

**Міністерство освіти і наук України**  
**Національний університет “Львівська політехніка”**



Кафедра ЕОМ

**ДОСЛІДЖЕННЯ БАЗОВИХ КОНСТРУКЦІЙ МОВИ PYTHON**

**Методичні вказівки**  
до лабораторної роботи № 7 з курсу “Кросплатформні засоби  
програмування”  
для студентів базового напрямку 6.123 - “Комп’ютерна інженерія”

Затверджено  
на засіданні кафедри  
”Електронні обчислювальні  
машини”  
Протокол № 1 від 28.08.2023 р.

Львів – 2023

**Дослідження базових конструкцій мови Python:** Методичні вказівки до лабораторної роботи № 7 з курсу “Кросплатформні засоби програмування” для студентів базового напрямку 6.123 - “Комп’ютерна інженерія” / Укладач: Олексів М.В. – Львів: Національний університет “Львівська політехніка”, 2023, 15 с.

**Укладач**

Олексів М. В., к.т.н.

**Рецензент**

**Відповідальний за випуск:**

Мельник А. О., професор, завідувач  
кафедри

# ДОСЛІДЖЕННЯ БАЗОВИХ КОНСТРУКЦІЙ МОВИ PYTHON

**Мета:** ознайомитися з базовими конструкціями мови Python.

## ТЕОРЕТИЧНІ ВІДОМОСТІ

Python програми – це набір \*.py файлів. Кожен файл – це окремий модуль. Модулі підключаються за допомогою ключового слова `import` після якого вказується назва файлу без розширення `py`. Щоб доступитися з одного модуля до іншого вони мають бути в одному каталозі. Якщо модуль лежить в іншому каталозі, то щоб до нього доступитися цей каталог має бути оформлений як пакет. Для цього достатньо у нього помістити порожній файл `__init__.py`. При підключенні пакету виконується вміст файлу `__init__.py`, який може містити визначення імен і інші необхідні для роботи з пакетом конфігураційні дії.

### Коментарі

Python має лише рядкові коментарі. Коментарем у Python є текст після символу '#':  
# Comment

### Форматування коду

Код у Python виділяється у блоки за допомогою 4-ох пробілів відносно попереднього блоку або одного символу табуляції. Пробіли і табуляцію не можна змішувати.

### Запуск на виконання програми мовою Python

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

### Типи даних, оголошення змінних та операції над ними

Python підтримує наступні типи даних.

*Таблиця 1.*

**Типи даних Python**

Текстовий тип:	<code>str</code>
Числові типи:	<code>int, float, complex</code>
Послідовності:	<code>list, tuple, range</code>
Типи-відповідності (Mapping type):	<code>dict</code>
Множини:	<code>set, frozenset</code>
Булівські типи:	<code>bool</code>
Бінарні типи:	<code>bytes, bytearray, memoryview</code>
Ніякий тип (None Type):	<code>NoneType</code>

Детальніше типи і операції над ними описані тут: <https://docs.python.org/3/library/stdtypes.html#built-in-typ>. Для визначення типу змінної слід виконати команду: `type(<змінна>)`

## Оголошення змінних

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

Таблиця 2.

Способи оголошення змінних

Приклад оголошення змінної	Тип оголошеної змінної
<code>x = "Slava Ukraini"</code>	<code>str</code>
<code>x = 5</code>	<code>int</code>
<code>x = 3.14</code>	<code>float</code>
<code>x = 1j</code>	<code>complex</code>
<code>x = ["One", "two", "three"]</code>	<code>list</code>
<code>x = ("One", "two", "three", 1, 6.25)</code>	<code>tuple</code> (кортеж: незмінний, гетерогенний (може містити елементи різних типів), впорядкований, з дублюваннями тип даних)
<code>x = range(6)</code> <code>range(0, 6)</code> <code>range(0, 6, 2)</code>	<code>range</code> (діапазон: список елементів в певному діапазоні з певним кроком у форматі: початкове значення, кінцеве значення, крок).
<code>x = {"name": "Ivan", "age": 20}</code>	<code>dict</code>
<code>x = {"One", "two", "three"}</code>	<code>set</code>
<code>x = frozenset({"One", "two", "three"})</code>	<code>frozenset</code>
<code>x = True</code> або <code>False</code>	<code>bool</code>
<code>x = b"Hello World"</code>	<code>bytes</code>
<code>x = bytearray(10)</code>	<code>bytearray</code>
<code>x = memoryview(bytes(16))</code>	<code>memoryview</code>
<code>x = None</code>	<code>NoneType</code>

