**Звіт про виконання практичних завдань до лекцій з курсу Технології програмування на мові Python**

Звіт до Теми №1

**Функції та змінні**

Під час виконання практичного завдання до Теми №1 було надано варіанти рішення до наступних задач:

Завдання 1:

**Перетворення рядка**

Необхідно рядок, що має вигляд "abcdefg123" перетворити наступним чином "321gfedcba", вважаючи сталою довжину рядку в 10 символів.

Хід виконання завдання:

Спочатку створив змінну «str» і надав їй значення «abcdefg123». Потім за допомогою команди «str[:10][::-1]» розвернув рядок і зберіг його довжину в 10 символів.

Текст програми:

str = "abcdefg123"

print(str[:10][::-1])

Посилання на github: <https://github.com/dmitrijpryhodko/TP-KB-221-Dmytrii-Pryhodko/blob/main/topic_01/task1.py>

Знімок екрану з посилання на github:

str = "abcdefg123"

print(str[:10][::-1])

Завдання 2:

**Розібрати методи в Python**

Необхідно створити код, який буде виводити деякі методи в Python.

Хід виконання завдання:

Спочатку я створив змінну, а потім прописав потрібний метод. Так я зробив декілька разів, застосовуючи кожного разу різний метод.

Текст програми:

a = "wElcOMe to School"

print(a.capitalize())

#.capitalize() - перетворює рядок так, щоб він починався з букви у верхньому регістрі, а решта букв переходила у нижній регістр

sometext = "wElcOMe to schOoL"

print(sometext.title())

#title() - пише кожне слово у рядку з великої літери

txt = " Welcome to school "

print(txt.strip())

#strip() - видаляє пробіли перед і після рядка

b = "welcome to school"

print(b.upper())

#upper - переводить всі символи рядка у верхній регістр

с = "WELCOME To scOOL"

print(b.lower())

#lower - переводить всі символи рядка у нижній регістр

list = ['cherry', 'melon', 'apple', 'pear']

list.append('lemon')

print(list)

#append - додає елемент в кінець списку

list = ['cherry', 'melon', 'apple', 'pear','apple']

x = list.count('apple')

print(x)

#count - рахує скільки разів зустрічається певний елемент в списку

list = ['cherry', 'melon', 'pear', 'apple']

list.insert(2, 'lemon')

print(list)

#insert - додає елемент до списку по індексу

Посилання на github: <https://github.com/dmitrijpryhodko/TP-KB-221-Dmytrii-Pryhodko/blob/main/topic_01/task2.py>

Знімок екрану з посилання на github:

a = "wElcOMe to School"

print(a.capitalize())

#.capitalize() - перетворює рядок так, щоб він починався з букви у верхньому регістрі, а решта букв переходила у нижній регістр

sometext = "wElcOMe to schOoL"

print(sometext.title())

#title() - пише кожне слово у рядку з великої літери

txt = "   Welcome to school  "

print(txt.strip())

#strip() - видаляє пробіли перед і після рядка

b = "welcome to school"

print(b.upper())

#upper - переводить всі символи рядка у верхній регістр

с = "WELCOME To scOOL"

print(b.lower())

#lower - переводить всі символи рядка у нижній регістр

list = ['cherry', 'melon', 'apple', 'pear']

list.append('lemon')

print(list)

#append - додає елемент в кінець списку

list = ['cherry', 'melon', 'apple', 'pear','apple']

x = list.count('apple')

print(x)

#count - рахує скільки разів зустрічається певний елемент в списку

list = ['cherry', 'melon', 'pear', 'apple']

list.insert(2, 'lemon')

print(list)

#insert - додає елемент до списку по індексу

Завдання 3:

**Знайти дискримінант**

Потрібно написати код, який буде вираховувати дискримінант.

Хід виконання завдання:

Спочатку я створив функцію «findD», а потім в цю функцію додав змінну, яка має значення формули дискримінанта. Потім створив ще 3 змінні, які будуть приймати необхідні для розрахунку значення. Ще створив змінну «D» яка буде виводити кінцевий результат. І написав функцію виводу на екран.

