

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет

ЗВІТ
ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 1
з навчальної дисципліни
“Скриптові мови програмування (Python)”
АРИФМЕТИЧНІ ВИРАЗИ, УПРАВЛЯЮЧІ КОНСТРУКЦІЇ
ТА МАСИВИ У PYTHON

ВИКОНАВ
студент академічної групи КН-24
Мельник Д.С.

ПЕРЕВІРИВ
асистент кафедри кібербезпеки та
програмного забезпечення
Ткаченко О.С

Тема: Арифметичні вирази, управляючі конструкції та масиви у Python

Мета: навчитися створювати найпростіші програми на Python, використовуючи оператори вибору і циклів, арифметичні вирази та масиви

Варіант 20

1)
$$z = \frac{\sqrt{m}-\sqrt{n}}{m}$$

Числа m та n вводяться користувачем у консолі Python.

2) Визначити, чи являється число n досконалим. Досконале число – натуральне число, яке дорівнює сумі всіх своїх дільників, напр., 6 ($1 + 2 + 3 = 6$), 28 ($1 + 2 + 4 + 7 + 14 = 28$).

3) Дано одновірний масив, що складається з N дійсних елементів.

- Знайти мінімальний елемент.
- Обчислити добуток не нульових елементів масиву.
- Вивести додатні елементи на екран у зворотному порядку.

Завдання 1

$$z = \frac{\sqrt{m}-\sqrt{n}}{m}$$

Лістинг програми

```
1 import math
2
3 m = int(input("Введіть M\n"))
4 n = int(input("Введіть N\n"))
5
6 if m <= 0 or n < 0:
7     print("ERROR!!")
8 else:
9     z = (math.sqrt(m) - math.sqrt(n)) / m
10 print(z)
```

Опис принципу роботи

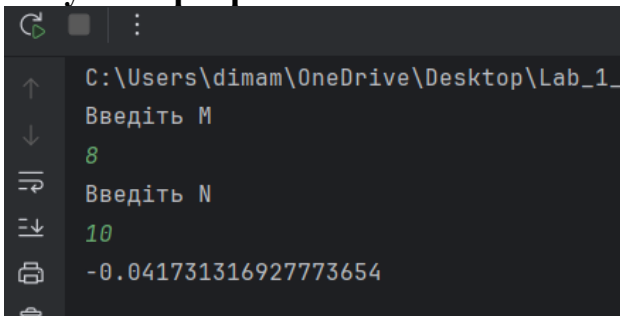
Цей код виконує наступні дії:

1. Зчитує два цілі числа (m і n) від користувача.
2. Перевіряє коректність введених значень:
 - Якщо $m \leq 0$ або $n < 0$, виводить "ERROR!!" і завершує виконання.
 - Це зроблено для запобігання математичним помилкам, оскільки:
 - Корінь ($\text{math.sqrt}(n)$) не можна обчислювати для від'ємного n .
 - Ділення на m , якщо $m = 0$, спричинило б помилку.
3. Обчислює значення z за формулою:
$$z = \frac{\sqrt{m}-\sqrt{n}}{m}$$
4. Виводить результат z

Вхідні дані: m, n

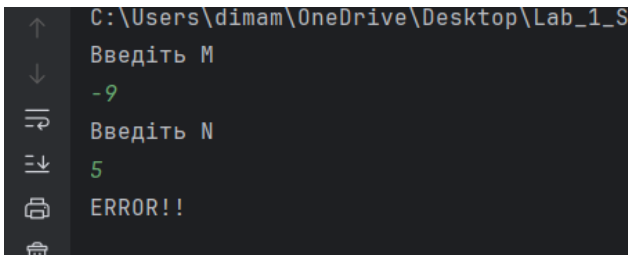
Вихідні дані: z

Запуски програми



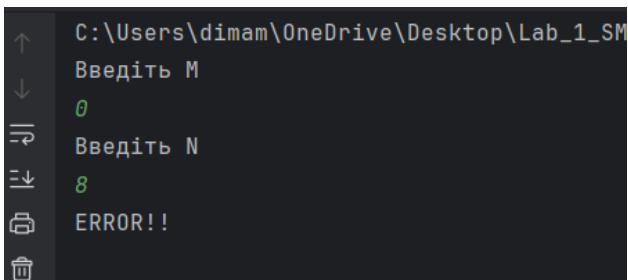
```
C:\Users\dimam\OneDrive\Desktop\Lab_1_...
Введіть M
8
Введіть N
10
-0.041731316927773654
```