Для зміни типу змінної після ініціалізації слід використовувати оператор приведення типу: `x = int(1)`.

## Рядки

Букви у рядку можуть займати різну кількість байт в залежності від кодування, яке використовується. З рядками (тип `str`) можна робити наступні операції:

- Оголошувати багаторядкові рядки між `"""`:

```
a = """ Multiline
      string"""
```

- Отримувати частини тексту вказуючи діапазон індексів букв:

```
b = "Hello World!"
print(b[3:7]) # виведеться 'lo W'
print(b[-1]) # виведеться '!'
```

```
print(b[0:-2]) # виведеться "Hello Worl"
print(b[::-1]) # виведеться ревертований рядок '!dlroW olleH'
```

- Викликати методи обробки рядків, оскільки тип `str` є класом.
- Здійснювати конкатенацію рядків (не рекомендовано, так як призводить до втрат пам'яті):  
`b = "Hello" + "World!"`
- Здійснювати форматування рядків:  
`val1 = 1`  
`val2 = 2`  
`val3 = 33.1`  
`text = "Val1 = {0}, Val2 = {1}, Val3 = {2}"`  
`print(text.format(val1, val2, val3))`
- Використовувати f-рядки:  
`val1 = 1`  
`val2 = 2`  
`val3 = 33.1`  
`text = f"Val1 = {val1}, Val2 = {val2}, Val3 = {val3}"`  
`print(text)`

## Списки

Списки також використовуються для заміни масивів, так як масивів у Python немає. Основні операції над списками:

- Оголошення списку:  
`x = []` # порожній список  
`x = list()` # порожній список  
`x = ["One", "two", "three"]` # ініціалізований список
- Доступ до елементів списку відбувається по індексу. Індксація починається з 0 :  
`x[1]`
- Присвоєння:  
`x[1] = "New value"`
- Визначення розміру списку:  
`len(x)`

## Основні методи модифікації списків

Метод	Призначення
<code>append()</code>	Додати елемент(и) в кінець списку
<code>clear()</code>	Очистити список
<code>copy()</code>	Отримати копію списку
<code>count()</code>	Підрахунок кількості елементів з заданим значенням
<code>extend()</code>	Додати елементи до кінця списку, або будь що, що можна проітерувати.
<code>index()</code>	Повертає індекс першого значення заданого параметром
<code>insert()</code>	Додати елемент в задану позицію
<code>pop()</code>	Повертає значення заданого параметром індексу елемента і видаляє його зі списку
<code>remove()</code>	Видаляє елементи з вказаним значенням зі списку
<code>reverse()</code>	Розташовує елементи списку в зворотному порядку
<code>sort()</code>	Сортує список

## Словники

Словник – це набір пар елементів ключ-значення. Ключі мають бути унікальні.  
Оголошення словника:

```
x = {} # порожній словник
x = dict() # порожній словник
x = {
    "name": "Ivan",
    "age": 20
}
```

- Доступ до елемента словника за ключем:

```
x["age"]
```

- Отримання списку ключів:

```
x.keys()
```

- Модифікація значень:

```
x["age"] = 33
x.update({"name": "Petro"})
```

- Додавання елементів:

```
x["height"] = 180
x.update({"weight": 82})
```

- Видалення елементів:

```
x.pop("weight")
del x["height"]
```

- Видалення змінної словника:

```
del x
```

Таблиця 4.