Текст програми:

def findD(a, b, c):

discriminant = b\*\*2 - 4 \* a \* c

return discriminant

# Приклад використання:

a = int(input("Please enter start point: "))

b = int(input("Please end point: "))

c = int(input("please enter mult: "))

D = findD(a, b, c)

print(f"Дискримінант рівняння дорівнює {D}")

Посилання на github: <https://github.com/dmitrijpryhodko/TP-KB-221-Dmytrii-Pryhodko/blob/main/topic_01/task3.py>

Знімок екрану з посилання на github:

def findD(a, b, c):

    discriminant = b\*\*2 - 4 \* a \* c

    return discriminant

# Приклад використання:

a = int(input("Please enter start point: "))

b = int(input("Please end point: "))

c = int(input("please enter mult: "))

D = findD(a, b, c)

print(f"Дискримінант рівняння дорівнює {D}")

Висновок: На даному практичному завданні я розібрався з деякими функціями та змінними в Python та навчився розміщувати файли на Github.

**Звіт про виконання практичних завдань до лекцій з курсу Технології програмування на мові Python**

Звіт до теми №2

**Умовний перехід**

Під час виконання практичного завдання до Теми №2 було надано варіанти рішення до наступних задач:

**Завдання 1:** Написати функцію пошуку коренів квадратного рівняння використовуючи функцію розрахунку дискримінанту з попередньої теми та умовні переходи.

Хід виконання завдання

Функція «find\_roots» використовує умовні переходи для підрахунку кількості та значень коренів квадратного рівняння, а також використовує функцію «findD» для розрахунку дискримінанта. Якщо дискримінант більше нуля, то виконується пошук коренів, якщо дисримінант дорівнює нулю, то тоді виконується пошук одного кореня. У інших випадках програма виведе напис: «Немає дійсних коренів».

Код програми

import math

def findD(a, b, c):

discriminant = b\*\*2 - 4 \* a \* c

return discriminant

def find\_roots(a, b, c):

discriminant = findD (a, b, c)

if discriminant > 0:

# Два різних корені

root1 = (-b + math.sqrt(discriminant)) / (2\*a)

root2 = (-b - math.sqrt(discriminant)) / (2\*a)

return (root1, root2)

elif discriminant == 0:

# Один подвійний корінь

root1 = -b / (2\*a)

return (root1,)

else:

# Немає дійсних коренів

return None

a = int(input("Введіть значення a: "))

b = int(input("Введіть значення b: "))

c = int(input("Введіть значення c: "))

roots = find\_roots(a, b, c)

if roots:

if len(roots) == 2:

print(f"Два корені: {roots[0]} та {roots[1]}")

else:

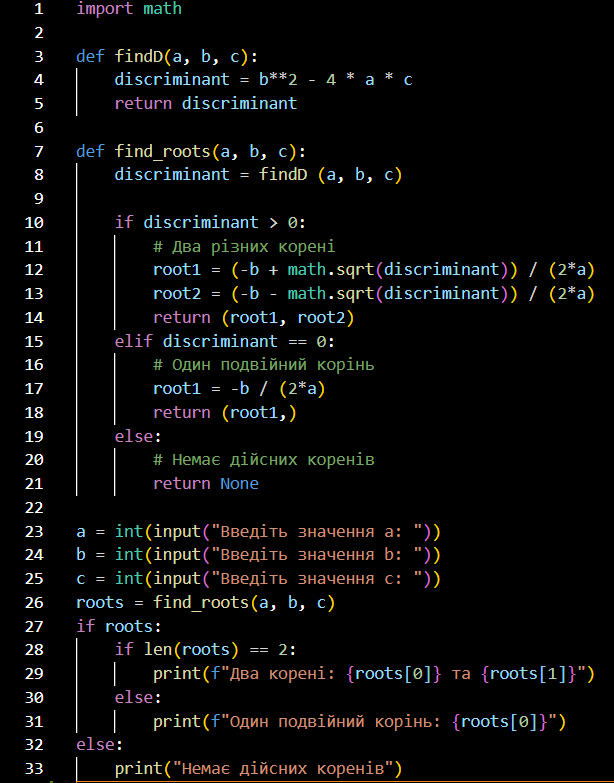
print(f"Один подвійний корінь: {roots[0]}")

else:

print("Немає дійсних коренів")

Посилання на github: <https://github.com/dmitrijpryhodko/TP-KB-221-Dmytrii-Pryhodko/blob/main/topic_02/task1.py>

Скріншот коду програми



**Завдання 2:** Написати програму калькулятор використовуючи **if else** конструкцію.

Хід виконання завдання

Спочатку створив змінні, які будуть приймати значення першого та другого числа. У нашому випадку це змінна «a» і «b». Потім створив змінну, яка буде приймати дію, яку потрібно виконати над числами. Далі створив 4 функції, для кожної з дій. Це додавання, віднімання, ділення та множення. І за допомого конструкцій if та elif задав коли необхідно буде виконувати які дії.

