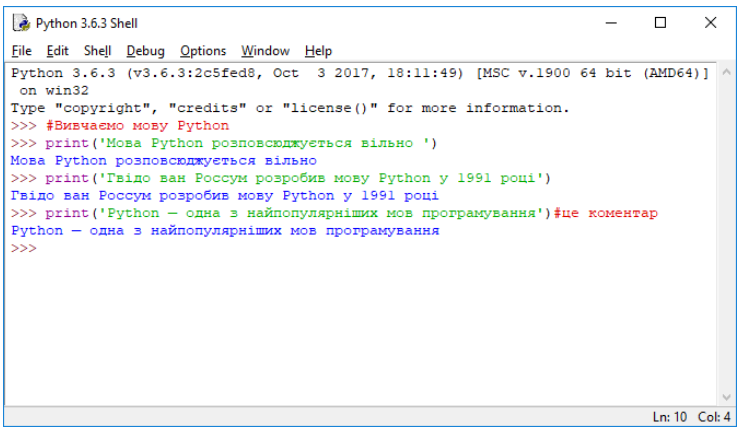


Рис. 3. *Програма мовою Python*

Інтерактивний режим роботи є досить зручним. Але програмний код, який уводиться в інтерактивному режимі, не зберігається: він зникає одразу після того, як інтерпретатор мови **Python** його виконає. Для повторного виконання програмного коду потрібно увести код заново, що є суттєвим недоліком інтерактивного режиму. Програмні коди необхідно зберігати у файлах, які, називають модулями.

Модуль — це будь-який файл із програмним кодом. Файл програмного коду інтерпретатор може виконувати необмежену кількість разів. Після запуску коду на виконання інтерпретатор виконує послідовно одну інструкцію за іншою у порядку їх розташування і видає результат на екран монітора. Якщо у програмному коді буде виявлено синтаксичну помилку, відповідне повідомлення виводиться на екран.

Хід роботи

ЗАВДАННЯ 1.

Виконати завдання 1 – 9

https://colab.research.google.com/drive/1qYafRJyQRiuF0KjVBZrY6CPKOfsF9Yx0?usp=sharing

ЗАВДАННЯ 2. Розробіть і виконайте програмний код (і збережіть як файл **zav_2.py**), який би запитував у користувача:

Його ім'я: "Як тебе звати?"

Вік: "Скільки тобі років?"

Місце проживання: "Де ти живеш?"

А потім має вивести три рядки

"Мене звати ….."

"Мені … років"

"Я *місце проживання*"

Замість слів *ім'я, вік, місце проживання* повинні бути відповідні дані, що введені користувачем. Додайте ще декілька повідомлень до даної програми.

Контрольні запитання

1. Для чого застосовується коментар у програмному коді?
2. Яке позначення має знак запрошення в інтерактивному режимі?
3. Які переваги має інтерактивний режим виконання програмного коду?
4. Як можна запустити інтерактивний режим інтерпретатора **IDLE**?
5. Як правильно ввести декілька команд в одному рядку?
6. Які недоліки має інтерактивний режим інтерпретатора?
7. Чому доцільно зберігати програмний код у файлі?
8. Для чого у програмному коді останньою зазначається команда **input**()?

Лабораторна робота № 2

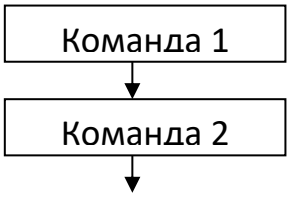
ТЕМА: Проектування та налагодження програм оброблення простих типів даних. Реалізація лінійних алгоритмів.

МЕТА: *ознайомитися з інтерфейсом інтегрованого середовища, та засвоїти основні команди для створення та редагування програм*

ТЕОРЕТИЧНІ ВІДОМОСТІ

Лінійний алгоритм *– алгоритм, який містить лише вказівки безумовного виконання деяких операцій, без повторень або розгалужень (просте слідування).*

Слідування означає, що дії виконуються послідовно одна за одною.



Введення-виведення даних

Виведення даних. Функція **print**()

Функція **print ()** має кілька “прихованих” аргументів, які задаються за замовчуванням в момент виклику:

print(...)

print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

де **value** – перераховуються об’єкти *через кому*, що необхідно вивести на екран;

sep – *рядок-розділювач* між рядками. За замовчуванням стоїть пробіл;

end – *рядок*, що розміщено після останнього об’єкту. За замовчуванням – *перехід на новий рядок*.

Функція **print()** додає пробіл між кожним виведеним об’єктом, а також символ нового рядка в кінці:

```
>>> print(10,20,30,40,50)
10 20 30 40 50
```

```
>>> print('a',"b",'c')
a b c
```

Щоб змінити розділювач (за замовчуванням, це **пропуск**) між елементами, які виводить функція **print()**, треба використати її необов'язковий параметр **sep** і вказати розділювач:

```
>>> print(10,20,30,40,50,sep=';')
10;20;30;40;50
```

можна додати керуючу послідовність (символ перенесення рядка '\n')

```
>>> print(1,2,3,4,5,sep='\n')
1
2
3
4
5
```

Також, використовуючи атрибут **end** та вказавши в якості властивості декілька символів перенесення рядка, можна отримати необхідну кількість порожніх рядків:

```
>>> print("Як справи?", end="\n\n\n")
Як справи?


>>> print('Добре')
Добре
```

Введення даних. Функція **input()**

Введення даних з клавіатури в програму здійснюється за допомогою функції **input()**

Якщо ця функція виконується, то потік виконання програми зупиняється в очікуванні даних, які користувач *повинен ввести за допомогою клавіатури*.

```
>>> input()
```

⇨ зупинка програми для введення даних

```
>>> input()
2020
'2020'
```

Після введення даних і натискання **Enter**, функція **input()** завершує своє виконання і *повертає результат*, який є ***рядком символів***, введених користувачем.

Коли програма, що виконується пропонує користувачеві що небудь ввести, то користувач може не зрозуміти, що робити. Треба *повідомити*, введення яких саме даних очікує програма. З цією метою функція **input()** може приймати ***необов'язковий аргумент-запрошення*** (записаний в одинарних ‘ ’ або подвійних “ ” лапках) ***рядкового типу***.

Наприклад,

```
>>> input('Введіть значення змінної a=')
Введіть значення змінної a=
```

```
>>> input('Введіть значення змінної a=')
Введіть значення змінної a=100
'100'
```

При виконанні функції повідомлення Введіть значення змінної a= буде з'являтися на екрані і інформувати користувача про дані, які потрібно ввести.

Зауваження, дані повертаються у *вигляді рядка*, навіть якщо було введено число.

Якщо потрібно *отримати число*, то результат виконання функції **input()** змінюють за допомогою функцій **int()** або **float()**

Наприклад, ввести ціле число

```
>>> int(input('Введіть число:'))
Введіть число:
```

```
>>> int(input('Введіть число:'))
Введіть число:123
123
```

Ввести дійсне число

```
>>> float(input("Введіть число:"))
Введіть число:
```

```
>>> float(input("Введіть число:"))
Введіть число:12.5
12.5
```

Введення в один рядок даних одного типу:

```
>>> a, b = input().split()
1 2.5
```

Після введення можна виконати перетворення змінних до потрібного типу, по-черзі після їх зчитування:

```
>>> a=int(a)
>>> a
1
>>> b=float(b)
>>> b
2.5
```

Якщо всі змінні *мають один тип*, можна використати функцію **map(),** яка має два обов'язкових параметри: *перший* – тип даних, *другий* рядок, який перетворюється, *наприклад*,

```
>>> a, b, c = map(int, input('Введіть значення:').split())
Введіть значення:1 2 3
```

Функція **split()** (від англ. split - розколоти).

Модуль math

Окрім основних арифметичних операцій та функцій наведених вище, **Python** надає програмісту математичну бібліотеку **math**, яка містить додаткові математичні функції і сталі. Для використання бібліотеки її необхідно імпортувати командою

```
>>> import math
```

Якщо необхідно використовувати функцію часто, можна присвоїти її локальній змінній:

```
>>> import math
>>> s=math.sin; c=math.cos; p=math.pi
```

Інший варіант інструкції **from ім'я_модуля import *** імпортує всі імена з модуля, за винятком таких, що починаються із символу **"_".**

```
>>> from math import* або >>> from math import sin, cos
```

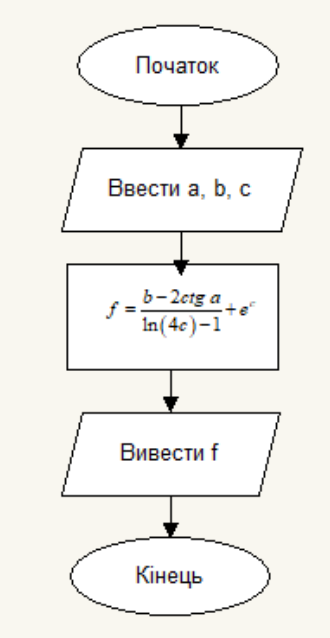
Функції та сталі з бібліотеки math

Функція	Опис
math.pi	Константа $\pi = 3.141592$ ...
math.e	Константа $e = 2.718281$ ...
math.sqrt(x)	Обчислення функції $\sqrt{x}$
math.exp(x)	Обчислення функції e^x
math.log(x)	Обчислення функції $\ln x$
math.log(x, a)	Обчислення функції $\log_a x$
math.log1p(x)	Обчислення функції $\ln(1+x)$
math.log2(x)	Обчислення функції $\log_2 x$
math.log10(x)	Обчислення функції $\lg x$
math.cos(x)	Обчислення функції $\cos x$
math.sin(x)	Обчислення функції $\sin x$
math.tan(x)	Обчислення функції $\mathrm{tg}\, x$
math.acos(x)	Обчислення функції $\arccos x$
math.asin(x)	Обчислення функції $\arcsin x$
math.atan(x)	Обчислення функції $\mathrm{arctg}\, x$
math.atan2(y, x)	Обчислення функції $\mathrm{arctg}\frac{y}{x}$
math.cosh(x)	Обчислення функції $\mathrm{ch}\, x$
math.sinh(x)	Обчислення функції $\mathrm{sh}\, x$
math.tanh(x)	Обчислення функції $\mathrm{th}\, x$
math.hypot(x, y)	Обчислення $\sqrt{x^2+y^2}$
math.degrees(x)	Повертає результат перетворення x з радіан у градуси
math.radians(x)	Повертає результат перетворення x з градусів у радіани
math.factorial(n)	Обчислення n! для цілого та невід'ємного значення аргументу.

math.trunc(x)	Дійсне число, яке є результатом відкидання дробової частини x
math.ceil(x)	Найменше ціле, яке більше або дорівнює x
math.floor(x)	Найбільше ціле, яке менше або дорівнює x

Щоб отримати повний список можна звернутися до документації https://docs.python.org/uk/3/library/math.html з модуля **math**.

Приклад 1. Обчислення значення функції $f=\frac{b-2ctg\,a}{\ln(4c)-1}+e^{c}$ при $a=2,5;\ b=8;\ c=0,4$




```
#Обчислити значення функції
from math import*
print("введіть a,b,c")
a=float(input("a="))
b=float(input("b="))
c=float(input("c="))
f=((b-2*(cos(a)/sin(a)))/(log(4*c-1)))+exp(c)
print("Значення функції f=",'%.3f'%f)
input()
```

Хід роботи

ЗАВДАННЯ 1. Обчислення значення функції

1. *Скласти блок-схему алгоритму розв'язування задачі.*
2. *Написати програму реалізації алгоритму з обов'язковими коментарями*
3. *Проаналізувати достовірність отриманих результатів обчислень.*

Варіанти завдань

№	Розрахункова формула	Контрольні значення змінних
1.	$f=\frac{4\sin(a+bc)}{tg(c-b)+a^3}+\frac{b}{c}$	$a=2;\ b=4,5;\ c=6$
2.	$f=\frac{\sqrt{bc+a^2}}{\ln a}+\cos b$	$a=6,7;\ b=-5;\ c=0,5$
3.	$f=\frac{-b-\sqrt{b^2+4ac}}{2a}+\lvert c\rvert$	$a=5;\ b=9;\ c=-2,5$
4.	$f=ab+\frac{b^a-\sin c}{arctg\,a}$	$a=0,2;\ b=4,5;\ c=8$
5.	$f=ctg\,c+\sqrt{\frac{c}{b}+a}$	$a=4;\ b=3,2;\ c=0,1$
6.	$f=\frac{c^b-4tg\,a}{\ln b}-\{bc\}$ {a}=a-[a]	$a=5;\ b=2;\ c=5,4$
7.	$f=\frac{a+\cos b}{3a+4\sqrt{a+c}}-5c$	$a=5;\ b=-4;\ c=1$
8.	$f=\frac{e^b+3\ln(a+c)}{arctg\,a}+[bc]$	$a=-5;\ b=3,2;\ c=9$
9.	$f=\frac{\sqrt{\frac{a}{b-a}}}{b^2+ab}-\lvert b+2\rvert$	$a=-1;\ b=-4;\ c=6,3$
10.	$f=\frac{a-2ctg\,b}{\ln(3c)+3}+a^b$	$a=1,5;\ b=8;\ c=0,4$
11.	$f=\frac{3\cos(a+bc)}{ctg(c-b)+a^3}+\left\{\frac{c}{b}\right\}$	$a=2;\ b=4,5;\ c=6$
12.	$f=\frac{\sqrt{bc+a^2}}{\ln b}+\sin a$	$a=6,7;\ b=-5;\ c=0,5$
13.	$f=\frac{-b+\sqrt{b^2+4ac}}{2a}-\lvert c\rvert$	$a=5;\ b=9;\ c=-2,5$
14.	$f=ab-\frac{a^c-\cos a}{arctg\,b}$	$a=0,2;\ b=4,5;\ c=8$

15.	$f = tg\,c - \sqrt{\frac{c}{b} + a}$	$a = 4;\ b = 3,2;\ c = 0,1$
16.	$f = \frac{c^b + 2tg\,a}{\ln c} + [abc]$	$a = 5;\ b = 2;\ c = 5,4$
17.	$f = \frac{a + tgb}{3a - 4\sqrt{a + c}} + 3\lvert b\rvert$	$a = 5;\ b = -4;\ c = 1$
18.	$f = \frac{e^a - 4\ln(c - a)}{arctg\,b} - bc$	$a = -5;\ b = 6,2;\ c = 9$
19.	$f = \frac{\sqrt{\frac{b}{b - a}}}{b^2 + abc} + \lvert b + a\rvert$	$a = -1;\ b = -4;\ c = 6,3$
20.	$f = \frac{b - 2ctg\,a}{\ln(4c) - 1} + e^c$	$a = 2,5;\ b = 8;\ c = 0,4$

ЗАВДАННЯ 2. Розробити програму для обчислення значень вказаних елементів трикутника, заданого величинами його сторін. Дано сторони a, b, і c.

1. *Скласти блок-схему алгоритму розв'язування задачі.*
2. *Написати програму реалізації алгоритму з обов'язковими коментарями.*
3. *Проаналізувати достовірність отриманих результатів обчислень.*

Варіанти завдань

Номер варіанта	**Величина, яку потрібно обчислити**	**Довжини сторін, см**		
		a	**b**	**c**
1.	Висота h_a, бісектриса w_β	12,3	14,5	16,7
2.	Медіана m_a, радіус описаного кола R	11,4	12,6	17,1
3.	Бісектриса w_α, радіус вписаного кола r	21,8	24,9	30,6
4.	Радіус описаного кола R, висота h_b	40,6	39,5	41,8
5.	Радіус вписаного кола r, медіана m_b	17,7	15,8	12,1
6.	Висота h_b, бісектриса w_α	17,7	15,8	12,1
7.	Медіана m_b, висота h_c	11,3	15,9	20,7

8.	Бісектриса w_β, бісектриса w_γ	15,5	18,4	19,2
9.	Радіус описаного кола R, бісектриса w_α	14,2	13,9	17,2
10.	Радіус вписаного кола r, медіана m_c	34,6	40,1	28,4
11.	Висота h_c, бісектриса w_γ	39,0	42,2	34,4
12.	Медіана m_c, висота h_a	68,4	63,2	70,8
13.	Бісектриса w_γ, радіус описаного кола R	28,1	26,7	33,2
14.	Радіуси описаного кола R і вписаного r кіл	41,9	49,8	36,3
15.	Радіус вписаного кола r, висота h_b	17,8	20,5	11,6

Контрольні запитання

1. Для чого застосовується коментар у програмному коді?
2. Яке позначення має знак запрошення в інтерактивному режимі?
3. Які переваги має інтерактивний режим виконання програмного коду?
4. Як можна запустити інтерактивний режим інтерпретатора **IDLE**?
5. Як правильно увести декілька команд в одному рядку?
6. Як оформлюється блок команд у мові Python?
7. Як можна запустити на виконання файл програмного коду?
8. Як можна викликати на екран файл програмного коду?
9. Наведіть приклад найпростішого програмного коду мовою Python.
10. Як зберегти файл програмного коду?
11. Які недоліки має інтерактивний режим інтерпретатора?
12. Для чого у програмному коді останньою зазначається команда **input**()?

Лабораторна робота № 3

ТЕМА: Реалізація розгалужених алгоритмів

МЕТА: *вміти створювати програми розгалужених алгоритмів*

ТЕОРЕТИЧНІ ВІДОМОСТІ

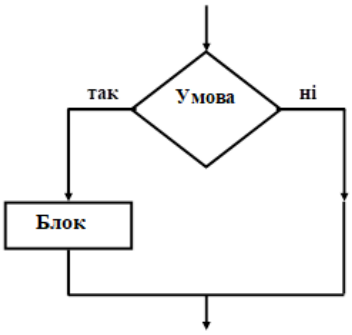
Існують три типи розгалуження: *одноальтернативне* (неповне розгалуження), *двоальтернативне* (повне розгалуження) і *багатоальтернативне* (вибір, варіант).

Одноальтернативне (неповне розгалуження)

У мові **Python** цей тип розгалуження реалізується оператором умовного переходу в неповній формі:

if <*логічний вираз*>**:**

<*блок інструкцій*>



Приклад 1.

```
#більше з двох
a=int(input('a='))
b=int(input('b='))
if a>b:
    print('max=',a)
input()
```

У мові програмування **Python** інструкція **if** включає такі елементи:

- ключове слово **if**;
- умова (вираз, обчислення якого дає одне з двох значень: **True** або **False**);

– двокрапка;
– блок коду з відступом, який починається в наступному рядку.

Ознакою блоку коду є збільшення відступу від лівого краю (4 пробіли або **tab**). Блоки можуть містити інші блоки. Ознакою кінця блоку слугує зменшення відступу до нуля або до величини відступу зовнішнього блоку.

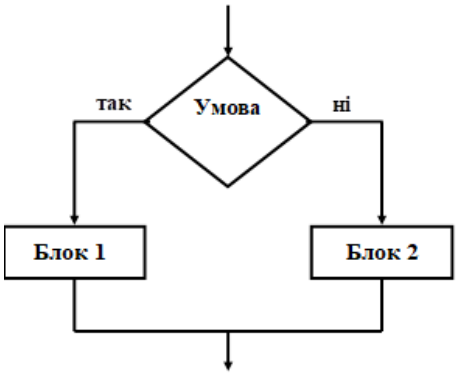
Двоальтернативне (повне розгалуження)

if <*логічний вираз*>**:**

<*блок_1 інструкцій*>

else:

<*блок_2 інструкцій*>



Приклад 2.

```
#більше з двох
a=int(input('a='))
b=int(input('b='))
if a>b:
    print('max=',a)
else:
    print('max=',b)
input()
```

У мові програмування **Python** команда else не має умови та складається із таких елементів:

– ключове слово else;
– двокрапка;
– блок коду з відступом, що починається на наступному рядку.

Багатоальтернативне розгалуження. У мові **Python**:

У цьому типі розгалуження реалізується розгалуження з багатьма варіантами вибору **elif** (скорочено від **else if**).

if *<вираз-селектор>= =<значення_1>***:**

<блок_1>

elif *<вираз>***= =***<значення_2>***:**

<блок_2>

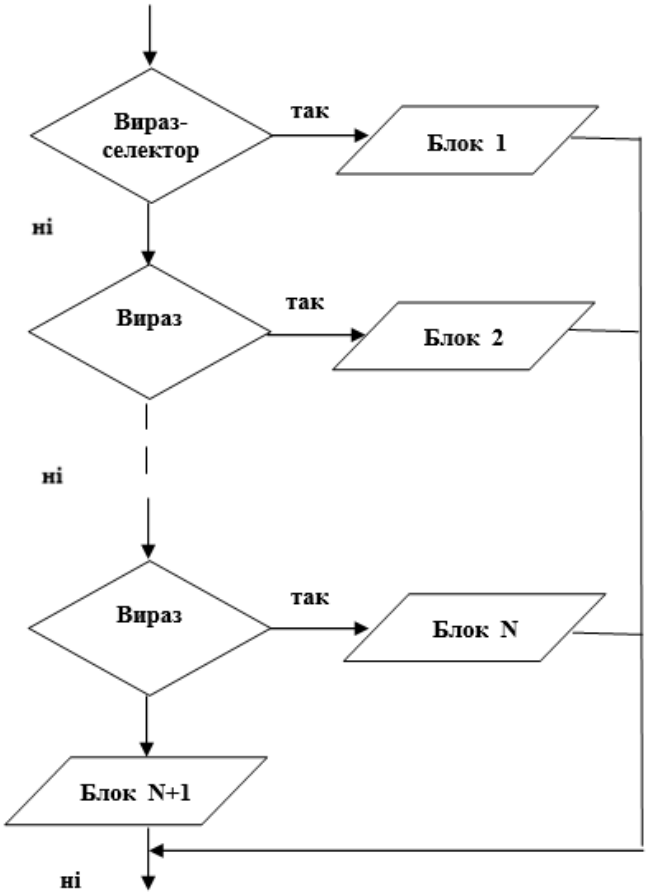
…

elif *<вираз>= =<значення_n>***:**

<блок_n>

else:

<блок_n+1>



Вираз-селектор повинен мати ціле значення, кожне значення виразу повинно бути унікальним константним виразом. Значення виразу порівнюється з кожним значенням в операторах **elif** у порядку їх розташування. Якщо певне значення вираз співпадає зі значенням оператора **elif**, то виконується той блок операторів, що розташований безпосередньо за цим оператором. Якщо значення виразу не співпадає з жодним значенням в операторах **elif**, виконується блок **N+1**.

Приклад 3. *Скласти програму, яка за номером місяця виводить назву місяця або пору року.*

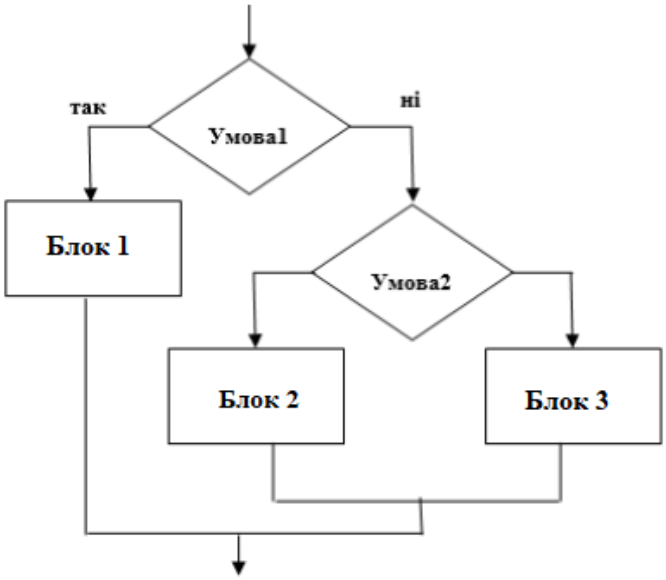
```
#Місяці
n=int(input('Введіть номер місяця n='))
if n==1:
    print('Січень')
elif n==2:
     print('Лютий')
else:
    print('Це не номер місяця')
input()
```

На практиці зустрічаються випадки, коли виконання або невиконання події залежить не від однієї умови, а від декількох.

Один оператор **if** може входити до складу іншого. У цьому випадку говорять про вкладеність операторів.

Вкладені оператори умовного переходу — це оператори умовного переходу, які входять до складу інших операторів умовного переходу.

Існують різні конструкції вкладених операторів. Один із варіантів конструкції вкладених операторів умовного переходу подано на графічній схемі



Приклад 4. *Скласти програму розв'язування квадратного рівняння:* $ax^2 + bx + c = 0$.

```
#Квадратне рівняння
import math
a=int(input('a='))
b=int(input('b='))
c=int(input('c='))
d=b**2-4*a*c
if d>0:
    x1=(-b+math.sqrt(d))/(2*a)
    x2=(-b-math.sqrt(d))/(2*a)
    print('x1=',x1)
    print('x2=',x2)
else:
    if d==0:
        print('x=',-b/(2*a))
    else:
        print('немає розв\'язків')
input()
```

або

```
#Квадратне рівняння
import math
a=int(input('a='))
b=int(input('b='))
c=int(input('c='))
d=b**2-4*a*c
if d>0:
    x1=(-b+math.sqrt(d))/(2*a)
    x2=(-b-math.sqrt(d))/(2*a)
    print('x1=',x1)
    print('x2=',x2)
elif d==0:
    print('x=',-b/(2*a))
else:
    print('немає розв\'язків')
input()
```

Хід роботи

ЗАВДАННЯ 1. Скласти програму обчислення значення функції. *Використання оператора розгалуження if.*

1. *Скласти блок-схему алгоритму розв'язування задачі.*

2. *Написати програму реалізації алгоритму з обов'язковими коментарями*

3. *Проаналізувати достовірність отриманих результатів обчислень.*

Варіанти завдань:

№	**Функція** $y = f(x)$
1.	$y = \begin{cases} \sin^2 x, \text{ якщо } x < 0, \\ \ln(x+1), \text{ якщо } x \geq 0 \end{cases}$
2.	$y = \begin{cases} \cos x^2, \text{ якщо } x < -1, \\ e^{-x^2}, \text{ якщо } x \geq -1 \end{cases}$
3.	$y = \begin{cases} \ln\lvert x\rvert, \text{ якщо } x \leq -3, \\ arctgx, \text{ якщо } x > -3 \end{cases}$
4.	$y = \begin{cases} xe^x, \text{ якщо } x > 1, \\ \sqrt{1-x}, \text{ якщо } x \leq 1 \end{cases}$
5.	$y = \begin{cases} x^3, \text{ якщо } x > 1, \\ x + \sin(x+1), \text{ якщо } x \leq 1 \end{cases}$

6.	$y=\begin{cases}\frac{1}{x}, \text{ якщо } x<0,\\ \ln(3+x^2), \text{ якщо } x\geq 0\end{cases}$
7.	$y=\begin{cases}\sin(\cos x), \text{ якщо } x\leq 0,\\ \cos(\ln x), \text{ якщо } x>0\end{cases}$
8.	$y=\begin{cases}\ln^2\lvert x\rvert, \text{ якщо } x<0,\\ -\ln(x^3+1), \text{ якщо } x\geq 0\end{cases}$
9.	$y=\begin{cases}e^{-x}+e^x, \text{ якщо } x<0,\\ \ln(\sin x+2), \text{ якщо } x\geq 0\end{cases}$
10.	$y=\begin{cases}\ln(-x+1), \text{ якщо } x\leq 0,\\ arctg(\ln x), \text{ якщо } x>0\end{cases}$
11.	$y=\begin{cases}\sin^2(arctgx), \text{ якщо } x<3,\\ \sqrt{x^3-10}, \text{ якщо } x\geq 3\end{cases}$
12.	$y=\begin{cases}\sqrt{1-\cos x}, \text{ якщо } x\leq 0,\\ e^{-arctgx}, \text{ якщо } x>0\end{cases}$
13.	$y=\begin{cases}\sin^3 x\cdot\cos^2 x, \text{ якщо } x<-1,\\ \ln\lvert x+2\rvert\cdot e^{x+1}, \text{ якщо } x\geq -1\end{cases}$
14.	$y=\begin{cases}e^{-x}+e^x, \text{ якщо } x<0,\\ \ln(\sin x+2), \text{ якщо } x\geq 0\end{cases}$
15.	$y=\begin{cases}2x, \text{ якщо } x<2,\\ 1-3x, \text{ якщо } 2\leq x<10,\\ 5\sin x, \text{ якщо } x\geq 10\end{cases}$
16.	$y=\begin{cases}x, \text{ якщо } x<0,\\ 5, \text{ якщо } 0\leq x<5,\\ 5\ln x, \text{ якщо } x\geq 5\end{cases}$
17.	$y=\begin{cases}2e^x, \text{ якщо } x<0,\\ 3\sin x, \text{ якщо } 0\leq x<1,\\ 5\ln x, \text{ якщо } x\geq 1\end{cases}$
18.	$y=\begin{cases}3x, \text{ якщо } x<1,\\ \cos x, \text{ якщо } 1\leq x<3,\\ -x, \text{ якщо } x\geq 3\end{cases}$

19.	$y=\begin{cases} x^2, \text{ якщо} -2 \le x \le 2, \\ 4e^x, \text{ якщо } x<-2, x>2 \end{cases}$

ЗАВДАННЯ 2. Використання оператора **if** та оператора розгалуження з багатьма варіантами вибору **elif** (скорочено від **else if**).

1. *Скласти блок-схему алгоритму розв'язування задачі.*

2. *Написати програму реалізації алгоритму з обов'язковими коментарями*

3. *Проаналізувати достовірність отриманих результатів обчислень.*

Варіанти завдань

1. Скласти програму визначення за довжинами сторін трикутника його виду: рівнобедрений, рівносторонній, довільний.

2. Скласти програму розв'язування біквадратного рівняння:

$$ax^4+bx^2+c=0.$$

3. Скласти програму обчислення значення виразу:

$$\sqrt{x^3-\sqrt{x-1}}.$$

4. Скласти програму визначення найбільшого з трьох чисел.

5. Скласти програму розв'язування системи трьох лінійних рівнянь методом Крамера.

6. Скласти програму визначення найменшого з трьох чисел.

7. Скласти програму визначення кількості від'ємних чисел з трьох заданих.

8. Скласти програму визначення довжин сторін трикутника (перевірити нерівність трикутника) за координатами його вершин.

9. Скласти програму, яка перевіряє, чи натуральне чотиризначне число є більшим за число, записане у зворотному порядку.

10. Скласти програму перевірки твердження „існує трикутник з довжинами сторін a,b,c".

11. Скласти програму, яка за номером квартири визначити в якому під'їзді та на якому поверсі вона знаходиться, якщо задано кількість під'їздів та кількість поверхів в будинку.

12. Скласти програму, яка перевіряє чи поміститься круг площею S_1 у квадрат площею S_2.

13. Скласти програму, яка визначає чи пройде цеглина з ребрами a, b, c у квадратний отвір зі стороною d.

14. Скласти програму визначення маршруту за номером трамвая.

15. Скласти програму визначення дня тижня за його номером.

16. Скласти програму визначення літери за її номером в алфавіті.

17. Скласти програму, яка за номером місяця визначає кількість днів у ньому.

18. Скласти програму, яка виводить розклад занять для заданого дня тижня.

19. Скласти програму визначення маршруту за номером тролейбуса.

20. Є дані про 5 товарів, які визначаються числовим кодом. Скласти програму отримання довідки про ціну і кількість товару на складі.

21. Скласти програму, яка назвою групи визначає кількість студентів у ній.

22. Скласти програму визначення пункту відправлення за номером поїзда.

23. Скласти програму визначення пункту призначення за номером поїзда.

24. Скласти програму, яка визначає чи введена українська літера є голосною чи приголосною.

25. Скласти програму визначення пори року за номером місяця.

26. Голландський медик Рейнхолд Лаесле обчислив індекс маси тіла людини, який дозволяє об'єктивно визначити, чи потрібно схуднути. Індекс обчислюється так: вагу тіла (в кг) поділити на зріст (в м) в квадраті. Якщо результат менше 20 – недостатня вага. Від 20 до 25 – нормальна вага. Від 25 до 30 – надмірна вага. Від 30 до 40 – негайно вжити заходів для схуднення. Скласти програму для оцінки вагової характеристики людини.

27. Скласти програму, яка обчислює довжину відрізка за його назвою (m_a, b_a, h_a),

якщо $m_a = \frac{1}{2}\sqrt{2\left(b^2 + c^2\right) - a^2}$, $b_a = \frac{\sqrt{bc\left(\left(b+c\right)^2 - a^2\right)}}{b+c}$, $h_a = \frac{2}{c}\sqrt{p\left(p-a\right)\left(p-b\right)\left(p-c\right)}$.

28. Скласти програму, яка виводить назву області за назвою міста.

29. Скласти програму, яка виводить назву країни за назвою столиці.

30. Скласти програму, яка за назвою країни визначає назву її континенту.

31. Скласти програму, яка за назвою річки визначає її довжину.

32. Скласти програму, яка обчислює площу фігури за її номером:

$$S = \begin{cases} ab\sin\alpha, & n = 1, \\ \frac{ab}{2}\sin\alpha, & n = 2, \\ \frac{\left|d_1^2 - d_2^2\right|}{4} tg\,\alpha, & n = 3, \\ \frac{\pi d^2}{4}, & n = 4. \end{cases}$$

33. Скласти програму, яка обчислює площу фігури за її номером:

$$S = \begin{cases} ab, & n = 1, \\ \frac{ah}{2}, & n = 2, \\ \frac{a+b}{2}h, & n = 3, \\ \pi R^2, & n = 4. \end{cases}$$

Контрольні запитання

1. Який алгоритм називається розгалуженим?
2. Які існують типи розгалужень?
3. Які логічні операції ви знаєте?
4. Як записують скорочену форму команди розгалуження?
5. Як записують повну форму команди розгалуження?
6. Що таке вкладеність операторів?

Лабораторна робота № 4 - 5

ТЕМА: Реалізація циклічних алгоритмів
МЕТА: *вміти створювати програми циклічних алгоритмів*

ТЕОРЕТИЧНІ ВІДОМОСТІ

Реалізація циклічних алгоритмів мовою Python

Для запису алгоритмів із повторенням (циклів) мовою **Python** використовують два види операторів циклу: *з параметром* та *з умовою*.

Повторення (цикл) — це алгоритмічна структура, за допомогою якої та сама послідовність дій виконується багаторазово для різних значень змінних.

