Міністерство освіти і науки України

Державний університет ,,Житомирська політехніка”

Кафедра ІПЗ

Група: ІПЗ-23-1

Програмування мовою Python

Лабораторна робота №5

«ФУНЦІЇ»

Виконав: Риженко Я.В.

Прийняв: Желізко В. В.

**Мета роботи:** ознайомитися основами функціонального програмування і

використання користувацьких функцій в мові Python

**Хід роботи**

**Завдання 1.** Користувач вводить дві сторони трьох прямокутників. Вивести їх

площі.

**Листинг програми:**

for i in range(3):  
 a = float(input(f"Введіть першу сторону прямокутника {i + 1}: "))  
 b = float(input(f"Введіть другу сторону прямокутника {i + 1}: "))  
 print(f"Площа прямокутника {i + 1}: {round(a\*b,3)}")

**Результат виконання:**

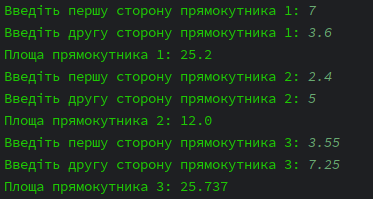
****

Рис. 1. Результат.

**Завдання 2.** Дано катети двох прямокутних трикутників. Написати функцію

обчислення довжини гіпотенузи цих трикутників. Порівняти і вивести яка з

гіпотенуз більше, а яка менше.

**Листинг програми:**

import math  
  
def hypotenuse(a, b):  
 return math.sqrt(a\*\*2 + b\*\*2)  
  
cathetus1 = float(input("Введіть перший катет для першого трикутника: "))  
cathetus2 = float(input("Введіть другий катет для першого трикутника: "))  
hyp1 = round(hypotenuse(cathetus1, cathetus2), 3)  
  
cathetus3 = float(input("Введіть перший катет для другого трикутника: "))  
cathetus4 = float(input("Введіть другий катет для другого трикутника: "))  
hyp2 = round(hypotenuse(cathetus3, cathetus4), 3)  
  
print(f"Гіпотенуза першого трикутника: {hyp1}")  
print(f"Гіпотенуза другого трикутника: {hyp2}")  
  
if hyp1 > hyp2:  
 print("Гіпотенуза першого трикутника більша.")  
elif hyp1 < hyp2:  
 print("Гіпотенуза другого трикутника більша.")  
else:  
 print("Гіпотенузи однакові.")

**Результат виконання:**

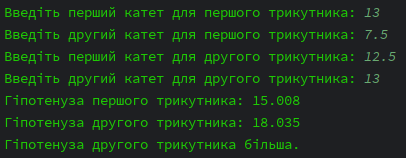
****

Рис. 2. Результат.

**Завдання 3.** Задано коло (x-a)2 + (y-b)2 = R2 і точки Р (р1, р2), F (f1, f1), L (l1, l2).

З'ясувати і вивести на екран, скільки точок лежить всередині кола. Перевірку,

чи лежить точка всередині кола, оформити у вигляді функції.

**Листинг програми:**

def is\_inside\_circle(x, y, a, b, R):  
 return (x - a)\*\*2 + (y - b)\*\*2 < R\*\*2  
  
a = float(input("Введіть координату a центру кола: "))  
b = float(input("Введіть координату b центру кола: "))  
R = float(input("Введіть радіус кола R: "))  
  
points = [  
 ('P', float(input("Введіть р1: ")), float(input("Введіть р2: "))),  
 ('F', float(input("Введіть f1: ")), float(input("Введіть f2: "))),  
 ('L', float(input("Введіть l1: ")), float(input("Введіть l2: ")))  
]  
  
count\_inside = 0  
for name, x, y in points:  
 if is\_inside\_circle(x, y, a, b, R):  
 print(f"Точка {name} знаходиться всередині кола")  
 count\_inside += 1  
 else:  
 print(f"Точка {name} не знаходиться всередині кола")  
  
print(f"Кількість точок, що лежать всередині кола: {count\_inside}")

**Результат виконання:**

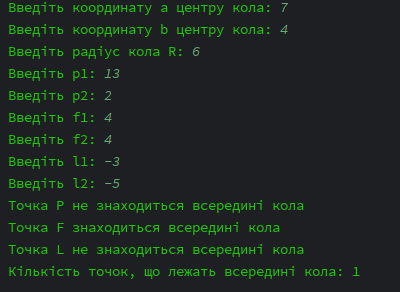
****

Рис. 3. Результат.