#### Основні методи модифікації словників

Метод	Призначення
<code>clear()</code>	Видаляє усі елементи словника
<code>copy()</code>	Повертає копію словника
<code>fromkeys()</code>	Повертає новий словник на базі вказаних ключів
<code>get()</code>	Повертає значення визначеного ключа
<code>items()</code>	Повертає список, що містить tuple (кортеж) для кожної пари ключ-значення
<code>keys()</code>	Повертає список ключів
<code>pop()</code>	Видаляє елемент з вказаним ключем і повертає його значення
<code>popitem()</code>	Видаляє останню додану пару ключ-значення
<code>setdefault()</code>	Повертає значення вказаного ключа. Якщо ключа не існує, то додає ключ з вказаним значенням
<code>update()</code>	Оновлює словник вказаними значеннями
<code>values()</code>	Повертає список значень словника

Словники і списки зручно використовувати при роботі з json і серіалізації даних.

Таблиця 5.

**Математичні операції**

Назва операції	Оператор
Додавання	+
Віднімання	-
Множення	*
Ділення	/
Цілочисельне ділення	//
Піднесення до степеня	**
Остача від ділення	%

**Оператори**  
**Умовний оператор if-else**

Синтаксис умовного оператора if-else:

```
if <умова>:
    [оператор]
elif <умова>:
    [оператор]
else:
    [оператор]
```

Таблиця 6.

**Умови**

Тип умови	Приклад
Рівність	a == b
Не рівність	a != b
Менше	a < b
Менше або рівне	a <= b
Більше	a > b
Більше або рівне	a >= b

Приклад:

```
if a<b:
    c=1
elif a>b:
    c=2
else:
    c=3
```

Якщо гілка не має містити операцій, то можна вказати порожній оператор pass:

```
if a<b:
    c=1
```



```
else:
    pass
```

Таблиця 7.

### Логічні оператори

Оператор	Приклад
або	or
і	and
ні	not

```
if a<b and b>0:
    c=1 # ця змінна буде доступна також і після виходу з if
else:
    c=2
```

Скорочені форми запису (умовні вирази):

```
if a > b: c=2
c=2 if a > b else c=1
c=2 if a > b else c=1 if a == b else c=3
```

### Цикл while

Синтаксис циклу while:

```
while <умова>:
    <оператори>
```

Приклад:

```
i = 1
while i < 5:
    i += 1
```

### Цикл for

Синтаксис циклу for:

```
for x in <ітератор>:
    <оператори>
[else
    <оператори>]
```

Цикл for у python має деякі відмінності в порівнянні з іншими мовами програмування. Розглянемо принцип його роботи. Змінній x по чергові присвоюються елементи, що знаходяться у ітераторі, і для кожного з них виконуються оператори тіла циклу. Після завершення виконання циклу виконується блок операторів після else.

Приклади застосування циклу for:

1. Цикл for використовується для ітерування по послідовностях, таких як список, кортеж, словник, множина, рядок.

```
nums = ["one", "two", "three"]
for x in nums:
    print(x)
```

```
for x in "Hello":
    print(x)
```

2. Ітерування по діапазону значень (аналог звичайного циклу for).

```
for x in range(5): # створюється послідовність 0,1,2,3,4
    print(x)
```

```
for x in range(0, 5): # створюється послідовність 0,1,2,3,4
    print(x)
```

```
for x in range(0, 5, 2): # створюється послідовність 0,2,4
    print(x)
```

Перед початком виконання циклу range створює набір всіх значень, необхідних для циклу, тому це займає більше часу, ніж коли значення генеруються лише для окремої ітерації. Тому, доцільніше використовувати не range, а генератор:

```
def my_range(start, end, step):
    while start <= end:
        yield start
        start += step
```

```
for x in my_range(1, 10, 0.5):
    print(x)
```

У цьому прикладі з'являється оператор yield, який використовується для реалізації генераторів. Він перериває виконання функції і повертає своє поточне значення з функції. При наступному виклику функції вона почне виконуватися з наступного оператора після yield.