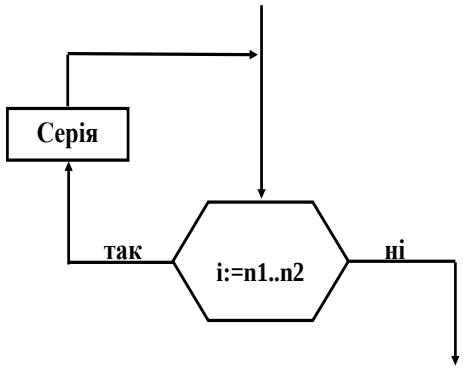
1. Цикли з параметрами

Серію інструкцій, які виконуються багаторазово під час виконання циклу, називають *тілом циклу*. У циклах із параметром кількість повторень інструкцій тіла циклу заздалегідь відома.

Цикли з параметром у мові **Python** реалізуються оператором циклу **for** (для), який має таку загальну структуру:

for <змінна циклу> **in** <об'єкт> :

<блок інструкцій тіла циклу>



де:

<*об'єкт*> — може бути рядок, список, словник та інші типи даних, які підтримують реалізацію циклу. Тип об'єкта може також створити програміст самостійно;

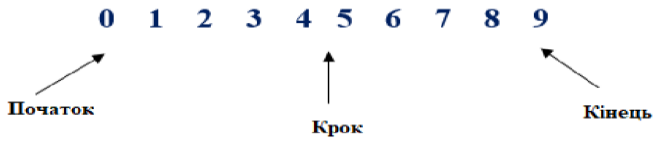
<*змінна циклу*> — поточне значення об'єкта. Початкове її значення — це перший елемент об'єкта. У другому циклі ця змінна набуде значення другого елемента об'єкта і так далі до останнього.

На початку виконання циклу змінна циклу набуває значення першого елемента об’єкта. Оператор **in** генерує логічне значення **True**, і виконується блок інструкцій тіла циклу. На цьому кроці змінна циклу набуде значення другого елемента об’єкта й також буде генеруватися значення **True**, у результаті чого буде виконано блок інструкцій тіла циклу. Після того як усі елементи об’єкта будуть перебрані, оператор **in** згенерує значення **False**, блок інструкцій тіла циклу не виконається, а управління буде передано першій інструкції, що розташована безпосередньо за блоком інструкцій тіла циклу.

Блок інструкцій тіла циклу буде виконуватися до тих пір, поки змінна циклу послідовно не набуде усіх значень, що містяться в об’єкті.

Функція **range ()**

Досить часто при розробці програм необхідно отримати послідовність (діапазон) цілих чисел, наприклад від 0 …9:



Для вирішення цього завдання в **Python** передбачена функція **range ()**, що створює послідовність (діапазон) чисел. В якості аргументів функція приймає: *початкове значення діапазону* (за замовчуванням 0), *кінцеве значення* (НЕ включно) і *крок* (за замовчуванням 1).

Загальна структура цієї функції така:

range ([<початок>,] <кінець> [, <крок>])

Обов’язковим є лише параметр **кінець**.

Якщо ([<початок>,]=0 і [, <крок>])=1 то їх можна опускати.

Якщо викликати функцію, то результату ми не побачимо:

```
>>> range(0,10,1)
range(0, 10)
>>> range(10)
range(0, 10)
```

Для створення діапазону чисел від 0 до 9 необхідно використовувати цикл **for**:

```
>>> for i in range(0,10,1):
        print(i, end=' ')# виведення чисел в рядок через пробіл


0 1 2 3 4 5 6 7 8 9
```

де ([<початок>,]=0 , [, <кінець>])=10, [, <крок>])=1), кінцеве значення (НЕ включно).

```
>>> for i in range(0,10,1):
        print(i )# виведення чисел у стовпець


0
1
2
3
4
5
6
7
8
9
```

Задане останнє значення ніколи не є частиною створеного списку, функція створює список із десяти елементів, які відповідають індексам послідовності довжиною 10 елементів.

```
>>> for i in range(10):
...     print(i,end=" ")
...
...
0 1 2 3 4 5 6 7 8 9
```

Можливо також задати початок списку іншим числом та вказати інший крок (навіть від’ємний):

```
>>> for i in range(2,20,2):
...     print(i,end=" ")
...
...
2 4 6 8 10 12 14 16 18
```

```
>>> for i in range(-10,-100,-30):
...     print(i,end=" ")
...
...
-10 -40 -70
```

Можна отримати діапазон в зворотному порядку, зверніть увагу на аргументи функції **range ()**:

```
>>> for i in range(20,2,-2):
...     print(i,end=" ")
...
...

20 18 16 14 12 10 8 6 4
```

Приклад 1. Знайти суму чисел на інтервалі від 1 до 9, від 1 до 10 :

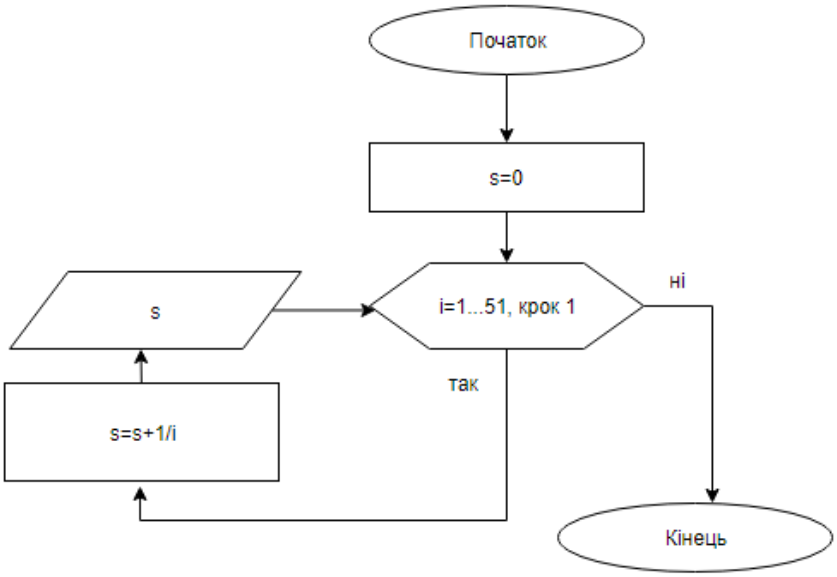
```
>>> S=0
>>> for i in range(1, 10):
        S=S+i
        print('S=',S)

S= 1
S= 3
S= 6
S= 10
S= 15
S= 21
S= 28
S= 36
S= 45
```

```
>>> S=0
>>> for i in range(1, 11):
        S=S+i
        print('S=',S)

S= 1
S= 3
S= 6
S= 10
S= 15
S= 21
S= 28
S= 36
S= 45
S= 55
```

```
>>> for i in range(1, 11):
        S=S+i

>>> S
55
```

Приклад 2.

```
#Обчислити суму s=1+1/2+1/3+..+1/50 використовуючи оператор for
s=0
for i in range (1,51,1):
    s=s+1/i
    print ('s='+'%.3f'% s)
input()
```



У мові Python використовуються арифметичні **оператори з присвоюванням**

Оператор	Позначення	Приклад
Збільшення значення змінної на вказану величину	+=	x+=x (еквівалентно x=x+2)
Зменшення значення змінної на вказану величину	-=	x-=x (еквівалентно x=x-2)
Множення значення змінної на вказану величину	*=	x*=x (еквівалентно x=x*2)
Ділення значення змінної на вказану величину	/=	x/=x (еквівалентно x=x/2)

Попередній приклад, підрахунку суми натуральних чисел від 1 до 9 можемо переписати у такий спосіб:

```
suma = 0
for i in range(1, 10):
    suma += i
print(suma)
input()
```

Приклад 3. Скласти програму для обчислення добутку двох натуральних чисел m і n, використовуючи тільки операцію додавання.

```
m = int(input("m = "))
n = int(input("n = "))
s = 0
for i in range(m):
    s += n
print(s)
```

2. Цикли з умовою

У циклах з умовою кількість виконання інструкцій тіла циклу заздалегідь невідома: вона завершується після досягнення певної умови. Існує два види циклів з умовою: *цикли з передумовою* і *цикли з післяумовою*.

- ***Цикли з передумовою***

У таких алгоритмах спочатку перевіряється умова, а потім виконуються оператори тіла циклу. Якщо умова має значення **True** (Так), виконання операторів тіла циклу продовжується. Як тільки умова набуде значення **False** (Ні), виконання операторів тіла циклу припиняється, й управління передається першому оператору, розташованому за оператором циклу.

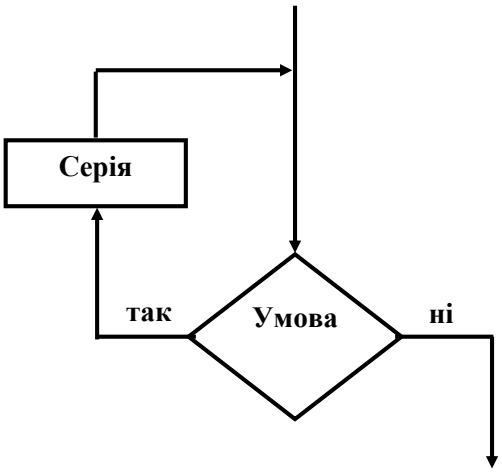
У мові **Python** цикли з передумовою реалізуються оператором **while**, який має таку структуру:

<початкове значення>

while <умова>:

<блок інструкцій тіла циклу>

<зміна початкового значення>



Приклад 4.

```
#Обчислити суму s=1+1/2+1/3+..+1/50 використовуючи оператор while
i=1 #початкове значення параметру
s=0
while (i<=50):
    s=s+1/i
    i=i+1
print ('s='+'%.3f'% s)
input()
```

- ***Цикли з післяумовою***

У таких алгоритмах спочатку виконуються оператори тіла циклу, а потім перевіряється умова. Якщо умова має значення **False** (Ні), виконання операторів тіла циклу продовжується.

Як тільки умова набуде значення **True** (Так), виконання операторів тіла циклу припиняється, й управління передається першому оператору, розташованому за оператором циклу. У мові **Python** відсутній оператор, який безпосередньо реалізує такий варіант циклу.

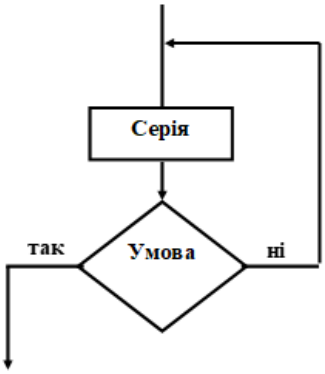
У мові **Python** цикли з післяумовою можна реалізувати такою конструкцією оператора **while**:

<початкове значення>

while True:<блок інструкцій тіла циклу>

<зміна початкового значення>

if <умова>: **break**



Приклад 5.

```
#Обчислити суму s=1+1/2+1/3+..+1/50 використовуючи оператор while
i=1 #початкове значення параметру
s=0
while True:
    s=s+1/i
    i=i+1
    if (i>50):break
print ('s='+'%.3f'% s)
input()
```

Оператори continue і break

Оператори **continue** і **break** застосовуються всередині операторів тіла циклу та призначені для їх переривання.

- Оператор **continue** перериває цикл і повертає управління на початок циклу. Це дозволяє перейти до наступної ітерації циклу до завершення виконання всіх інструкцій у середині циклу.

Приклад 6. Вивести всі парні числа від 6 до 30, крім чисел у діапазоні від 14 до 24.

```
>>> for i in range(6,31,2):
        if 12<i<26:     # якщо числа 12<i<26
                continue #перехід на нову ітерацію
        print(i)


6
8
10
12
26
28
30
```

- Оператор **break** здійснює переривання та вихід із циклу навіть у тому випадку, коли всі ітерації ще не виконані.

Приклад 7. У цьому прикладі використовується нескінченний цикл введення чисел та обчислення їх суми. Але цикл переривається, якщо буде введено слово «кінець».

```
S=0
while True:#нескінченний цикл
        a=input('Введіть число')
        if a=='кінець':
                break
        a=int(a)#перетворення рядка в число
        S=S+a
print('S=',S)
input()
```

```
Введіть число1
Введіть число2
Введіть число3
Введіть числокінець
S= 6
```

До операторів циклів **while** та **for** може застосовуватися оператор **else**.

<початкове значення> **while** <умова>) **:** <блок інструкцій тіла циклу> <зміна початкового значення> **[else :** <інструкції, які не виконуються, якщо не використовується оператор break> **]**	**for** <змінна циклу> **in** <об'єкт> : <блок інструкцій тіла циклу> **[else :** <блок інструкцій> *# виконується, якщо не використовується оператор break* **]**

Вкладені цикли

Циклічні алгоритми, розглянуті раніше, не містять у собі інших циклів, тому їх називають **простими циклами**.

Вкладені цикли — це такі цикли, що містяться в іншому циклі.

Цикл, що входить до складу іншого циклу, називають **внутрішнім**, а цикл, який містить інший цикл, — **зовнішнім**.

Приклад 8. Обчислити площі 30 прямокутників, сторони яких набувають таких значень: i = 3, 4, 5, 6, 7; j = 2, 4, 6, 8, 10, 12.

```
>>> for i in range (3,8,1):# зовнішній цикл
        for j in range (2,13,2):# внутрішній цикл
                s=i*j
                print('s=',s)
```

Хід роботи

ЗАВДАННЯ 1.

1. *Скласти блок-схему алгоритму розв'язування задачі.*

2. *Написати програму реалізації алгоритму з обов'язковими коментарями*

3. *Проаналізувати достовірність отриманих результатів обчислень.*

Варіанти завдань

1. Знайти суму кубів всіх натуральних чисел від 5 до 50.

2. Знайти суму всіх двозначних непарних чисел.

3. Знайти суму всіх двозначних парних чисел.

4. Знайти суму $\sum_{k=1}^{n} k^k$.

5. Знайти суму квадратів всіх натуральних чисел від 10 до 100.

6. Знайти суму $\sum_{k=1}^{n} \frac{1}{k!}$.

7. Знайти суму квадратів всіх двозначних парних чисел.

8. Знайти суму $\sum_{k=1}^{n} \frac{2}{k^3}$.

9. Знайти суму квадратів всіх двозначних непарних чисел.

10. Дано натуральне число n. Визначити суму цифр у числі.

11. Знайти всі двозначні числа, що діляться одночасно на 2 і на 3.

12. Знайти суму $\sum_{k=1}^{n} \frac{1}{k^2}$.

13. Знайти всі тризначні числа, що діляться одночасно на 4 і на 5.

14. Знайти всі тризначні числа, що діляться одночасно на 16 і на 24.

15.Дано натуральне число n. Визначити $(2n+1)!$

16.Знайти всі двозначні числа, для яких число дорівнює квадратові суми його цифр.

17.За даним натуральним значенням змінної n обчислити:

$$\underbrace{\sqrt{2+\sqrt{2+\ldots\sqrt{2}}}}_{n\ разів}$$

18.За даним натуральним значенням змінної n обчислити:

$$\underbrace{\sqrt{5+\sqrt{5+\ldots\sqrt{5}}}}_{n\ разів}$$

19.Знайти суму $s=\frac{1}{1\cdot 2}+\frac{1}{2\cdot 3}+\frac{1}{3\cdot 4}+\cdots+\frac{1}{n(n+1)}$.

ЗАВДАННЯ 2. *Протабулювати задану функцію*, тобто обчислити значення функції на відрізку $[a;\ b]$ в точках $x_i=a+ih$, де $h=\frac{b-a}{k}$, k – задане натуральне число. Результати обчислень вивести у вигляді пар чисел $x,\ y$. Число k вводиться з клавіатури.

1. *Скласти блок-схему алгоритму розв'язування задачі.*

2. *Написати програму реалізації алгоритму з обов'язковими коментарями*

3. *Проаналізувати достовірність отриманих результатів обчислень.*

Варіанти завдань

№	Функція	Відрізок
1	$y=\sin\left(x^2+4\right)$	$[0;\ 5]$
2	$y=\ln\left(x^2+x+1\right)$	$[-2{,}5;\ 2{,}5]$
3	$y=e^{3x^2+1}$	$[0;\ 2]$
4	$y=\cos\left(x^2+4\ln x\right)$	$[1;\ 6]$

5	$y = x^{\log_2(x^2+1)}$	$[2,2;\ 8,3]$
6	$y = e^{\lvert\sin(x^2+4)\rvert}$	$[-3;\ 2]$
7	$y = \dfrac{\sqrt{\lvert x^2-4\rvert}}{e^x}$	$[-5;\ 3]$
8	$y = \dfrac{\sqrt{\lvert x^2-4x+3\rvert}}{\sin x}$	$[-3,2;\ -2]$
9	$y = \sin\left(x^3 + 4\log_3 x\right)$	$[2;\ 9]$
10	$y = \cos\left(x^2\right) - \sin^2 x$	$[-2,5;\ 5]$
11	$y = \cos\left(x^2 + 4\lvert x-1\rvert\right)$	$[-1;\ 4,5]$
12	$y = \log_3\left(x^2 + x + 2\right)$	$[-2,1;\ 2,5]$
13	$y = e^{3x^2+\lvert x+2\rvert}$	$[-4;\ 2]$
14	$y = \sin\left(\ln x + e^x\right)$	$[0,1;\ 5]$
15	$y = 3^{\log_2(2x^2-2)}$	$[2,2;\ 5,2]$
16	$y = 3^{\lvert\sin(x^2-1)\rvert}$	$[-2;\ 5,2]$
17	$y = \dfrac{\sqrt{\lvert x^2-1\rvert}}{3^{x^2-1}}$	$[-2;\ 4]$
18	$y = \dfrac{\sqrt{\lvert x^2-5x-6\rvert}}{\cos x}$	$[-1,6;\ 1,6]$
19	$y = \sin\left(\log x^3 + \lvert x\rvert\right)$	$[1,2;\ 4,2]$
20	$y = \sin\left(e^x\right) - \ln^2 x$	$[1,5;\ 5,5]$

Контрольні запитання

1. Що називається циклом? Які типи циклів ви знаєте?
2. Призначення функції **range**()?
3. Коли застосовують оператор **for**?
4. Як записується цикл **for** у випадку зростаючих і спадних значень параметра?
5. Що таке цикл з післяумовою?

Лабораторна робота № 6

ТЕМА: **Рекурентні формули**

МЕТА: *вміти створювати програми для обробки простих типів даних із використанням рекурентних формул*

ТЕОРЕТИЧНІ ВІДОМОСТІ

Серед задач, для розв'язування яких необхідна побудова циклічних конструкцій, велику частку займають задачі додавання, до яких належать задачі обчислення n-го члена (суми n членів) арифметичних та геометричних прогресій та обчислення суми нескінченних числових рядів.

Рекурентний спосіб задання послідовностей

Рекурентна формула (від лат. *recurrentis* – що повертається) – формула, що зводить обчислення n-го члена якої-небудь послідовності до обчислення декількох попередніх її членів.

Рекурентна формула формула, що виражає загальний (n-й) член послідовності через попередні її члени; як-от:

$$a_n = 2a_{n-1} + 3a_{n-2}.$$

Прикладами рекурентних формул є формули для обчислення наступного елемента арифметичної чи геометричної прогресії, якщо відомо попередній елемент і різниця чи знаменник прогресії:

$$a_{n+1} = a_n + d,\ n = 0,\ 1,\ 2,\ldots$$

$$b_{n+1} = b_n \cdot q,\ n = 0,\ 1,\ 2,\ldots$$

У задачах про прогресії зазвичай відомо значення першого елемента, різниці чи знаменника, що достатньо для їхнього розв’язування.

Алгоритм обчислення n-го члена прогресії

Означення арифметичної прогресії:

A[0]=A;

A[n]=A[n-l]+d; n=1,2,...

Задаємо позначення:

A - перший член послідовності; **d** - різниця арифметичної прогресії; **N** - номер поточного члена прогресії; **K** -задана кількість членів; **S** - сума N членів прогресії.

Алгоритм обчислення n-го члена послідовності та суми n членів складається з виконання таких дій:

1. Задаються значення першого члена послідовності A, різниці d, кількості членів K, які потрібно обчислити;
2. Задаються початкові значення лічильника членів послідовності (N:=0) і суми (S:=0);
3. Поки номер N члена послідовності, який обчислено, не досягне значення K - заданої кількості членів, повторюються дії: номер поточного члена збільшується на 1; обчислюється значення наступного члена A; обчислене значення A додається до суми S.

Приклад 1. *Дана послідовність 3, 7, 11, 15... Які значення потрібно надати A, d, K, щоб отримати суму 10 членів цієї послідовності?*

Рекурентна залежність*:* $\mathbf{A_n = A_{n-1} + 4; A_0 = -1.}$

Оскільки задана рекурентна залежність для послідовності, то для знаходження суми перших **K** членів необхідно обчислити значення кожного члена послідовності до **K-го** включно, а потім знайти їх суму.

Змінна **d** відповідає за крок арифметичної прогресії, тобто за різницю між кожним членом прогресії та попереднім членом. В даному випадку, якщо перший член прогресії дорівнює **3**, а другий - **7**, то **d** буде рівним **7-3=4**.

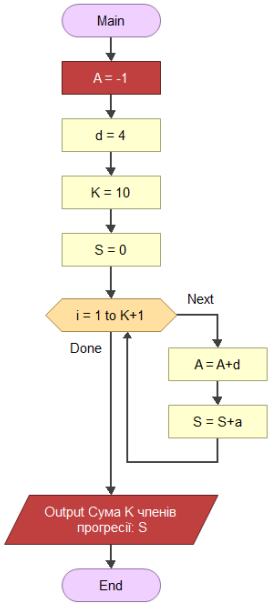
У загальному випадку формула члена арифметичної прогресії має вигляд **A=A+d**, де **A**- перший член прогресії, **d** - крок арифметичної прогресії, а **i** - номер члена прогресії.

Варіант програми з циклом **For:**

```
# задаємо значення першого члена та різницю арифметичної прогресії
A=-1
d=4

# задаємо кількість членів прогресії, для яких потрібно знайти суму
K=10

# обчислюємо суму перших K членів прогресії
S = 0
for i in range(1,K+1):    # або for i in range(K):
    A=A+d
    S=S+A

# виводимо результат
print("Сума {} членів прогресії дорівнює {}".format(K, S))
```

Оболонка

```
>>> %Run -c $EDITOR_CONTENT
Сума 10 членів прогресії дорівнює 210
```



Варіант програми з циклом **while:**

```
A = 3
d = 4
K = 10  # Кількість елементів
sum = 0
n = 0
while n < K:
    # Виводимо поточний елемент послідовності
    print(A)
    sum += A
    # Обчислюємо наступний елемент послідовності
    A += d
    n += 1
# Сума перших 10 елементів
print("Сума перших 10 елементів: ", sum)
```

Оболонка

```
>>> %Run -c $EDITOR_CONTENT
3
7
11
15
19
23
27
31
35
39
Сума перших 10 елементів:  210
```

Метод ітерацій

Ітераціями називаються послідовні уточнення значення деякої величини, отримані в результаті застосування рекурентних формул.

Уточнення кореня – проведення обчислень для перевірки досягнення даної точності.

Наприклад, квадратний корінь будь-якого додатного числа n наближено можна визначити за такою рекурентною формулою:

$$b_{i+1} := (b_i + n / b_i) / 2,\ i = 1,\ 2,\ ...,\ \text{де}\ b_1 = n.$$

Алгоритм:

1. Позначити через b_1 будь-яке початкове наближення до шуканої величини, ввести n і присвоїти $b_1 := n.$
2. Позначити через e задану точність і присвоїти $e := 0,01,\ i := 1.$
3. Обчислити $b_2 := (b_1 + n / b_1) / 2.$
4. Доки $|b_2 - b_1| > e$, виконати обчислення

$$b_1 := b_2,$$

$$b_2 := (b_1 + n / b_1) / 2,$$

$$i := i + 1.$$

5. Після виходу з циклу **while** останнє значення (b_2) прийняти за відповідь.
6. Вивести кількість ітерацій i.

Особливістю ітераційних циклів є те, що наперед невідома кількість циклів, які виконуються. Обчислення в циклі припиняється при досягненні заданої точності.

Ітераційні цикли будуються за допомогою операторів умовного і безумовного переходів, а також використовуються структури організації циклів з передумовою чи з післяумовою.

Приклад 2. Знайти корінь рівняння cos(x)+2x-5=0 на інтервалі [0;1] з точністю ε =0,0001, користуючись методом ітерацій.

Запишемо рівняння у вигляді $x_{i+1}=(5-\cos(x_i))/2$ і наведемо початкове значення кореня x=0,5.

Таблиця ідентифікації змінних:

Змінна	x_i	x_{i+1}	**t**	**e**	**Кількість ітерацій**
Ідентифікатор	x	y	t	e	n

```
# Методом ітерацій
from math import  cos
x=0.5
e=0.0001
n=0
while True:
    y=(5-cos(x))/2
    t=abs(x-y)
    x=y
    n=n+1
    if (t<e):break
print('Відповідь:')
print('Корінь рівняння y=',y)
print('Кількість ітерацій n=:',n)
input()
```

```
Відповідь:
Корінь рівняння y= 2.9946074951082116
Кількість ітерацій n=: 7
```

Числа Фібоначчі

У математиці серед «золотих» чисел почесне місце посіли числа Фібоначчі (Леонардо Пізанський, XIII ст.). Кажуть, що свого часу відомий учений в послідовності цих чисел відобразив закон розмноження кроликів. Ці числа мають такий вигляд:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

Можна помітити, що, починаючи з третього числа, кожне наступне дорівнює сумі двох попередніх.

1 + 1 = 2

1 + 2 = 3

2 + 3 = 5

3 + 5 = 8 і т. д.

Тобто, починаючи з третього члена цієї послідовності існує така залежність:

$$f_n = f_{n-2} + f_{n-1}$$

Цю задачу сформулював і розв'язав італієць Фібоначчі у 1228 році.

Приклад 3. Знайти *N*-й член послідовності Фібоначі

Є послідовність чисел, де перші два числа – це 1 та 1, треба знайти інші. Третім числом буде 2. Розв'язок дають числа, які отримують за таким правилом: кожне наступне число (c) в послідовності є сумою двох попередніх (a та b). Ці числа називаються числами Фібоначчі. Визначимо дванадцять чисел послідовності.

```
# Числа Фібоначчі
print ('Числа Фібоначчі')
n=int(input('n=')) #кількість чисел в послідовності
a=1 # Список чисел Фібоначчі
b=1 # (напочатку має дві одиниці)
print (1)
print (1)
for i in range(1,n+1):
    c=a+b
    a=b
    b=c
    print(c)
input()
```

```
Числа Фібоначчі
n=10
1
1
2
3
5
8
13
21
34
55
89
144
```

Метод Ньютона для визначення кореня рівняння

Нехай потрібно обчислити корінь рівняння $f(x) = n$. Рекурентна формула методу Ньютона для заданого рівняння така:

$$x_{i+1} = x_i - \left(f(x_i) - n \right) / f'(x_i),$$

де $i = 1, 2, \ldots$, а x_1 – визначене „на око” початкове значення кореня. Воно має бути достатньо близьким до шуканого кореня. Визначити його можна, побудувавши графік функції $f(x) - n$. Цикл закінчити, якщо $|x_{i+1} - x_i| < e$, де e – задана точність, а f' – це похідна від функції f.

```
# Метод Ньютона
import math
print ('Введіть точність <0.1')
e=float(input('e='))
n=1
x1=-21 #початкове наближення кореня x1
x=x1-(math.sin(x1)-x1-20)/(math.cos(x1)-1)#друге наближення кореня x_2
while True:
    x1=x #i-те наближення кореня x_i
    x=x-(math.sin(x)-x-20)/(math.cos(x)-1)#i+1-е наближення кореня x_{i+1}
    n=n+1#лічильник кількості ітерацій
    if abs(x-x1)<e:break
print('x=',x)
print ('Кількість ітерацій n=',n)
input()
'
Введіть точність <0.1
e=0.0001
x= -20.89118901566636
Кількість ітерацій n= 3
```

Хід роботи

ЗАВДАННЯ 1. Обчислити суму членів послідовності. Скласти програму для обчислення з точністю до e суми членів послідовності, заданої формулою $a_n = f(x)$ $(n = 1, 2, \ldots)$. Додавати члени послідовності слід до тих пір, поки виконується умова $|a_n| \geq e$. *У варіантах 1-10 для організації циклу використати оператор while (з передумовою), у варіантах 11 – 20 для організації циклу використати оператор while (з післяумовою).*

1. *Скласти блок-схему алгоритму розв’язування задачі.*
2. *Написати програму реалізації алгоритму з обов’язковими коментарями.*
3. *Проаналізувати достовірність отриманих результатів обчислень.*

Варіанти завдань:

№	$a_n = f(x)\ (n = 1, 2, \ldots)$	e
1.	$a_n = \dfrac{n^2}{n+2}$	0,001
2.	$a_n = \sin\dfrac{1}{n}$	0,001
3.	$a_n = \dfrac{1}{n!}$	0,0001
4.	$a_n = \dfrac{n^3}{n+1} \cdot e^{-n}$	0,0001
5.	$a_n = \dfrac{\ln(n+1)}{n}$	0,001
6.	$a_n = \dfrac{(-1)^n}{n} \cdot \cos n$	0,001
7.	$a_n = \dfrac{arctg\, n}{n}$	0,001
8.	$a_n = \dfrac{3^n}{n!}$	0,0001
9.	$a_n = (-1)^{n-1} \cdot \dfrac{\sin n}{n^2}$	0,001
10.	$a_n = (-1)^n \cdot \dfrac{4^n}{(2n)!}$	0,0001
11.	$a_n = n^3 \cdot e^{-n^2}$	0,0001
12.	$a_n = (-1)^n \cdot \dfrac{1}{n^2+1}$	0,0001
13.	$a_n = (-1)^{n+1} \cdot \dfrac{0{,}5^{2n-1}}{2n-1}$	0,001
14.	$a_n = (-1)^n \cdot \dfrac{(n+5)^2}{n!}$	0,001
15.	$a_n = \arcsin\dfrac{1}{n+2}$	0,0001
16.	$a_n = \sin\dfrac{1}{n}$	0,001

17.	$a_n = \frac{1}{n!}$	0,0001
18.	$a_n = \frac{n^3}{n+1} \cdot e^{-n}$	0,0001
19.	$a_n = \frac{\ln(n+1)}{n}$	0,001
20.	$a_n = \frac{n}{n^2+1}$	0,001

ЗАВДАННЯ 2. *Знаходження членів послідовностей.*

1. *Скласти блок-схему алгоритму розв'язування задачі.*
2. *Написати програму реалізації алгоритму з обов'язковими коментарями.*
3. *Проаналізувати достовірність отриманих результатів обчислень.*

Варіанти завдань

1. Обчисліть 12 значень елементів послідовності, яка утворюється за допомогою такої рекурентної формули: $a_{n+2} = a_{n+1} - a_n$, де $a_0 = 1,\ a_1 = 2.$

2. Обчисліть 14 значень елементів послідовності, яка утворюється за допомогою такої рекурентної формули: $a_{n+2} = (a_{n+1} + a_n)/2$, де $a_0 = 1,\ a_1 = 2.$

3. Обчисліть 12 значень елементів послідовності, яка утворюється за допомогою такої рекурентної формули: $a_{n+2} = a_{n+1} + a_n$, де $a_0 = 1,\ a_1 = 2.$

4. Обчисліть 10 значень елементів послідовності, яка утворюється за допомогою такої рекурентної формули: $a_{n+2} = (2a_{n+1} - 3a_n)^2$, де $a_0 = 1,\ a_1 = 3.$

5. Обчисліть 12 значень елементів послідовності, яка утворюється за допомогою такої рекурентної формули: $a_{n+1} = 3a_n - 2$, де $a_0 = 1,\ a_1 = 2.$

6. Обчисліть 14 значень елементів послідовності, яка утворюється за такої рекурентної формули: $a_{n+2} = 4a_{n+1} - 3a_n$, де $a_0 = 1,\ a_1 = 2.$

7. Обчисліть 12 значень елементів послідовності, яка утворюється за допомогою такої рекурентної формули: $a_{n+2} = 3a_{n+1} + 2a_n$, де $a_0 = 1,\ a_1 = 2.$

8. Обчисліть 12 значень елементів послідовності, яка утворюється за допомогою такої рекурентної формули: $a_{n+2} = (a_{n+1} + a_n)^2$, де $a_0 = 1,\ a_1 = 2.$

9. Обчисліть 10 значень елементів послідовності, яка утворюється за допомогою такої рекурентної формули: $a_{n+2} = 7a_{n+1} - 3a_n$, де $a_0 = 1,\ a_1 = 3.$

10. Обчисліть 12 значень елементів послідовності, яка утворюється за допомогою такої рекурентної формули: $a_{n+1} = (3a_n - 2)^2$, де $a_0 = 1,\ a_1 = 2.$

11. Обчисліть 14 значень елементів послідовності, яка утворюється за допомогою такої рекурентної формули: $a_{n+2} = 5a_{n+1} - 7a_n$, де $a_0 = 1,\ a_1 = 2.$

12. Обчисліть 12 значень елементів послідовності, яка утворюється за допомогою такої рекурентної послідовності, яка утворюється за допомогою такої рекурентної формули: $a_{n+2} = (3a_{n+1} + 2a_n)^2 + 1$, де $a_0 = 1,\ a_1 = 2.$

13. Обчисліть 12 значень елементів послідовності, яка утворюється за допомогою рекурентної формули: $a_{n+2} = 3a_{n+1} + 7a_n$, де $a_0 = 2,\ a_1 = 3.$

14. Обчисліть 10 значень елементів послідовності, яка утворюється за допомогою такої рекурентної формули: $a_{n+2} = (2a_{n+1} - 3a_n)^3 + 2$, де $a_0 = 1,\ a_1 = 3.$

15. Обчисліть 12 значень елементів послідовності, яка утворюється за допомогою такої рекурентної послідовності, яка утворюється за допомогою такої рекурентної формули: $a_{n+1} = 3a_n - 2a_{n-1} + 1$, де $a_0 = 1,\ a_1 = 2.$

16. Обчисліть 14 значень елементів послідовності, яка утворюється за допомогою такої рекурентної формули: $a_{n+2} = 4a_{n+1} - 3a_n + 2$, де $a_0 = 2,\ a_1 = 2.$

17. Перший член і знаменник зростаючої геометричної прогресії дорівнюють відповідно b_0 і q. Виведіть десять перших членів прогресії.

18. Перший член і різниця арифметичної прогресії дорівнюють відповідно a_0 і d (задайте їх довільно). Виведіть десять перших членів прогресії.