Рисунок 1



```
C:\Users\dimam\OneDrive\Desktop\Lab_1_...
Введіть M
-9
Введіть N
5
ERROR!!
```

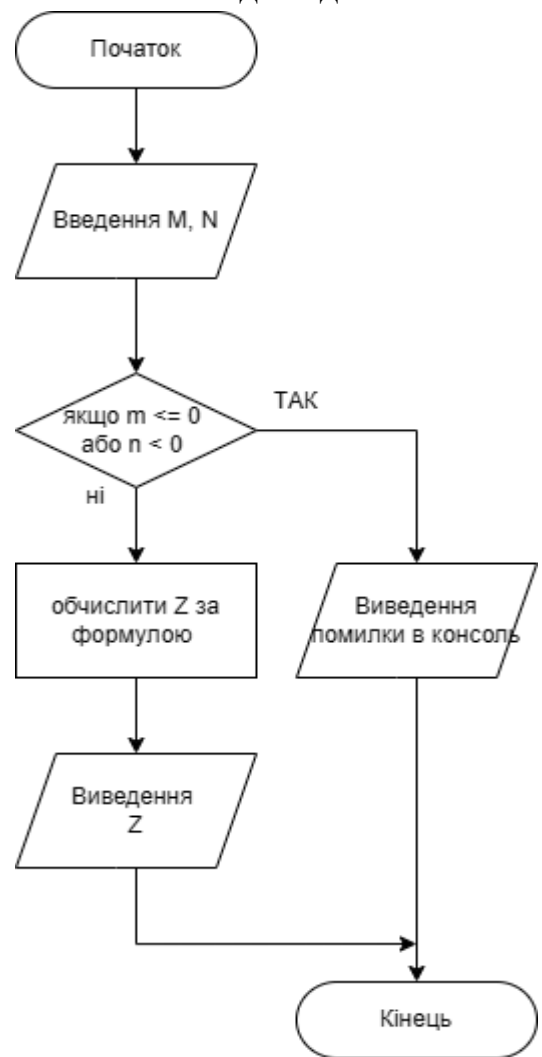
Рисунок 2



```
C:\Users\dimam\OneDrive\Desktop\Lab_1_...
Введіть M
0
Введіть N
8
ERROR!!
```

Рисунок 3

Блок-схема до задачі



Завдання 2

Визначити, чи являється число n досконалим. Досконале число – натуральне число, яке дорівнює сумі всіх своїх дільників, напр., $6 (1 + 2 + 3 = 6)$, $28 (1 + 2 + 4 + 7 + 14 = 28)$.

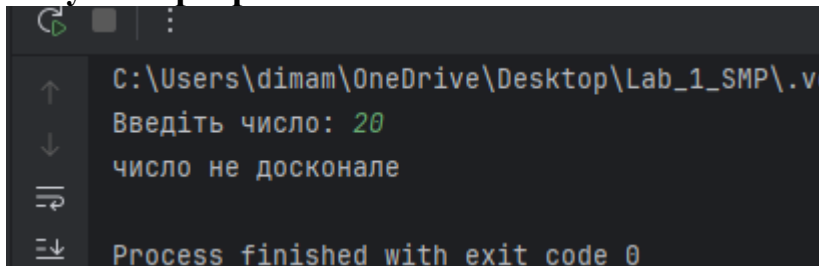
Програма перевіряє, чи є введене число досконалим. Досконале число — це таке число, яке дорівнює сумі всіх своїх дільників, окрім самого себе.