**Завдання 4.** Дано числа X, Y, Z, Т - довжини сторін чотирикутника. Обчислити

його площу, якщо кут між сторонами довжиною X і Y - прямий.

**Листинг програми:**

import math  
  
  
def calculate\_quadrilateral\_area(X, Y, Z, T):  
 # Площа першого трикутника зі сторонами X і Y (прямокутний трикутник)  
 area1 = 0.5 \* X \* Y  
 # Обчислення діагоналі між точками X і Y за теоремою Піфагора  
 diagonal = math.sqrt(X \*\* 2 + Y \*\* 2)  
  
 # Обчислення площі другого трикутника за формулою Герона  
 s = (diagonal + Z + T) / 2 # Півпериметр трикутника  
 area2 = math.sqrt(s \* (s - diagonal) \* (s - Z) \* (s - T))  
 # Загальна площа чотирикутника  
 total\_area = area1 + area2  
 return total\_area  
  
X = float(input("Введіть довжину X: "))  
Y = float(input("Введіть довжину Y: "))  
Z = float(input("Введіть довжину Z: "))  
T = float(input("Введіть довжину T: "))  
  
print(f"Площа чотирикутника: {round(calculate\_quadrilateral\_area(X, Y, Z, T), 3)}")

**Результат виконання:**

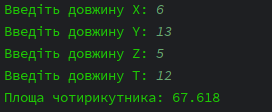
****

Рис. 4. Результат.

**Завдання 5.** Знайти всі натуральні числа, що не перевищують заданого n, які

діляться на кожне із заданих користувачем чисел.

**Листинг програми:**

def find\_divisible\_numbers(n, divisors):  
 return [i for i in range(1, n + 1) if all(i % d == 0 for d in divisors)]  
  
n = int(input("Введіть число n: "))  
divisors = list(map(int, input("Введіть числа, на які повинні ділитися знайдені числа, через пробіл: ").split()))  
  
result = find\_divisible\_numbers(n, divisors)  
print(f"Натуральні числа, що не перевищують {n} і діляться на всі задані числа: {result}")

**Результат виконання:**

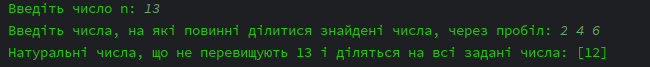
****

Рис. 5. Результат.

**Завдання 6.** Скласти програму для знаходження чисел з інтервалу [М, N], що

мають найбільшу кількість дільників.

**Листинг програми:**

def count\_divisors(num):  
 divisors = 0  
 for i in range(1, num + 1):  
 if num % i == 0:  
 divisors += 1  
 return divisors  
  
def find\_number\_with\_max\_divisors(M, N):  
 max\_divisors = 0  
 numbers = []  
 for i in range(M, N + 1):  
 divisors = count\_divisors(i)  
 if divisors > max\_divisors:  
 max\_divisors = divisors  
 numbers = [i]  
 elif divisors == max\_divisors:  
 numbers.append(i)  
 return numbers, max\_divisors  
  
M = int(input("Введіть початок інтервалу M: "))  
N = int(input("Введіть кінець інтервалу N: "))  
  
numbers, max\_divisors = find\_number\_with\_max\_divisors(M, N)  
print("Числа з найбільшою кількістю дільників:", numbers)  
print(f"Кількість дільників: {max\_divisors}")

**Результат виконання:**

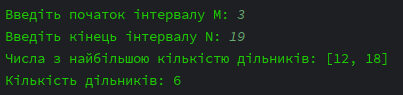
****

Рис. 6. Результат.

**Завдання 7.** Написати функцію для пошуку всіх простих чисел від 0 до N з

можливістю вибору формату представлення результату (списком; рядками в

стовпчик; просто вивести кількість простих чисел.