Код програми

a = int(input("Enter a: "))

b = int(input("Enter b: "))

operation = input("Operation: ")

def sum(a, b):

return a + b

def min(a, b):

return a - b

def div(a, b):

if b != 0:

return a / b

if b == 0:

print("На нуль не ділиться")

def mult(a, b):

return a \* b

match operation:

case "+":

result = sum(a , b)

case "-":

result = min (a , b)

case "/":

result = div (a , b)

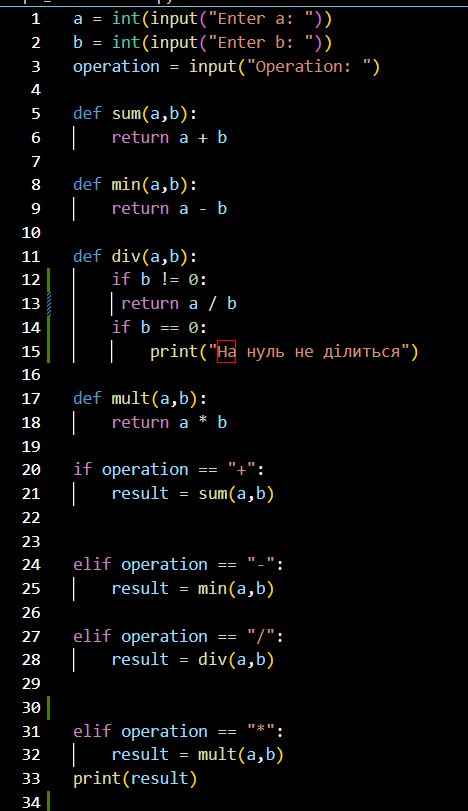
case "\*":

result = mult (a , b)

print(result)

Посилання на github: <https://github.com/dmitrijpryhodko/TP-KB-221-Dmytrii-Pryhodko/blob/main/topic_02/task2.py>

Скріншот коду програми



**Завдання 3:** Написати програму калькулятор використовуючи **match** конструкцію

Хід виконання завдання

Спочатку створив змінні, які будуть приймати числа та дії над ними. Потім використовуючи конструкцію **match** написав які дії потрібно виконувати при виборі тієї чи іншої математичної операції.

Код програми

a = int(input("Enter a: "))

b = int(input("Enter b: "))

operation = input("Operation: ")

def sum(a, b):

return a + b

def min(a, b):

return a - b

def div(a, b):

if b != 0:

return a / b

if b == 0:

print("На нуль не ділиться")

def mult(a, b):

return a \* b

match operation:

case "+":

result = sum(a , b)

case "-":

result = min (a , b)

case "/":

result = div (a , b)

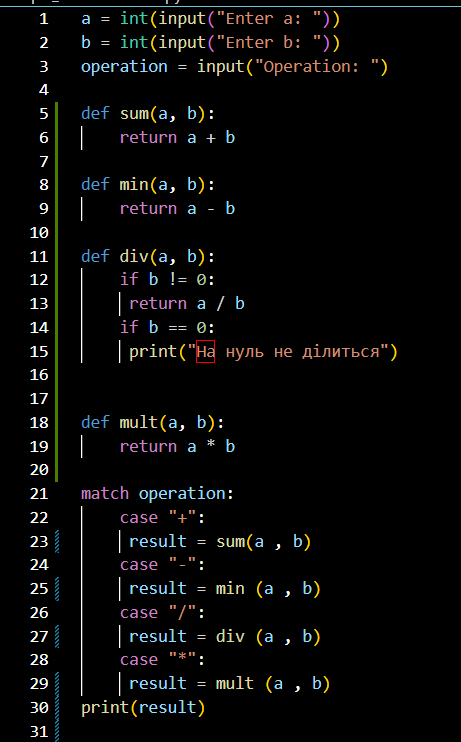
case "\*":

result = mult (a , b)

print(result)