19. Перший член і знаменник спадної геометричної прогресії дорівнюють відповідно b_0 і q. Виведіть десять перших членів прогресії.

20. Обчисліть 12 значень елементів послідовності, яка утворюється за допомогою такої рекурентної формули: $a_{n+2} = 3a_{n+1} + 2a_n - 1$, де $a_0 = 1,\ a_1 = 1.$

Контрольні запитання

1. Що таке рекурентна формула?
2. Що таке ітерації?
3. Який вигляд має рекурентна формула для обчислення кореня квадратного з додатного числа?
4. Розкрийте суть методу **Ньютона**?
5. Як утворюються числа **Фібоначчі**?
6. Запишіть рекурентну формулу для обчислення чисел **Фібоначчі**.

Лабораторна робота № 7

ТЕМА: Проектування та налагодження програм оброблення одновимірних списків (масивів).

МЕТА: вміти створювати програми обробки та сортування одновимірних масивів

ТЕОРЕТИЧНІ ВІДОМОСТІ

Масив – це структура даних, що являє собою однорідну (за типом), фіксовану (за розміром і конфігурацією) сукупність елементів, упорядкованих за номерами.

Масив визначається ім'ям (*ідентифікатором*) і *кількістю індексів* (*номерів*), що потрібні для визначення місцезнаходження необхідного елементу масиву.

Ім'я масиву є *єдиним* для всіх його елементів.

У програмуванні кількість індексів масиву (таблиці) називають *розмірністю масиву*.

Список — це певна сукупність об'єктів будь-якого типу в квадратних дужках, які відокремлюються один від одного комою. Об'єктами списку є числа, рядки та список. У списках можна змінювати значення його елементів, збільшувати та зменшувати кількість елементів, здійснювати пошук потрібних елементів і впорядковувати його елементи.

Отже, список є об'єктом, що змінюється. Самі списки можуть бути вкладені в інші типи об'єктів.

```
[5, 'python', 'pascal', 21, [1, 6, 3]]
```

Списки деякою мірою нагадують масиви, але в масивах значення елементів можуть бути лише одного типу. Мова **Python** має значну кількість операцій, функцій і методів опрацювання списків.

Списки можуть бути *одновимірними* і *багатовимірними*.

Одномірні списки. Позиція елемента у списку задається індексом, який починається з нуля. Списки можна створювати простим перерахуванням елементів списку в квадратних дужках:

```
>>> a=[5,  21, 1, 2, 3]
>>> a
[5, 21, 1, 2, 3]
```

Доступ до кожного елемента одновимірного списку (масиву) здійснюється за індексом:

<ім'я списку> [номер]

Наприклад, **a**[0] =5; (елементу списку (масиву) **a** з індексами 0 надано значення 5).

Методи списків

Мова Python має значну кількість операцій, функцій і методів опрацювання списків.

1. append(): додає елемент в кінець списку.

```
my_list = [1, 2, 3]
my_list.append(4)
print(my_list)  # [1, 2, 3, 4]
```

Оболонка

```
>>> %Run -c $EDITOR_CONTENT
  [1, 2, 3, 4]
```

2. extend(): додає елементи з іншого списку в кінець поточного списку.

```
my_list = [1, 2, 3]
other_list = [4, 5, 6]
my_list.extend(other_list)
print(my_list)  # [1, 2, 3, 4, 5, 6]
```

Оболонка

```
>>> %Run -c $EDITOR_CONTENT
  [1, 2, 3, 4, 5, 6]
```

3. insert(): додає елемент на задану позицію у списку.

```
my_list = [1, 2, 3]
my_list.insert(1, 4)
print(my_list)  # [1, 4, 2, 3]
```

Оболонка

```
>>> %Run -c $EDITOR_CONTENT
  [1, 4, 2, 3]
```

4. remove(): видаляє перший зустрічний елемент зі списку.

```
my_list = [1, 2, 3, 2]
my_list.remove(2)
print(my_list)  # [1, 3, 2]
```

Оболонка

```
>>> %Run -c $EDITOR_CONTENT
  [1, 3, 2]
```

5. pop(): видаляє та повертає останній елемент списку.

```
my_list = [1, 2, 3]
last_item = my_list.pop()
print(last_item)  # 3
print(my_list)    # [1, 2]
```

Оболонка

```
>>> %Run -c $EDITOR_CONTENT
  3
  [1, 2]
```

6. index(): повертає індекс першого зустрічного елемента у списку.

```
my_list = [1, 2, 3, 2]
index_of_2 = my_list.index(2)
print(index_of_2)  # 1
```

Оболонка

```
>>> %Run -c $EDITOR_CONTENT
  1
```

7. count(): повертає кількість зустрічних елементів у списку.

```
my_list = [1, 2, 3, 2]
count_of_2 = my_list.count(2)
print(count_of_2)  # 2
```

Оболонка

```
>>> %Run -c $EDITOR_CONTENT
  2
```

8. sort(): сортує елементи у списку.

```
my_list = [3, 1, 2]
my_list.sort()
print(my_list)  # [1, 2, 3]
```

Оболонка

```
>>> %Run -c $EDITOR_CONTENT
  [1, 2, 3]
```

9. reverse(): змінює порядок елементів у списку на протилежний.

```
my_list = [1, 2, 3]
my_list.reverse()
print(my_list)  # [3, 2, 1]
```

Оболонка

```
>>> %Run -c $EDITOR_CONTENT
  [3, 2, 1]
```

У списку є також деякі вбудовані функції **Python**, які можна використовувати для обчислення різних статистичних даних:

1) **len()** - повертає кількість елементів у списку;
2) **sum()** - повертає суму всіх елементів у списку;
3) **min**() - повертає найменший елемент у списку;
4) **max**() - повертає найбільший елемент у списку.

Приклад 1. Знайти середнє арифметичне значення у одновимірному списку (масиву).

```
my_list = [1, 2, 3, 4, 5]
average = sum(my_list) / len(my_list)
print(average)
```

Оболонка

```
>>> %Run -c $EDITOR_CONTENT
  3.0
```

Заповнення та виведення одновимірного списку (масиву):

Заповнення одновимірного списку (масиву):	**Виведення елементів одновимірного списку (масиву):**
1) *за формулою:* `n=int(input("n="))` `a=[]` `for i in range (n+1):` `  a=i*i+2` `  print(a)` `input()` //будь-яка інша формула	***1)*** *виведення у стовпчик:* `for i in range (n):` `  print(a[i])`
2) *з клавіатури:* `n=int(input("n="))` `a=[]` `for i in range (n):` `  print("a[",i,"]=",sep=" ",end=" ")` `  a.append(int(input()))` `input()`	***2)*** *виведення у рядок:* `for i in range (n):` `  print(a[i],sep=" ",end=" ")`
3) *випадково (генератором випадкових чисел) із проміжку [a, b]:* `import random` `n=int(input("n="))` `a=[]` `for i in range (n):` `  a.append(random.randint(1,11))` `print(a)`	

Приклад 2. Піднести до квадрату всі елементи даного одновимірного списку (масиву).

```
#Піднести до квадрату всі елементи одновимірного масиву
# створюємо новий список
a = []
for i in range(7):
    # додаємо введені елементи до списку в області виконання проекту
    print ("a[",i,"]=",sep="",end="")
    a.append(int(input()))
    # змінюємо значення введеного елемента списку
    a[i] = a[i] ** 2
# виводимо результуючий список
print(a)
```

Модуль **random** містить функції, які генерують випадкові числа. На початку використання функцій необхідно цей модуль імпортувати у програму за допомогою команди import random. Деякі функції модуля **random:**

Функція	Призначення
random ()	Генерує випадкове число 0.0 до 1.0
	`>>> import random` `>>> random.random()` `0.019682079595158553`
uniform(початок, кінець)	Генерує дійсне випадкове число у діапазоні від «початок» до «кінець»
	`>>> import random` `>>> random.uniform(20,30)` `25.40198690187999`
randint (початок, кінець)	Генерує ціле випадкове число у діапазоні від «початок» до «кінець»
	`>>> import random` `>>> random.randint(5,10)` `8`
choice(послідовність)	Вибирає з послідовності (рядка, списку або кортежу) випадковий елемент
	`>>> import random` `>>> random.choice("Інформатика")` `'и'`

Щоб отримати повний список можна звернутися до документації https://docs.python.org/uk/3/library/random.html з модуля **random**.

Приклад 3. Заповнення масиву за допомогою модуля **random**

```
#Піднести до квадрату всі елементи одновимірного масиву
# створюємо новий список
import random
a = []
for i in range(10):
    # додаємо введені елементи до списку  модуль random
    a.append(random.randint(1,11))
print(a)
for i in range(10):
    # змінюємо значення введеного елемента списку
    a[i] = a[i] ** 2
# виводимо результуючий список
print(a)
```

Пошук мінімального (максимального) елемента масиву здійснюється послідовним порівнянням елементів масиву з поточним мінімумом – мінімальним елементом із відсканованої частини масиву. Якщо поточний елемент масиву менший від поточного мінімуму, то значення поточного мінімуму змінюється на значення поточного елемента. Алгоритм пошуку максимального елемента є аналогічним.

Приклад 4. Знайти мінімальний та максимальний елементи в масиві за допомогою вбудованих функцій **max** () і **min** ().

```
arr = [4, 2, 7, 1, 9, 3]

min_elem = min(arr)
max_elem = max(arr)

print("Мінімальний елемент:", min_elem)
print("Максимальний елемент:", max_elem)
```

```
Оболонка
>>> %Run -c $EDITOR_CONTENT
  Мінімальний елемент: 1
  Максимальний елемент: 9
```

Приклад 5. Знайти мінімальний та максимальний елементи в масиві за допомогою алгоритму пошуку:

```
arr = [4, 2, 7, 1, 9, 3]

# Знаходимо мінімальний елемент
min_elem = arr[0]
for i in range(1, len(arr)):
    if arr[i] < min_elem:
        min_elem = arr[i]

# Знаходимо максимальний елемент
max_elem = arr[0]
for i in range(1, len(arr)):
    if arr[i] > max_elem:
        max_elem = arr[i]

print("Мінімальний елемент:", min_elem)
print("Максимальний елемент:", max_elem)
```

```
Оболонка
>>> %Run -c $EDITOR_CONTENT
  Мінімальний елемент: 1
  Максимальний елемент: 9
```

Упорядкування масиву

Упорядкування (сортування) – це розміщення елементів за певним порядком. Елементи масиву можна впорядковувати за зростанням, спаданням значень та за іншими ознаками.

Є багато методів упорядкування масивів, які поділяються на **прямі** і **покращені**. До **прямих** методів належать:

- *метод обміну (метод "бульбашки");*
- *метод екстремальних елементів;*
- *метод вставок.*

До **покращених** методів належать: метод Шелла, Хоара, пірамідального впорядкування, злиття та ін.

Кожний метод характеризується кількістю порівнянь і перестановок елементів, які асимптотично залежать від кількості елементів n (інколи цю характеристику називають порядком чи складністю методу), обсягом використаної пам’яті та часом виконання програми.

Покращені методи впорядкування використовують меншу кількість порівнянь та перестановок елементів. Як правило вони більш ефективні для упорядкування великих масивів даних. Для невеликих масивів даних краще використовувати прямі методи, які не потребують додаткової оперативної пам’яті.

Метод обміну ("бульбашки")

«*Бульбашкове*» сортування полягає в тому, що елементи масиву порівнюються *попарно* та *послідовно*. У цьому методі спочатку розглядають перших два елементи масиву. Якщо перший елемент більший від другого, то їх міняють місцями. Далі другий елемент порівнюють з третім і, якщо другий більший від третього, то їх також міняють місцями і т.д. Якщо у масиві n елементів, то після n-1 порівнянь максимальний елемент буде розташовано в кінці масиву.

Далі розглядають даний масив без останнього елемента і застосовують до нього ту саму процедуру, в результаті чого другий за величиною елемент масиву буде розташовано на передостанньому місці і т.д. Таким чином, упорядковані елементи будуть накопичуватися в кінці масиву і за n-1 таких процедур масив буде впорядковано за зростанням.

Наприклад, `a= [4, 5, 2, 1, 3]`

```
a[0]=4
a[1]=5
a[2]=2
a[3]=1
a[4]=3
Впорядкування - Метод Бульбашки
[4, 2, 5, 1, 3]
[4, 2, 1, 5, 3]
[4, 2, 1, 3, 5]
[2, 4, 1, 3, 5]
[2, 1, 4, 3, 5]
[2, 1, 3, 4, 5]
[1, 2, 3, 4, 5]
```

Назва методу "бульбашки" пояснюється тим, що елементи переміщаються до кінця масиву подібно до повітряних бульбашок у воді.

Приклад 6. Піднести до квадрату всі елементи даного одновимірного масиву.

Виконати сортування елементів масиву за зростанням, використовуючи метод обміну.

```
#Піднести до квадрату всі елементи одновимірного масиву
#Виконати сортування елементів масиву з зростанням, метод "Бульбашки"
# створюємо новий список
n=5
a = []
for i in range(n):
      # додаємо введені елементи до списку в області виконання проекту
     print ("a[",i,"]=",sep="",end="")
     a.append(int(input()))
print ('Впорядкування - Метод Бульбашки')
for j in range(n-1):
    for i in range(n-j-1):
        if a[i] > a[i+1]:
            a[i+1], a[i] = a[i], a[i+1]
    for i in range(n):
        print
print ('Впорядкований масив a=',a)
for i in range(n):
    # змінюємо значення введеного елемента списку
     a[i] = a[i] ** 2
     # виводимо результуючий список
print('Масив до квадрату')
print(a)
```

Метод екстремальних елементів

У цьому методі спочатку знаходять мінімальний (максимальний) елемент масиву і міняють його місцями з першим (останнім) елементом залежно від типу впорядкування. Далі розглядають масив без першого елемента, в якому роблять те ж саме і т. д. Таким чином, упорядковані елементи будуть накопичуватися на початку масиву. Якщо у масиві n елементів, то після n-1 кроків масив буде впорядковано за зростанням (спаданням).

Наприклад, `a= [4, 5, 2, 1, 3]`

```
a[0]=4
a[1]=5
a[2]=2
a[3]=1
a[4]=3
Впорядкування - Метод екстремальних (min) елементів
j= 0
[1, 5, 4, 2, 3]
j= 1
[1, 2, 5, 4, 3]
j= 2
[1, 2, 3, 5, 4]
j= 3
[1, 2, 3, 4, 5]
Впорядкований масив a= [1, 2, 3, 4, 5]
```

Для того, щоб описаний алгоритм виконував упорядкування масиву за спаданням, необхідно логічний вираз **a[i]<a[min]** замінити на **a[i]>a[min].** Можна також замінити ідентифікатор **min** на ідентифікатор **max**.

Приклад 7. Піднести до квадрату всі елементи одновимірного масиву Виконати сортування елементів масиву за зростанням методом мінімальних елементів.

```
#Піднести до квадрату всі елементи одновимірного масиву
#Виконати сортування елементів масиву за зростанням, методом мінімальних елементів
# створюємо новий список
n=5
a = []
for i in range(n):
      # додаємо введені елементи до списку в області виконання проекту
     print ("a[",i,"]=",sep="",end="")
     a.append(int(input()))
print ('Впорядкування - Метод екстремальних (min) елементів')
for j in range(n-1):
    min=j
    for i in range(j,n):
        if a[i] < a[min]:
            a[i], a[min] = a[min], a[i]
print ('Впорядкований масив a=',a)
for i in range(n):
    # змінюємо значення введеного елемента списку
     a[i] = a[i] ** 2
     # виводимо результуючий список
print('Масив до квадрату')
print(a)
```

Результат виконання програми:

```
a[0]=4
a[1]=5
a[2]=2
a[3]=1
a[4]=3
Впорядкування - Метод екстремальних (min) елементів
Впорядкований масив a= [1, 2, 3, 4, 5]
Масив до квадрату
[1, 4, 9, 16, 25]
```

Сортування елементів списку в **Python** реалізується за допомогою **методу sort ()**, який має таку загальну структуру: **sort ([key = None][,reverse =False]).** Як бачимо, параметри є не обов'язковими. За замовчуванням сортування виконується за зростанням значень елементів з урахуванням регістра. Для сортування за зменшенням слід указати другий параметр таким: **reverse = True.** Метод **sort ()** перетворює старий список у новий.

```
>>> al = [17, 100, 20, 3, 8, 16, 25]
>>> al.sort (reverse = True)
>>> al
[100, 25, 20, 17, 16, 8, 3]
```

Метод **sorted (**послідовність **['key=None][reverse =False])** — призначений для сортування списку і збереження старого списку.

Приклад 8. Сортування за зростанням

```
#Піднести до квадрату всі елементи одновимірного масиву
# створюємо новий список
a = []
for i in range(7):
    # додаємо введені елементи до списку в області виконання проекту
    print ("a[",i,"]=",sep="",end="")
    a.append(int(input()))
    # змінюємо значення введеного елемента списку
    a[i] = a[i] ** 2
# виводимо результуючий список
print(a)
for i in range(10):
        a.sort()
print('Відсортований масив за зростанням: ',a)
print('max = {0}; min = {1}'.format(a[-1], a[0]))
```

Приклад 9. Сортування за спаданням

```
#Піднести до квадрату всі елементи одновимірного масиву
# створюємо новий список
a = []
for i in range(7):
    # додаємо введені елементи до списку в області виконання проекту
    print ("a[",i,"]=",sep="",end="")
    a.append(int(input()))
    # змінюємо значення введеного елемента списку
    a[i] = a[i] ** 2
# виводимо результуючий список
print(a)
for i in range(10):
    a.sort()
print('Впорядкований список за зростанням: ', a)
a.sort(reverse = True)
print('Впорядкований список за спаданням: ', a)
```

Хід роботи

ЗАВДАННЯ 1. Обробка одновимірних списків (масивів).

1. *Скласти блок-схему алгоритму розв'язування задачі.*

2. *Написати програму реалізації алгоритму з обов'язковими коментарями.*

3. *Проаналізувати достовірність отриманих результатів обчислень.*

Варіанти завдань

1. Збільшити вдвічі всі додатні елементи даного одновимірного масиву. Виконати сортування елементів масиву за спаданням, використовуючи метод обміну.
2. Підрахувати кількість від'ємних елементів у даному одновимірному масиві. Виконати сортування елементів масиву за зростанням, використовуючи метод обміну.
3. Підрахувати кількість додатних елементів у даному одновимірному масиві. Виконати сортування елементів масиву за спаданням, використовуючи метод екстремальних елементів.
4. Підрахувати суму від'ємних елементів у даному одновимірному масиві. Виконати сортування елементів масиву за зростанням, використовуючи метод екстремальних елементів.

5. Піднести до кубу всі елементи даного одновимірного масиву. Виконати сортування елементів масиву за спаданням, використовуючи метод обміну.

6. Підрахувати суму додатних елементів у даному одновимірному масиві. Виконати сортування елементів масиву за зростанням, використовуючи метод обміну.

7. Підрахувати середнє арифметичне всіх елементів у даному одновимірному масиві. Виконати сортування елементів масиву за спаданням, використовуючи метод екстремальних елементів.

8. Підрахувати середнє арифметичне всіх додатних елементів у даному одновимірному масиві. Виконати сортування елементів масиву за зростанням, використовуючи метод екстремальних елементів.

9. Підрахувати середнє арифметичне всіх від'ємних елементів у даному одновимірному масиві. Виконати сортування елементів масиву за спаданням, використовуючи метод обміну.

10. Знайти найбільший елемент одновимірного масиву. Виконати сортування елементів масиву за зростанням, використовуючи метод обміну.

11. Знайти найменший елемент одновимірного масиву. Виконати сортування елементів масиву за спаданням, використовуючи метод екстремальних елементів.

12. Знайти різницю між найбільшим і найменшим елементами одновимірного масиву. Виконати сортування елементів масиву за зростанням, використовуючи метод екстремальних елементів.

13. Знайти суму квадратів елементів одновимірного масиву. Виконати сортування елементів масиву за спаданням, використовуючи метод обміну.

14. Знайти довжину заданого вектора ($|a| = \sqrt{a_1^2 + a_2^2 + ... + a_n^2}$). Виконати сортування елементів масиву (вектора) за зростанням, використовуючи метод обміну.

15. Знайти скалярний добуток двох векторів (обов'язково мають однакову розмірність). Виконати сортування елементів масивів за спаданням, використовуючи метод екстремальних елементів.

16. Знайти всі числа, які входять у масив лише один раз. Виконати сортування елементів масиву за зростанням, використовуючи метод екстремальних елементів.

17. Знайти кількість різних чисел даного одновимірного масиву. Виконати сортування елементів масиву за спаданням, використовуючи метод обміну.

18. Замінити всі додатні елементи у даному одновимірному масиві їх номерами. Виконати сортування елементів масиву за спаданням, використовуючи метод екстремальних елементів.

19. Замінити всі від'ємні елементи у даному одновимірному масиві їх номерами. Виконати сортування елементів масиву за зростанням, використовуючи метод екстремальних елементів.

20. Піднести до квадрату всі елементи даного одновимірного масиву. Виконати сортування елементів масиву за зростанням, використовуючи метод обміну.

 **Контрольні запитання** 

1. Що називається масивом?
2. Змінні яких типів можуть бути елементами масиву?
3. З яких етапів складається робота з масивами?
4. Для чого використовується модуль **random**?
5. Які методи сортування масивів ви знаєте?
6. У чому полягає суть методу обміну?
7. У чому полягає суть методу екстремальних елементів?

Лабораторна робота № 8

ТЕМА: Проектування та налагодження програм оброблення двовимірних списків (масивів).

МЕТА: вміти створювати програми обробки та сортування двовимірних масивів

ТЕОРЕТИЧНІ ВІДОМОСТІ

Багатовимірні списки

У багатовимірних (або вкладених) списках кожна група елементів списку береться у квадратні дужки.

Наприклад, двовимірний список можна створити так:

```
>>> al = [[1, 2, 3], [4, 78, 45], [5, 6, 7]]
>>> al
[[1, 2, 3], [4, 78, 45], [5, 6, 7]]
```

Але для наочності краще записувати так:

```
>>> al=[
        [1, 2, 3],
        [4, 78, 45],
        [5, 6, 7]]
>>> al
[[1, 2, 3], [4, 78, 45], [5, 6, 7]]
```

У мові **Python** нумерація рядків і стовпців починається з нуля. Елементи двовимірного списку (масиву) беруться у квадратні дужки, елементи кожного рядка теж беруться у квадратні дужки, які відокремлюються комою. В середині рядка його елементи також відокремлюються комою. Доступ до кожного елемента двовимірного масиву здійснюється за двома індексами:

<ім’я списку> [номер_рядка] [номер_стовпця]

Наприклад, звернутися до першого елемента першої групи двовимірного списку можна так:

```
>>> al = [[1, 2, 3], [4, 78, 45], [5, 6, 7]]
>>> al[1][1]
78
```

(елементу масиву **a** з індексами 1 і 1 надано значення 78).

Для виведення матриці використовують структуру вкладених циклів **for** та змінних з іменами: **i** – *рядки* (вкладених списків) і **j** – *стовпці* (елементів вкладених списків).

Заповнення та виведення двовимірного масиву:

Заповнення двовимірного масиву:

1) *за формулою:*

```
n=int(input("n="))
m=int(input("m="))
a=[[]*n for i in range(m)]
for i in range (n):
    for j in range (m):
        a=i+j+4
```

2) *з клавіатури:*

```
n=int(input("n="))
m=int(input("m="))
a=[[]*n for i in range(m)]
for i in range(n):
    for j in range(m):
        print("a[",i,",",j,"]=",sep=" ",end=" ")
        a[i].append(int(input()))
```

3) *випадково (генератором випадкових чисел) із проміжку [a, b]:*

```
import random
n=int(input("n="))
m=int(input("m="))
a=[[]*n for i in range(m)]
for i in range(n):
    for j in range(m):
        a[i].append(random.randint ( 1, 100))
```

або

```
import random
n=int(input("n="))
m=int(input("m="))
a=[[]*n for i in range(m)]
for i in range(n):
    for j in range(m):
        a[i].append(random.randint ( 1, 100))
        print ( "{:<3d}".format(a[i][j]), end = "" )
    print()
```

Виведення елементів двовимірного масиву

1) *виведення у стовпчик:*

```
for i in range(n):
    for j in range(m):
        print(a[i][j])
```

2) *виведення у рядок:*

```
for i in range(n):
    for j in range(m):
        print(a[i][j],sep=" ",end=" ")
```

3) *у вигляді таблиці*

```
for i in range(n):
    for j in range(m):
        print ( (a[i][j]), end =" " )
    print()
```

або

```
for i in range(n):
    for j in range(m):
        print ( "{:<3d}".format(a[i][j]), end = "" )
    print()
```

Приклад 1. Піднести до квадрату всі елементи двовимірного масиву.

```
import random
n=int(input("n="))
m=int(input("m="))
a=[[]*n for i in range(m)]
for i in range(n):
    for j in range(m):
        print("a[",i,",",j,"]=",sep=" ",end=" ")
        a[i].append(int(input()))
for i in range(n):
    for j in range(m):
        a[i][j]=a[i][j]**2
for i in range(n):
    for j in range(m):
        print ( a[i][j], end =" " )
    print()
input()
```

Приклад 2. Створити матрицю 3×4.

```
m = int(input("Введіть кількість рядків: "))
n = int(input("Введіть кількість стовпців: "))

matrix = []  # створюємо пусту матрицю

# використовуємо вкладений цикл для введення кожного елементу матриці
for i in range(m):
    row = []  # створюємо пустий рядок для поточного рядка матриці
    for j in range(n):
        elem = float(input(f"Введіть елемент [{i}][{j}]: "))
        row.append(elem)  # додаємо елемент до поточного рядка матриці
    matrix.append(row)  # додаємо поточний рядок до матриці

print("Введена матриця:")
for row in matrix:
    print(row)
```

```
Оболонка ×
>> %Run -c $EDITOR_CONTENT
Введіть кількість рядків: 3
Введіть кількість стовпців: 4
Введіть елемент [0][0]: 1
Введіть елемент [0][1]: 2.3
Введіть елемент [0][2]: 4.8
Введіть елемент [0][3]: 6
Введіть елемент [1][0]: 3
Введіть елемент [1][1]: 5
Введіть елемент [1][2]: 1.5
Введіть елемент [1][3]: 2
Введіть елемент [2][0]: 4
Введіть елемент [2][1]: 5
Введіть елемент [2][2]: 6
Введіть елемент [2][3]: 7
Введена матриця:
[1.0, 2.3, 4.8, 6.0]
[3.0, 5.0, 1.5, 2.0]
[4.0, 5.0, 6.0, 7.0]
```

Приклад 3. Заповнити матрицю **3×4** випадковими дійсними числами з діапазону [0, 1):

```
import random

m = int(input("Введіть кількість рядків: "))
n = int(input("Введіть кількість стовпців: "))

matrix = []  # створюємо пусту матрицю

# використовуємо вкладений цикл для заповнення кожного елементу матриці
for i in range(m):
    row = []  # створюємо пустий рядок для поточного рядка матриці
    for j in range(n):
        elem = random.random() # генеруємо випадкове дійсне число
        row.append(elem)  # додаємо елемент до поточного рядка матриці
    matrix.append(row)  # додаємо поточний рядок до матриці

print("Згенерована матриця:")
for row in matrix:
    print(row)
```

Оболонка

```
>>> %Run -c $EDITOR_CONTENT
  Введіть кількість рядків: 3
  Введіть кількість стовпців: 4
  Згенерована матриця:
  [0.3199263591899888, 0.6867596530638015, 0.5691836481527199, 0.6786062611025442]
  [0.5178093053470545, 0.9335443072461643, 0.311785413359403, 0.9521785202207208]
  [0.6186957533612077, 0.8169330940751278, 0.9620780437112928, 0.559041192422157]
```

Приклад 4. Заповнити матрицю 3×4 випадковими дійсними числами з діапазону (-1, 1):

```
import random

m = int(input("Введіть кількість рядків: "))
n = int(input("Введіть кількість стовпців: "))
a = float(input("Введіть початок діапазону: "))
b = float(input("Введіть кінець діапазону: "))

matrix = []  # створюємо пусту матрицю

# використовуємо вкладений цикл для заповнення кожного елементу матриці
for i in range(m):
    row = []  # створюємо пустий рядок для поточного рядка матриці
    for j in range(n):
        elem = round(random.uniform(a, b), 2)  # генеруємо випадкове дійсне число з точністю до двох знаків після коми
        row.append(elem)  # додаємо елемент до поточного рядка матриці
    matrix.append(row)  # додаємо поточний рядок до матриці

print("Згенерована матриця:")
for row in matrix:
```

Оболонка

```
>>> %Run -c $EDITOR_CONTENT
  Введіть кількість рядків: 3
  Введіть кількість стовпців: 4
  Введіть початок діапазону: -1
  Введіть кінець діапазону: 1
  Згенерована матриця:
  [-0.19, -0.15, -0.56, 0.43]
  [-0.91, -0.61, -0.4, 0.2]
  [-0.28, 0.26, -0.53, 0.89]
```

Властивості квадратної матриці:

1. Матриця, в якій в кількість рядків дорівнює кількості стовпців називається квадратною матрицею.

```python
import random

n = int(input("Введіть розмір квадратної матриці: "))

# Створюємо пусту матрицю з розміром n x n
matrix = [[0 for j in range(n)] for i in range(n)]

# Заповнюємо матрицю випадковими значеннями в діапазоні від 0 до 9
for i in range(n):
    for j in range(n):
        matrix[i][j] = random.randint(0, 9)

# Виводимо матрицю на екран
for i in range(n):
    for j in range(n):
        print(matrix[i][j], end=" ")
    print()
```

Оболонка

```
>>> %Run -c $EDITOR_CONTENT
Введіть розмір квадратної матриці: 3
7 2 3
5 9 0
9 8 9
```

2. Для звернення до елементів матриці, потрібно використувати два індекси **matrix[i][j]**

Наприклад, щоб отримати значення елементу матриці в третьому рядку та другому стовпці, **matrix** [2][1]=9

i \ j			
	3	**9**	**0**
	2	**9**	**6**
	0	**9**	**2**

3. Якщо номер рядка елемента співпадає з номером стовпця (**i = j**) це означає, що елемент лежить на головній діагоналі матриці

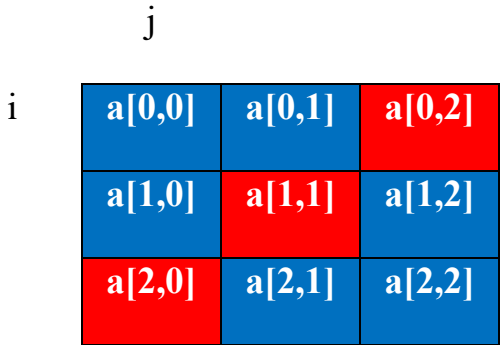
i \ j			
	a[0,0]	**a[0,1]**	**a[0,2]**
	a[1,0]	**a[1,1]**	**a[1,2]**
	a[2,0]	**a[2,1]**	**a[2,2]**

```python
import random

n = int(input("Введіть розмір квадратної матриці: "))

# Створюємо пусту матрицю з розміром n x n
matrix = [[0 for j in range(n)] for i in range(n)]

# Заповнюємо матрицю випадковими значеннями в діапазоні від 0 до 9
for i in range(n):
    for j in range(n):
        matrix[i][j] = random.randint(0, 9)

# Виводимо матрицю на екран
for i in range(n):
    for j in range(n):
        print(matrix[i][j], end=" ")
    print()
for i in range(n):
    value = matrix[i][i]  # елемент знаходиться на головній діагоналі
    print(value)

```

Оболонка

```
>>> %Run -c $EDITOR_CONTENT
Введіть розмір квадратної матриці: 4
4 4 5 3
4 5 4 7
8 6 8 4
6 8 8 6
4
5
8
6
```

4. У квадратній матриці, коли сума індексів номеру рядка та номеру стовпця дорівнює n-1 (тобто i+j=n-1), елемент знаходиться на побічній діагоналі матриці.