**Листинг програми:**

def is\_prime(n):  
 if n < 2:  
 return False  
 for i in range(2, int(n \*\* 0.5) + 1):  
 if n % i == 0:  
 return False  
 return True  
  
def find\_primes(N, format\_type='list'):  
 primes = [i for i in range(2, N + 1) if is\_prime(i)]  
  
 if format\_type == 'list':  
 print(" ".join(map(str, primes)))  
 elif format\_type == 'column':  
 for prime in primes:  
 print(prime)  
 elif format\_type == 'count':  
 print(f"Кількість простих чисел: {len(primes)}")  
  
N = int(input("Пошук всіх простих чисел від 0 до "))  
format\_type = input("Виберіть формат виведення результату (list/column/count): ")  
find\_primes(N, format\_type)

**Результат виконання:**

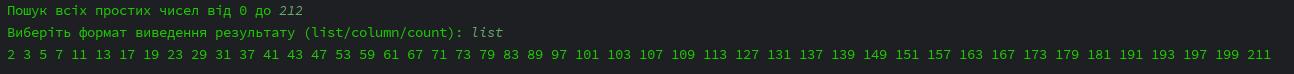
****

Рис. 7. Результат.

**Завдання 8.** Дано список з випадкових натуральних чисел довільної довжини.

Написати програму, що формуватиме з заданого другий список, що міститиме

тільки значення від MIN+bottom до MAX-upper. Де MIN і MAX – відповідно

найменше і найбільше число в списку, а botton і upper – нижня і верхня межа

значень вибірки нового списку. Програма має містити обробку винятків на

випадок введення символів невірного типу, дробових чисел, вихід за межі

мінімального і максимального значення.

**Листинг програми:**

def filter\_list(nums, bottom, upper):  
 try:  
 MIN = min(nums)  
 MAX = max(nums)  
 result = [num for num in nums if MIN + bottom <= num <= MAX - upper]  
 return result  
 except ValueError:  
 print("Список не може бути порожнім")  
 return []  
  
try:  
 nums = list(map(int, input("Введіть список чисел (через пробіл): ").split()))  
 bottom = int(input("Введіть нижню межу bottom: "))  
 upper = int(input("Введіть верхню межу upper: "))  
  
 filtered\_nums = filter\_list(nums, bottom, upper)  
 print("Отриманий відфільтрований список:", filtered\_nums)  
except ValueError:  
 print("Введено неправильні дані")

**Результат виконання:**

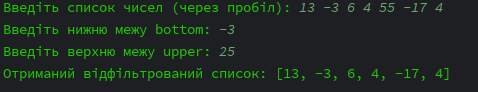
****

Рис. 8. Результат.

**Завдання 9.** Для завдань 6 – 8 написати декоратор, що дозволить визначати час

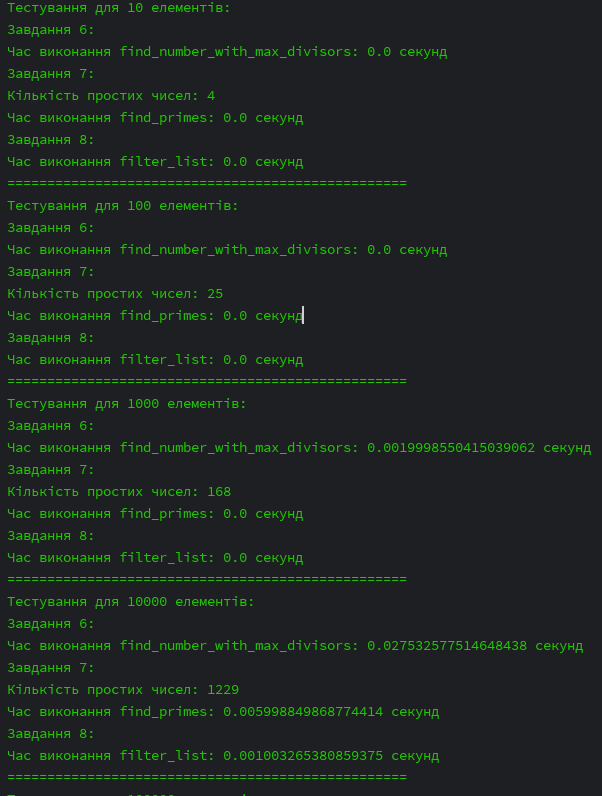
виконання програми. Виконати перевірку часу виконання написаних функцій

для 10\*\*n елементів при n <=6 з кроком в n. Тобто визначити час виконання функцій для десятків, сотень, тисяч і так до мільйону елементів.