Посилання на github: <https://github.com/dmitrijpryhodko/TP-KB-221-Dmytrii-Pryhodko/blob/main/topic_02/task3.py>

Скріншот коду програми



Висновок: На цьому практичному завданні я створив програму для пошуку квадратних коренів та калькулятор, застосовуючи 2 різних способи. Вивчив конструкцію **match.**

Звіт до теми №3

**Завдання 1:** Написати програму калькулятор з постійними запитами на введення нових даних та операцій. За основу взяти програму калькулятор з попередньої теми. Реалізувати механізм завершення програми після отримання відповідної команди.

Хід виконання завдання

Спочатку написав функції для кожної дії. Потім написав цикл «while», який буде запитувати у користувача про елементи прикладу, дії над ними та вихід. Написав конструкцію для виконання математичних операцій та вивів результат.

Код програми

def add(a, b):

return a + b

def min(a, b):

return a - b

def mult(a, b):

return a \* b

def div(a,b):

if b != 0:

return a / b

if b == 0:

print("На нуль не ділиться")

return None

while True:

exit = str(input("Enter 'q' for quit or 'Enter' for continue: "))

if exit == 'q':

break

a = float(input("Enter a: "))

b = float(input("Enter b: "))

operation = input("Operation (+, -, \*, /): ")

if operation == "+":

result = add(a, b)

elif operation == "-":

result = min(a, b)

elif operation == "/":

result = div(a, b)

elif operation == "\*":

result = mult(a, b)

else:

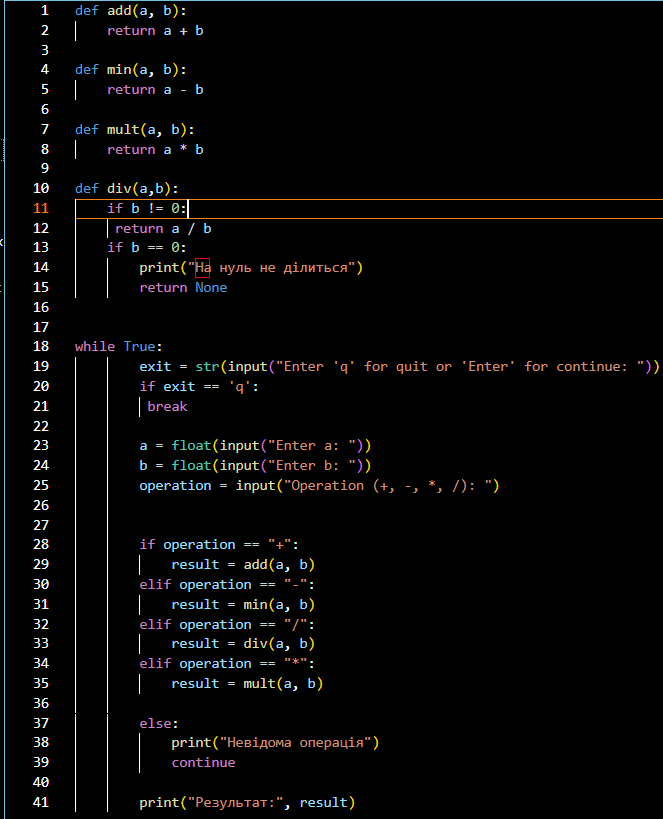
print("Невідома операція")

continue

print("Результат:", result)

Посилання на github: <https://github.com/dmitrijpryhodko/TP-KB-221-Dmytrii-Pryhodko/blob/main/topic_03/task1.py>

Скріншот коду програми



**Завдання 2:** Написати програму тестування функцій списків таких як: extend(), append(), insert(id, val), remove(val), clear(), sort(), reverse(), copy()

Хід виконання завдання

Написав приклад із застосуванням кожної функції та пояснив що робить кожна з них.