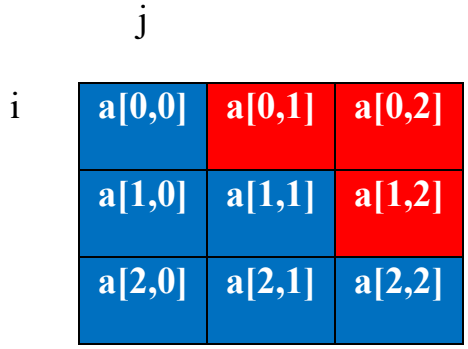



```python
import random

n = int(input("Введіть розмір квадратної матриці: "))

# Створюємо пусту матрицю з розміром n x n
matrix = [[0 for j in range(n)] for i in range(n)]

# Заповнюємо матрицю випадковими значеннями в діапазоні від 0 до 9
for i in range(n):
    for j in range(n):
        matrix[i][j] = random.randint(0, 9)

# Виводимо матрицю на екран
for i in range(n):
    for j in range(n):
        print(matrix[i][j], end=" ")
    print()
for i in range(n):
    j = n - i - 1  # номер стовпця, який відповідає номеру рядка i на побічній діагоналі
    value = matrix[i][j]  # елемент знаходиться на побічній діагоналі
    print(value)

```

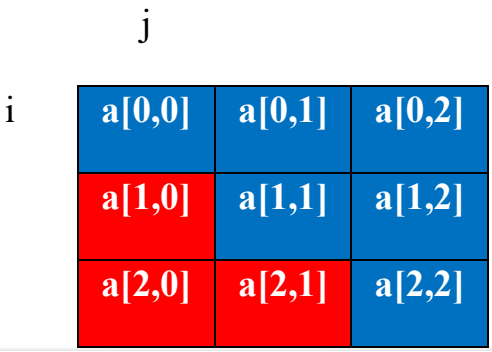
Оболонка

```
>>> %Run -c $EDITOR_CONTENT
Введіть розмір квадратної матриці: 6
1 8 6 2 3 8
7 2 7 9 5 2
3 3 6 0 3 4
4 7 1 5 2 1
2 4 6 8 6 7
0 2 0 2 8 3
8
5
0
1
4
0
```

5. Елементи, які знаходяться *вище головної діагоналі*: **i < j**,



а *нижче i > j*




```python
import random

n = int(input("Введіть розмір квадратної матриці: "))

# Створюємо пусту матрицю з розміром n x n
matrix = [[0 for j in range(n)] for i in range(n)]

# Заповнюємо матрицю випадковими значеннями в діапазоні від 0 до 9
for i in range(n):
    for j in range(n):
        matrix[i][j] = random.randint(0, 9)

# Виводимо матрицю на екран
for i in range(n):
    for j in range(n):
        print(matrix[i][j], end=" ")
    print()
upper_elems = []  # створюємо пустий список для елементів над головною діагоналлю
lower_elems = []  # створюємо пустий список для елементів під головною діагоналлю
for i in range(n):
    for j in range(n):
        if i < j:  # елемент над головною діагоналлю
            upper_elems.append(matrix[i][j])  # додаємо елемент до списку елементів над головною діагоналлю
        elif i > j:  # елемент під головною діагоналлю
            lower_elems.append(matrix[i][j])  # додаємо елемент до списку елементів під головною діагоналлю
print("Елементи над головною діагоналлю:", upper_elems)
print("Елементи під головною діагоналлю:", lower_elems)
```

Оболонка

```
>>> %Run -c $EDITOR_CONTENT
Введіть розмір квадратної матриці: 3
7 7 8
3 7 0
1 8 5
Елементи над головною діагоналлю: [7, 8, 0]
Елементи під головною діагоналлю: [3, 1, 8]
```

Приклад 5. Створити масив 3х3 і вивести його елементи у вигляді матриці

```
a = [[2,3,4],[4,3,7],[8,3,5]]
for row in a:
    for item in row:
        print(item, "", end="")
    print()
```

Результат виконання програми:

```
2 3 4
4 3 7
8 3 5
```

Приклад 6. Заповнити двовимірний масив розміром 2 х 2 випадковими числами і вивести його елементи. Обчислити частинні суми і вивести їх на екран.

```
from random import randint
a = [[] * 4 for i in range(4)]
for i in range(4):
    for j in range(4):
        a[i].append(randint(1,5))
        print("{:<3d}".format(a[i][j]), end="")
    print()
s=0
for i in range(4):
    for j in range(4):
        s=s+a[i][j]
        print(s)
```

```
3  5
3  5
Частинні суми
3
8
11
16
```

Хід роботи

ЗАВДАННЯ 1. Обробка двовимірних масивів

1. *Скласти блок-схему алгоритму розв'язування задачі.*
2. *Написати програму реалізації алгоритму з обов'язковими коментарями.*
3. *Проаналізувати достовірність отриманих результатів обчислень.*

Варіанти завдань

1. Знайти суму елементів головної діагоналі квадратної матриці (слід матриці). Виконати сортування непарних стовпців за спаданням.
2. Знайти добуток елементів головної діагоналі квадратної матриці. Виконати сортування непарних стовпців за зростанням.
3. Знайти суму квадратів елементів двовимірного масиву. Виконати сортування парних стовпців за спаданням.
4. Підрахувати кількість від'ємних елементів у даному двовимірному масиві. Виконати сортування парних стовпців за зростанням.
5. Домножити всі елементи матриці на мінімальний елемент. Виконати сортування непарних стовпців за спаданням, а парних – за зростанням.
6. Підрахувати кількість додатних елементів у даному двовимірному масиві. Виконати сортування парних стовпців за спаданням, а непарних – за зростанням.
7. Знайти суму кожного рядка матриці. Виконати сортування непарних стовпців за спаданням.
8. Підрахувати середнє арифметичне всіх елементів у даному двовимірному масиві. Виконати сортування непарних стовпців за зростанням.
9. Підрахувати середнє арифметичне всіх додатних елементів у даному двовимірному масиві. Виконати сортування парних стовпців за спаданням.
10. Підрахувати середнє арифметичне всіх від'ємних елементів у даному двовимірному масиві. Виконати сортування парних стовпців за зростанням.
11. Знайти найбільший елемент двовимірного масиву і поміняти його місцями з першим. Виконати сортування непарних рядків за спаданням, а парних – за зростанням.
12. Знайти найменший елемент двовимірного масиву і поміняти його місцями з останнім. Виконати сортування непарних рядків за спаданням, а парних – за зростанням.
13. Знайти суму всіх від'ємних елементів матриці. Виконати сортування непарних рядків за спаданням.

14. Знайти транспоновану матрицю до даної. Виконати сортування непарних рядків за зростанням.

15. Знайти суму елементів квадратної матриці, які знаходяться над головною діагоналлю. Виконати сортування парних рядків за спаданням.

16. Знайти найбільший за модулем елемент двовимірного масиву. Виконати сортування парних рядків за зростанням.

17. Знайти найменший за модулем елемент двовимірного масиву. Виконати сортування непарних стовпців за спаданням.

18. Замінити всі елементи квадратної матриці, які знаходяться нижче головної діагоналі, нулями. Виконати сортування непарних стовпців за зростанням.

19. Замінити всі елементи квадратної матриці, які знаходяться вище головної діагоналі, нулями. Виконати сортування парних стовпців за спаданням.

20. Знайти добуток від'ємних елементів головної діагоналі матриці. Виконати сортування парних стовпців за спаданням, а непарних – за зростанням.

Контрольні запитання

1. Що називається масивом?
2. Змінні яких типів можуть бути елементами масиву?
3. З яких етапів складається робота з масивами?
4. Для чого використовується модуль **random**?
5. Які методи сортування масивів ви знаєте?
6. У чому полягає суть методу обміну?
7. У чому полягає суть методу екстремальних елементів?

Лабораторна робота № 9

ТЕМА: Проектування та налагодження програм обробки рядків.

МЕТА: *навчитися використовувати рядковий тип даних, навчитися користуватися методами рядкового типу*

ТЕОРЕТИЧНІ ВІДОМОСТІ

У **Python** символи представлені у кодуванні **UTF**-8.

Символьний літерал (або символьна стала) – це конкретний символ із таблиці кодування.

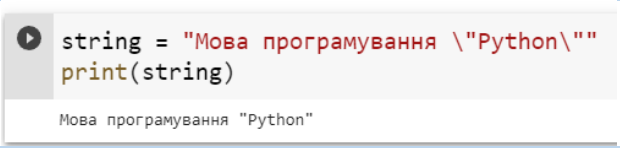
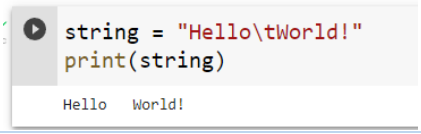
Символьні літерали беруть у апострофи або подвійні лапки, щоб відрізняти їх від ідентифікаторів, цифр та інших символьних знаків.

```
B = "B" # Змінна B набуває значення символьного літералу "B"
```

Для задавання **символу апострофа** у **Python** використовують літерал **" ' ",** а для задавання символу подвійних лапок – літерал **' " '**

Спеціальні або екрановані послідовності (*escape-послідовності*) **у Python** дозволяють задавати символи, які неможливо ввести з клавіатури.

Символ	Значення
\n	Продовження рядка на наступний `string = "Hello\nWorld!"` `print(string)` `Hello` `World!`
\\	Обернена коса риска (зберігає \) `string = "Мова Python\\C++"` `print(string)` `Мова Python\C++`
\’	Апостроф (зберігає ’) `string = 'Пам\'ять'` `print(string)` `Пам'ять`
\”	Подвійні лапки (зберігає “ ”)

	
\t	Табуляція 

Рядок – це послідовність символів. Рядки у **Python** належать до незмінюваних (***immutable***) типів. Рядкові літерали (тобто послідовність символів) описують одним з таких способів:

- апострофами або подвійними лапками для рядків, що розташовуються у одному фізичному рядку;
- потрійними апострофами або потрійними подвійними лапками для рядків, що розташовуються у кількох фізичних рядках.

```
>>> 'Рядок-літерал'
"Рядок-літерал"
'''Це рядок-літерал, який
розташовується у кількох рядках'''
"""Це також рядок-літерал, який
розташовується у кількох рядках"""
```

Способи створення рядків:

1. Створення рядка за допомогою літерала.

```
>>> s = "" # Порожній рядок
s = "a" # Рядок, що складається з одного елементу.
f = '''Рядки це
набір символів.'''
```

2. Створення рядка за допомогою перетворення об’єкта у рядок використовуючи функцію **str**.

```
>>> s = str(2018)
>>> print (s)
2018
```

Для рядків визначено інструкцію **input** введення рядка з клавіатури

```
>>> s = input('Введіть рядок')
Введіть рядок Привіт
```

Індексація рядків

- До символів рядка можна звертатися за їхніми індексами;

P	y	t	h	o	n
S[0]	S[1]	S[2]	S[3]	S[4]	S[5]

- Якщо вказати від'ємне значення індексу, то номер буде відраховуватися з кінця, починаючи з номера -1.

P	y	t	h	o	n
S[-6]	S[-5]	S[-4]	S[-3]	S[-2]	S[-1]

```
>>> s="python"
>>> s[0]
'p'
>>> s[-1]
'n'
```

Всі символи рядка можна послідовно перебрати використовуючи цикл по колекції ***for***

Вбудовані функції рядків

Операція	Значення
ord(s)	Перетворює символ в число (код символа) s з таблиці кодування `print ('Введіть символ')` `s=input()` `print(ord(s))` `Введіть символ` `s` `115`
chr(n)	Повертає символ вказаного коду **n**, що відповідає у таблиці кодування коду

	>>> n=115 >>> chr(n) 's'
int(“c”)	Перетворення рядка в int >>> int("100") 100
str(n)	Перетворює тип об’єкта на string >>> str(25) '25'
len(s)	Повертає довжину рядка >>> s="інформатика" >>> len(s) 11

Зріз рядків (string slice)

Зріз (string slice) – вилучення з цього рядка одного символу або деякого фрагмента (підрядка).

Форми зрізів:

1. Взяття одного символу рядка, а саме, **S[i]** – це зріз, що складається з одного символу, який має номер **i**, при цьому вважаючи, що нумерація починається з числа **0.**

```
>>> s="string"
>>> s[1]
't'
>>> s[-1]
'g'
```

*Якщо ж номер символу в зрізі рядка S більше або дорівнює **len**(S), або менше, ніж - **len**(S), то при зверненні до цього символу рядка відбудеться помилка IndexError: string index out of range.*

2. Зріз з двома параметрами: S[i:j] повертає підрядок, що починається з **i**–го символу та закінчується **j-1**-м символом, не включаючи його.

```
>>> s="python"
>>> s[0:2]
'py'
```
або
```
>>> s="python"
>>> s[-6:-4]
'py'
```

Можна використовувати як додатні, так і від'ємні індекси в одному зрізі, *наприклад,*

```
>>> s="python"
>>> s[1:-1]
'ytho'
```

S[1: -1] - це рядок без першого і останнього символу (зріз починається з символу з індексом 1 і закінчуватись індексом -1, не включаючи його).

3. Зріз S[i:], якщо *опустити другий параметр* (але поставити двокрапку), то зріз береться до кінця рядка.

Наприклад, щоб видалити з рядка перший символ

```
>>> s="python"
>>> s[1:]
'ython'
```
або
```
>>> s="python"
>>> s[2:]
'thon'
```

4. Зріз S[:j], якщо *опустити перший параметр*, то зріз береться від початку рядка.

Наприклад, щоб видалити з рядка останній символ

```
>>> s="python"
>>> s[:-1]
'pytho'
```
або
```
>>> s="python"
>>> s[:-3]
'pyt'
```

5. Зріз S[:], виділяється весь рядок.

6. Зріз з трьома параметрами S[i: j: k] виділення фрагмента рядка.

i – початок // за замовчуванням початок - 0

j – кінець // номер індексу останнього символу j-1

k – крок // дорівнює 1

- якщо величина $k \geq 0$, то рядок розглядається з початку до кінця (зліва направо);
- якщо величина $k < 0$, то рядок розглядається з кінця до початку (справа наліво)

При такій формі витягуються всі елементи рядка починаючи з позиції ***i***, завершуючи позицією ***j***-1 включно зі зміщенням (кроком) ***k***.

```
>>> S = '0123456789'
>>> S[1:9:2]
'1357'
```
```
>>> S = '0123456789'
>>> S[0:10:3]
'0369'
```

Якщо ***k***<0, то порядок меж ***i, j*** змінюється на протилежний.

```
>>> S = '0123456789'
>>> S[9:0:-2]
'97531'
```

```
>>> S = '0123456789'
>>> S[9:0:-1]
'987654321'
```

7. Зріз виду S[*i* : : *k*] *При даній формі середня межа пропускається. Якщо* k>=*0, то рядок обробляється з позиції* i *до кінця рядка. Якщо* k<*0, то рядок обробляється з позиції* i *до початку рядка у зворотному порядку.*

8. Зріз виду S[: *j* : *k*] При такій формі відсутня перша межа ***i***. Якщо значення ***k***>=0, то і приймається рівним початку рядка (***i***=0). Якщо значення ***k***<0, то рядок розглядається з кінця до початку, а значення і приймається рівним кінцю рядка.

9. Зріз виду S[: : *k*]

```
>>> s="string"
>>> s[::2]
'srn'
```

```
>>> s[::-1]
'gnirts'
```

10. Зріз виду S[: :]

```
>>> for i in s:
        print(i)


P
y
t
h
o
n
>>> s[3:]
'hon'
>>> s[::-1]
'nohtyP'
>>> len(s)
6
>>> max(s)
'y'
>>> min(s)
'P'
```

Операції над рядками

Конкатенація (щеплення)	
Операція	**Опис**
s1 + s2	Конкатенація рядків s1 та s2, тобто створення нового рядка `>>> s1="Мова"` `>>> s2="Python"` `>>> s=s1+s2` `>>> s` `'МоваPython'` `>>> s=s2+"!"` `>>> s` `'Python!'` s=s1+” ”+s2
s * n **n * s**	n зчеплених копій рядка s `>>> s1 = "інформатика"` `>>> s2 = " це цікаво"` `>>> print(s1 * 2)` `інформатикаінформатика` `>>> s="*"` `>>> s*10` `'**********'`
Перевірка входження символів до рядка	
c in S ***#символ***	Повертає значення **True**, якщо символ **і** міститься у рядку S `>>> S = "інформатика"` `>>> 'i' in S` `True` `>>> 'r' in S` `False`
c not in S ***#символ***	Повертає значення **True**, якщо символ **а** не міститься у рядку S `>>> s="String"` `>>> 'a' not in s` `True`

Методи роботи з рядками	
<відокремлювач>.join(послідовність)	Перетворює рядкові елементи послідовності на рядок `>>> " ".join(("Мова","програмування","Python"))` `'Мова програмування Python'`
<рядок>.upper()	Замінює в рядку всі малі букви на великі `>>> s="python"` `>>> print(s.upper())` `PYTHON`
<рядок>.lower()	Замінює в рядку всі великі букви на малі

	>>> s="PYTHON" >>> s.lower() 'python'
<рядок>.capitalize()	Замінює першу букву рядка на велику >>> s="вінниця" >>> s.capitalize() 'Вінниця'
Методи пошуку та заміни в рядку	
<рядок>.count(підрядок)	Визначає кількість входжень підрядка в рядок [13] string = "мова Python - це мова програмування" x = string.count("мова") print(x) 2 string = "мова Python - це мова програмування" x = string.count("мова", 0, 4) print(x) 1
<рядок>.find (підрядок, початок, кінець)	Пошук підрядка в рядку, якщо підрядок знайдено, повертається номер позиції, з якої починається підрядок, інакше – повертається значення -1 string = "Hello, Python" x = string.find("Python") print(x) 7 string = "Hello, Python" print(string.find("a")) -1
<рядок>.index(підрядок, початок, кінець)	Відрізняється від методу **find**() лише тим, що у випадку, коли підрядок у рядку відсутній, генерується виняток ValueError string = "Hello Python" print(string.find("q")) print(string.index("q")) -1 ValueError Traceback (most recent call last) <ipython-input-26-686038f0df5f> in <module>() 2 3 print(string.find("q")) ----> 4 print(string.index("q")) ValueError: substring not found

<рядок>.replace (*підрядок заміни*, *новий підрядок*)	Виконує заміну всіх входжень заданого підрядка в рядку на новий і повертає новий рядок s=’Python 3’ `>>> s.replace("3", "3.6.3")` `'Python 3.6.3'`

Приклад 1.

```
s=input('Введіть рядок')#Введіть рядок холестерин
# Виведемо третій символ цього рядка:
print (s[2])
# Виведемо передостанній символ цього рядка:
print (s[-2])
# Виведемо перші п'ять символів цього рядка:
print (s[0: 5])
#Виведемо весь рядок, крім останніх двох символів:
print (s[: - 2])
#Виведемо всі символи з парними індексами
#(вважаючи, що індексація починається з 0, тому символи виводяться починаючи з першого):
#(вважаючи, що індексація починається з 0, тому символи виводяться починаючи з першого):
print (s[:: 2])
#Виведемо всі символи з непарними індексами, тобто починаючи з другого символу рядка:
print (s[1 :: 2])
#Виведемо всі символи у зворотному порядку:
print (s[:: - 1])
#Виведемо всі символи рядка через один в зворотному порядку, починаючи з останнього:
print (s[:: - 2])
```

```
>>> s = input ()
холестерин
>>> s=input('Введіть рядок')
Введіть рядок холестерин
>>> print (s[2])
о
>>> print (s[-2])
и
>>> print (s[0: 5])
 холе
>>> print (s[: - 2])
 холестер
>>> print (s[:: 2])
 оетрн
>>> print (s[1 :: 2])
хлсеи
>>> print (s[:: - 1])
ниретселох
>>> print (s[:: - 2])
нртео
```

Хід роботи

ЗАВДАННЯ 1. Визначити довжину вашого прізвища, імені, по-батькові (вводяться послідовно через пробіл) та довжину рядка, що об'єднує всі ці слова в одне (через пробіл).

1. *Скласти блок-схему алгоритму розв'язування задачі.*
2. *Написати програму реалізації алгоритму з обов'язковими коментарями*
3. *Проаналізувати достовірність отриманих результатів обчислень.*

ЗАВДАННЯ 2.

1. *Скласти блок-схему алгоритму розв'язування задачі.*
2. *Написати програму реалізації алгоритму з обов'язковими коментарями*
3. *Проаналізувати достовірність отриманих результатів обчислень.*

Варіанти завдань

1. Шляхом вирізання та склеювання отримати зі слова „капелюх" 3 іменники в однині.
2. Шляхом вирізання та склеювання отримати зі слова „математика" 3 іменники в однині.
3. Шляхом вирізання та склеювання отримати зі слова „балкон" слова „лоб", „клан" і „колба".
4. Шляхом вирізання та склеювання отримати зі слова „пролісок" 3 іменники в однині.
5. Шляхом вирізання та склеювання отримати зі слова „омар" слова „томат" і „атом".
6. Шляхом вирізання та склеювання отримати зі слова „інформатика" 3 іменники в однині.
7. Шляхом вирізання та склеювання отримати зі слова „комбінаторика" 3 іменники в однині.

8. Шляхом вирізання та склеювання отримати зі слова „астрономія” 3 іменники в однині.

9. Шляхом вирізання та склеювання отримати зі слова „філософія” 3 іменники в однині.

10. Шляхом вирізання та склеювання отримати зі слова „щоденник” 3 іменники в однині.

11. Шляхом вирізання та склеювання отримати зі слова „малина” усі можливі слова (іменники в однині).

12. Шляхом вирізання та склеювання отримати зі слова „маскарад” 3 іменники в однині.

13. Шляхом вирізання та склеювання отримати зі слова „програмування” 3 іменники в однині.

14. Шляхом вирізання та склеювання отримати зі слова „телевізор” 3 іменники в однині.

15. Шляхом вирізання та склеювання отримати зі слова „магазин” 3 іменники в однині.

16. Нехай дано деякий текст. Обчислити, скільки разів повторюється комбінація літер „на”.

17. Підрахувати в рядку кількість ком.

18. Підрахувати в рядку кількість пробілів.

19. Перевірити, чи вірно розставлені в тексті дужки (кількість відкриваючих дорівнює кількості закриваючих).

20. Підрахувати в слові кількість голосних літер.

21. У довільному слові замінити всі літери ’a’ на ’o’.

22. У довільному слові замінити всі літери ’b’ на ’c’.

23. Нехай дано деякий текст. Обчислити, скільки разів повторюється наперед заданий символ „a”.

Контрольні запитання

1. Якою може бути максимальна довжина рядка?
2. Як можна виконати порівняння рядків?
3. Яке слово називають паліндромом?
4. Яка функція визначає довжину рядка?
5. Назвіть операції над рядковими змінними?
6. Які є правила звертання до окремих символів рядка?

Лабораторна робота № 10

ТЕМА: Проектування та налагодження програм обробки структур.

МЕТА: *навчитися створювати нові типи даних у вигляді структур, а також розробляти алгоритми їх обробки засобами мови Python*

ТЕОРЕТИЧНІ ВІДОМОСТІ

Структури дозволяють об’єднувати під одним іменем дані різних типів. Це зручно для розв’язування різноманітних задач.

Структури – *набір різнотипних або однотипних даних.*

Приклади структур:

- Автомобіль: марка, колір, пробіг, рік виробництва тощо.
- Планшет: фірма, назва, операційна система, пам'ять, кількість ядер тощо.

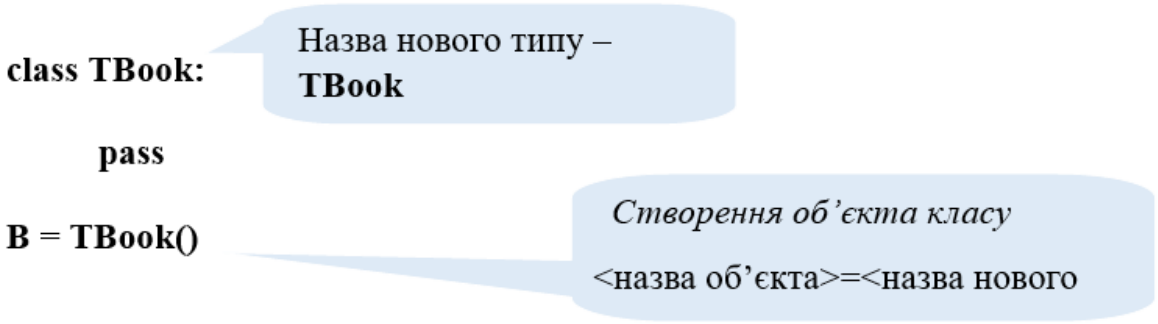
В **Python** для роботи з структурами використовують спеціальний тип даних - **клас**. В програмі можна створювати свої класи (нові типи даних).

Опис класу:

```
class <назва класу>:
    pass
```

На відміну від інших мов програмування (C, C ++ та ін.) при оголошенні класу не обов’язково відразу перераховувати всі поля структури, вони можуть додаватися по ходу виконання програми.

Наприклад, створити новий клас **TBook** – структуру за допомогою якої можна описати книги бібліотеки. Поля структури: ПІП автора (author), назва книги (title), кількість екземплярів в бібліотеці.



Можна також створити масив (список) об'єктів:

Books = []

for i in range(100):

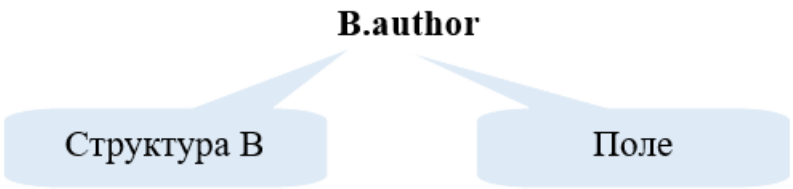
Books.append (TBook())

Доступ до елементів структури

Доступ до конкретного поля змінної типу структура дає складене ім'я вигляду:

<назва структури>.<назва поля>

Наприклад, **B.author** крапкою розділяється назва структури і назва поля.



Поля структури можна вводити з клавіатури

B.author = input()

B.title = input()

B.count = int(input())

Присвоєння нових значень:

B.author = "Шевченко Т.Г."

B.title = "Полтава"

B.count = 1

Щоб ввести або вивести структуру, треба **ввести/вивести** відповідні поля. Для **введення/виведення** масиву записів використовують цикл.

```
class TBook:
    pass
#структура книга
n=int(input('n='))
books=TBook()
books=[]
for i in range (n):
    books.append(TBook())
    books[i].author=input("Введіть ім\'я автора: ")
    books[i].title=input("Введіть назву: ")
    books[i].year=int(input("Рік видання: "))
book=input('Введіть автора книги ')
for i in range (n):
    if  books[i].author==book:
        print(books[i].title,'\t',books[i].year)
```

 **Хід роботи** 

 **ЗАВДАННЯ.**

1. *Скласти блок-схему алгоритму розв'язування задачі.*
2. *Написати програму реалізації алгоритму з обов'язковими коментарями*
3. *Проаналізувати достовірність отриманих результатів обчислень.*

Варіанти завдань

1. Структура містить такі поля: ПІП студента, курс, адресу, рік народження. Вивести на екран всіх тих студентів, хто навчається на 4 курсі.
2. Структура містить такі поля: ПІП студента, курс, номер групи, стипендія. Вивести на екран всіх тих студентів, які не отримують стипендію.
3. Структура: ПІП автора, назва книги, рік видання. Вивести на екран всі книги, які видані раніше введеного року (рік вводить користувач).
4. Структура: ПІП автора, назва книги, рік видання. За введеним ПІП автора виводити на екран всі його книги.
5. Структура містить такі поля: ПІП студента, курс, номер групи, місце проживання. Вивести на екран всіх тих студентів, які проживають у гуртожитку.
6. Структура містить такі поля: ПІП студента, курс, номер групи, стипендія. Вивести на екран всіх тих студентів, які отримують мінімальну стипендію.
7. Структура містить такі поля: ПІП студента, курс, номер групи, кількість пропусків з поважної причини і без причини. Вивести на екран всіх тих студентів, які пропустили більше 10 год без поважної причини.
8. Структура: ПІП автора, назва книги, рік видання. Вивести на екран всі книги, які видані пізніше введеного року (рік вводить користувач).
9. Структура містить такі поля: ПІП студента, курс, номер групи, заліки з п’яти предметів. Вивести на екран всіх тих студентів, які мають заборгованість з якогось предмета.
10. Структура містить такі поля: ПІП студента, курс, номер групи, оцінки з п’яти предметів. Вивести на екран всіх відмінників.
11. Структура містить такі поля: ПІП студента, курс, номер групи, стипендія. Вивести на екран всіх тих студентів, які отримують підвищену стипендію.
12. Структура містить такі поля: ПІП студента, курс, номер групи, місце проживання. Вивести на екран всіх тих студентів, які проживають на квартирі.

13. Структура містить такі поля: ПІП студента, курс, номер групи, заліки з п’яти предметів. Вивести на екран всіх тих студентів, які склали всі заліки.

14. Структура містить такі поля: ПІП студента, курс, номер групи, кількість пропусків з поважної причини і без причини. Вивести на екран всіх тих студентів, які пропустили більше 10 год з поважної причини.

15. Структура містить такі поля: ПІП студента, курс, номер групи, рік народження. Вивести на екран найстаршого студента.

16. Структура містить такі поля: назва команди, кількість поразок, кількість перемог, кількість нічиїх. Вивести на екран команду, яка найбільше разів перемогла.

17. Структура містить такі поля: назва команди, кількість поразок, кількість перемог, кількість нічиїх. Вивести на екран лідера чемпіонату.

18. Структура містить такі поля: назва команди, кількість поразок, кількість перемог, кількість нічиїх. Вивести на екран команду, яка жодного разу не програла.

19. Структура містить такі поля: ПІП студента, курс, номер групи, оцінки з п’яти предметів. Вивести на екран всіх «хорошистів» і відмінників.

20. Структура містить такі поля: ПІП студента, курс, номер групи, рік народження. Вивести на екран наймолодшого студента.

Контрольні запитання

1. Що таке структура?
2. Який загальний вигляд опису структури?
3. Які є підходи до опису структурної змінної?
4. Як здійснюється доступ до окремих полів структурної змінної?
5. Що таке вкладені структури? Як здійснюється доступ до полів вкладених структур?

Лабораторна робота № 11

ТЕМА: Файли в мові Python
МЕТА: *навчитися здійснювати операції читання та запису для файлів у мові Python*

ТЕОРЕТИЧНІ ВІДОМОСТІ

У **Python** є два типи файлів:

- *Бінарні (двійкові) файли*
- *Текстові файли*

<table>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>Двійкові файли відкриваються як послідовність байтів. При цьому відповідальність за коректну робота з даними повністю покладається на програму, що використовує цей файл. Вони зберігаються у .bin форматі.</td>
<td>Текстові файли відкриваються як послідовність рядків (символів), що міститься у файлі. При цьому фізичний рядок у файлі відповідає рядковому літералу у програмі. Вони можуть зберігатися у (.txt) форматі звичайного або форматованого тексту (.rtf).</td>
</tr>
<tr>
<td colspan="2">Створювати, змінювати та опрацьовувати двійкові файли можна як правило лише за допомогою спеціальної програми. З текстовими файлами можна працювати за допомогою будь-якого текстового редактора.</td>
</tr>
</table>

Будь-яку операцію з файлом можна розділити на три основні кроки:

1. Відкриття файлу
2. Виконання операцій з файлами (читання/запис)
3. Закриття файлу

Відкриття файлу

Відкриття файлу здійснюється інструкцією

f = open(file_name, mode)

де **f** – ім'я файлової змінної, **file_name** – ім'я файла, що відкривається, **mode** – режим роботи з файлом – це рядковий літерал, що будується за допомогою значень наведених у таблиці.

Режими роботи з файлом

Режим	**Опис**
”r”	Відкриття для читання. Якщо файла з таким ім’ям не існує, то інструкція породжує помилку (типове значення).
“w”	Відкриття для запису. Якщо файл з таким ім’ям вже існує, то цей буде перезаписаний новим.
“x”	Відкриття для запису. Якщо файл з таким ім’ям вже існує, то інструкція породжує помилку.
“a”	Відкриття для додавання даних у кінець файлу.
“+”	Відкриття на читання та запис.

“b”	Відкриття у двійковому режимі.
“t”	Відкриття у текстовому режимі (типове значення).

Значення режиму **mode** отримується об'єднанням значень наведених вище.

Наприклад, "**rb**" – відкриття існуючого двійкового файлу для читання. "**r**" і "**t**" – це типові значення, їх можна опускати при формуванні режиму **mode**. Типове значення параметра **mode** – це "**rt**".

Інструкція **open** ототожнює вміст файла **file_name** зі змінною **f**. Подальша робота з файлом відбувається лише через цю змінну. Крім цього інструкція **open** блокує файл для змін іншими програмами, щоб уникнути конфліктних ситуацій.

Закриття файлу

Робота з файлом відбувається через файлову змінну, що зберігає свої дані у оперативній пам'яті. Для того, щоб зберегти результат роботи з файлом на носій інформації, файл необхідно закрити. Крім цього операція закриття файлу повідомляє операційній системі, що файл розблокований і може використовуватися (для зміни) іншими програмами.

Для того, щоб закрити файл використовується інструкція

f.close()

де **f** – ім'я файлової змінної, пов’язаної з файлом.

Операції з файлами

Операція відкриття файлу породжує спеціальний об'єкт (що належить файловій змінній), що називається **маркером**. Маркер вказує на поточну позицію у файлі, тобто місце з якого відбувається читання чи запис даних. Під час відкриття файлу у режимі "**a**" (додавання) маркер встановлюється у позицію після останнього запису. Для інших режимів маркер встановлюється на

найперший запис. Будь-яка операція *читання/запису* змінює поточну позицію маркера.

Нехай **f** – файлова змінна, яка асоційована з файлом відкритим у відповідному режимі. Тоді виконуються операції наведені в таблиці.

Операції з файлами

Операція	**Опис**
f.read()	Повертає весь вміст файла
f.read(n)	Читає *n* байтів з поточного положення маркера
f.readline()	Читає та повертає поточний рядок з текстового файлу
f.readlines()	Повертає список всіх рядків текстового файлу
f.write(s)	Записує значення змінної *s* у файл
print(s,file=f)	Записує значення змінної *s* у файл
f.writelines(lineslist)	Записує список рядків *lineslist* у текстовий файл
f.tell()	Повертає значення поточної позиції маркера у файлі
f.seek(n)	Встановлює маркер у позицію *n* у файлі

Текстові файли

Файлова змінна для текстового файлу – це колекція його рядків. Отже, текстовий файл можна проходити по рядках, використовуючи цикл **for**. Нехай **f** – файлова змінна відкритого для читання текстового файлу. Тоді інструкція

for line **in** f:

process_iteration

виконує **process_iteration** для всіх рядків файлу **f**.

Хід роботи

ЗАВДАННЯ 1. За допомогою текстового редактора **Блокнот** створити текстовий файл **mas.txt** у вашій робочій папці, в якому записати 10 цілих чисел (через пробіл). Потрібно скласти програму, яка зчитує ці дані та обчислює вказані величини.

1. *Скласти блок-схему алгоритму розв'язування задачі.*
2. *Написати програму реалізації алгоритму з обов'язковими коментарями*
3. *Проаналізувати достовірність отриманих результатів обчислень.*

Варіанти завдань

1. Суму квадратів від'ємних чисел.
2. Добуток чисел, кратних 5.
3. Добуток від'ємних чисел.
4. Середнє арифметичне від'ємних чисел.
5. Суму квадратів додатних чисел.
6. Добуток додатних чисел.
7. Середнє арифметичне додатних чисел.
8. Суму квадратів парних чисел.
9. Добуток парних чисел.
10. Середнє арифметичне парних чисел.
11. Суму квадратів непарних чисел.
12. Добуток непарних чисел.
13. Середнє арифметичне непарних чисел.
14. Суму чисел, кратних 3.
15. Добуток чисел, кратних 3.

16. Середнє арифметичне чисел, кратних 3.

17. Суму чисел, кратних 4.

18. Добуток чисел, кратних 4.

19. Суму чисел, кратних 5.

20. Середнє арифметичне чисел, кратних 5.

 **ЗАВДАННЯ 2.** Модифікувати свою програму опрацювання структур (лабораторна робота № 10) таким чином, щоб введені дані були записані у файл на диску і зчитані з нього.

 Контрольні запитання 

1. Що таке файл?

2. Які типи файлів визначені в **Python**?

3. Що означає “закрити файл” ?

4. Як організувати читання даних з текстового файлу? виведення даних до текстового файлу?

5. З яких етапів складається робота з файлами?

6. Які є режими для роботи з файлами?