1. Спочатку програма запитує у користувача число.
2. Потім вона ініціалізує змінну, яка буде зберігати суму дільників числа.
3. За допомогою циклу програма перебирає всі числа від 1 до введеного числа мінус 1.
4. Якщо поточне число є дільником введеного (тобто, число ділиться на нього без залишку), то це число додається до змінної суми.
5. Коли цикл завершено, програма порівнює суму дільників з введеним числом.
6. Якщо сума дільників дорівнює введеному числу, програма виводить "число досконале".
7. Якщо сума не дорівнює числу, виводиться "число не досконале".

Лістинг задачі 2

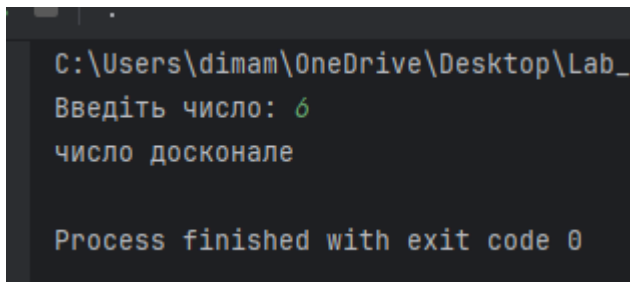
```
1 number_n = int(input("Введіть число: "))
2 temp = 0
3
4 for i in range(1, number_n):
5     if number_n % i == 0:
6         temp += i
7
8 if temp == number_n:
9     print("число досконале")
10 else:
11     print("число не досконале")
```

Запуски програми



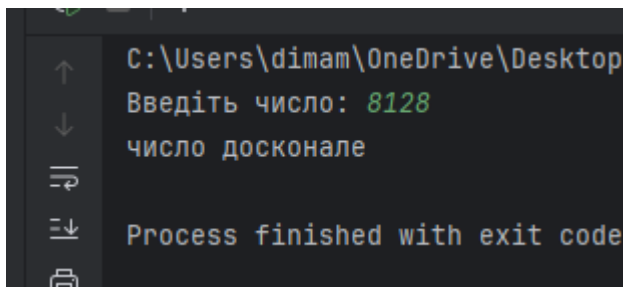
```
C:\Users\dimam\OneDrive\Desktop\Lab_1_SMP\.venv> python lab1_2.py
Введіть число: 20
число не досконале
Process finished with exit code 0
```

Рисунок 4



```
C:\Users\dimam\OneDrive\Desktop\Lab_1_SMP> python lab1_2.py
Введіть число: 6
число досконале
Process finished with exit code 0
```