Код програми

#1

students1 = ["Emma", "Jon", "Bob"] #студенти першої групи

students2 = ["Stiven", "Jake", "Lara"] #студенти другогї групи

students1.extend(students2)

print(students1)

#extend() - додає вказані елементи списку у кінець поточного списку

#2

students1 = ["Emma", "Jon", "Bob"]

students1.append("Tailer")

print(students1)

#append() - додає елемент до списку

#3

students1 = ["Emma", "Jon", "Bob"]

students1.insert(1, "Tailer")

print(students1)

#insert() - додає елемент до списку за індексом

#4

fruits = ["apple", "orange", "lemon", "cherry"]

fruits.remove("orange")

print(fruits)

#remove() - видаляє елемент з списку

#5

fruits = ["apple", "orange", "lemon", "cherry"]

fruits.clear()

print(fruits)

#clear() - повністю видаляє список

#6

def myFunc(e):

return len (e)

animals = ["bird", "tiger", "cat", "leo"]

animals.sort(key=myFunc)

print(animals)

animals.sort(reverse= True, key=myFunc)

print(animals)

#sort() - сортує список. у нашому випадку сортування відбулося за довжиною і в оберненому порядку

#7

fruits = ["apple", "orange", "lemon", "cherry"]

fruits.reverse()

print(fruits)

#reverse() - розвертає список

#8

fruits = ["apple", "orange", "lemon", "cherry"]

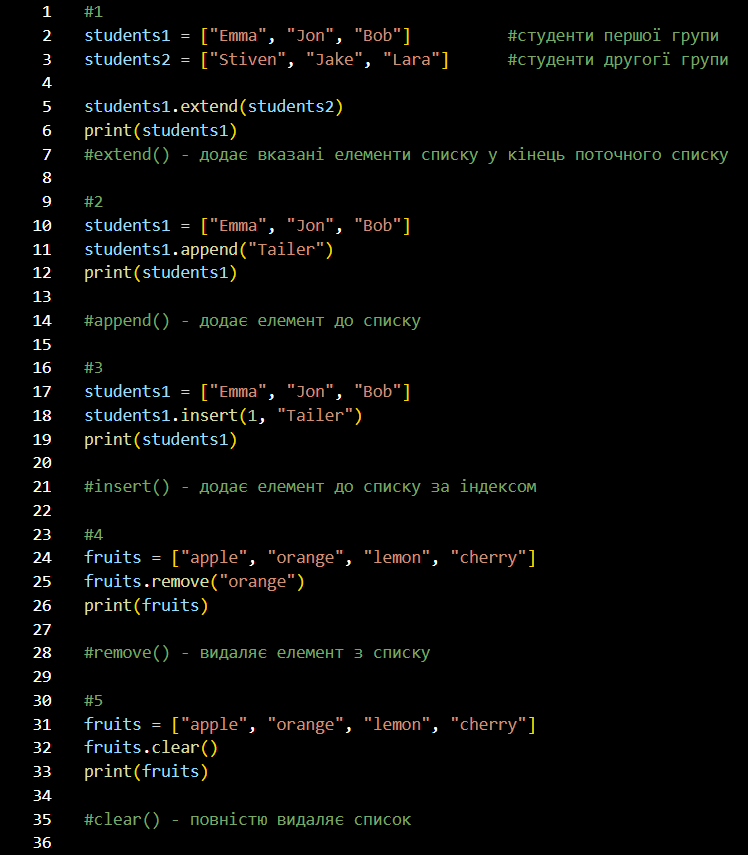
a = fruits.copy()

