Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



**Звіт**

до лабораторної роботи №7

З дисципліни: «Кросплатформні засоби програмування»

На тему: «ДОСЛІДЖЕННЯ БАЗОВИХ КОНСТРУКЦІЙ МОВИ PYTHON»

**Варіант 3**

Виконав: ст. гр. КІ-305

Гнідець В.М.

Прийняв:

Іванов Ю.С.

Львів 2023

**Мета роботи:** ознайомитися з базовими конструкціями мови Python та оволодіти навиками написання простих програм.

**Завдання:**

1.Написати та налагодити програму на мові Python згідно варіанту. Програма має задовольняти наступним вимогам:

• програма має розміщуватися в окремому модулі;

• програма має генерувати зубчастий список, який міститиме лише заштриховані області квадратної матриці згідно варіанту;

• розмір квадратної матриці і символ-заповнювач масиву вводяться з клавіатури;

• при не введені або введенні кількох символів-заповнювачів відбувається коректне переривання роботи програми;

• сформований масив вивести на екран;

• програма має містити коментарі.

2. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.

3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.

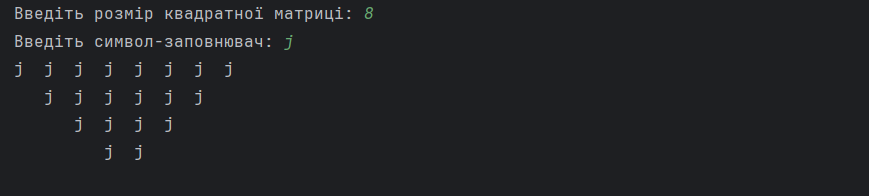
4. Дати відповідь на контрольні запитання.

**Варіант 3:** 

**Код програми**

import os  
import struct  
import math  
  
rows\_num = int(input("Введіть розмір квадратної матриці: "))  
filler = input("Введіть символ-заповнювач: ")  
#filler1 = input("Введіть символ-заповнювач для непарних рядків: ")  
  
if len(filler) == 0:  
 print("Не введено символ-заповнювач")  
 sys.exit(1)  
  
n\_subarrays = (rows\_num + 1) // 2  
newline\_count = (rows\_num // 2) - 1  
leading\_spaces = 0  
  
for i in range(n\_subarrays):  
 for \_ in range(leading\_spaces):  
 print(" ", end=" ")  
  
 num\_elements = rows\_num - 2 \* i  
 arr = []  
  
 # Вибір символу-заповнювача в залежності від парності рядка  
 #current\_filler = filler1 if i % 2 != 0 else filler  
  
 for \_ in range(num\_elements):  
 if len(filler) == 1:  
 arr.append(ord(filler))  
 print(chr(arr[0]), end=" ")  
 else:  
 print("Забагато символів-заповнювачів")  
 sys.exit(1)  
 print()  
 leading\_spaces += 1  
  
for \_ in range(newline\_count):  
 print()

**Результат роботи програми:**

****

**Контрольні питання**

1. ***Який вигляд має програма мовою Python?***

Програма мовою Python має текстовий вигляд і складається з послідовності інструкцій, які визначають поведінку програми. Основні елементи, які можна знайти в програмі Python, включають:

**1.Імпортування модулів**: Часто програми Python починаються з імпортування різних модулів або бібліотек, які містять функції та об'єкти, які можна використовувати в програмі. Наприклад:

******

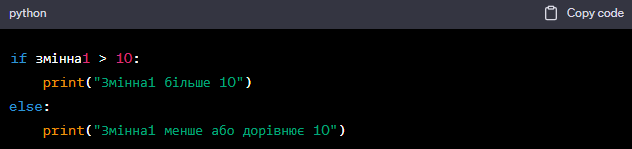
**2.Оголошення змінних**: Змінні використовуються для зберігання даних. Вони можуть бути оголошені і присвоєні значення в програмі. Наприклад:

******

**3.Використання функцій**: Функції визначаються за допомогою ключового слова **def** і використовуються для виконання певних операцій. Наприклад:

******

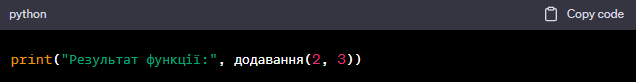
**4.Умовні оператори**: Умовні оператори використовуються для прийняття рішень на основі певних умов. Наприклад:

******

**5.Цикли**: Цикли використовуються для повторення певних дій кілька разів. Наприклад, цикл **for**:

******

**6.Вивід результатів**: Результати програми можуть виводитися на екран за допомогою функції **print()**. Наприклад:

******

Це загальний вигляд програми мовою Python. Фактичний вигляд програми буде залежати від її завдання і структури, яку розробник обрав для виконання цього завдання.