### **Оператори переривання потоку виконання**

Python 3 має 2 оператори переривання потоку виконання це – break і continue. Принцип їх роботи аналогічний до принципу роботи цих операторів у мові C/C++.

## Введення даних з клавіатури

Зчитування рядка зі стандартного пристрою введення `sys.stdin` (клавіатура) в мові Python здійснюється за допомогою функції

```
input([prompt])
```

Необов'язковий параметр `prompt`, призначений для вказання запрошення до введення, та буде виведений на стандартний пристрій виведення `sys.stdout` (екран).

Функція повертає введений користувачем рядок після натискання клавіші Enter.

Приклад використання:

```
змінна = input([prompt])
```

Оскільки функція повертає текстовий рядок, то щоб отримати результат іншого типу його треба явно привести до потрібного типу. Наприклад, щоб отримати результат типу `int` і присвоїти його змінній `a` треба зробити наступний виклик:

```
a = int(input("Enter a number"))
```

## Вивід даних на екран

Виведення на стандартний пристрій виведення `sys.stdout` (екран) можна здійснити функцією `print()`. Вона приймає наступні параметри:

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

де:

`*objects` – послідовність об'єктів розділених комами (змінні, рядки, константи), значення яких потрібно вивести

`sep=' '` – роздільник, який функція ставитиме між об'єктами, що виводяться (за замовчуванням – пробіл). Для виводу спеціальних символів використовують Escape-послідовності, наприклад:

`\'` – одинарна лапка

`\"` – подвійна лапка

`\?` – знак питання

`\\` – зворотний слеш

`\n` – новий рядок

`\t` – горизонтальна табуляція

`\v` – вертикальна табуляція

`end='\n'` – символ, що ставиться в кінці рядка (за замовчуванням – символ кінця рядка)

`file=sys.stdout` – виведення в файл. Об'єкт `file` повинен бути об'єктом з методом `write(str)`. `print()` можна використовувати тільки для текстових файлів.

`flush=False` – примусове очищення буфера виводу (за замовчуванням – не здійснюється, оскільки зазвичай визначається файлом).

Приклад:

```
a=b=3
print(a, b)
print("a+b=", a+b)
```

## КОНТРОЛЬНІ ПИТАННЯ

1. Який вигляд має програма мовою Python?
2. Як запустити на виконання програму мовою Python?
3. Які коментарі підтримує Python?
4. Які типи даних підтримує Python?
5. Як оголосити змінну?
6. Які керуючі конструкції підтримує Python?
7. Які операції підтримує Python?
8. Як здійснити ввід з консолі?
9. Як здійснити вивід у консоль?
10. Як здійснити приведення типів?

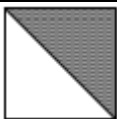

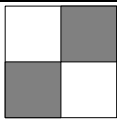
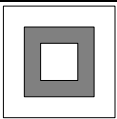


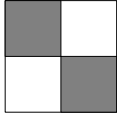
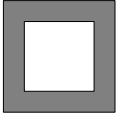



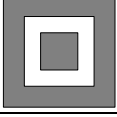
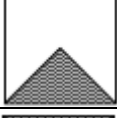
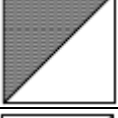
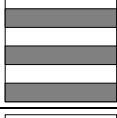
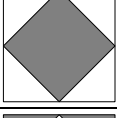


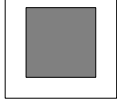
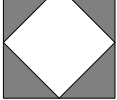
## ЛІТЕРАТУРА

1. Lutz M. Learning Python, 5th Edition / Mark Lutz. – O'Reilly, 2013. – 1643 p.
2. Васильєв О. Програмування мовою Python / Олексій Васильєв. – К: НК Богдан, 2019. – 504 с.
3. Python documentation [електронний ресурс]. – Режим доступу до документації: <https://docs.python.org/3/>

## 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 зубчатий список, який міститиме лише заштриховані області квадратної матриці згідно варіанту;
  - розмір квадратної матриці і символ-заповнювач масиву вводяться з клавіатури;
  - при не введенні 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.