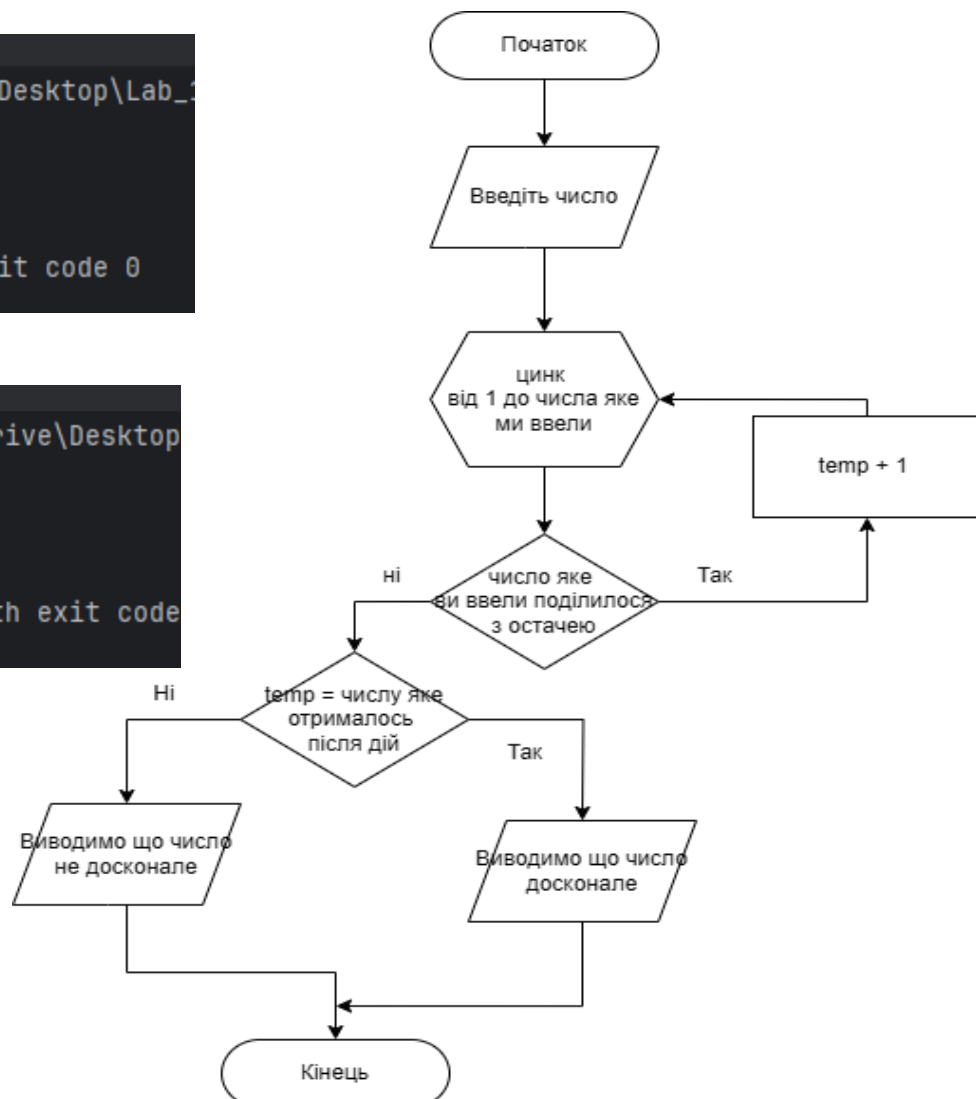
Рисунок 5



```
C:\Users\dimam\OneDrive\Desktop\Lab_1_SMP> python lab1_2.py
Введіть число: 8128
число досконале
Process finished with exit code 0
```

Рисунок 6

Блоксхема до задачі 2



Завдання 3

Дано одномірний масив, що складається з N дійсних елементів.

- Знайти мінімальний елемент.
- Обчислити добуток не нульових елементів масиву.
- Вивести додатні елементи на екран у зворотному порядку.

Ця програма працює наступним чином:

1. **Введення кількості елементів масиву:** Програма спочатку запитує користувача, скільки елементів має бути в масиві, і зчитує це значення через `input()`. Якщо кількість елементів менше або рівна нулю, програма виводить повідомлення про помилку і завершується.
2. **Створення масиву з випадковими числами:** За допомогою генератора списку, програма створює масив `array` з випадкових цілих чисел у діапазоні від -10 до 10 (включно). Кількість елементів масиву дорівнює значенню, яке ввів користувач.
3. **Пошук мінімального числа масиву:** За допомогою функції `min()` програма знаходить мінімальне число у масиві та зберігає його в змінну `min_num`.
4. **Отримання додатних чисел у зворотному порядку:** Програма створює новий список `dodatni_chusla`, що містить всі додатні числа масиву. Цей список потім перевертається у зворотний порядок за допомогою зрізу `[::-1]`.
5. **Обчислення добутку ненульових елементів:** Програма ініціалізує змінну `temp` значенням 1. Потім вона перебирає всі елементи масиву, перемножуючи всі ненульові числа (якщо елемент масиву не дорівнює нулю). Якщо хоча б один ненульовий елемент знайдено, змінна `zero_num_in_array` отримує значення `True`, що дозволяє вивести результат обчислення добутку ненульових елементів. Якщо ж всі елементи масиву рівні нулю, програма виведе повідомлення про відсутність ненульових елементів.
6. **Виведення результатів:** Наприкінці програма виводить:
 - початковий масив,
 - мінімальне число масиву,
 - додатні числа масиву у зворотному порядку,
 - добуток ненульових елементів (якщо є ненульові елементи).