Лабораторна робота № 12 - 13

ТЕМА: Функції в Python
МЕТА: *ознайомитися з особливостями створення та використання користувацьких функцій на мові Python, використанням глобальних та локальних змінних*

ТЕОРЕТИЧНІ ВІДОМОСТІ

У **Python** усі підпрограми є функціями. З багатьма функціями ми вже зустрічалися (тобто використовували) при написанні програм раніше, *наприклад*: **len(**string**)**, **max(**num**).**

Опис функції

Перш ніж використовувати функцію її необхідно створити (описати**)**.

Для опису функції у **Python** необхідно:

- задати ім'я функції (для іменованих функцій);
- описати аргументи функції (не обов'язково);
- описати програмний код функції;
- описати повернення результату (не обов'язково).

Функція – це ізольована частина коду програми, яка виконується лише під час виклику.

Структура запису функції має наступний вигляд:

def <ім'я функції> (<список параметрів>): **<тіло функції>** **[return <результат>]**	***Наприклад*:** `def hello():` `  print("Hello ")`

Для виклику функції використовуємо ім'я функції, а в дужках вказуємо параметри, які ми їй передаємо (якщо такі є).

Існують функції **з параметром** і **без параметрів**

Параметри функції – це імена об’єктів (змінних, списків тощо), які отримують конкретне значення (атрибути) під час звернення до функції.

Параметри відокремлюються один від одного комою. Якщо функція не має параметрів, вказують лише порожні круглі дужки

Ім’я функції – це ідентифікатор, який позначається латинськими літерами.

Тіло функції – це сукупність інструкцій, які реалізують певне завдання.

```
>>> def suma (): # функція без параметрів
        s=2*5
        print ("s=",s)

>>> suma() #звернення до функції
s= 10
```

Звернення до функції suma

```
>>> def vuras (a,b):
        c=a*2+b*5
        print("c=",c,sep="")
        return(c)

>>> x,y=5,20
>>> vuras(x,y)
c=110
110
>>> x,y=4,6
>>> vuras(x,y)
c=38
38
```

Звернення до функції suma здійснюється два рази

Якщо тіло функції не містить інструкцій, слід розмістити оператор **pass,** який жодних дій не виконує.

def <ім’я функції>:

<тіло функції>

[return <результат>]

Інструкція **return** повертає результат виконання функції до основної програми.

Якщо інструкції **return** не вказано, повертається значення **None**.

```
>>> def suma (a,b): # функція з двома параметрами
        c=a*2+b*2
        print ("c=",c)

>>> a,b=2,10 #Значання аргументів функції suma
>>> print(suma(a,b)) # перше звернення до функції
c= 24
None
```

Оголошення функцій визначається, як правило на початку програми, після імпортування необхідних модулів.

Змінні оголошені в середині функцій, називають **локальними**.

Глобальні змінні – це змінні, які оголошені в основній програмі, тобто за межами функції.

Локальні змінні у функціях і *глобальні змінні* можуть мати однакові імена.

Для запобігання плутанини краще не користуватися однаковими іменами.

Можливості функцій Python:

- в інструкції звернення до функції мати меншу кількість аргументів, ніж кількість параметрів, зазначених у процесі оголошення функції;

```
>>> from math import fmod
>>> def mod(x,y=3):  #функція з необов'язковими параметрами
        return fmod(x,y)

>>> a=mod(8)
>>> print(a)
2.0
>>> print(mod(27,4))
3.0
```

- передавати параметрам значення аргументів не послідовно один за одним, а в довільному порядку;

```
>>> def dob(x,y):
        return x*y

>>> print(dob(y=10,x=5))
50
```

- використовувати у функції змінну кількість параметрів;

```
>>> def dob(*a):
        d=1
        for i in a:
            d*=i
        return d

>>> print(dob(8,5))
40
>>> print(dob(6,6,5,4))
720
```

Мова Python реалізує також **анонімні функції** (лямбда-функції)

Вони не мають імені, а оголошуються за допомогою ключового слова **lambda**:

lambda[<параметр 1>[,…,<параметр N>]]:<значення, що повертається>

Значення, що повертається – це вираз, результат виконання якого повертається функцією.

```
>>> f1=lambda:2+5*5
>>> print(f1())
27
```

```
>>> f2=lambda a,b:a*b+4
>>> print(f2(5,9))
49
```

```
>>> f3=lambda a,b,c=5:a*2+4*b+c
>>> print(f3(4,2))
21
```

Рекурсивні функції

Рекурсією називається алгоритмічна конструкція, де підпрограма викликає сама себе. Рекурсія дає змогу записувати циклічний алгоритм, не використовуючи команду циклу. Максимальне число рекурсивних викликів підпрограми називається *глибиною рекурсії.*

Приклад 1. Обчислити N! факторіал.

```
>>> def fact(n):
        if n==0:
                return 1
        else:
                return n*fact(n-1)


>>> print ("Введіть число")
Введіть число
>>> n=int(input("n="))
n=5
>>> print(fact(n))
120
```

Приклад 2. Використовуючи функцію для обчислення суми S_n за формулою прямокутників інтеграл із заданою точністю ε (вводиться користувачем з клавіатури). Задана точність досягається, коли $|S_n - S_{n-1}| < \varepsilon$. Тоді S_n приймається за наближене значення інтеграла. Для оперативності перевірки правильності обчислень знайти $\int_0^1 x^m dx$, де $m=1$

```
import math
m=1
a=0
b=1
def S(n):
    h=(b-a)/n
    x=a
    Sum=0
    for i in range(n):
        x=x+h
        y=math.exp(m*math.log(x))
        Sum=Sum+y
        S=Sum*h
        return S
print('Введіть точність')
e=float(input('e='))
n=2
Integral=float(S(n-1))
while abs(S(n)-S(n-1))>=e:
    Integral=S(n)
    n=n+1
print('Інтеграл дорівнює Integral=',Integral)
input()
```

Хід роботи

ЗАВДАННЯ 1. *Створити функції для введення, та функцію обчислення елементів масиву. Ввести та опрацювати два масиви цілих чисел.*

1. *Скласти блок-схему алгоритму розв'язування задачі.*
2. *Написати програму реалізації алгоритму з обов'язковими коментарями*
3. *Проаналізувати достовірність отриманих результатів обчислень.*

Варіанти завдань

1. Суму квадратів від'ємних чисел.

2. Добуток від'ємних чисел.

3. Середнє арифметичне від'ємних чисел.

4. Суму квадратів додатних чисел.

5. Добуток додатних чисел.

6. Середнє арифметичне додатних чисел.

7. Суму квадратів парних чисел.

8. Добуток парних чисел.

9. Середнє арифметичне парних чисел.

10.Суму квадратів непарних чисел.

11.Добуток непарних чисел.

12.Середнє арифметичне непарних чисел.

13.Суму чисел, кратних 3.

14.Добуток чисел, кратних 3.

15.Середнє арифметичне чисел, кратних 3.

16.Суму чисел, кратних 4.

17.Добуток чисел, кратних 4.

18.Суму чисел, кратних 5.

19.Добуток чисел, кратних 5.

20.Середнє арифметичне чисел, кратних 5.

 **ЗАВДАННЯ 2.** Створити функції обчислення суми.

№	Вираз	№	Вираз	№	Вираз
1	$\sum_{k=1}^{7}\frac{2^k\sin(x+k)}{(x+1)^k}$	2	$\sum_{k=1}^{9}\frac{x^{k+1}}{(k+1)^x}$	3	$\sum_{k=1}^{12}\frac{\sin(kx)+k}{\sqrt[k]{x+0.1}+6k}$
4	$\sum_{k=1}^{9}\frac{\ln(x+1)}{(x+k)^k}$	5	$\sum_{k=1}^{9}\frac{\sin(2kx)+0.2}{2k+5}$	6	$\sum_{k=1}^{7}\frac{kx\cos(x+k)}{\ln(2+x)+2k}$
7	$\sum_{k=2}^{6}\frac{\sin(0.17x^k)}{2k+x}$	8	$\sum_{k=1}^{8}\frac{5\ln(2kx)}{\operatorname{arctg}(2x)+k^2}$	9	$\sum_{k=2}^{9}\frac{\operatorname{tg}(x)-x^2/k}{k^2-1}$
10	$\sum_{k=1}^{7}\frac{\sin(x^k-\pi)}{\ln k^2+0.3}$	11	$\sum_{k=1}^{12}\frac{\cos(kx)}{k}$	12	$\sum_{k=1}^{8}\frac{\sin x^k}{4k}$
13	$\sum_{k=1}^{7}\frac{\ln^k(3x)}{(2+x)^k}$	14	$\sum_{k=1}^{8}\sqrt[k]{\ln(x+1)}$	15	$\sum_{k=6}^{1}\frac{x^k}{k^3+x^{k+2}}$
16	$\sum_{k=1}^{11}\frac{\sin(x^k-1)}{4k^2+1}$	17	$\sum_{k=1}^{8}\frac{\ln x^{2k-1}}{2^k(2k-1)}$	18	$\sum_{k=2}^{9}\frac{\operatorname{tg}(e^x)}{3k^2+1}$
19	$\sum_{k=3}^{10}\frac{x^{k-1}\cos x}{12^k-1}$	20	$\sum_{k=3}^{11}\frac{\cos^{2+k}x}{2k-1}$	21	$\sum_{k=1}^{8}\sin^k(x)(k+\cos(x+2))$
22	$\sum_{k=2}^{10}\frac{\operatorname{arctg}^3(2kx)}{1.2\ln(k+x)}$	23	$\sum_{k=1}^{11}\frac{\sin(x)^k+0.3}{(2^k)}$	24	$\sum_{k=1}^{6}\frac{k^2\sin^2(x/k)-kx^2}{e^{kx}}$
25	$\sum_{k=1}^{12}\frac{\cos(x^k)}{(x+5)^k+k}$	26	$\sum_{k=2}^{9}\frac{\sin(x+1)+1.5}{\lg(5kx)+2.1}$	27	$\sum_{k=1}^{7}\frac{2(x+1)^{3-k}}{(k+1)^x+k^3}$
28	$\sum_{k=1}^{10}\cos\left(k^3-\frac{kx}{5}\right)$	29	$\sum_{k=1}^{7}\frac{x\sin(x-k)}{e^{2+x}+k}$	30	$\sum_{k=2}^{6}x\cdot\operatorname{arctg}\frac{x-4.4k}{x+\sin(x+k/5)}$

Контрольні запитання

1. Що таке функція і для чого її створюють?

2. Що таке формальні параметри?

3. Що таке фактичні параметри?

4. Де описують і діють глобальні змінні?

5. Де описують і діють локальні змінні?

6. Яке призначення функцій?

7. Який вигляд має функція?

8. Де записують функції?

9. Який вигляд має команда виклику функції?

Лабораторна робота № 14

ТЕМА: **Модуль tkinter. Графічні об’єкти (віджети) та їх властивості**

МЕТА: *навчитися створювати графічні об’єкти за допомогою модуля tkinter*

ТЕОРЕТИЧНІ ВІДОМОСТІ

Tkinter (від англ. **T**ool**k**it for **I**nterfaces) — багатоплатформна (*Windous, Linux, Apple Mac OS та ін.*) графічна бібліотека інтерфейсів на основі засобів **Tk**, написана Стіном Лумхольтом (Steen Lumholt) і Гвідо ван Россумом.

Для створення графічного інтерфейсу користувача необхідно дотримуватися послідовності дій:

1. Створити головне вікно (форму).

2. Створити віджети і конфігурувати їх властивості (опцій).

3. Визначити події, тобто те, на що буде реагувати програма.

4. Визначити обробники подій, тобто те, як буде реагувати програма.

5. Розташувати віджети в головному вікні.

6. Запустити цикл обробки подій.

Створення вікна tkinter

Імпортувати модуль **tkinter** і створити головне вікно. Інструкція імпортування модуля **import tkinter** або **from tkinter import***

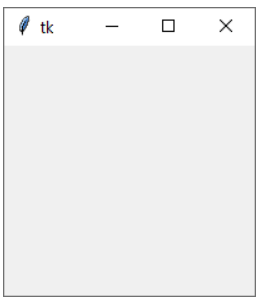
Екранна форма — це вікно, що містить візуальні графічні елементи або об’єкти управління, такі як меню, кнопки тощо.

Для створення вікна використовуємо клас **Tk**.

<ім’я змінної>=<назва класу>

Змінну пов’язану з об’єктом – вікно, зазвичай називають **root**.

```
from tkinter import *
root = Tk()
root.mainloop()
```

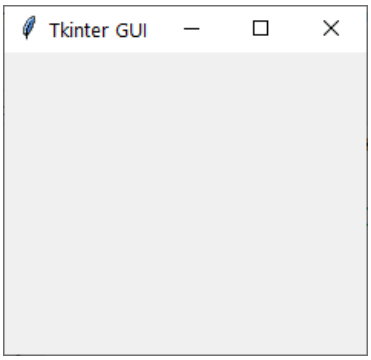


Кореневе вікно має заголовок за замовчуванням **tk**. Також має три системні кнопки, включаючи **Згорнути, Згорнути** та **Закрити**.

Зміна назви вікна

Щоб змінити заголовок вікна, ви використовуєте такий **title()** метод:

```
from tkinter import *
root = Tk()
root.title('Tkinter GUI')
root.mainloop()
```

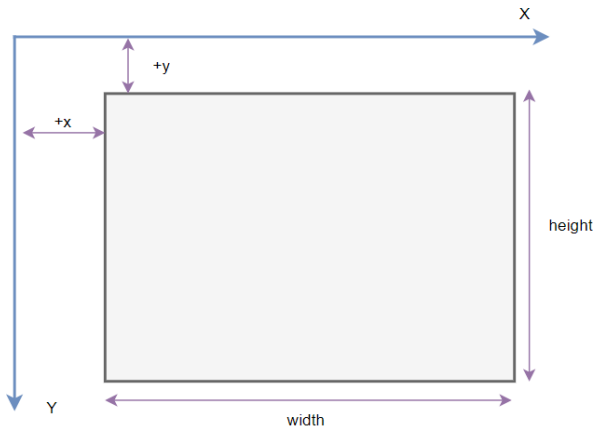


Зміна розмірів вікна

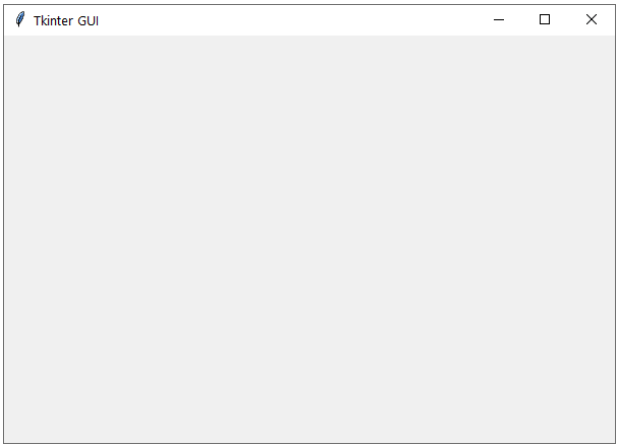
У **tkinter** положення та розмір вікна на екрані визначається геометрією.

<ім'я змінної>.geometry('widthxheight±x±y')

- **width**– ширина;
- **height** – висота;
- *x* – відступ від лівого краю;
- *y* – відступ від правого краю. Задається в пікселях. Параметри *x* та *y* – необов'язкові.



```
from tkinter import *
root = Tk()
root.title('Tkinter GUI')
root.geometry('600x400+50+50')
root.mainloop()
```



Центрування вікна на екрані

```
from tkinter import *
root = Tk()
root.title('Tkinter GUI')
window_width = 300
window_height = 200

# отримати розмір екрану
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()

# знайти центральну точку
center_x = int(screen_width/2 - window_width / 2)
center_y = int(screen_height/2 - window_height / 2)

# встановити положення вікна в центрі екрана
root.geometry(f'{window_width}x{window_height}+{center_x}+{center_y}')
root.mainloop()
```

Зміна розміру поведінки вікна

За замовчуванням ви можете змінити розмір ширини та висоти вікна. Щоб запобігти зміні розміру вікна, можна скористатися методом **resizable**():

Ім’я змінної.resizable(width,height)

```
from tkinter import *
root = Tk()
root.title('Tkinter GUI')
root.geometry('600x400+50+50')
root.resizable(False, False)
root.mainloop()
```

Для заборони зміни розмірів встановлюємо значення **width=0** та **height=0**.

Максимальний та мінімальний розміри. Методи minsize() і maxsize()

- ***minsize(*width,height*)*** - задає мінімальний розмір вікна (**width** – ширина, **height** – висота) у пікселях. Якщо не задавати – вікно не матиме обмежень у зменшені.

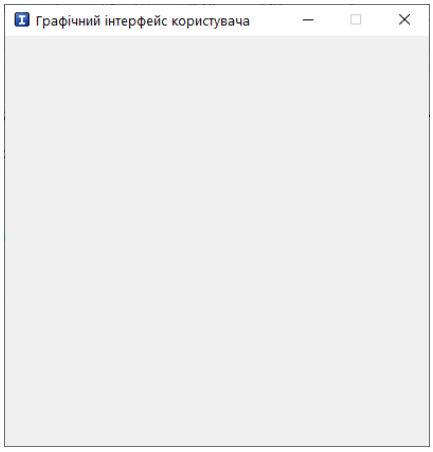
maxsize(*width,height*) - задає максимальний розмір вікна (**width** – ширина, **height** – висота) у пікселях. Якщо не задавати – вікно не матиме обмежень у збільшені.

Зміна значка за замовчуванням

У вікні **tkinter** відображається піктограма за замовчуванням. Щоб змінити цей значок за замовчуванням, виконайте такі дії:

- Підготуйте зображення у .ico форматі. Помістіть піктограму в папку, до якої можна отримати доступ із програми.
- Викличте **iconbitmap**() метод віконного об'єкта.

```
from tkinter import*
root=Tk()
root.geometry("400x400+300+300")
root.resizable(False,False)
root.title("Графічний інтерфейс користувача")
root.iconbitmap("as.ico")
mainloop()
```



Колір вікна

Щоб змінити колір вікна, можна скористатися командою:

Ім’я змінної [‘bg’]=’назва кольору’

```
root['bg'] ='green'
```

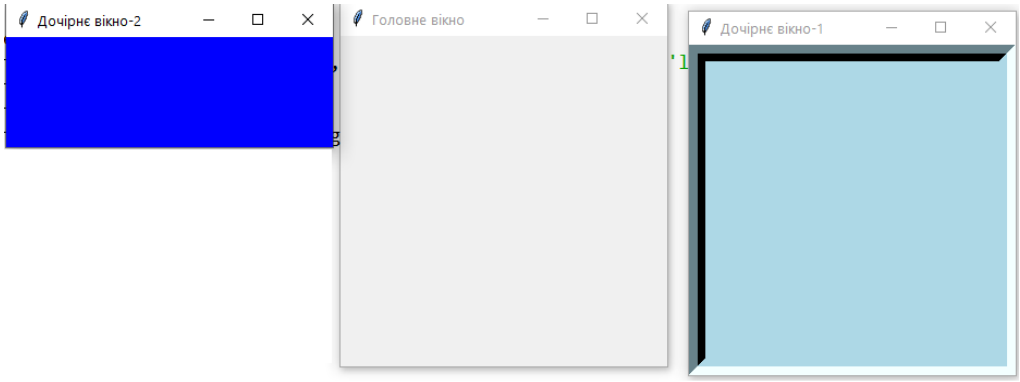
Вікно верхнього рівня

Вікно верхнього рівня є дочірнім головного вікна. Ці вікна найчастіше використовуються для групування об’єктів за призначенням. На екрані можна відобразити кілька таких вікон. Закриття головного вікна призведе й до закриття дочірнього, а закриття дочірніх не призводить до закриття головного вікна. Для створення вікна верхнього рівня викликається клас **Toplevel**. Структура конструктора створення дочірнього вікна:

<змінна>= Toplevel (<Параметр_1>,< Параметр_2>,…,< Параметр_n>)

Наприклад, створити два дочірніх вікна можна.

```
from tkinter import *
root = Tk()
root.title('Головне вікно')

root.geometry('300x300+400+200')
toproot1=Toplevel(root,relief=SUNKEN,width=200,height=100,bd=15,bg='lightblue')
toproot1.geometry('300x300+800+200')
toproot1.title('Дочірнє вікно-1')
toproot2=Toplevel(root,width=300,height=100,bd=10,bg='blue')
toproot2.title('Дочірнє вікно-2')

root.mainloop()
```



Відображення вікна на екрані. Метод mainloop()

Для завершення роботи з вікном необхідно використати метод **mainloop()**. Метод ***mainloop ()*** об'єкта ***Tk*** запускає головний цикл обробки подій, що в тому числі призводить до відображення головного вікна з усіма його елементами на екрані.

Синтаксис виклику метода:

Ім’я_змінної.mainloop()

! Дана інструкція має бути останньою в програмі.

Віджети

Візуальні елементи керування або **віджети** (від англ. *widget – window gadget*) – це графічні об'єкти, розташовані у вікні програми для показу або введення даних, виконання дій або полегшення роботи: *текстові поля, списки, перемикачі, кнопки, прапорці* тощо.

Button (*Кнопка*). *Кнопки* використовують для управління програмою. Кнопці відповідає клас **Button.** Структура конструктора створення кнопки:

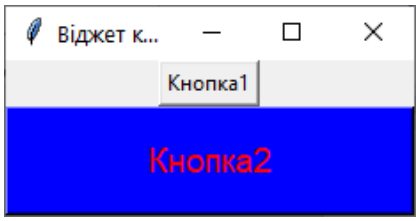
<змінна>=Button(<Параметр_1>,< Параметр_2>,…,< Параметр_n>)

Параметр_1 (якщо він вказаний) називається *батьківським*. Цей параметр визначає посилання на об'єкт, у якому створюється кнопка. Інші параметри визначають властивості кнопки та їх значення. Ці параметри оголошуються за структурою:

<властивість=значення>

- **text** – текст, який буде відображатися на кнопці
- **bg/ background**-фоновий колір кнопки
- **width, height -** ширина і висота кнопки
- **fg/ foreground** – колір тектсу на кнопці
- **font** – шрифт, наприклад, font=("Verdana", 13, "bold")
- **bd/ border-width** – товщина границі (за замовчуванням 2)
- **justify** – вирівнювання тексту. Значення LEFT вирівнювання тексту по лівому краю, CENTER - по центру, RIGHT - по правому краю
- **padx** – відступ від границі кнопки до її тексту зправа і зліва
- **pady** – відступ від границі кнопки до її тексту зверху і знизу

```
from tkinter import*
root = Tk()
root.title("Віджет кнопка")
button1=Button(root,text="Кнопка1")
button2=Button(root,
               text="Кнопка2",
               width=20,height=2,
               bg="blue",fg="red",
               font="Arial 14")
button1.pack()
button2.pack()
root.mainloop()
```



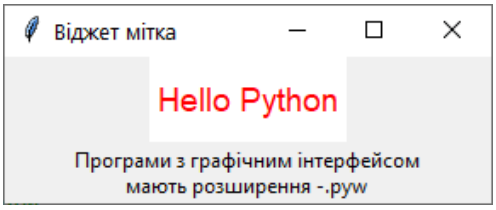
Label (*Мітка*). *Мітки* (написи) використовують для інформування користувача про результати виконання програмою дій або про дії які необхідно

виконати. Мітки можуть містити один або декілька рядків. Для створення мітки викликається клас **Label**

Структура конструктора створення мітки:

<змінна>= Label (<Параметр_1>,< Параметр_2>,…,< Параметр_n>)

- **bg/background** – фоновий колір
- **bd/border-width** – товщина границі мітки
- **fg/foreground** –колір тексту
- **font** – шрифт тексту
- **height, width –** висота, ширина
- **justify** – вирівнювання тексту. Значення LEFT вирівнювання тексту по лівому краю, CENTER - по центру, RIGHT - по правому краю
- **text** - встановлює текст мітки
- **textvariable** – втановлює прив'язку до елементу StringVar

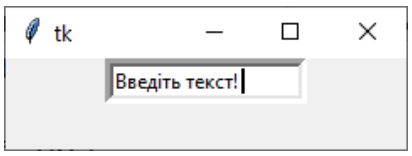



```
from tkinter import*
root = Tk()
root.title("Віджет мітка")
label1=Label(root,text="Hello Python",
                  width=10,height=2,
                  bg="white",fg="red",
                  font="Arial 14")
Text = "Програми з графічним інтерфейсом\nмають розширення -.pyw"
label2 = Label(text=Text, justify=CENTER)
label1.pack()
label2.pack()
root.mainloop()
```

Entry (*Однорядкове текстове поле*). Це поле використовується для введення користувачем тексту в один рядок. Для його створення викликається клас **Entry.**

Структура конструктора створення однорядкового текстового поля:

<змінна>= Entry (<Параметр_1>,< Параметр_2>,…,< Параметр_n>)




```
from tkinter import *
root = Tk()
edit1 = Entry(root,width=18,bd=5).pack()
root.mainloop()
```

- **bg/background** – фоновий колір
- **bd/border-width**– товщина границі текстового поля
- **fg/foreground** –колір тексту
- **font** – шрифт тексту
- **height, width –** висота, ширина
- **justify** – вирівнювання тексту. Значення LEFT вирівнювання тексту по лівому краю, CENTER - по центру, RIGHT - по правому краю
- **text** - встановлює текст мітки
- **textvariable** – втановлює прив'язку до елементу StringVar

Методи Entry

<ім'я змінної текстового поля>.**get()** – дозволяє отримати значення, що знаходиться в текстовому полі.

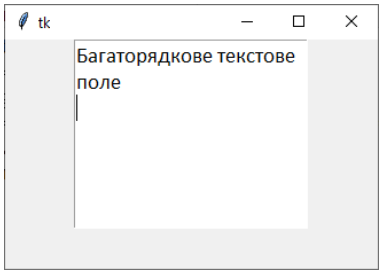
<ім'я змінної текстового поля>.**insert(index, str)** – виводить в текстове поле рядок, починаючи зі знакомісця з номером **index**.

<ім'я змінної текстового поля>.**delete(first, last)** – вилучає символи, починаючи зі знакомісця з номером **first** до знакомісця з номером **last** (нумерація символів з 0). Щоб вилучити весь текст, в якості другого параметра потрібно вказати **END**.

Бажано перед викликом методу **insert**() записувати виклик методу **delete**(), тобто перед виведенням очищувати текстове поле.

Text (*Багаторядкове текстове поле*). Це поле використовується для введення тексту користувачем. Для його створення викликається клас **Text.** Структура конструктора створення багаторядкового текстового поля:

<змінна>= Text (<Параметр_1>,< Параметр_2>,…,< Параметр_n>)




```
from tkinter import *
root = Tk()
root.geometry("200x200")
text1=Text(root,width=20,height=7,font="Calibri 14",wrap=WORD) # height=7 кількість рядків
text1.pack()
root.mainloop()
```

- **bg/background** – фоновий колір
- **bd/border-width**– товщина границі текстового поля
- **fg/foreground** –колір тексту
- **font** – шрифт тексту
- **height, width –** висота, ширина
- **justify** – вирівнювання тексту. Значення LEFT вирівнювання тексту по лівому краю, CENTER - по центру, RIGHT - по правому краю
- **text** - встановлює текст мітки
- **wrap** – визначає, що слова не будуть розриватися в процесі перенесення на новий рядок

Методи текстового поля Text

<ім'я змінної текстового поля>.**get**(початок, кінець) – *повертає фрагмент тексту від символу, позиція якого визначається першим параметром, до символу, позиція якого визначена другим параметром (нумерація рядків починається з 1, а символів в рядку починається з 0)*

< ім'я змінної текстового поля >.**insert**(позиція, текст) – *вставляє текст в поле перед символом індекс якого вказаний, як парметр позиції (нумерація починається з 0)*

< ім'я змінної текстового поля >.**delete**(початок, кінець) – *видаляє фрагмент тексту від символу, позиція якого визначається першим параметром, до символу,*

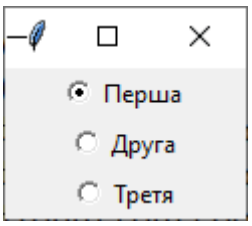
позиція якого визначена другим параметром (нумерація рядків починається з 1, а символів в рядку починається з 0)

Radiobutton *(Радіокнопки-перемикачі).* Радіокнопки-пермикачі призначені для вибору одного з кількох запропонованих варіантів. Радіокнопок має бути декілька, у кожен окремий момент можна увікнути лише одну радіокнопку.

Для створення викликається клас **Radiobutton.**

Структура конструктора створення перемикачів:

<змінна>= Radiobutton (<Параметр_1>,< Параметр_2>,…,< Параметр_n>)




```
from tkinter import*
root=Tk()
var = IntVar()                # створюємо змінну, яка приймає цілі значення
var.set(1)                    # встановлюємо перше значення для створеної змінної
rad1 = Radiobutton(root, text="Перша", variable=var, value=1)    # створюємо перемикач зі значенням 1
rad2 = Radiobutton(root, text="Друга", variable=var, value=2)    # створюємо перемикач зі значенням 2
rad3 = Radiobutton(root, text="Третя", variable=var, value=3)    # створюємо перемикач зі значенням 3

# розміщуємо перемикачі у вікні
rad1.pack()
rad2.pack()
rad3.pack()
```

Властивість **variable** зв’язує змінну з перемикачем, а властивість **value** визначає значення, яке буде передано змінній, якщо цю кнопку увімкнути.

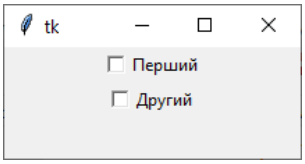
Tkinter не може використовувати будь-яку змінну для зберігання станів віджетів. Для цих цілей передбачені спеціальні класи-змінні пакета **tkinter**:

- **BooleanVar** (приймає булеві значення),
- **IntVar** (приймає цілі значення),
- **DoubleVar** (приймає дробові та цілі значення),
- **StringVar** (приймає рядкові значення).

Checkbutton (*Прапорці*). Прапорців, як і радіокнопок, може бути кілька. Одночасно ввімкненими можуть бути кілька прапорців. Для його створення викликається клас **Checkbutton.**

Структура конструктора створення перемикачів:

<змінна>= Checkbutton (<Параметр_1>,< Параметр_2>,…,< Параметр_n>)




```
from tkinter import *
root = Tk()
var1 = IntVar()                          # створення змінної цілого типу для першого прапорця
check_b1 = Checkbutton(root,
                       text="Перший",
                       variable=var1,    # значення змінної першого прапорця
                       onvalue=1,        # значення при включеному прапорці
                       offvalue=0)       # значення при вимкненому прапорці
check_b1.pack()

var2 = IntVar()                          # створення змінної цілого типу для другого прапорця
check_b2 = Checkbutton(root,
                       text="Другий",
                       variable=var2,    # значення змінної другого прапорця
                       onvalue=1,        # значення при включеному прапорці
                       offvalue=0)       # значення при вимкненому прапорці
check_b2.pack()
root.mainloop()
```

Властивість **variable** зв’язує змінну з прапорцем, а властивість **value** визначає значення, яке буде передано змінній, якщо цю кнопку увімкнути.

Listbox (*Список*). Це віджет, який представляє собою список, з елементів якого користувач може вибирати один або кілька пунктів.

Для його створення викликається клас **Listbox.**

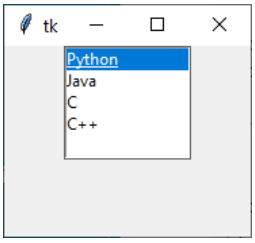
Структура конструктора створення списку:

<змінна>= Listbox (<Параметр_1>,< Параметр_2>,…,< Параметр_n>)

- **bg/background** – фоновий колір списку
- **bd/border-width**– товщина границі текстового поля
- **fg/foreground** –колір тексту
- **font** – шрифт тексту
- **height, width –** висота, ширина списку
- **text** – заголовок списку
- **selectmode** – визначає скільки елементів можна вибрати і перетягування миші впливає на вибір:

- SINGLE – можна вибрати лише один рядок, і не можна перетягувати курсор миші;

- EXTENDED – можна вибрати будь-яку суміжну групу рядків одночасно, натиснувши на перший рядок і перетягнувши на останній рядок.




```
from tkinter import *
root = Tk()
list1 = ["Python", "Java", "C", "C++"]                          # задаємо елементи, які повинні

listbox1 = Listbox(root, height=5, width=15, selectmode=EXTENDED)    # створюємо віджет – список

for i in list1:                                                 # додаємо елементи у віджет
    listbox1.insert(END, i)
listbox1.pack()
root.mainloop()
```

Методи Listbox

<ім'я змінної списку>.**get**(початок, кінець) – *повертає фрагмент списку від рядка, позиція якого визначається першим параметром, до рядка, позиція якого визначена другим параметром*

<ім'я змінної списку>.**insert**(позиція, назва_елемента) – *вставляє елемент в список після даної позиції, визначеної першим параметром (нумерація починається з 0)*

<ім'я змінної списку>.**delete**(початок, кінець) – *видаляє зі списку всі елементи.*

Scrollbar (*Смуга прокручування***).** Використовується для прокручування вмісту іншого віджета, *наприклад*, списку (**ListBox)** або багаторядкового текстового поля (**Text**). Перед створенням смуги прокручування необхідно визначити об'єкт (список або текстове поле), для якого вона створюється.

Структура конструктора створення **Scrollbar**:

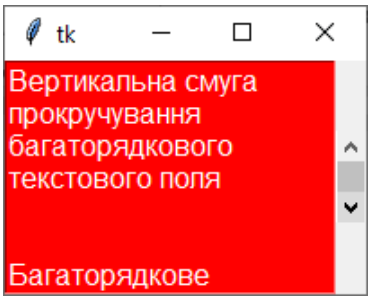
<змінна>= Scrollbar (<Параметр_1>,< Параметр_2>,…,< Параметр_n>)

Command – прив'язує смугу прокручування до віджета.

Значення:

- ім'я_віджета.**xview** – горизонтальне розташування смуги прокручування;
- ім'я_віджета.**yview** – вертикальне розташування смуги прокручування

```
from tkinter import *
root = Tk()
text = Text(width=20,          # ширина текстового поля
            height=7,          # висота текстового поля
            bg="red",          # колір фону текстового поля
            fg='white',        # колір тексту текстового поля
            font='14',wrap=WORD)
text.pack(side=LEFT)
# створення і розміщення скроллера (смуги прокручування вертикальної)
scroll = Scrollbar(command=text.yview)
text.grid(row=0,column=0)
scroll.grid(row=0,column=1)
root.mainloop()
```



Вікно повідомлень

Пакет **tkinter** містить модуль **messagebox**, який надає доступ до вікон повідомлень.