**Листинг програми:**

import time  
import random  
import math  
  
# Декоратор для вимірювання часу виконання  
def timer(func):  
 def wrapper(\*args, \*\*kwargs):  
 start\_time = time.time()  
 result = func(\*args, \*\*kwargs)  
 end\_time = time.time()  
 print(f"Час виконання {func.\_\_name\_\_}: {end\_time - start\_time} секунд")  
 return result  
 return wrapper  
  
# Завдання 6: Пошук числа з найбільшою кількістю дільників  
def count\_divisors(num):  
 divisors = 0  
 for i in range(1, int(math.sqrt(num)) + 1):  
 if num % i == 0:  
 divisors += 1  
 if i != num // i:  
 divisors += 1  
 return divisors  
  
@timer  
def find\_number\_with\_max\_divisors(M, N):  
 max\_divisors = 0  
 numbers = []  
 for i in range(M, N + 1):  
 divisors = count\_divisors(i)  
 if divisors > max\_divisors:  
 max\_divisors = divisors  
 numbers = [i]  
 elif divisors == max\_divisors:  
 numbers.append(i)  
 return numbers, max\_divisors  
  
# Завдання 7: Пошук простих чисел  
def is\_prime(n):  
 if n < 2:  
 return False  
 for i in range(2, int(n \*\* 0.5) + 1):  
 if n % i == 0:  
 return False  
 return True  
  
@timer  
def find\_primes(N, format\_type='list'):  
 primes = [i for i in range(2, N + 1) if is\_prime(i)]  
  
 if format\_type == 'list':  
 print(" ".join(map(str, primes)))  
 elif format\_type == 'column':  
 for prime in primes:  
 print(prime)  
 elif format\_type == 'count':  
 print(f"Кількість простих чисел: {len(primes)}")  
  
# Завдання 8: Фільтрація списку чисел  
@timer  
def filter\_list(nums, bottom, upper):  
 try:  
 MIN = min(nums)  
 MAX = max(nums)  
 result = [num for num in nums if MIN + bottom <= num <= MAX - upper]  
 return result  
 except ValueError:  
 print("Список не може бути порожнім")  
 return []  
  
# Тестування для різних значень n  
for n in range(1, 7):  
 size = 10 \*\* n  
 print(f"Тестування для {size} елементів:")  
  
 # Тест для завдання 6  
 M = 1  
 N = size  
 print("Завдання 6:")  
 find\_number\_with\_max\_divisors(M, N)  
  
 # Тест для завдання 7  
 print("Завдання 7:")  
 find\_primes(size, 'count')  
  
 # Тест для завдання 8  
 nums = random.sample(range(1, size + 1), size)  
 bottom, upper = random.randint(1, size // 10), random.randint(1, size // 10)  
 print("Завдання 8:")  
 filter\_list(nums, bottom, upper)  
  
 print("=" \* 50)

**Результат виконання:**

****

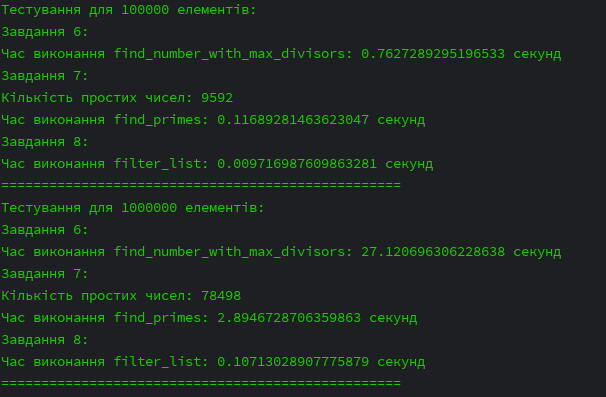
****

Рис. 9. Результат.

**Висновок:** У цій лабораторній роботі ми ознайомились з функціями в мові Python.

***Посилання на Git****: :* *https://github.com/JanRizhenko/Python\_Labs*