Приклад виконання:

1. Користувач вводить кількість елементів масиву, наприклад, 5.
2. Програма генерує масив, наприклад, `[-1, 2, 0, 5, -3]`.
3. Мінімальне число масиву: -3.
4. Додатні числа у зворотному порядку: `[5, 2]`.
5. Добуток ненульових елементів: $-1 * 2 * 5 * -3 = 30$.

Лістинг програми

```
import random

num_array = int(input("Введіть кількість елементів в масиві:\n"))
if num_array <= 0:
    print("Помилка! Кількість елементів має бути більше 0")
    exit()

array = [round(random.uniform(-10,10), 2) for _ in range(num_array)]
min_num = min(array) #мінімальне число масиву

dodatni_chusla = [num for num in array if num > 0]
dodatni_chusla= dodatni_chusla[::-1]

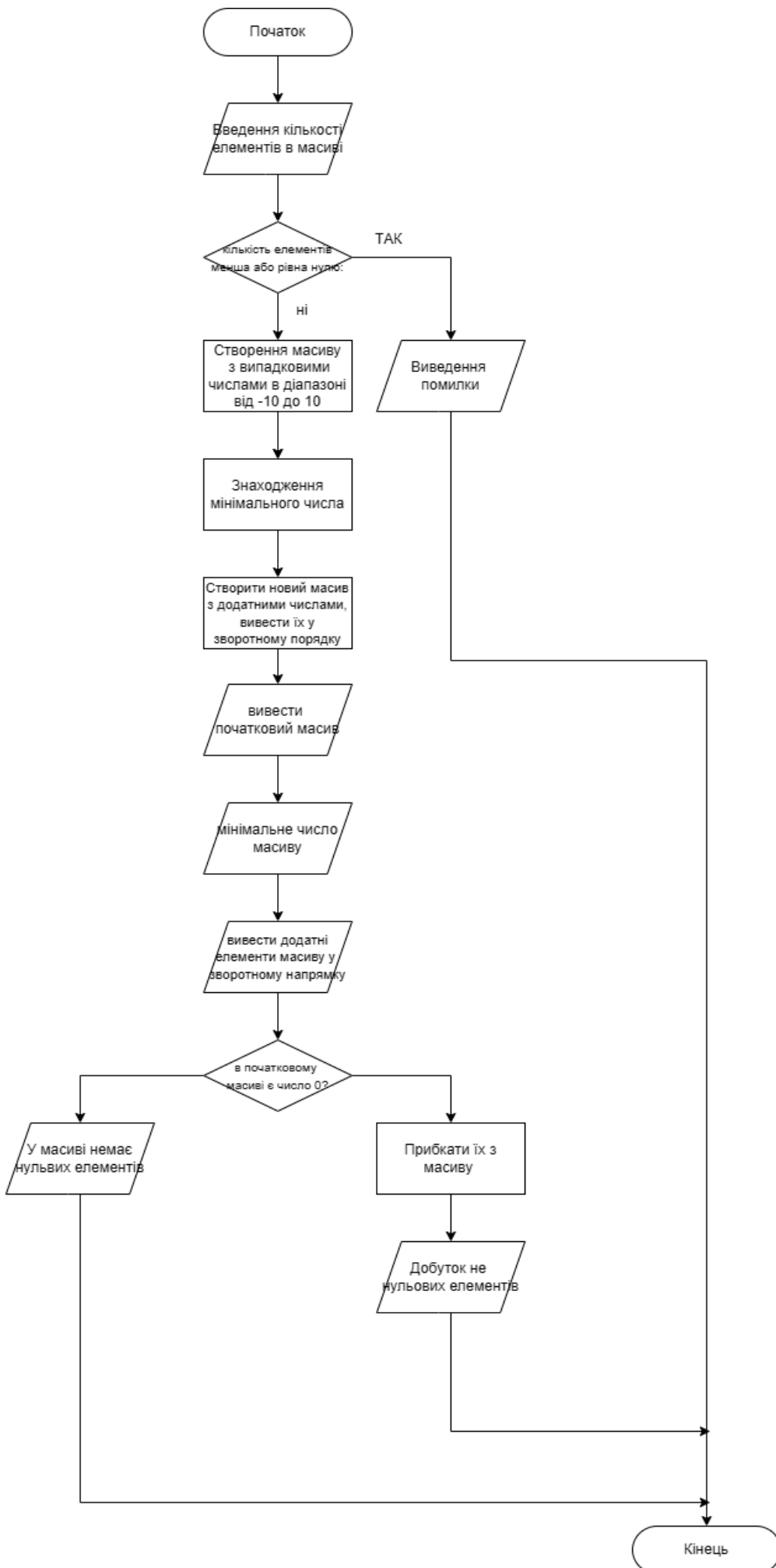
zero_num_in_array = False
temp = 1

for num in array:
    if num != 0:
        temp *= num
        zero_num_in_array = True

print("Початковий масив:\n", array)
print("мінімальне число масиву\n", min_num)
print("додатні елементи масиву у зворотному порядку\n", dodatni_chusla)

if zero_num_in_array:
    print("Добуток ненульових елементів:\n", temp)
else:
    print("У масиві немає ненульових елементів\n")
```