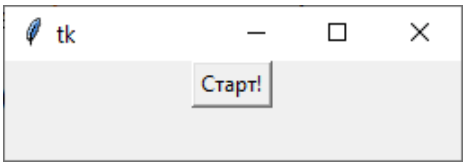
from tkinter import messagebox

Для того, щоб згенерувати вікно повідомлення, потрібно для об'єкта **messagebox** викликати функції:

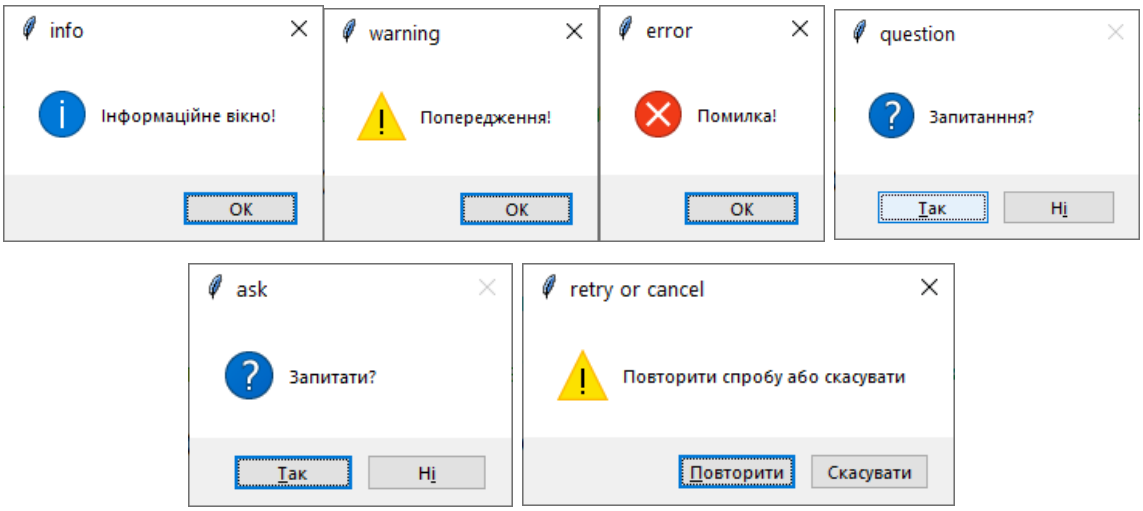
- **showinfo,**
- **showwarning,**
- **showerror,**
- **askquestion,**
- **askyesno,**
- **askretrycancel.**

Синтаксис виклику:

messagebox. функція (заголовок, текст_повідомлення)



```
from tkinter import *
from tkinter import messagebox

root = Tk()

def Showmessage():
    messagebox.showinfo("info", "Інформаційне вікно!")
    messagebox.showwarning("warning", "Попередження!")
    messagebox.showerror("error", "Помилка!")
    messagebox.askquestion("question", "Запитанння?")
    messagebox.askyesno("ask", "Запитати?")
    messagebox.askretrycancel("retry or cancel", "Повторити спробу або скасувати")

Button(root, text="Старт!", command=Showmessage).pack()

root.mainloop()
```



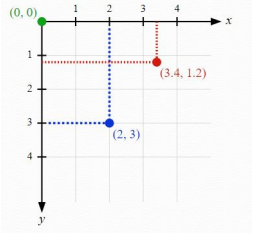
Полотно для малювання Canvas

У **tkinter** зображення створюється в межах полотна — об’єкта класу **Canvas**, який входить до модуля **tkinter**.

Об’єкт Canvas (полотно) — це віджет бібліотеки **tkinter**, на якому можуть бути розміщені інші віджети: геометричні фігури, малюнки, фото, графічні зображення тощо.

Canvas має систему координат, початок яких розташовується в лівому верхньому куті полотна. Першою завжди вказується координата x, а другою — координата y.

Для задавання положення точок на полотні використовують координати. Будь-яка точка може бути задана парою чисел (X, Y), де X — відстань від точки до лівого краю полотна, Y — відстань від точки до верхнього краю полотна.



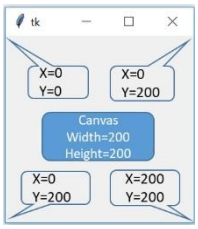
Перш ніж записувати методи для малювання, потрібно створити полотно. Синтаксис створення об’єкта класу **Canvas**:

змінна = **Canvas**(батьківський віджет, width = значення, height = значення)

де width — ширина полотна; height — його висота (в пікселях).

Наприклад, створити полотно розміром 200 х 200 пікселів:

```
from tkinter import*
root=Tk()
canvas=Canvas(root,width=200,height=200)
canvas.pack()
root.mainloop()
```



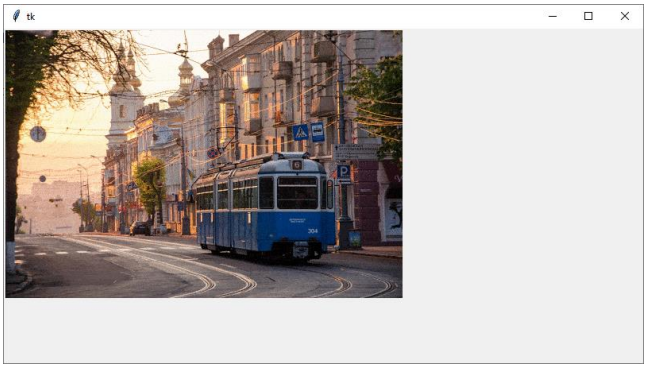
Колір полотна за замовчуванням світло-сірий, змінити фон можна за допомогою методу **config**.

```
canvas.config(bg='#49F477')
```

Виведення зображень із графічних файлів. Метод create_image

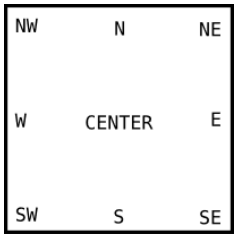
Засобами **tkinter** можна вивести на полотно зображення з графічного файла. Для цього потрібно ім’я графічного файла завантажити до змінної за допомогою функції **PhotoImage** (file='ім'я файла') і викликати метод **create_image**.

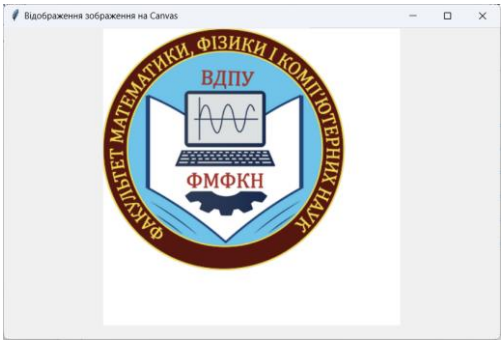
Вивести на полотно рисунок vin.gif, який міститься в папці з кодом програми:



```
from tkinter import *
root = Tk()
canvas= Canvas(root,width=500,height=500)
photo = PhotoImage(file="vin.gif")
canvas.create_image(0,0,anchor=NW,image=photo)
canvas.pack(anchor=NW)
root.mainloop()
```

Параметр **anchor** визначає розташування рисунка на полотні. Значення **NW** вказує на верхній лівий кут полотна.





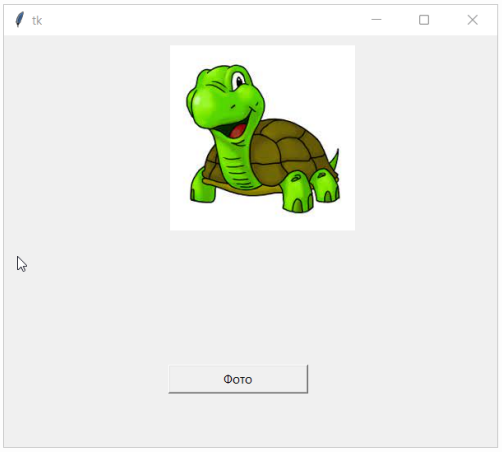

```
import tkinter as tk
root = tk.Tk()
root.title("Відображення зображення на Canvas")

# Створення Canvas
canvas = tk.Canvas(root, width=400, height=400, bg="white")
canvas.pack()

# Завантаження зображення у форматі PNG
image_path = "logo1.png"
photo = tk.PhotoImage(file=image_path)

# Додавання зображення до Canvas з прив'язкою NW
canvas.create_image(0, 0, image=photo, anchor=tk.NW)

root.mainloop()
```

Приклад 1

```
from tkinter import *
from tkinter import Tk, Canvas
from PIL import ImageTk, Image
root =Tk()
canvas = Canvas(root, width=300, height=300)
canvas.pack()
def picture1():
    im = Image.open('shake.jpg')
    canvas.image = ImageTk.PhotoImage(im)
    canvas.create_image(0, 0, image=canvas.image, anchor='nw')
def picture2():
    im = Image.open('turtle.jpg')
    canvas.image = ImageTk.PhotoImage(im)
    canvas.create_image(0, 0, image=canvas.image, anchor='nw')
button1 = Button(root, text='shacke',width=20,command=picture1)
button1.pack()
button2 = Button(root, text='turtle', width=20,command=picture2)
button2.pack()
root.mainloop()
```

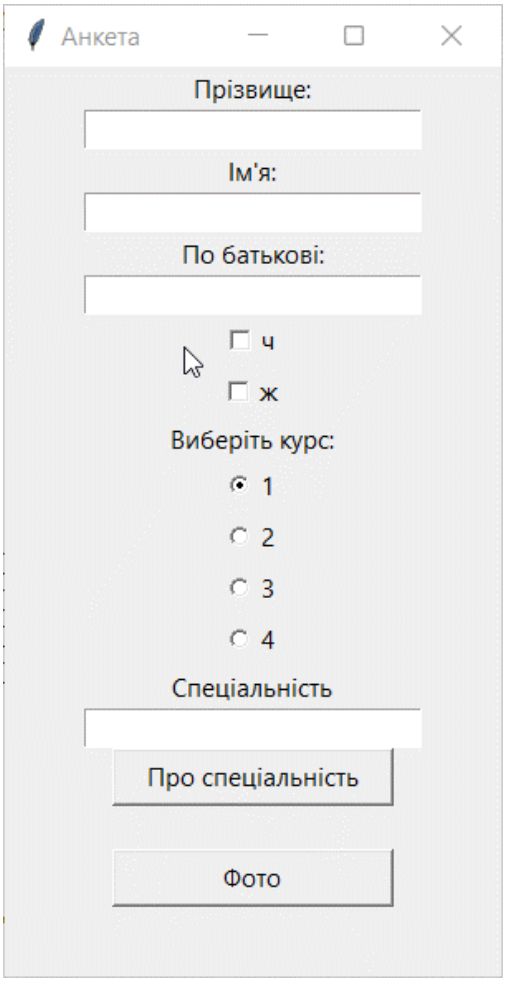
Приклад 2

```
from tkinter import*
from tkinter import messagebox
from tkinter import Tk, Canvas
from PIL import ImageTk, Image
from tkinter import filedialog
root=Tk()
root.geometry('600x500')
def openpicture():
    global img
    filename=filedialog.askopenfilename()
    img=ImageTk.PhotoImage(Image.open(filename))
    label1.config(image=img)
label1=Label(root,text='Фото',justify =CENTER,image=None)
label1.place(x=200,y=10)
button1=Button(root,text="Фото",width=20,command=openpicture)
button1.place(x=200,y=400)
root.mainloop()
```



Хід роботи

ЗАВДАННЯ 1. Створити проект «Анкета»



1. Імпортуйте графічну бібліотеку **tkinter**.Створіть головне вікно, задавши йому ім’я root (*або будь-яке інше*), та вивести вікно використавши метод **mainloop()** для відображення вікна після запуску програми.

```
from tkinter import *
root = Tk()
# віджети
root.mainloop() # команда відображення вікна при запуску
```

Надалі усі інші команди повинні міститись між цими двома рядками.

2. Заголовок вікна – «Анкета»;
3. Розміри вікна: **width** – 300, **width** – 550, відступ з лівого краю *x*=500, відступ від правого краю y=100 пікселів;
4. Колір вікна– обрати колір із запропонованої таблиці кольорів на власний смак;

5. Розмісти у вікні текст «Прізвище». Для цього створіть змінну *label1* та задайте для неї тип віджета класу **Label,** розташуйте створений віджет у головному

вікні використовуючи метод **pack**() без параметрів. Вкажіть у дужках відповідні параметри.

```
label1 = Label(root, text="Прізвище:")
label1.pack()
```

6. Створити текстове поле класу Entry для введення прізвища та розташуйте його у головному вікні використовуючи метод **pack**() без параметрів. Задай назву цього об'єкта – *edit1*. Вкажіть у дужках відповідні параметри.

```
edit1=Entry(root,width=25)
edit1.pack()
```

7. Аналогічно задай та розташуйте текст «Ім'я» і текстове поле для введення ім'я (змінні *label2* та *edit2*).

8. Аналогічно задай та розташуйте текст «По батькові» і текстове поле для введення даного тексту (змінні *label3* та *edit3*).

9. Задайте та розташуйте у вікні прапорець з текстом «ч».

```
checkbutton1 = Checkbutton(root, text="ч")
checkbutton1.pack()
```

10. Аналогічно задай прапорець з текстом «ж» (checkbutton2).

11. Створіть текстову мітку

```
label4=Label(root, text="Виберіть курс:")
label4.pack()
```

12. Створіть чотири перемикачі **Radiobutton** з номерами, курси навчання

```
result1=IntVar()
result1.set(1)
Radiobutton1=Radiobutton(root, text=1,variable=result1, value=1).pack()
Radiobutton2=Radiobutton(root, text=2,variable=result1, value=2).pack()
Radiobutton3=Radiobutton(root, text=3,variable=result1, value=3).pack()
Radiobutton4=Radiobutton(root, text=4,variable=result1, value=4).pack()
```

13. Створити мітку та поле для введення

```
label5=Label(text='Спеціальність')
label5.pack()
edit4=Entry(root,width=25)
edit4.pack()
```

14. Створити кнопку «Про спеціальність»

```
button1=Button(root, text="Про спеціальність",width=20,command=inf)
button1.pack()
```

15. Створити мітку, для розміщення фото

```
label6=Label()
label6.pack()
```

16. Створити кнопку «Фото»

```
button2=Button(root, text="Фото",width=20, command=show)
button2.pack()
```

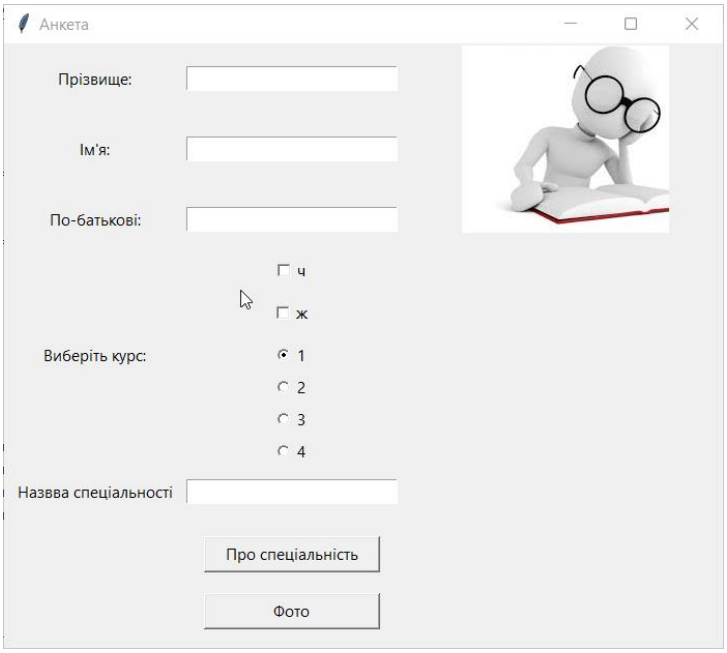
17.Створіть функцію обробника подій для кнопки «Фото» на початку програми

```
def show():
    global photo
    photo = PhotoImage(file="11.gif")
    label5.configure(image = photo)
```

18.Створіть функцію обробника подій для кнопки «Про спеціальність» на початку програми

```
def inf():
    s=edit4.get()
    if s=="111M":
       messagebox.showinfo("Про спеціальність", edit4.get() + '''\nСпеціальность 111 Математика\n
       Освітня програма Комп'ютерна математика''')
    elif s=="COM":
        messagebox.showinfo("Про спеціальність", edit4.get() + '''\nСпеціальность 014 Середня освіта\n
        Освітня програма Середня освіта. Математика, інформатика''')
    else:
          messagebox.showinfo("Про спеціальність", edit4.get() + "\nТакої спеціальності на факультеті немає")
```

19.Модифікувати програму, задати додаткові параметри для віджетів, для розташування віджетів у вікні використати метод **grid**() або **place**()



Контрольні запитання

1. Для чого призначений модуль **tkinter**?
2. Що таке графічний інтерфейс користувача?
3. Які основні об’єкти містить модуль **tkinter**?
4. За допомогою якої команди створюються віджети?
5. Назвіть загальний порядок створення графічного інтерфейсу користувача?
6. Що таке функція? Яку загальну структуру має функція?
7. Яку загальну структуру має конструктор створення кнопки?
8. Поясніть призначення параметрів мітки?
9. Для чого призначені текстові поля, назвіть параметри та методи?
10. Для чого призначені радіокнопки, перемикачі?

Лабораторна робота № 15

ТЕМА: Python модуль tkinter. Опрацювання подій

МЕТА: *навчитися опрацьовувати події*

ТЕОРЕТИЧНІ ВІДОМОСТІ

Додатки з графічним інтерфейсом користувача подійно-орієнтовані. Тобто та чи інша частина програмного коду починає виконуватися лише тоді, коли виконується та чи інша подія.

Подія – *зміна властивостей об'єкта, взаємодія між об'єктами, створення нового або знищення існуючого об'єкта.*

Зовнішня дія на віджет називається **подією**.

Події

1. Події, що виникають у результаті дії на мишу.

'<Button-1>' – клацання ЛКМ;

'<Button-3>' – клацання ПКМ;

'<Double-Button-1>' – подвійне клацання ЛКМ;

'<MouseWheel>' – прокручування колесика мишки;

'<Motion>' – рух курсора миші.

2. Події, що виникають внаслідок дії на клавіші.

Клавіші букв записують у лапках, *наприклад* **'<W>'**.

Для неалфавітних клавіш існують спеціальні зарезервовані слова. *Наприклад*, натискання клавіши **Enter** позначається '<**Return**>' , а клавіша «*пробіл*» – '<**space**>'

Для клавіш, які натискаються одночасно, теж існують спеціальні позначення. *Наприклад*, натискання клавіш **Ctl+Shift** позначається **'<Control-Shift>'**, а натискання клавіш **Ctrl+z** – **'<Control-z>'**.

3. Події, що виникають внаслідок зміни властивостей інших об'єктів.

Об’єкти реагують на події, які виконуються в програмі. Кожна подія приводить до виконання методу об’єкта або описаної користувачем функції.

Опрацювання подій часто реалізують у вигляді функцій, які викликаються під час виникнення певної події.

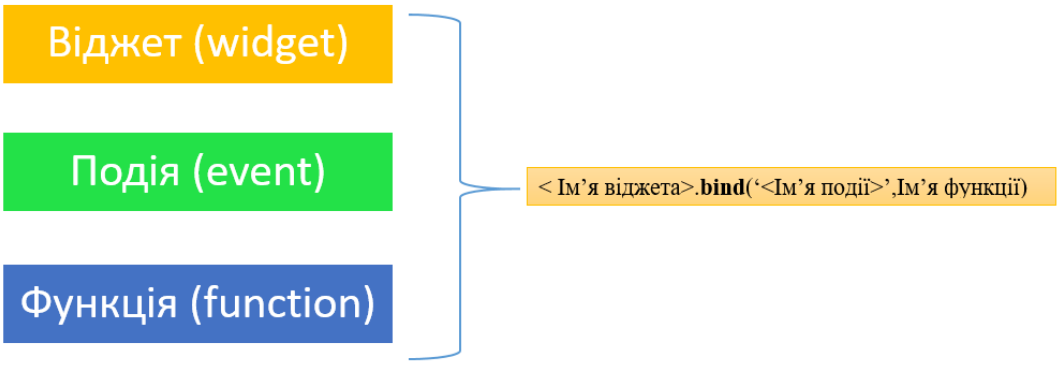
Функція має структуру:

def <назва функції>(<назва події>)

<тіло функції>

Функція пов’язана з певною подією, називається **обробником події.**

Щоб можна було виконати функцію опрацювання події, необхідно зв’язати її із самою подією. Зв’язування реалізується за допомогою методу **bind**().

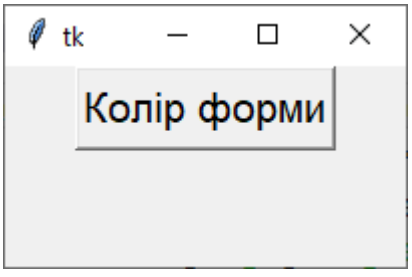


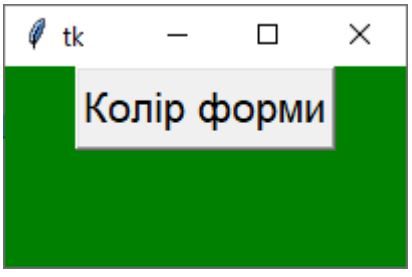
Де ***Ім’я функції*** – функція, в якій запрограмована реакція на вказану подію.

Функція, яка викликається при настанні події, повинна приймати один аргумент. Це об'єкт класу ***event***, в якому описані властивості події, що відбулася.

Метод ***bind*** описується в кінці програми, перед методом ***mainloop()***

Приклад 1. Віджет — кнопка, подія — натиснення на кнопку лівою клавішею миші, дія — зміна кольору фону вікна.





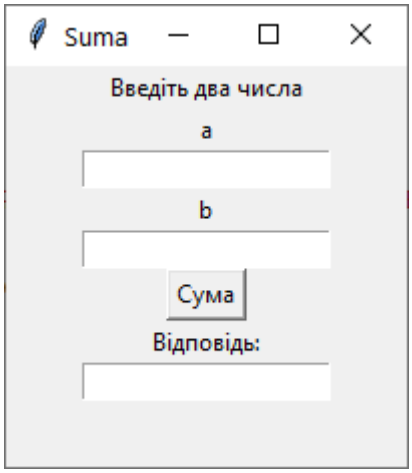
```
from tkinter import*
def button_click(event):
    root['bg'] ='green'
root=Tk()
root.geometry('200x100')
button1=Button(root,text='Колір форми',width=10,font='Arial 16')
button1.bind("<Button-1>",button_click)
button1.pack()
root.mainloop()
```

Щоб прикріпити до віджета **Button** обробник події, необхідно при створенні цього об‘єкту в переліку атрибутів указати параметр **command** і присвоїти йому посилання на метод, який буде виконуватися при натисканні.

```
from tkinter import*
def button_click():
    root['bg'] ='green'
root=Tk()
root.geometry('200x100')
button1=Button(root,text='Колір форми',width=10,font='arial 16', command = button_click)
button1.pack()
root.mainloop()
```

!!! `button1.bind_all('<space>',button_click)` # для клавіатури

Приклад 2. Обчислити суму двох чисел.



```
from tkinter import*
root=Tk()
root.title("Suma")
root.geometry('200x200+500+200')
a=StringVar()
b=StringVar()
suma = StringVar()
def Suma():
    summa = float(a.get()) + float(b.get()) #читати значення з зв'язаних змінних
    suma.set('%.0f '%summa)
lab1=Label(root,text="Введіть два числа").pack()
lab2=Label(root,text="a").pack()
edit1=Entry(textvariable=a).pack()
edit1=a.get()
lab3=Label(root,text="b").pack()
edit2=b.get()
edit2=Entry(textvariable=b).pack()
buton=Button(root,text='Сума',command=Suma).pack()
lab4=Label(root,text="Відповідь:").pack()
edit3=Entry(textvariable=suma).pack()
root.mainloop()
```

Хід роботи

ЗАВДАННЯ 1. Проект «**Функція**».

Варіанти завдань

№	Розрахункова формула	Контрольні значення змінних
1.	$f=\frac{4\sin(a+bc)}{tg(c-b)+a^3}+\frac{b}{c}$	$a=2;\ b=4,5;\ c=6$
2.	$f=\frac{\sqrt{bc+a^2}}{\ln a}+\cos b$	$a=6,7;\ b=-5;\ c=0,5$
3.	$f=\frac{-b-\sqrt{b^2+4ac}}{2a}+\lvert c\rvert$	$a=5;\ b=9;\ c=-2,5$
4.	$f=ab+\frac{b^a-\sin c}{arctg\,a}$	$a=0,2;\ b=4,5;\ c=8$
5.	$f=ctg\,c+\sqrt{\frac{c}{b}+a}$	$a=4;\ b=3,2;\ c=0,1$
6.	$f=\frac{c^b-4tg\,a}{\ln b}-\{bc\}$	$a=5;\ b=2;\ c=5,4$

7.	$f=\dfrac{a+\cos b}{3a+4\sqrt{a+c}}-5c$	$a=5;\ b=-4;\ c=1$
8.	$f=\dfrac{e^b+3\ln(a+c)}{arctg\,a}+[bc]$	$a=-5;\ b=3,2;\ c=9$
9.	$f=\dfrac{\sqrt{\dfrac{a}{b-a}}}{b^2+ab}-\lvert b+2\rvert$	$a=-1;\ b=-4;\ c=6,3$
10.	$f=\dfrac{a-2ctg\,b}{\ln(3c)+3}+a^b$	$a=1,5;\ b=8;\ c=0,4$
11.	$f=\dfrac{3\cos(a+bc)}{ctg(c-b)+a^3}+\left\{\dfrac{c}{b}\right\}$	$a=2;\ b=4,5;\ c=6$
12.	$f=\dfrac{\sqrt{bc+a^2}}{\ln b}+\sin a$	$a=6,7;\ b=-5;\ c=0,5$
13.	$f=\dfrac{-b+\sqrt{b^2+4ac}}{2a}-\lvert c\rvert$	$a=5;\ b=9;\ c=-2,5$
14.	$f=ab-\dfrac{a^c-\cos a}{arctg\,b}$	$a=0,2;\ b=4,5;\ c=8$
15.	$f=tg\,c-\sqrt{\dfrac{c}{b}+a}$	$a=4;\ b=3,2;\ c=0,1$
16.	$f=\dfrac{c^b+2tg\,a}{\ln c}+[abc]$	$a=5;\ b=2;\ c=5,4$
17.	$f=\dfrac{a+tgb}{3a-4\sqrt{a+c}}+3\lvert b\rvert$	$a=5;\ b=-4;\ c=1$
18.	$f=\dfrac{e^a-4\ln(c-a)}{arctg\,b}-bc$	$a=-5;\ b=6,2;\ c=9$
19.	$f=\dfrac{\sqrt{\dfrac{b}{b-a}}}{b^2+abc}+\lvert b+a\rvert$	$a=-1;\ b=-4;\ c=6,3$
20.	$f=\dfrac{b-2ctg\,a}{\ln(4c)-1}+e^c$	$a=2,5;\ b=8;\ c=0,4$

ЗАВДАННЯ 2. Проект «Трикутник». Розробити проект для обчислення значень вказаних елементів трикутника, заданого величинами його сторін. Дано сторони a, b, і c.

Номер варіанта	**Величина, яку потрібно обчислити**	**Довжини сторін, см**		
		a	**b**	**c**
1.	Висота h_a, бісектриса w_β	12,3	14,5	16,7
2.	Медіана m_a, радіус описаного кола R	11,4	12,6	17,1
3.	Бісектриса w_α, радіус вписаного кола r	21,8	24,9	30,6
4.	Радіус описаного кола R, висота h_b	40,6	39,5	41,8
5.	Радіус вписаного кола r, медіана m_b	17,7	15,8	12,1
6.	Висота h_b, бісектриса w_α	17,7	15,8	12,1
7.	Медіана m_b, висота h_c	11,3	15,9	20,7
8.	Бісектриса w_β, бісектриса w_γ	15,5	18,4	19,2
9.	Радіус описаного кола R, бісектриса w_α	14,2	13,9	17,2
10.	Радіус вписаного кола r, медіана m_c	34,6	40,1	28,4
11.	Висота h_c, бісектриса w_γ	39,0	42,2	34,4
12.	Медіана m_c, висота h_a	68,4	63,2	70,8
13.	Бісектриса w_γ, радіус описаного кола R	28,1	26,7	33,2
14.	Радіуси описаного кола R і вписаного r кіл	41,9	49,8	36,3
15.	Радіус вписаного кола r, висота h_b	17,8	20,5	11,6

ЗАВДАННЯ 3. Проект «Сума».

№	Завдання
1	$\sum_{i=5}^{10}(-1)^i\cdot\frac{x\cdot i!}{y+(i-1)!}$, для $x=1$ та $y=2$
2	$\sum_{i=2}^{8}(-1)^i\cdot\frac{x\cdot(i-1)!}{y+i!}$, для $x=1$ та $y=3$
3	$\sum_{i=3}^{9}(-1)^i\cdot\frac{x^{(i-1)}-y^{(i+1)}}{(i-1!)}$, для $x=2$ та $y=2$
4	$\sum_{i=2}^{6}(-1)^i\cdot\frac{x\cdot(i+1)}{y+(i-1)!}$, для $x=2$ та $y=3$
5	$\sum_{i=2}^{7}(-1)^i\cdot\frac{x\cdot(i+1)}{y+i!}$, для $x=2$ та $y=2$
6	$\sum_{i=3}^{8}(-1)^i\cdot\frac{(x+i)^2+(i-1)!}{(y+i)!}$, для $x=2$ та $y=1$
7	$\sum_{i=2}^{7}(-1)^i\cdot\frac{x^{(i+1)}+y^{(i-1)}}{(i-1)!}$, для $x=1$ та $y=2$
8	$\sum_{i=5}^{9}(-1)^i\cdot\frac{y}{x}\cdot\frac{(i+1)!}{(i-1)!}$, для $x=2$ та $y=3$
9	$\sum_{i=2}^{5}(-1)^i\cdot\frac{x^{(y+i)}+i!}{(i+1)!}$, для $x=2$ та $y=1$
10	$\sum_{i=4}^{8}(-1)^i\cdot\frac{(x+y)\cdot(i-1)!}{(x-y)\cdot(i+1)!}$, для $x=3$ та $y=1$
11	$\sum_{i=2}^{6}(-1)^i\cdot\frac{x}{y+(i-1)!}$, для $x=4$ та $y=1$
12	$\sum_{i=6}^{10}(-1)^i\cdot\frac{(x/y)!}{(i+1)!}$, для $x=2$та $y=4$
13	$\sum_{i=3}^{7}(-1)^i\cdot\frac{(x-y)\cdot(i-1)!}{(x+y)!}$, для $x=3$ та $y=2$
14	$\sum_{i=6}^{11}(-1)^i\cdot\frac{(x\cdot y)^{(i-1)}}{(x+y)}\cdot\frac{i!}{(i+1)!}$, для $x=4$ та $y=2$
15	$\sum_{i=2}^{6}(-1)^i\cdot\frac{y^{(x+i)}}{(i+1)!}$, для $x=1$ та $y=3$
16	$\sum_{i=4}^{9}(-1)^i\cdot\frac{(x+i)!}{(y+i)!+(i-1)!}$, для $x=2$ та $y=1$
17	$\sum_{i=3}^{8}(-1)^i\cdot\frac{(x+y)^2}{(i+1)!}$, для $x=4$ та $y=2$
18	$\sum_{i=7}^{10}(-1)^i\cdot\frac{(x-i)!}{(y+i)!}$, для $x=2$ та $y=1$

Контрольні запитання

1. Що таке подія?
2. На які групи поділяють події у мові **Python**?
3. Як позначається подія клацання лівою кнопкою миші?
4. Яку дію слід виконати в програмі для опрацювання події?
5. Яке призначення має метод **grid()?**
6. Яке призначення має метод **set()?**

Лабораторна робота № 16

ТЕМА: **Python модуль tkinter. Створення меню**

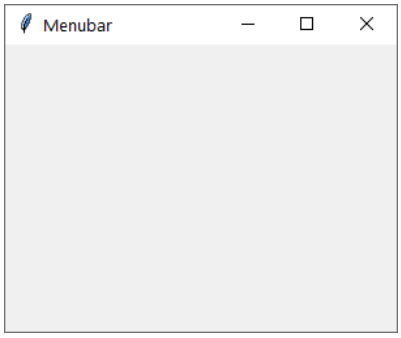
МЕТА: *навчитися створювати меню*

ТЕОРЕТИЧНІ ВІДОМОСТІ

Меню є об'єктом бібліотеки **tkinter**. Для створення меню викликається клас **Menu**.

Наприклад,

```
from tkinter import *
root=Tk()
root.title("Menubar")
root.geometry('300x200+500+200')
mainmenu = Menu(root) # Екземпляр класу Menu
root.config(menu=mainmenu)
root.mainloop()
```

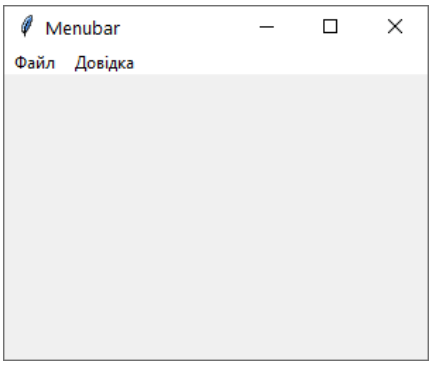


Меню може містити кілька пунктів, наприклад **Файл, Довідка** тощо.

Для додавання підпункту (команди) до пункту меню використовується метод **add_command().** *Наприклад*, щоб розмістити в пункті меню **File** команду **Close**, слід виконати команду:

<змінна посилання на основне меню>.add_command(label=«Файл»)

Наприклад,



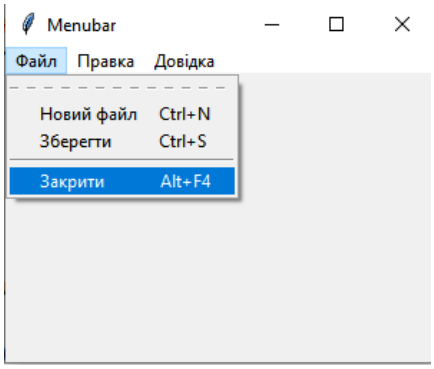

```
from tkinter import *
root=Tk()
root.title("Menubar")
root.geometry('300x200+500+200')
mainmenu = Menu(root) # Екземпляр класу Menu
root.config(menu=mainmenu)
mainmenu.add_command(label='Файл') # метод add_command додає пункти меню
mainmenu.add_command(label='Довідка')
root.mainloop()
```

Кожен пункт меню може містити список, що розкривається, тобто кілька підпунктів (команд).