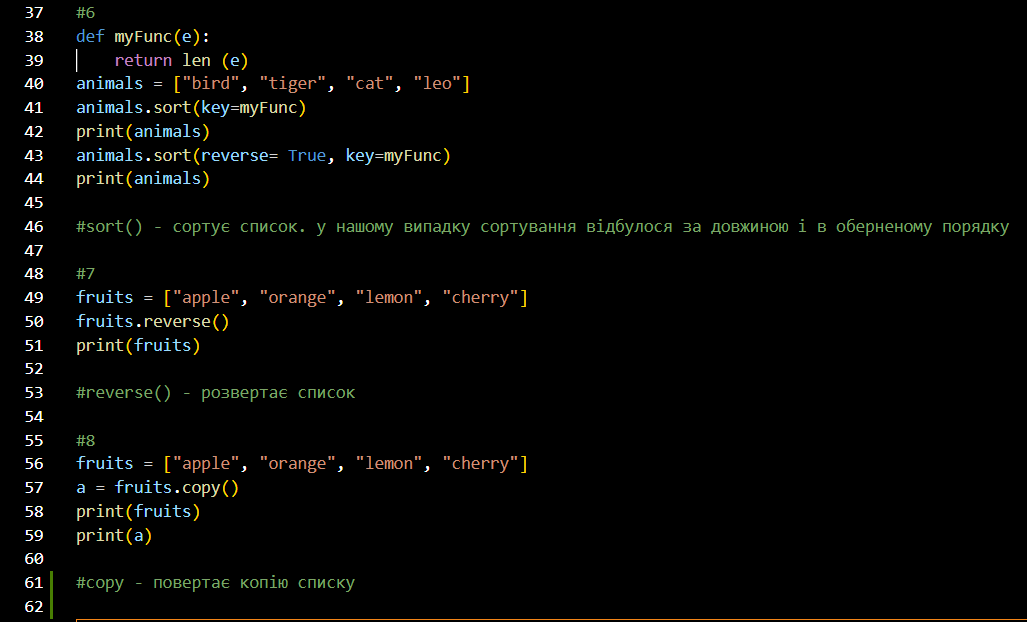
print(fruits)

print(a)

Посилання на github: <https://github.com/dmitrijpryhodko/TP-KB-221-Dmytrii-Pryhodko/blob/main/topic_03/task2.py>

Скріншот коду програми





**Завдання 3:** Написати програму тестування функцій словників таких як: update(), del(), clear(), keys(), values(), items()

Хід виконання завдання

У даній програмі я написав приклади з використанням кожної з функцій

Код програми

#1

student = {'Name': 'Dima', 'Age': 18}

student.update({'Course': 2})

print(student)

#update() - вставляє вказані елементи до словника

#2

student = {'Name': 'Dima', 'Age': 18, 'Surname': 'Pryhod`ko'}

del student['Age']

print(student)

#del() - видаляє елемент із словника

#3

student = {'Name': 'Dima', 'Age': 18, 'Surname': 'Pryhod`ko'}

student.clear()

print(student)

#clear() - видаляє весь словник

#4

student = {'Name': 'Dima', 'Age': 18, 'Surname': 'Pryhod`ko'}

x = student.keys()

print(x)

#keys() - повертає об'єкт представлення

#5

student = {'Name': 'Dima', 'Age': 18, 'Surname': 'Pryhod`ko'}

x = student.values()

print(x)

#values() - повертає дані зі словника

#6

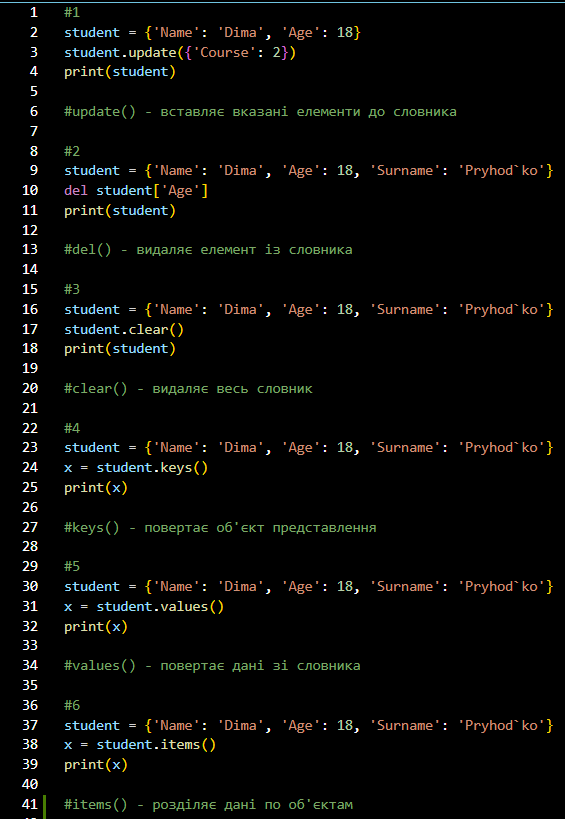
student = {'Name': 'Dima', 'Age': 18, 'Surname': 'Pryhod`ko'}

x = student.items()

print(x)