1. ***Як запустити на виконання програму мовою Python?***

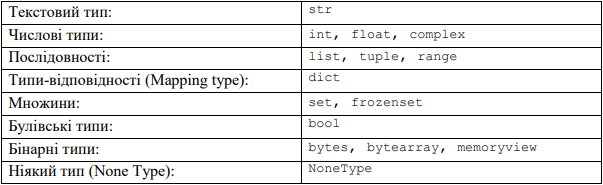
Для запуску на виконання програми мовою Python слід виконати в командному рядку: python.exe .py. Запустивши інтерпретатор Python.exe, можна вводити з командного рядка програму по-рядково і зразу отримувати результат виконання.

***3. Які коментарі підтримує Python?***

Python має лише рядкові коментарі. Коментарем у Python є текст після символа ‘#’:

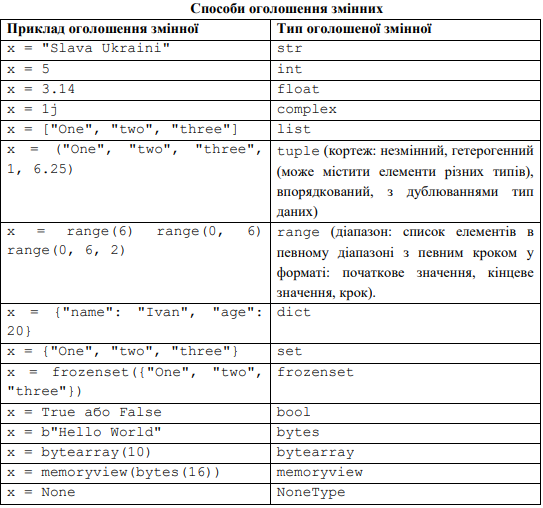
# Comment

***4. Які типи даних підтримує Python?***

******

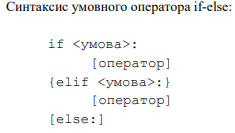
***5. Як оголосити змінну?***

Змінна може бути оголошена в будь-якому місці і має бути обов’язково проініціалізована. Тип змінної визначається значенням, яким вона ініціалізована.

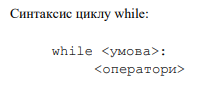
******

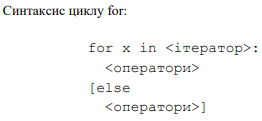
***6. Які керуючі конструкції підтримує Python?***

**1. Умовні конструкції**:

******

**2.Цикли**:

******

******

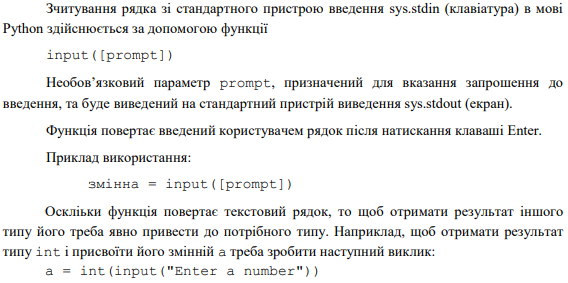
***7. Які операції підтримує Python?***

Python підтримує широкий спектр операцій, які можуть виконуватися над різними типами даних. Основні операції включають в себе:

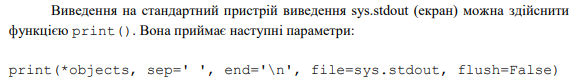
1. **Арифметичні операції**:
   * Додавання (**+**): Додає два числа.
   * Віднімання (**-**): Віднімає одне число від іншого.
   * Множення (**\***): Перемножує два числа.
   * Ділення (**/**): Ділить перше число на друге.
   * Остача від ділення (**%**): Повертає залишок від ділення.
   * Цілочисельне ділення (**//**): Ділить перше число на друге, повертаючи цілу частину результату.
2. **Логічні операції**:
   * І (**and**): Повертає True, якщо обидва операнди є True.
   * Або (**or**): Повертає True, якщо хоча б один із операндів є True.
   * Не (**not**): Інвертує значення операнду.
3. **Порівняння**:
   * Рівність (**==**): Порівнює два операнди на рівність.
   * Нерівність (**!=**): Порівнює два операнди на нерівність.
   * Більше (**>**): Перевіряє, чи перший операнд більший за другий.
   * Менше (**<**): Перевіряє, чи перший операнд менший за другий.
   * Більше або рівне (**>=**): Перевіряє, чи перший операнд більший або рівний другому.
   * Менше або рівне (**<=**): Перевіряє, чи перший операнд менший або рівний другому.
4. **Присвоєння**:
   * Присвоєння (**=**): Присвоює значення змінній.
5. **Рядкові операції**:
   * Конкатенація (**+**): Об'єднує два рядки.
   * Повторення (**\***): Повторює рядок задану кількість разів.
   * Довжина рядка (**len()**): Повертає кількість символів у рядку.
6. **Спискові операції**:
   * Індексація (**[]**): Дозволяє отримати доступ до елементів списку за їх індексом.
   * Зріз (**[start:stop:step]**): Дозволяє витягнути підсписок із списку.
   * Додавання (**+**): Об'єднує два списки.
   * Повторення (**\***): Повторює список задану кількість разів.
   * Довжина списку (**len()**): Повертає кількість елементів у списку.
7. **Операції зі змінними типами даних**:
   * Приведення типів: Можливість змінювати тип даних, наприклад, перетворювати число на рядок.

Це лише декілька прикладів операцій, які підтримує Python. Мова також має багато інших операцій і функцій для роботи з різними типами даних і структурами.

1. ***Як здійснити ввід з консолі?***

******

1. ***Як здійснити вивід у консоль?***

******

***10. Як здійснити приведення типів?***

У мові програмування Python можливе приведення (конвертування) типів даних з одного типу в інший. Це може бути корисним, коли вам потрібно виконати операції або операції порівняння між різними типами даних або коли ви хочете змінити тип даних для подальших обчислень або операцій.

Для приведення типів у Python використовуються функції та конструктори типів. Ось декілька основних способів приведення типів:

**1.Приведення до цілих чисел (integer)**:

Ви можете використовувати функцію **int()** для перетворення значення у ціле число. Наприклад:

рядок = "123"

ціле\_число = int(рядок)

**2.Приведення до дійсних чисел (float)**:

Ви можете використовувати функцію **float()** для перетворення значення у дійсне число. Наприклад:

рядок = "3.14"

дійсне\_число = float(рядок)

**3.Приведення до рядків (string)**:

Ви можете використовувати функцію **str()** для перетворення значення у рядок. Наприклад:

число = 42

рядок = str(число)

**4.Приведення до списків (list), кортежів (tuple) і множин (set)**:

Ви можете використовувати конструктори типів, такі як **list()**, **tuple()**, і **set()** для перетворення інших ітерабельних об'єктів (наприклад, рядків або списків) у відповідний тип даних. Наприклад:

рядок = "Hello"

список = list(рядок)

кортеж = tuple([1, 2, 3])

множина = set([1, 2, 3, 2]) # Приведення до множини видаляє дублікати

**5.Приведення до булевого типу (bool)**:

Булевий тип можна отримати за допомогою конструктора **bool()**. Будь-яке значення, крім 0 або пустого рядка, буде перетворено в **True**, а 0 або пустий рядок буде перетворено в **False**. Наприклад:

значення = 42

булеве\_значення = bool(значення)

**6.Інші специфічні випадки**:

В деяких випадках, наприклад, для приведення списку рядків до рядка, використовують операції, які специфічні для даного типу даних.

Це декілька прикладів приведення типів у мові програмування Python. Важливо враховувати, що не всі операції приведення типів можуть бути можливими, і деякі операції можуть призводити до втрати даних або небажаних наслідків, тому слід бути обережним при їх використанні.

**Висновок**

Досліджував базові конструкції мови Python.