Для розміщення в головному меню пункту меню використовується метод **add_cascade**. *Наприклад*, щоб розмістити пункт меню **Файл** у головному вікні, слід виконати команду:

<змінна посилання на головне вікно>. add_cascade (label=”Файл”, menu=<змінна посилання на основне меню>)

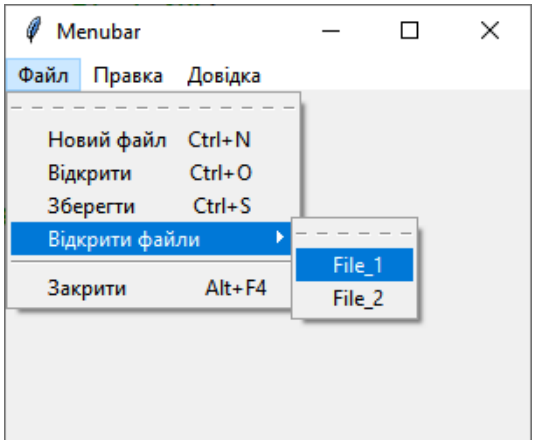
Наприклад,




```
from tkinter import *
root=Tk()
root.title("Menubar")
root.geometry('300x200+500+200')
#створення об'єкта меню Menu у головному вікні
mainmenu=Menu(root)
root.config(menu=mainmenu)
# Файл
mainmenu1=Menu(mainmenu)
mainmenu.add_cascade(label="Файл",menu=mainmenu1)
mainmenu1.add_command(label="Новий файл      Ctrl+N")
mainmenu1.add_command(label="Зберегти           Ctrl+S")
mainmenu1.add_separator()
mainmenu1.add_command(label="Закрити             Alt+F4")
#Правка
mainmenu2=Menu(mainmenu)
mainmenu.add_cascade(label="Правка",menu=mainmenu2)
mainmenu2.add_command(label="Вирізати")
mainmenu2.add_command(label="Копіювати")
#Довідка
mainmenu3=Menu(mainmenu)
mainmenu.add_cascade(label="Довідка",menu=mainmenu3)
mainmenu3.add_command(label="Help")
mainmenu3.add_command(label="Про програму")
root.mainloop()
```

Мова Python дозволяє створювати вкладені меню, тобто підпункт може містити власні підпункти. Для цього створюється ще одне меню, яке за допомогою методу **add_cascade** зв’язується з відповідними підпунктами (командами) меню.

Наприклад,

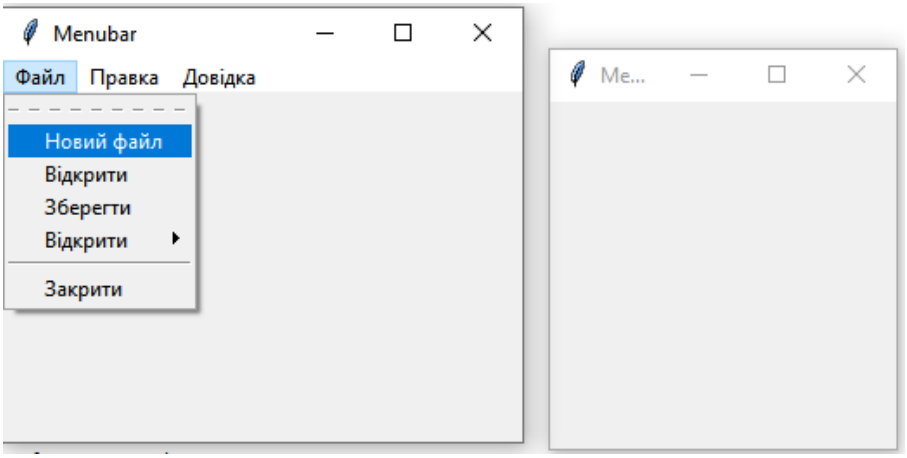



```
from tkinter import *
root=Tk()
root.title("Menubar")
root.geometry('300x200+500+200')
#створення об'єкта меню Menu у головному вікні
mainmenu=Menu(root)
root.config(menu=mainmenu)
# Файл
mainmenu1=Menu(mainmenu)
mainmenu.add_cascade(label="Файл",menu=mainmenu1)
mainmenu1.add_command(label="Новий файл    Ctrl+N")
mainmenu1.add_command(label="Відкрити          Ctrl+O")
mainmenu1.add_command(label="Зберегти          Ctrl+S")
#Файл - Відкрити - ...
mainmenu1_1=Menu(mainmenu1)
mainmenu1.add_cascade(label="Відкрити файли",menu=mainmenu1_1)
mainmenu1_1.add_command(label="File_1") # Створення вкладеного меню
mainmenu1_1.add_command(label="File_2")
mainmenu1.add_separator()
mainmenu1.add_command(label="Закрити              Alt+F4")

#Правка
mainmenu2=Menu(mainmenu)
mainmenu.add_cascade(label="Правка",menu=mainmenu2)
mainmenu2.add_command(label="Вирізати")
mainmenu2.add_command(label="Копіювати")
#Довідка
mainmenu3=Menu(mainmenu)
mainmenu.add_cascade(label="Довідка",menu=mainmenu3)
mainmenu3.add_command(label="Help")
mainmenu3.add_command(label="Про програму")
root.mainloop()
```

Зазвичай підпункти меню зв’язуються з відповідною функцією, яка виконує ту чи іншу дію. Зв’язування функцій із командами меню виконується за

допомогою параметра **command** методу **add_command**. Функція викликається під час клацання лівою кнопкою миші по відповідному підпункту меню.

Наприклад,



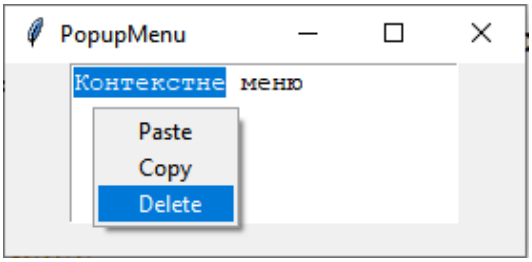

```
from tkinter import *
root=Tk()
#
def new_window():
    window=Toplevel(root)
def close_window():
    root.destroy()
#
root.title("Menubar")
root.geometry('300x200+500+200')
#створення об'єкта меню Menu у головному вікні
mainmenu=Menu(root)
root.config(menu=mainmenu)
# Файл
mainmenu1=Menu(mainmenu)
mainmenu.add_cascade(label="Файл",menu=mainmenu1)
mainmenu1.add_command(label="Новий файл",command=new_window)
mainmenu1.add_command(label="Відкрити")
mainmenu1.add_command(label="Зберегти")
```

У **tkinter** можна створити спливаюче меню (контекстне меню).

Спливаючі меню - це контекстні меню, які з’являються під час взаємодії з користувачем. Це меню може бути показано в будь-якому місці вікна.

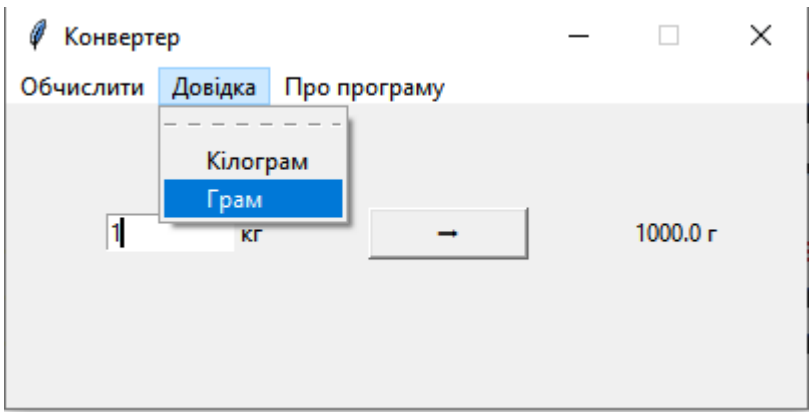
Для цього екземпляр меню зв’язується не через опцію **menu** до батьківського віджету, а до меню застосовується метод **post**, аргументами якого є координати того місця, де має з'являтися меню.

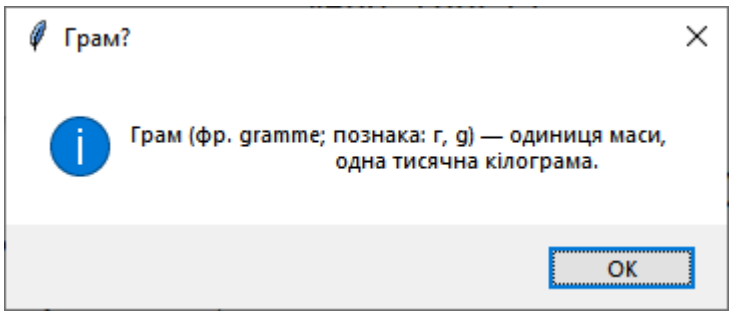
Наприклад,




```
from tkinter import *
def do_popup(event):
    global x,y
    x=event.x
    y=event.y
    popupmenu.post(event.x_root,event.x_root)
x=y=0
def insert_text():
    pass
def Copy_text():
    pass
def delete_text():
    text.delete(1.0, END)
 root = Tk()
root.title("PopupMenu")
# Контекстне меню
popupmenu = Menu(root, tearoff = 0)
popupmenu.add_command(label ="Paste",command=insert_text)
popupmenu.add_command(label ="Copy",command=Copy_text)
popupmenu.add_command(label ="Delete",command=delete_text)
text = Text(width=25, height=5).pack()
#
root.bind("<Button-3>", do_popup)
root.mainloop()
```

Наприклад,





http://unicode.org/charts/PDF/U2700.pdf

З діаграми символів Unicode 7.0 для різних символів і стрілок:

http://unicode.org/charts/PDF/U2B00.pdf

```
from tkinter import *
import tkinter.messagebox
#
root = Tk()
root.geometry("400x150")
root.resizable(width=False, height=False)
root.title("Конвертер")
#
kilogramme=StringVar()
def kilo_to_gramme():
    kilogramme = edit1.get()
    gramme = 1000 * float(kilogramme)
    label2["text"] = f"{round(gramme, 2)} г"
def dov_1():
     tkinter.messagebox.showinfo("Кілограм?", '''Кілогра́м (англ. kilogram, kilogramme) — одиниця
           вимірювання маси в Міжнародній системі одиниць (SI) та деяких інших метричних системах. ''')
def dov_2():
     tkinter.messagebox.showinfo("Грам?", '''Грам (фр. gramme; познака: г, g) — одиниця маси,
                                 одна тисячна кілограма.''')
def dov_3():
     tkinter.messagebox.showinfo("Про програму?", '''Програма переводить кілограми в грами''')
def func1():
    root['bg']='green'
def func2():
    root['bg']='red'

x=y=0
def do_popup(event):
     global x,y
     x=event.x
     y=event.y
     popupmenu.post(event.x_root,event.x_root)
#
mainmenu=Menu(root)
root.config(menu=mainmenu)
#
mainmenu1=Menu(mainmenu) # створення пункту меню Обчислити в основному меню
mainmenu.add_cascade(label="Обчислити",menu=mainmenu1)
mainmenu1.add_command(label="Перевести кг\N{RIGHTWARDS BLACK ARROW}г",command=kilo_to_gramme)
#
mainmenu2=Menu(mainmenu) # створення пункту меню Довідка в основному меню
mainmenu.add_cascade(label="Довідка",menu=mainmenu2)
mainmenu2.add_command(label="Кілограм",command=dov_1)
mainmenu2.add_command(label="Грам",command=dov_2)
#
mainmenu3=Menu(mainmenu) # створення пункту меню Про програму в основному меню
mainmenu.add_cascade(label="Про програму",menu=mainmenu3)
mainmenu3.add_command(label="Про програму",command=dov_3)
#
```

```
popupmenu = Menu(root, tearoff = 0) # контекстне меню
popupmenu.add_command(label ="Зелений",command=func1)
popupmenu.add_command(label ="Червоний",command=func2)
#
frame1 = Frame(master=root)
edit1 = Entry(master=frame1, width=10,textvariable=kilogramme)
label1 = Label(master=frame1, text="кг")
#
edit1.grid(row=0, column=0, sticky="e")
label1.grid(row=0, column=1, sticky="w")
#
button1 = Button(master=root,text="\N{RIGHTWARDS BLACK ARROW}",width=10,command=kilo_to_gramme)
label2 = Label(master=root, text="г")
#
frame1.grid(row=0, column=0, padx=50)
button1.grid(row=0, column=1, pady=50)
label2.grid(row=0, column=2, padx=50)
#
root.bind("<Button-3>", do_popup)
root.mainloop()
```


Хід роботи

ЗАВДАННЯ 1.

Варіанти завдань

1. Знайти добуток елементів масиву, що розміщені на парних позиціях. Виконати сортування елементів масиву за зростанням, використовуючи метод екстремальних елементів.

2. Знайти кількість різних чисел даного одновимірного масиву. Виконати сортування елементів масиву за спаданням, використовуючи метод обміну.

3. Замінити всі додатні елементи у даному одновимірному масиві їх номерами. Виконати сортування елементів масиву за спаданням, використовуючи метод екстремальних елементів.

4. Замінити всі від'ємні елементи у даному одновимірному масиві їх номерами. Виконати сортування елементів масиву за зростанням, використовуючи метод екстремальних елементів.

5. Піднести до квадрату всі елементи даного одновимірного масиву кратні 5. Виконати сортування елементів масиву за зростанням, використовуючи метод обміну.

6.Збільшити вдвічі всі елементи даного одновимірного масиву кратні 4. Виконати сортування елементів масиву за спаданням, використовуючи метод обміну.

7.Підрахувати кількість від'ємних елементів у даному одновимірному масиві. Виконати сортування елементів масиву за зростанням, використовуючи метод обміну.

8.Підрахувати кількість додатних елементів у даному одновимірному масиві. Виконати сортування елементів масиву за спаданням, використовуючи метод екстремальних елементів.

9.Підрахувати суму від'ємних елементів у даному одновимірному масиві. Виконати сортування елементів масиву за зростанням, використовуючи метод екстремальних елементів.

10.Піднести до кубу всі елементи даного одновимірного масиву кратні 5. Виконати сортування елементів масиву за спаданням, використовуючи метод обміну.

11.Знайти довжину заданого вектора ($|a| = \sqrt{a_1^2 + a_2^2 + ... + a_n^2}$). Виконати сортування елементів масиву (вектора) за зростанням, використовуючи метод обміну.

12.Знайти скалярний добуток двох векторів (обов'язково мають однакову розмірність). Виконати сортування елементів масивів за спаданням, використовуючи метод екстремальних елементів.

13.Підрахувати суму додатних парних елементів у даному одновимірному масиві. Виконати сортування елементів масиву за зростанням, використовуючи метод обміну.

14.Підрахувати середнє арифметичне всіх елементів масиву кратних 3 у даному одновимірному масиві.

15.Підрахувати середнє арифметичне всіх додатних елементів у даному одновимірному масиві. Виконати сортування елементів масиву за зростанням, використовуючи метод екстремальних елементів.

16. Підрахувати середнє арифметичне всіх від'ємних елементів у даному одновимірному масиві. Виконати сортування елементів масиву за спаданням, використовуючи метод обміну.
17. Знайти найбільший елемент одновимірного масиву. Виконати сортування елементів масиву за зростанням, використовуючи метод обміну.
18. Знайти найменший елемент одновимірного масиву. Виконати сортування елементів масиву за спаданням, використовуючи метод екстремальних елементів.
19. Знайти різницю між найбільшим і найменшим елементами одновимірного масиву. Виконати сортування елементів масиву за зростанням, використовуючи метод екстремальних елементів.
20. Знайти суму квадратів елементів одновимірного масиву. Виконати сортування елементів масиву за спаданням, використовуючи метод обміну.

 **Контрольні запитання** 

1. Який метод використовується для створення меню?
2. Для чого призначений метод **add_command**()?
3. Поясніть порядок створення вкладеного меню.
4. Як зв'язуються підпункти меню з відповідними функціями?

Лабораторна робота № 17

ТЕМА: **Python модуль tkinter. Діалогові вікна.**

МЕТА: *розглянути призначення діалогових вікон, навчитися використовувати діалогові вікна під час написання програм*

ТЕОРЕТИЧНІ ВІДОМОСТІ

Діалогові вікна призначені для виведення повідомлень користувачу й отримання від нього відповідей та для керування об'єктами.

Існують різні типи діалогових вікон.

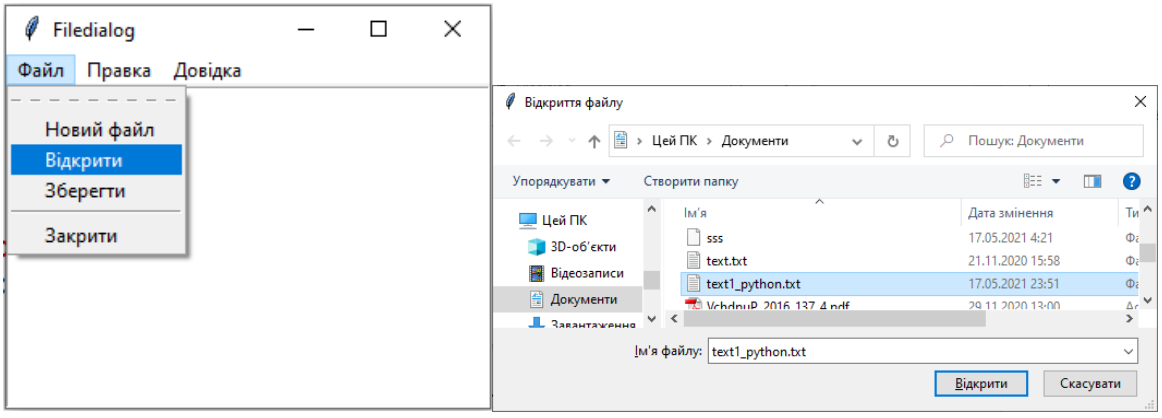
```
from tkinter import filedialog

filedialog.asksaveasfilename()
filedialog.asksaveasfile()
filedialog.askopenfilename()
filedialog.askopenfile()
filedialog.askdirectory()
filedialog.askopenfilenames()
filedialog.askopenfiles()
```

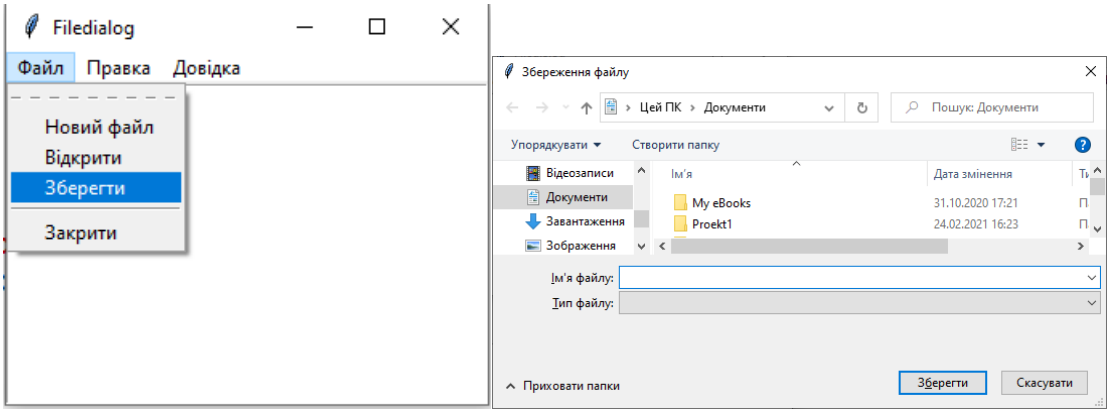
Розглянемо дві функції з модуля **filedialog** – **askopenfilename ()** і **asksaveasfilename ()**.

Для їх створення слід, окрім модуля **tkinter**, імпортувати модуль **tkinter.filedialog**.

```
from tkinter import *
from tkinter.filedialog import *
```

Вікно для *відкриття файлів* створюють за допомогою – **askopenfilename**(), а для *збереження файлів* за допомогою – **asksaveasfilename**().






```
from tkinter import *
from tkinter.filedialog import *
import fileinput

root = Tk()
root.title("Filedialog")
root.geometry('300x200+500+200')

# Функція відкриття файлу
def Open_file():
    open_file=askopenfilename()  # Вікно відкриття файлу
    for i in fileinput.input(open_file):
        text.insert(END,i)

# Функція збереження файлу
def Save_file():
    save_file=asksaveasfilename() # Вікно збереження файлу
    readtext=text.get(1.0,END)
    open_writeF=open(save_file,"w")
    open_writeF.write(readtext)
    open_writeF.close()
```

Об’єкт **open_file** відкриває вікно для відкриття файлів, а **save_file** — вікно для збереження файлів.

```
import fileinput
```

Метод **input** модуля **fileinput** може приймати в якості аргумента адресу файла, читати його вміст, формуючи список рядків.

```
import tkinter.font as tkFont
```

fontExample = tkFont.Font(family="Arial", size=16, weight="bold", slant="italic")

Параметри **Font**:

- **family** - сімейство шрифтів,наприклад Arial, Courier

- **size** - размір шрифту (в пунктах)
- **weight** - товщина, normal або bold
- **slant** – спосіб накреслення шрифту: italic
- **underline** - підкреслення шрифту, False або True

```
from tkinter.colorchooser import askcolor
```

Хід роботи

ЗАВДАННЯ 1.

1.Імпортуй графічну бібліотеку **tkinter**.

```
from tkinter import *
```

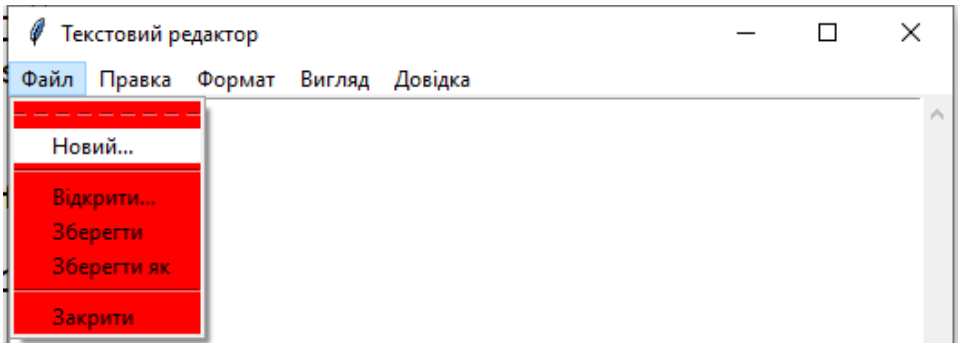
2.Створи головне вікно.

```
root = Tk()
root.title("Текстовий редактор")
root.minsize(width=500, height=400)
#


#
root.mainloop()
```

3.Створіть головне меню

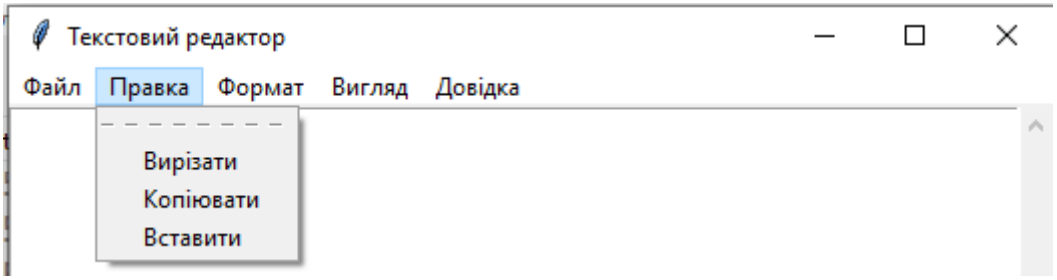
```
#Меню
mainmenu = Menu(root)
root.config(menu=mainmenu)
```

4.Додайте пункт меню **Файл**



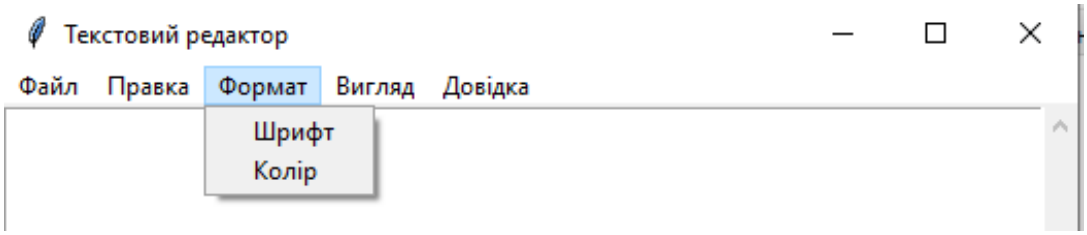
```
#Файл
mainmenu1 = Menu(mainmenu, background='red',
                 foreground='black',
                 activebackground='white',
                 activeforeground='black')
mainmenu.add_cascade(label="Файл", menu=mainmenu1)
mainmenu1.add_command(label="Новий...")
mainmenu1.add_separator()
mainmenu1.add_command(label="Відкрити...")
mainmenu1.add_command(label="Зберегти")
mainmenu1.add_command(label="Зберегти як")
mainmenu1.add_separator()
mainmenu1.add_command(label="Закрити")
```

5.Додайте пункт меню **Правка**



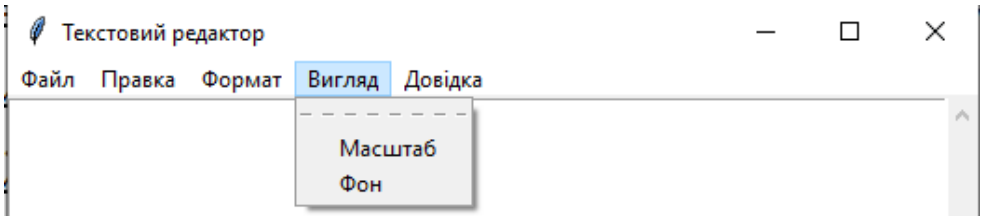

```
#Правка
mainmenu2=Menu(mainmenu)
mainmenu.add_cascade(label="Правка",menu=mainmenu2)
mainmenu2.add_command(label="Вирізати")
mainmenu2.add_command(label="Копіювати")
mainmenu2.add_command(label="Вставити")
```

6.Додайте пункт меню **Формат**



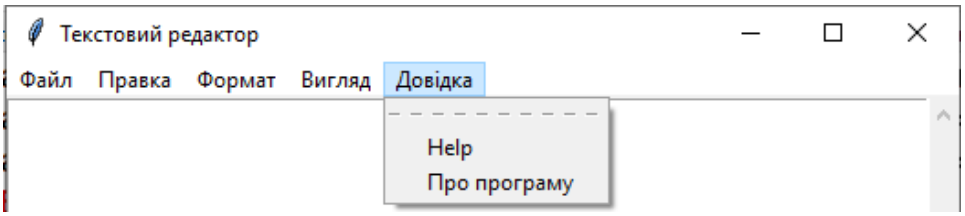

```
#Формат
check= BooleanVar()
check.set(True)
check= BooleanVar()
check.set(False)
mainmenu3=Menu(mainmenu,tearoff=0)
mainmenu.add_cascade(label="Формат",menu=mainmenu3)
mainmenu3.add_checkbutton(label="Шрифт", onvalue=1, offvalue=0, variable=check)
mainmenu3.add_checkbutton(label='Колір', onvalue=1, offvalue=0, variable=check)
```

7.Додайте пункт меню **Вигляд**




```
#Вигляд
mainmenu4=Menu(mainmenu)
mainmenu.add_cascade(label="Вигляд",menu=mainmenu4)
mainmenu4.add_command(label="Масштаб")
mainmenu4.add_command(label="Фон")
```

8.Додайте пункт меню **Довідка**




```
mainmenu5=Menu(mainmenu)
mainmenu.add_cascade(label="Довідка",menu=mainmenu5)
mainmenu5.add_command(label="Help")
mainmenu5.add_command(label="Про програму")
```

9.Створити багаторядкове текстове поле **Text,** з вертикальною смугою прокручування.

```
# Багаторядкове текстове поле
text = Text(root, width=50, height=20, font=" Courier 12")
scroll = Scrollbar(root)
scroll.pack(side=RIGHT, fill=Y)
text.pack(side=LEFT, fill=Y)
scroll.config(command=text.yview)
text.config(yscrollcommand=scroll.set)

text.pack()
```

10. Імпортуйте модуль для роботи з системними діалоговими вікнами

```
from tkinter.filedialog import *
```

11. Та імпортуйте додаткові модулі для виклику вікна повідомлення, кольору, шрифту, модуль для читання вмісту файла

```
from tkinter.filedialog import *
from tkinter.messagebox import *
import fileinput
from tkinter.colorchooser import askcolor
import tkinter.font as tkFont
```

12. Створіть функцію **new_file().**

```
FILE_NAME = NONE

def new_file():
    global FILE_NAME
    FILE_NAME = "Untitled"
    text.delete('1.0', END)
```

13.Зв’яжіть створену функцію через параметр **command** пункту меню «Файл – Новий».

```
mainmenu1.add_command(label="Новий...", command=new_file)
```

14. Створіть функцію **open_file()** для виклику діалогового вікна для відкриття файлу

```
def open_file():
    global FILE_NAME
    inp = askopenfile(mode="r")
    if inp is None:
        return
    FILE_NAME = inp.name
    data = inp.read()
    text.delete('1.0', END)
    text.insert('1.0', data)
```

15. Зв’яжіть створену функцію через параметр **command** підпункта «Відкрити», який відкриватиме файл

```
mainmenu1.add_command(label="Відкрити...", command=open_file)
```

16. Створіть функцію **save_file()** для виклику діалогового вікна для збереження файлу

```
def save_file():
    data = text.get('1.0', END)
    out = open(FILE_NAME, 'w')
    out.write(data)
    out.close()
```

17.Зв’яжіть створену функцію через параметр **command** підпункта «Зберегти»

```
mainmenu1.add_command(label="Зберегти", command=save_file)
```

18.Створіть функцію **save_as().** Зв’яжіть створену функцію через параметр **command** пункту меню «**Зберегти як**»

```
def save_as():
    out = asksaveasfile(mode='w', defaultextension='txt')
    data = text.get('1.0',END)
    try:
        out.write(data.rstrip())
    except Exception:
        showerror(title="Error", message="Помилка збереження файлу")
```

19.Створіть функцію **close_win().** Зв’яжіть створену функцію через параметр **command** пункту меню «Закрити»

```
def close_win():
  if askyesno("Закрити", "Закрити вікно?"):
    root.destroy()
```

20. Створити функцію **font().** Зв’яжіть створену функцію через параметр **command** пункту меню «Формат – Шрифт».

```
def font():
    fontExample = tkFont.Font(family="Arial", size=14, weight="bold", slant="italic")
    text.configure(font=fontExample)
```

21.Створити функцію **color().** Зв’яжіть створену функцію через параметр **command** пункту меню «Формат – Колір».

```
def color():
    result = askcolor()
    text.configure(fg = result[1])
    print(result[1])
```

22. Створити функцію **fon().** Зв’яжіть створену функцію через параметр **command** пункту меню «Вигляд – Фон».

```
def fon():
    result = askcolor()
    text.configure(bg = result[1])
    print(result[1])
```

23. Створити функцію **size_win(),** для зміни розміру вікна**.** Зв’яжіть створену функцію через параметр **command** пункту меню «Вигляд – Масштаб».

24. Створити функцію **about().** Зв’яжіть створену функцію через параметр **command** пункту меню «Довідка – Про програму». Використайте **showinfo**

25. Створити функцію **help().** Зв’яжіть створену функцію через параметр **command** пункту меню «Довідка – Help». Використайте **showinfo,** або **Toplevel.**

26. Створити контекстне меню, для віджета **Text**. Контекстне меню включає наступні пункти: *Фон, Масштаб, Шрифт, Колір тексту.*

 **Контрольні запитання** 

1. Для чого призначені діалогові вікна?

2. За допомогою якого методу створюється вікно відкриття вікна файлів?

3. Для чого призначено метод **askcaveasfilename**?

4. Поясніть порядок завантаження файлів у текстове поле.

5. Поясніть порядок створення діалогових вікон із повідомленями?

Лабораторна робота № 18

ТЕМА: Python модуль tkinter. Графічні примітиви. Робота зі звуком. Анімація.

МЕТА: *навчитися використовувати модуль для відтворення звуку, модуль time для створення покадрової анімації, навчитися створювати прості графічні примітиви*

ТЕОРЕТИЧНІ ВІДОМОСТІ

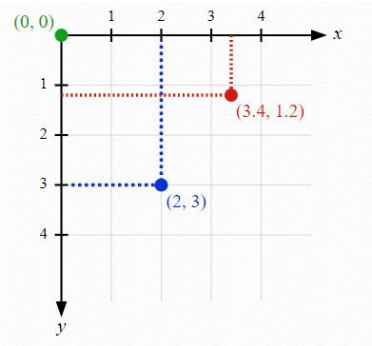
Полотно для малювання Canvas

У **tkinter** зображення створюється в межах полотна — об'єкта класу **Canvas**, який входить до модуля **tkinter**.

Об'єкт Canvas (полотно) — це віджет бібліотеки **tkinter**, на якому можуть бути розміщені інші віджети: геометричні фігури, малюнки, фото, графічні зображення тощо.

Canvas має систему координат, початок яких розташовується в лівому верхньому куті полотна. Першою завжди вказується координата x, а другою — координата y.

Для задавання положення точок на полотні використовують координати. Будь-яка точка може бути задана парою чисел (X, Y), де X — відстань від точки до лівого краю полотна, Y — відстань від точки до верхнього краю полотна.



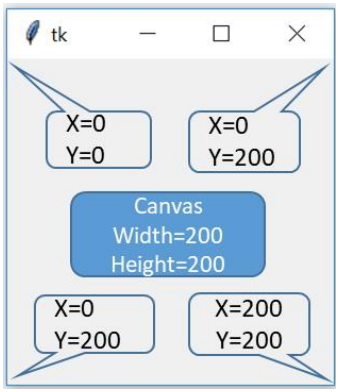
Перш ніж записувати методи для малювання, потрібно створити полотно.

Синтаксис створення об'єкта класу **Canvas**:

змінна = **Canvas**(батьківський віджет, width = значення, height = значення)

де width — ширина полотна; height — його висота (в пікселях).