Посилання на github: <https://github.com/dmitrijpryhodko/TP-KB-221-Dmytrii-Pryhodko/blob/main/topic_03/task3.py>

Скріншот коду програми



**Завдання 4:** Маючи відсортований список, написати функцію пошуку позиції для вставки нового елементу в список.

Хід виконання завдання

Ми оголошуємо функцію find\_pos, яка приймає два параметри: list - це відсортований список, в якому ми будемо шукати позицію для вставки нового елементу. target - це новий елемент, позицію для вставки якого ми хочемо знайти. Ми ініціалізуємо дві змінні left і right, які вказують на лівий та правий краї відсортованого списку. На початку left вказує на перший елемент (індекс 0), а right вказує на останній елемент (індекс len(list) - 1). Ми використовуємо цикл while, щоб виконувати бінарний пошук. Цей цикл продовжується, доки left не стає більшим або рівним right. У кожній ітерації циклу ми обчислюємо середній індекс mid між left і right.Ми порівнюємо значення в середині списку list[mid] з цільовим значенням target. Якщо list[mid] = target, це означає, що елемент вже присутній у списку, і ми повертаємо індекс mid. Якщо list[mid] менше за target, то ми зміщуємо ліву межу left вправо до mid + 1, тим самим ігноруючи ліву частину списку. Якщо list[mid] більше за target, то ми зміщуємо праву межу right вліво до mid - 1, тим самим ігноруючи праву частину списку. Після завершення циклу бінарного пошуку, ми повертаємо значення left, оскільки це і є позиція для вставки нового елементу у відсортований список. У головній частині коду ми задаємо відсортований список list та новий елемент new\_element, який ми хочемо вставити. Ми викликаємо функцію find\_pos для пошуку позиції для вставки new\_element у list і отримуємо позицію в змінну position. Ми використовуємо метод insert() для вставки new\_element на знайдену позицію у list. Нарешті, ми виводимо відсортований список після вставки нового елементу, щоб побачити результат. Цей код допомагає знайти позицію для вставки нового елементу у відсортований список і вставляє його, зберігаючи впорядкованість списку.

Код програми

def find\_pos(list, target):

left, right = 0, len(list) - 1

while left <= right:

mid = (left + right) // 2

if list[mid] == target:

return mid # Елемент вже присутній у списку

elif list[mid] < target:

left = mid + 1

else:

right = mid - 1

return left # Позиція для вставки нового елементу

list = [1, 3, 5, 7, 9]

new\_element = int(input("Введіть новий елемент: "))

position = find\_pos(list, new\_element)

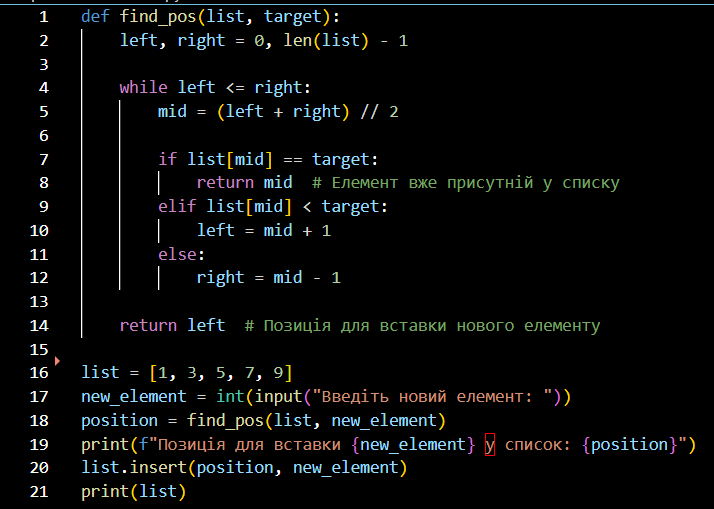
print(f"Позиція для вставки {new\_element} у список: {position}")

list.insert(position, new\_element)

print(list)

Посилання на github: <https://github.com/dmitrijpryhodko/TP-KB-221-Dmytrii-Pryhodko/blob/main/topic_03/task4.py>

Скріншот коду програми



**Висновок:** На даному практичному завданні я «модернізував» програму «калькулятор», яку я робив на минуле завдання. Розібрався з деякими функціями списку і словника. Та написав програму сортування.