## ВАРІАНТИ ЗАВДАНЬ

№		№		№		№	
1		6		11		16	
2		7		12		17	
3		8		13		18	
4		9		14		19	
5		10		15		20	

## ПОРЯДОК ВИКОНАННЯ

1. Переконайтеся, що у вас є встановлені інтерпретатор Python і середовище для розробки Microsoft Visual Studio Code (VS Code).
2. Створіть папку для проекту на диску.
3. Відкрийте середовище VS Code.
4. Відкрийте вкладку Extensions (Ctrl + Shift + X) і переконайтеся, що у VS Code встановлено модуль розширення під назвою “Python”.
5. Відкрийте створену на кроці 2 папку, вибравши її з меню “File” -> “Open Folder”.
6. Створіть новий Python модуль (файл з розширенням .py), вибравши в меню “File” -> “New File” -> “Python file” і вказавши його ім’я.
7. Скопіюйте код демонстраційної програми, що наведена в методичних вказівках, у новостворений модуль і збережіть його.
8. Створіть віртуальне середовище Python, натиснувши Ctrl+Shift+P та вибравши зі списку команду “Python: Create Environment” -> “Venv” -> <шлях до встановленого інтерпретатора Python>. Після її виконання має з’явитися папка .venv у вашій робочій папці і вибраний інтерпретатор з віртуального середовища у нижньому правому куті рядка стану VS Code.
9. Дослідіть демонстраційну програму у відлагоджувачі.
10. Створіть власний модуль. Для цього повторіть крок 6 і збережіть новостворений модуль у робочій папці.
11. У створеному файлі напишіть програму згідно завдання.
12. Запустіть програму на виконання. Для цього слід вибрати підпункт меню “Run” -> “Start Without Debugging”.
13. Встановіть точки переривання, запусіть налагоджувач (“Run” -> “Start Debugging”) та покроково дослідіть процес виконання програми.

## ДЕМОНСТРАЦІЙНА ПРОГРАМА

```
#####
# Copyright (c) 2023 Lviv Polytechnic National University. All Rights Reserved.
#
# This program and the accompanying materials are made available under the terms
# of the Academic Free License v. 3.0 which accompanies this distribution, and is
# available at https://opensource.org/license/afl-3-0-php/
#
# SPDX-License-Identifier: AFL-3.0
#####

import sys

rows_num = int(input("Введіть розмір квадратної матриці: "))
lst = []
filler = input("Введіть символ-заповнювач: ")

for i in range(rows_num):
    lst.append([])
    for j in range(i+1):
        if len(filler) == 1:
            lst[i].append(ord(filler))
            print(chr(lst[i][j]), end=" ")
        elif len(filler) == 0:
            print("Не введено символ-заповнювач")
            sys.exit(1)
        else:
            print("Забагато символів-заповнювачів")
            sys.exit(1)
    print()
```

НАВЧАЛЬНЕ ВИДАННЯ

**ДОСЛІДЖЕННЯ БАЗОВИХ КОНСТРУКЦІЙ МОВИ PYTHON**

**МЕТОДИЧНІ ВКАЗІВКИ**

до лабораторної роботи № 7 з дисципліни “ Кросплатформні засоби програмування ”  
для студентів базового напрямку 6.123 - “Комп’ютерна інженерія”

**Укладач**

Олексів М. В., к.т.н.

**Редактор**

**Комп’ютерне верстання**

Здано у видавництво . Підписано до друку  
Формат 70х100/16. Папір офсетний. Друк на різнографі  
Умовн. друк. арк. Обл.-вид. арк..  
Тираж прим. Зам..

Видавництво Національного університету “Львівська політехніка”  
*Реєстраційне свідоцтво ДК №751 від 27.12.2001 р.*

Поліграфічний центр Видавництва  
Національного університету “Львівська політехніка”

*Вул. Ф. Колесси, 2. Львів, 79000*