Наприклад, створити полотно розміром 200 х 200 пікселів:




```
from tkinter import*
root=Tk()
canvas=Canvas(root,width=200,height=200)
canvas.pack()
root.mainloop()
```

Колір полотна за замовчуванням світло-сірий, змінити фон можна за допомогою методу **config**.

```
canvas.config(bg='#49F477')
```

Методи полотна Canvas модуля tkinter

Геометричні фігури, що розміщуються на полотні, називають ***графічними примітивами***.

Для кожного примітива існує відповідний метод створення.

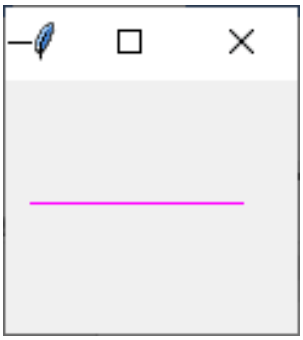
Методи створення графічних примітивів

Метод створення ліній — create_line()

Має таку загальну структуру:

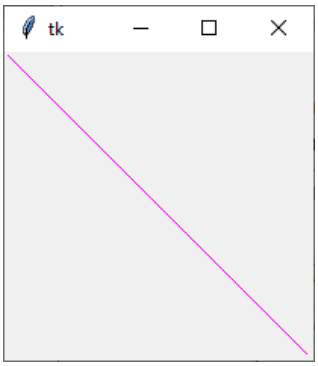
<змінна >.create_line (кординати початку і кінця лінії [,інші параметри])

Приклад 1. Намалювати лінію.



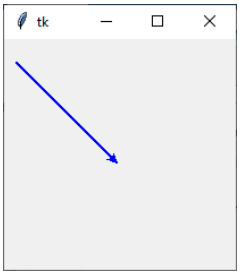
```
from tkinter import *
root=Tk()
canvas= Canvas(root,width=100,height=100)
canvas.pack()
x1=0
y1=50
x2=90
y2=50
canvas.create_line(x1,y1, x2,y2, fill="#ff00ff")
canvas.mainloop()
```

Приклад 2. Намалювати лінію від верхнього лівого кута полотна до правого нижнього кута.




```
from tkinter import *
root=Tk()
canvas= Canvas(root,width=200,height=200)
canvas.pack()
x1=0
y1=0
x2=200
y2=200
canvas.create_line(x1,y1, x2,y2, fill="#ff00ff")
canvas.mainloop()
```

Приклад 3.



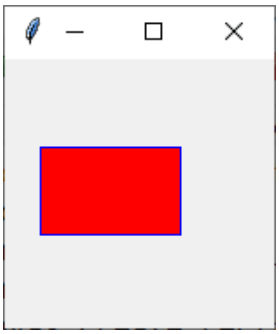

```
from tkinter import *
root=Tk()
canvas= Canvas(root,width=200,height=200)
canvas.pack()
canvas.create_line(10,20, 100,110,width=2,fill="blue",arrow=LAST)
canvas.mainloop()
```

Метод створення прямокутника — create_rectangle()

Загальна структура інструкції створення прямокутника така:

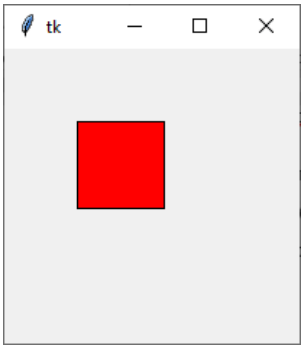
<змінна >.create_rectangle (координати верхнього лівого кута, координати правого нижнього кута [,інші параметри])

Приклад 4.



```
from tkinter import *
root=Tk()
canvas=Canvas(root,width=150,height=150)
canvas.pack()
canvas.create_rectangle(20,50,100,100,fill='red',outline="blue")
root.mainloop()
```

Приклад 5.

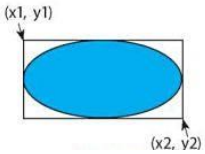



```
from tkinter import *
root=Tk()
canvas=Canvas(root,width=200,height=200)
canvas.pack()
canvas.create_rectangle(50,50,110,110,fill='red')
root.mainloop()
```

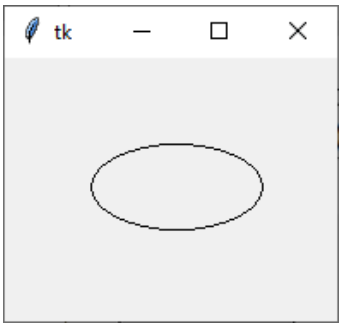
Метод створення еліпса — create_oval()

Загальна структура інструкції створення еліпса така:

<змінна >.create_oval (*координати верхнього лівого кута й координати правого нижнього кута неіснуючого прямокутника, у який буде вписано еліпс [,інші параметри]***)**

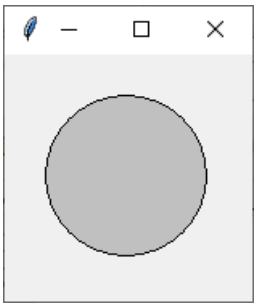


Приклад 6.




```
from tkinter import *
root = Tk()
canvas= Canvas(root,width=190,height=150)
canvas.pack()
canvas.create_oval(50,50,150,100)
root.mainloop()
```

Приклад 7.



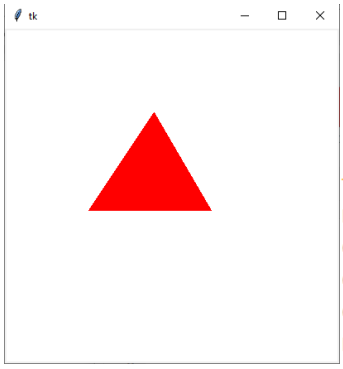
```
from tkinter import *
root=Tk()
canvas=Canvas(root,width=150,height=150)
canvas.pack()
canvas.create_oval(25,25,125,125,fill='#c0c0c0')
root.mainloop()
```

Метод створення багатокутника — create_polygon()

Структура інструкції створення багатокутника така:

<змінна >.create_polygon(координати вершин [,інші параметри])

Приклад 8.



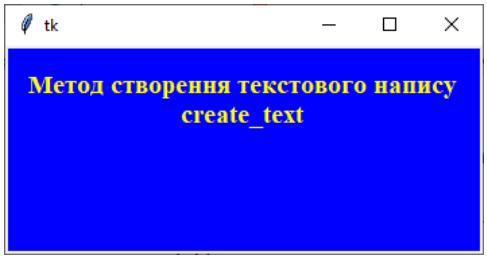
```
from tkinter import *
root = Tk()
canvas= Canvas(root,width=400,height=400,bg='white')
canvas.pack()
canvas.create_polygon([180,100],[100,220],[250,220],fill='red')
root.mainloop()
```

Метод створення текстового напису — create_text()

Інструкція створення має таку загальну структуру:

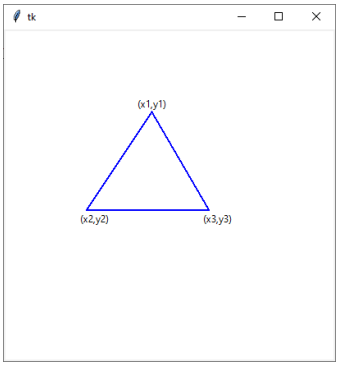
<змінна >.create_text(координати напису, text=«напис»)

Приклад 9.




```
from tkinter import *
root = Tk()
canvas= Canvas(root,width=350,height=150,bg="blue")
canvas.pack()
canvas.create_text(175,40,fill='yellow',
                   justify=CENTER,
                   font='Times 14 bold',
                   text='Метод створення текстового напису\ncreate_text')
root.mainloop()
```

Приклад 10.



```
from tkinter import *
root = Tk()
canvas= Canvas(root,width=400,height=400,bg='white')
canvas.pack()
canvas.create_polygon([180,100],[100,220],[250,220],fill='white',outline='blue',width=2)
canvas.create_text(180,90,text='(x1,y1)')
canvas.create_text(110,230,text='(x2,y2)')
canvas.create_text(260,230,text='(x3,y3)')
root.mainloop()
```

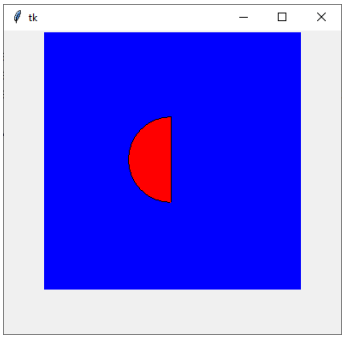
Метод створення дуги, сектора, сегмента — create_arc()

Значення **CHORD** параметра **style** у методі означає, що буде створено сегмент, значення **ARC** — дуга, а за замовчуванням цього параметра створюється сектор. За допомогою параметрів **start** і **extent** можна задати кут фігури.

Приклад 11.

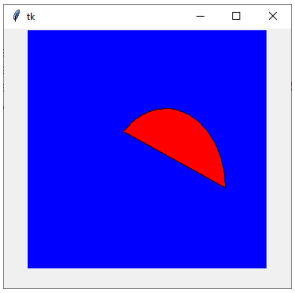
```
from tkinter import *
root = Tk()
canvas= Canvas(root,width=150,height=150)
canvas.pack()
canvas.create_arc(10,10,130,130,start=90,extent=300,fill='blue')
root.mainloop()
```

Приклад 12.



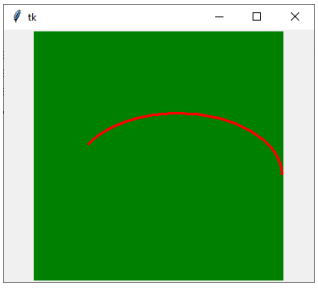
```
from tkinter import *
root = Tk()
canvas= Canvas(root,width=300,height=300,bg="blue")
canvas.pack()
canvas.create_arc(100,200,200,100,start=90,extent=180,style=CHORD,fill='red')
root.mainloop()
```

Приклад 13.

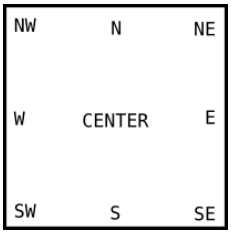


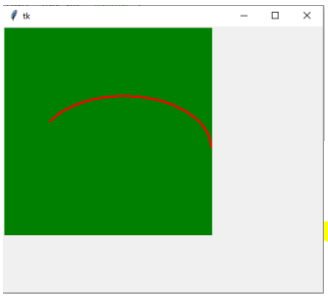
```
from tkinter import *
root = Tk()
canvas= Canvas(root,width=300,height=300,bg="blue")
canvas.pack()
canvas.create_arc(100,100,250,300,start=0,extent=135,style=CHORD,fill='red')
root.mainloop()
```

Приклад 14



```
from tkinter import *
root = Tk()
canvas= Canvas(root,width=300,height=300,bg="green")
canvas.pack()
canvas.create_arc(50,100,300,250,start=0,extent=150,style=ARC,width=3,outline='red')
root.mainloop()
```





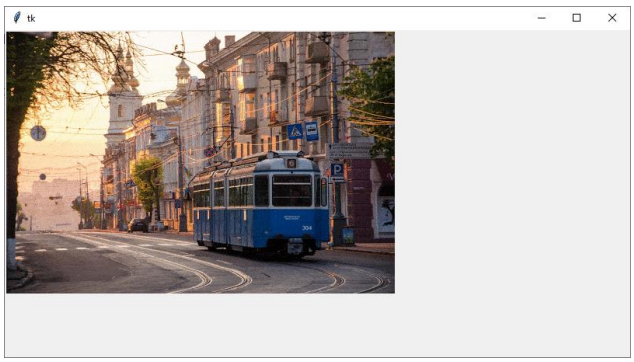
```
from tkinter import *
root = Tk()
canvas= Canvas(root,width=300,height=300,bg="green")
canvas.pack(anchor=NW)
canvas.create_arc(50,100,300,250,start=0,extent=150,style=ARC,width=3,outline='red')
root.mainloop()
```

Виведення зображень із графічних файлів. Метод create_image

Засобами **tkinter** можна вивести на полотно зображення з графічного файла. Для цього потрібно ім’я графічного файла завантажити до змінної за допомогою функції **PhotoImage** (file='ім’я файла') і викликати метод **create_image**.

Приклад 15.

Вивести на полотно рисунок **vin.gif**, який міститься в папці з кодом програми:

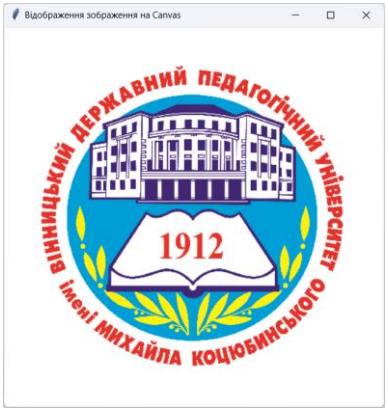


```
from tkinter import *
root = Tk()
canvas= Canvas(root,width=500,height=500)
photo = PhotoImage(file="vin.gif")
canvas.create_image(0,0,anchor=NW,image=photo)
canvas.pack(anchor=NW)
root.mainloop()
```

Параметр **anchor** визначає розташування рисунка на полотні. Значення **CENTER** вказує, що центр зображення буде розташований у точці з координатами (250, 250) на Canvas.

Приклад 16.

```
import tkinter as tk

root = tk.Tk()
root.title("Відображення зображення на Canvas")

# Створення Canvas
canvas = tk.Canvas(root, width=500, height=500, bg="white")
canvas.pack()

# Завантаження зображення у форматі PNG
image_path = "logo.png"
photo = tk.PhotoImage(file=image_path)

# Додавання зображення до Canvas
canvas.create_image(250, 250, image=photo, anchor=tk.CENTER)

root.mainloop()
```



Методи move() і itemconfig()

Методи **move**() і **itemconfig**() допомагають у процесі виконання програми змінювати властивості об’єктів, розташованих на полотні, переміщувати їх на полотні й виконувати деякі інші операції. Щоб застосувати ці методи, попередньо об’єктам необхідно надати імена (ідентифікатори). Присвоїти об’єкту ідентифікатор можна за допомогою такої інструкції:

<ідентифікатор>=<змінна полотна>.create_<назва об’єкта>(параметри)

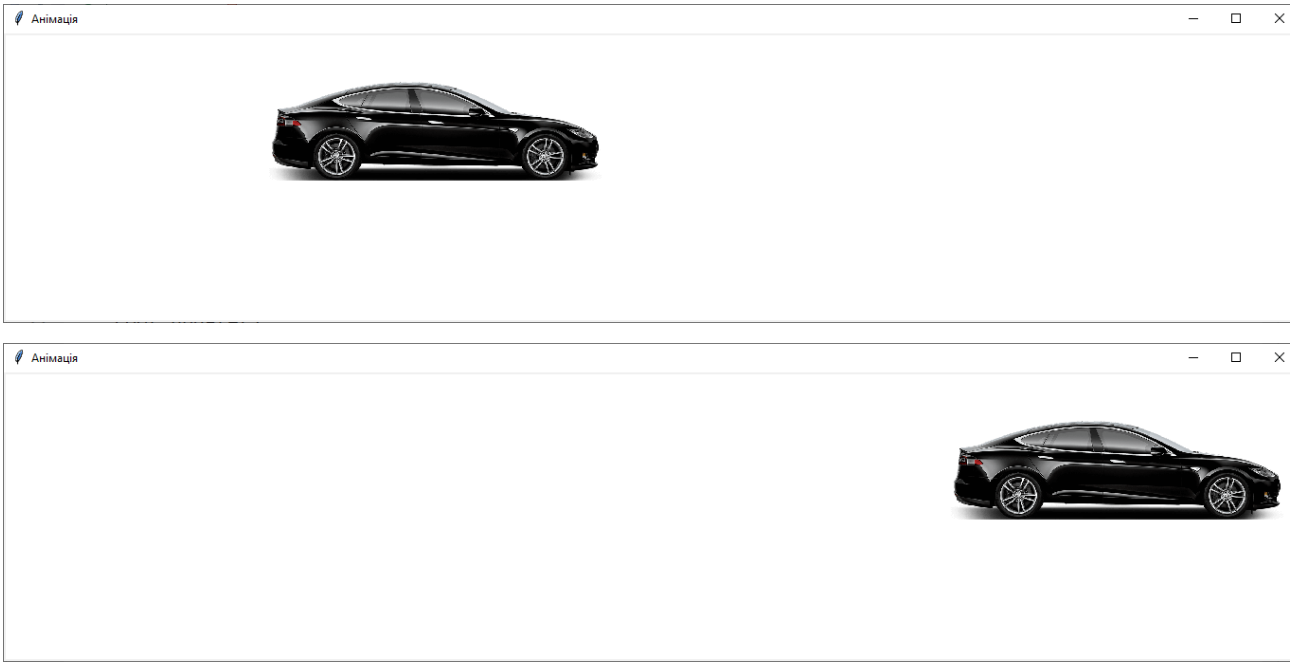
Метод move() – використовується для переміщення об’єктів по осях *x* і *y* на вказану кількість пікселів.

<змінна полотна>.move(< ідентифікатор об’єкта >, x, y)

Метод itemconfig(), змінює перелічені в методі властивості.

Метод delete() видаляє з полотна об’єкт з вказаним ідентифікатором або тегом. Якщо в методі вказати значення **all**, із полотна буде видалено всі об’єкти.

Приклад 17.

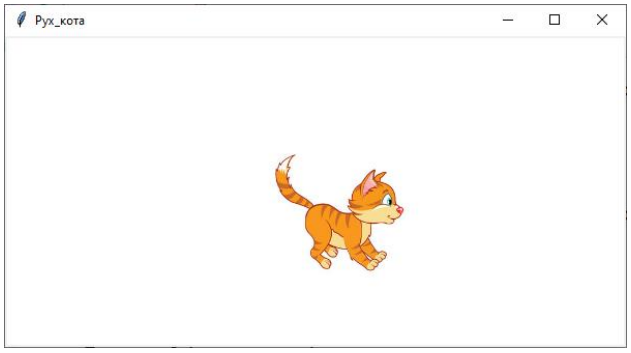



```
from tkinter import *
import time
root = Tk()
root.title("Анімація")
canvas = Canvas(root,height=300,width=1500,bg='white')
canvas.pack()
img = PhotoImage(file='tesla_car.png')
move1=canvas.create_image(10,10,image=img,anchor=NW)
canvas.image = img
for i in range(1,100):
    canvas.move(move1,10,0)
    root.update()
    time.sleep(0.01)
root.mainloop()
```

Приклад 18.

```
from tkinter import *
import time

root = Tk()
root.geometry("500x500")
root.title("Рух")
canvas = Canvas(root, width=500,height=500,bg='white')
canvas.pack()

ball1 = canvas.create_oval(5,5,60,60, fill='red')
for i in range(60): # рух вперед
    canvas.move(1,5,0)
    root.update()
    time.sleep(0.05)
for i in range(60): # рух назад
    canvas.move(1,-5, 0)
    root.update()
    time.sleep(0.05)
root.mainloop()
```

Приклад 19.




```
from tkinter import *
import time
def kadr1(x,y):
    img = PhotoImage(file='cat1.png')
    canvas.create_image(x+10,y+10,image=img,anchor=NW)
    canvas.image = img
    root.update()
def kadr2(x,y):
    img = PhotoImage(file='cat2.png')
    canvas.create_image(x+10,y+10,image=img,anchor=NW)
    canvas.image = img
    root.update()
def kadr3(x,y):
    img = PhotoImage(file='cat3.png')
    canvas.create_image(x+10,y+10,image=img,anchor=NW)
    canvas.image = img
    root.update()

root = Tk()
root.geometry("600x300")
root.title("Рух_кота")
canvas = Canvas(root,height=500,width=1000,bg='white')
canvas.pack()

for x in range(0,400,50):
    canvas.delete("all")
    kadr1(x,100)
    time.sleep(0.1)
    canvas.delete("all")
    kadr2(x+5,100)
    time.sleep(0.1)
    canvas.delete("all")
    kadr3(x+5,100)
    time.sleep(0.1)

root.mainloop()
```

Робота зі звуком

Для роботи з звуком в **tkinter** використовують вбудовану бібліотеку **winsound.** https://docs.python.org/2/library/winsound.html

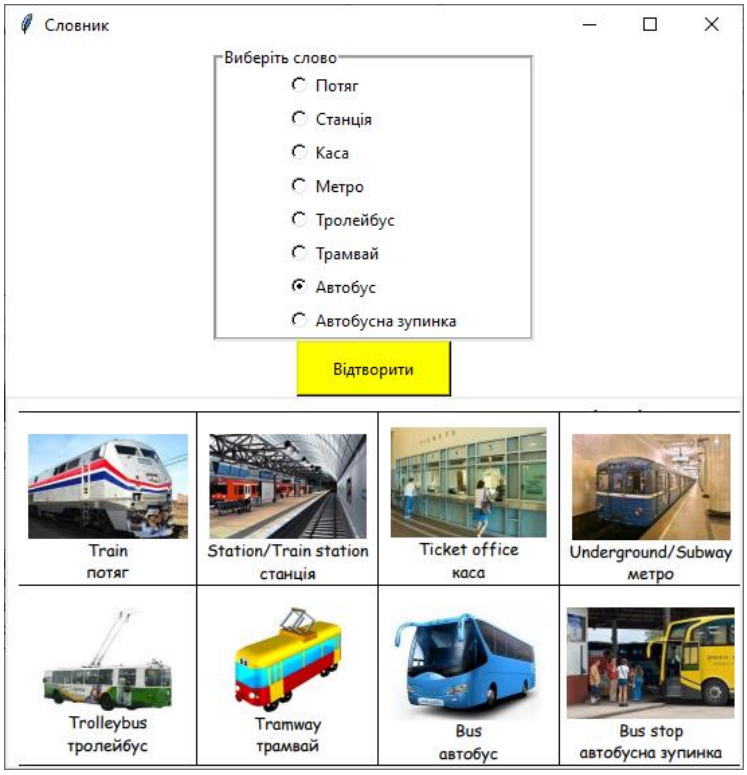
```
import winsound
```

За допомогою функції **PlaySound** можна даного модуля, можна відтворити звукові файли в форматі **.wav**. В функцію передаються два параметри: <name_file> і <flag> — **winsound.SND_FILENAME**, який потрібно **API**-інтерфейсу платформи для звернення до вихідного файлу.

```
winsound.PlaySound('1.wav', winsound.SND_FILENAME)
```

flag	Призначення
SND_FILENAME	Звернення до файлу формата .wav
SND_LOOP	Відтворення звуку декілька разів
SND_ASYNC	Асинхронне відтворення звуку
SND_NOSTOP	Заборона на переривання звуку

Приклад 20.



```
from tkinter import*
import winsound
def play():
    if var.get() == 0:
        winsound.PlaySound('1.wav', winsound.SND_FILENAME)
    elif var.get() == 1:
        winsound.PlaySound('2.wav', winsound.SND_FILENAME)
    elif var.get() == 2:
        winsound.PlaySound('3.wav', winsound.SND_FILENAME)
    elif var.get() == 3:
        winsound.PlaySound('4.wav', winsound.SND_FILENAME)
    elif var.get() == 4:
        winsound.PlaySound('5.wav', winsound.SND_FILENAME)
    elif var.get() == 5:
        winsound.PlaySound('6.wav', winsound.SND_FILENAME)
    elif var.get() == 6:
        winsound.PlaySound('7.wav', winsound.SND_FILENAME)
    elif var.get() == 7:
        winsound.PlaySound('8.wav', winsound.SND_FILENAME)
root=Tk()

root.minsize(width=500, height=400)
root.title("Словник")
root['bg']='white'
frame = LabelFrame(root, text='Виберіть слово', padx=50, bg="white",bd=3)
frame.pack()
var = IntVar()
var.set(0)
Radiobutton1 = Radiobutton(frame,bg="white",text="Потяг",
                   variable=var, value=0).pack(anchor=W)
Radiobutton2 = Radiobutton(frame,bg="white",text="Станція",
                     variable=var, value=1).pack(anchor=W)
Radiobutton3 = Radiobutton(frame,bg="white",text="Каса",
                    variable=var, value=2).pack(anchor=W)
Radiobutton4 = Radiobutton(frame,bg="white",text="Метро",
                     variable=var, value=3).pack(anchor=W)
Radiobutton5 = Radiobutton(frame,bg="white",text="Тролейбус",
                    variable=var, value=4).pack(anchor=W)

Radiobutton6 = Radiobutton(frame,bg="white",text="Трамвай",
                   variable=var, value=5).pack(anchor=W)
Radiobutton7 = Radiobutton(frame,bg="white",text="Автобус",
                  variable=var, value=6).pack( anchor=W)
Radiobutton8 = Radiobutton(frame,bg="white",text="Автобусна зупинка",
                   variable=var, value=7).pack(anchor=W)

button = Button(text="Відтворити",width=15, height=2,bg="yellow",command=play).pack()

canvas = Canvas(root, width=545, height=273, bg='white')
canvas.pack()

img = PhotoImage(file='1.png')
canvas.create_image(10,10,image=img,anchor=NW)
canvas.image = img

root.mainloop()
```

Хід роботи

ЗАВДАННЯ 1. Створіть навчальний проект «Вивчаємо англійську мову» з відтворенням звуку у форматі **.wav**

Використайте ілюстровані словники (виберіть тему за бажанням):

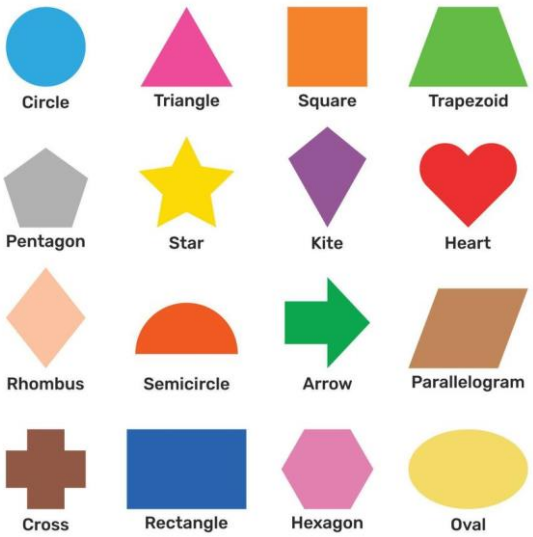
http://www.enrucafe.com/uk/freelessons/vocabularyfreelessons.html

ЗАВДАННЯ 2. Створити анімацію переміщається зображення (.png) зліва направо (переміщення машини та ін.).

Наприклад, https://replit.com/join/gcawlchlnc-galinakovtoniu1

ЗАВДАННЯ 3.

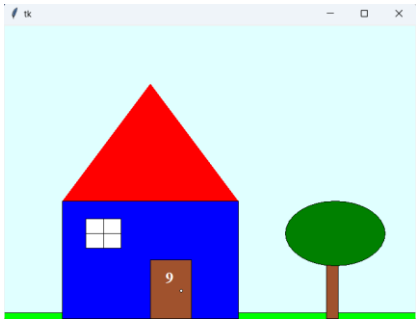
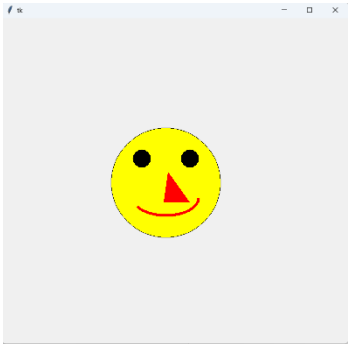
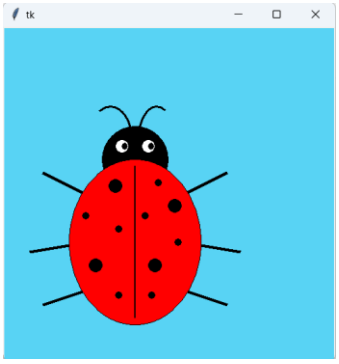
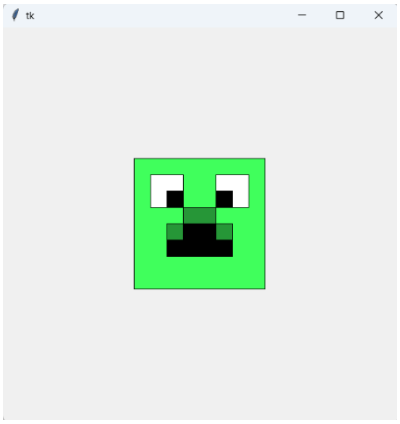
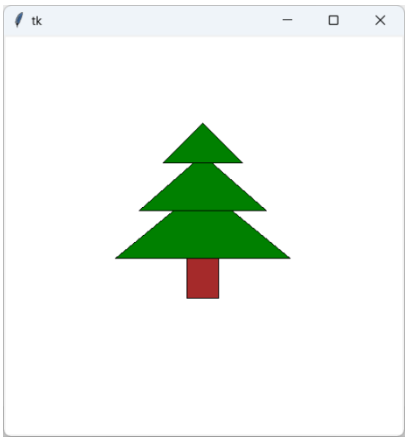
3.1. Використовуючи методи **tkinter** намалюйте дві фігури.



Координатна площина

https://drive.google.com/file/d/1vgWBVoaHNIjTHe0ZpFlxH9sPMuVhDaZW/view?usp=share_link

3.2 Створити власне зображення, *наприклад*

https://drive.google.com/drive/folders/1tUhPSzDQK-GbGoX32wUyYpUXUYyEmX2Z?usp=share_link

Контрольні запитання

1. Для чого призначений об’єкт **Canvas**? За допомогою якої команди створюється полотно в головному вікні?
2. Які параметри можуть розміщуватися в об’єкті **Canvas**?
3. Для чого призначений метод **delete**()?
4. Яке призначення методу **create_ rectangle()**?
5. Що називають тегами, яке їх призначення?

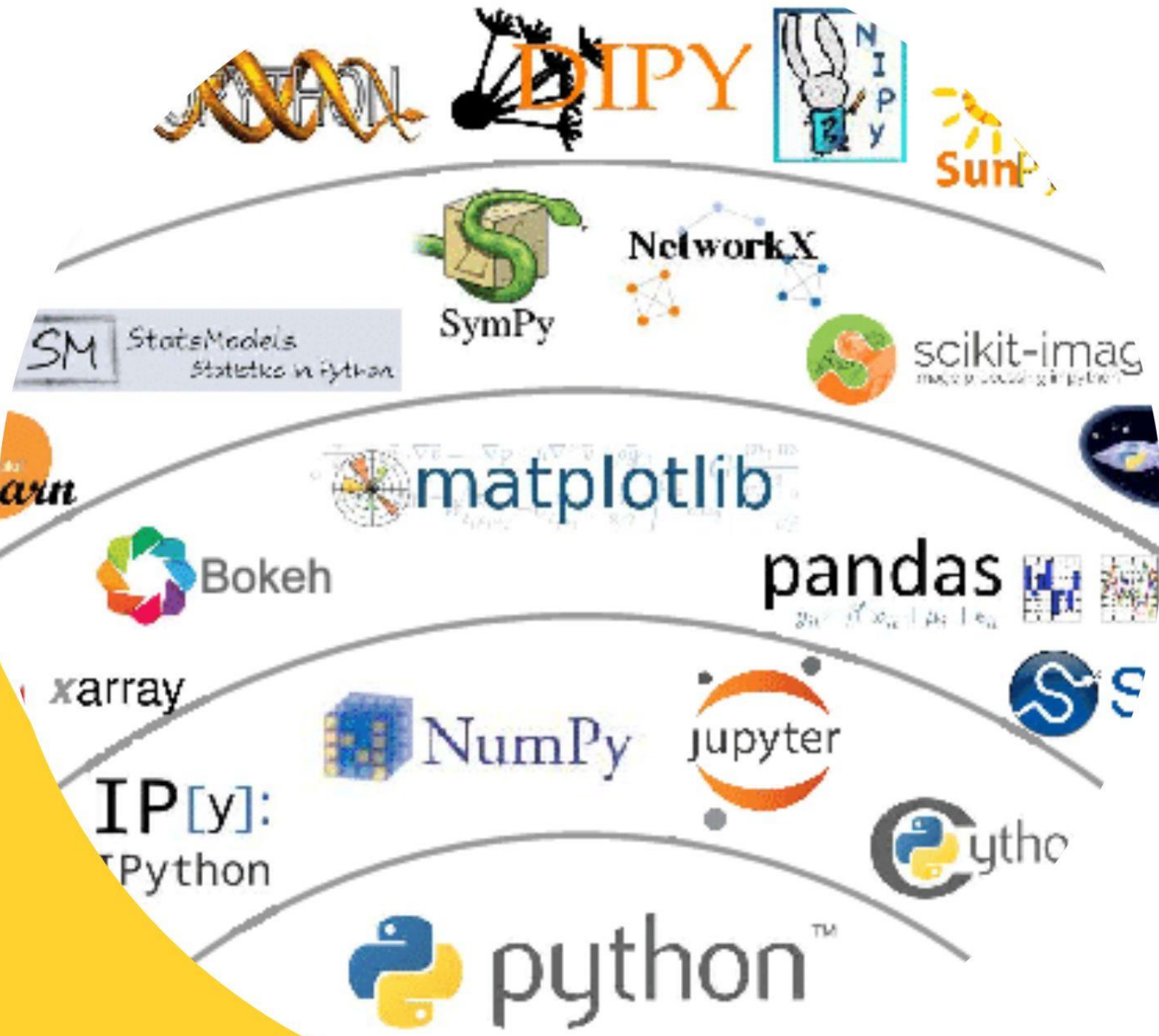
Навчальне видання

Ковтонюк Галина Миколаївна

ОСНОВИ ПРОГРАМУВАННЯ МОВОЮ PYTHON. ЛАБОРАТОРНИЙ ПРАКТИКУМ ЗІ ШКІЛЬНОГО КУРСУ ІНФОРМАТИКИ

для студентів предметної спеціальності
014.04 Середня освіта (Математика)

Формат 60×84/16.
Папір офсетний. Гарнітура Times New Roman.
Ум. друк. арк. 7,2
Виготовлювач ФОП Рогальська І.О.
м. Вінниця, вул. Хмельницьке шосе, 145
тел. (0432) 43-51-39, 65-80-80
E-mail: dilo_vd@ukr.net
Свідоцтво ВОЗ № 635744 від 01.03.2010 р.




```
import tkinter as tk

root = tk.Tk()
w = root.winfo_height
root.geometry("400x400+500+400")

lab1 = tk.Label(root,
    text="Hello",
    font="Arial 20",
    bg="yellow")
lab1.pack()
```