Блок-схема до 3 завдання



Запуски завдання 3

```
C:\Users\dimam\OneDrive\Desktop\Lab_1_SMP\.venv\Scripts>
Введіть кількість елементів в масиві:
5
Початковий масив:
[-8.16, -3.13, 5.1, 7.74, -0.63]
мінімальне число масиву
-8.16
додатні елементи масиву у зворотному порядку
[7.74, 5.1]
Добуток ненульових елементів:
-635.164449696
```

```
Введіть кількість елементів в масиві:
20
Початковий масив:
[6.65, 4.45, -6.67, 8.7, 6.76, 3.45, -8.46, -2.97, -3.04, 4.9, -3.73, 3.67, -3.62, -3.48, -4.46, -3.75, -5.19, -0.11, -0.85, 2.99]
мінімальне число масиву
-8.46
додатні елементи масиву у зворотному порядку
[2.99, 3.67, 4.9, 3.45, 6.76, 8.7, 4.45, 6.65]
Добуток ненульових елементів:
62728947945.04563
```

```
C:\Users\dimam\OneDrive\Desktop\Lab_1_SMP\.venv\Scripts>
Введіть кількість елементів в масиві:
3
Початковий масив:
[-4.52, 1.05, -2.61]
мінімальне число масиву
-4.52
додатні елементи масиву у зворотному порядку
[1.05]
Добуток ненульових елементів:
12.387059999999998
```

1. Особливості та переваги мови Python:

- Простий і зрозумілий синтаксис.
- Підтримує різні парадигми програмування (об'єктно-орієнтоване, процедурне, функціональне).
- Велика стандартна бібліотека та активна спільнота розробників.

2. Основні принципи синтаксису мови Python:

- Відступи для визначення блоків коду.
- Відсутність крапки з комою в кінці рядків.
- Використання двокрапки для початку блоків коду.

3. Як здійснюється введення/виведення даних у мові Python:

- `print()` — для виведення даних.
- `input()` — для введення даних.

4. Синтаксис циклу `for` у мові Python:

`for` елемент `in` послідовність:
 блок коду

5. Принципи роботи зі списками у мові Python:

- Списки є змінюваними (мутабельними).
- Підтримують індексацію та зрізи.
- Можуть містити елементи різних типів даних.