Звіт до теми №4

Задачі: Розширити програму калькулятор функцією запитів даних для виконання операцій від користувача, що обробляє виняткові ситуації та розширити функцію ділення обробкою виняткової ситуації ділення на нуль.

Хід роботи

До вже створеної програми «Калькулятор» необхідно було додати винятки. В дану програму я додав виняток «ValueError» та «ZeroDivisionError». Це було зроблено для того, щоб коли користувач вводитиме замість числа щось інше, то замість помилки буде отримувати повідомлення. З діленням на нуль я зробив аналогічні дії.

Код програми

def add(a, b):

return a + b

def min(a, b):

return a - b

def mult(a, b):

return a \* b

def div(a,b):

try:

return a / b

except ZeroDivisionError:

print("Ділення на нуль неможливе")

return None

while True:

exit = str(input("Enter 'q' for quit or 'Enter' for continue: "))

if exit == 'q':

break

try:

a = float(input("Enter a: "))

b = float(input("Enter b: "))

operation = input("Operation (+, -, \*, /): ")

if operation == "+":

result = add(a, b)

elif operation == "-":

result = min(a, b)

elif operation == "/":

result = div(a, b)

elif operation == "\*":

result = mult(a, b)

else:

print("Невідома операція")

continue

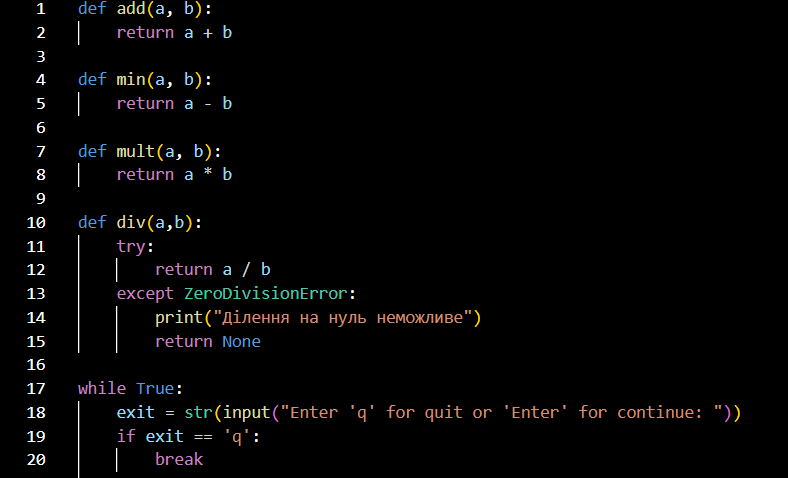
print("Результат:", result)

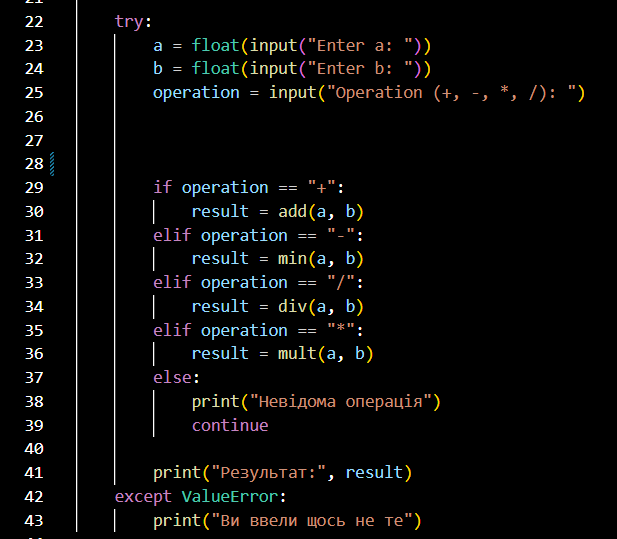
except ValueError:

print("Ви ввели щось не те")

Посилання на github: <https://github.com/dmitrijpryhodko/TP-KB-221-Dmytrii-Pryhodko/blob/main/topic_04/task1.py>

Скріншот коду програми





**Висновок:** На даному практичному завданні я розібрався з деякими винятками в Python та додав їх до своєї раніше створеної програми «Калькулятор».