**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:

**Перетворення рядка**

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

Хід виконання завдання:

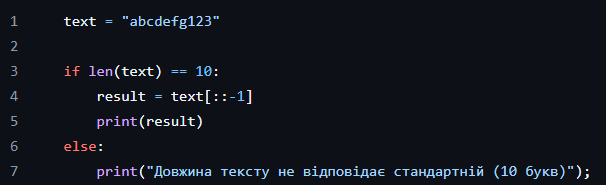
1. Запускаю Visual Studio Code та ознайомлююсь з поставленим завданням.
2. Задаю змінну «text» та присвоюю їй значення «abcdefg123».
3. Тепер використовую функції if та else для виведення лише того тексту, довжина якого становить 10 букв. Для віддзеркалення тексту використовую синтаксис text[::-1]. Якщо текст задовільняє умову, він записується у зворотному порядку, а якщо ні, то виводиться повідомлення: «Довжина тексту не відповідає стандартній (10 букв)».

Текст програми:

|  |
| --- |
| text = "abcdefg123"  if len(text) == 10:      result = text[::-1]      print(result)  else:      print("Довжина тексту не відповідає стандартній (10 букв)"); |

Посилання на github:

<https://github.com/jekaproc/TP-KB-241-Protsenko-Evgeniy/blob/main/topic_01/task01.py>

 Знімок екрану з посилання на github:

**Тестування методів strip, capitalize, title, upper, lower**

Дослідити методи strip, capitalize, title, upper, lower та застосувати їх до тексту.

Хід виконання завдання:

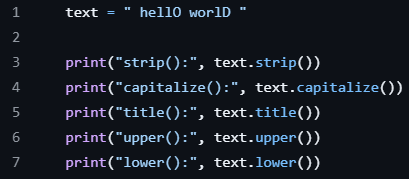
1. Запускаю Visual Studio Code, ознайомлююсь з поставленим завданням та методами для тестування.
2. Задаю змінну «text» та присвоюю їй значення « hellO worlD », поставивши пробіли по боках та записавши останні букви слів з великої літери.
3. Прописую виведення слів з нового рядка навпроти назви використаного метода.

Текст програми:

|  |
| --- |
| text = " hellO worlD "  print("strip():", text.strip())  print("capitalize():", text.capitalize())  print("title():", text.title())  print("upper():", text.upper())  print("lower():", text.lower()) |

Посилання на github:

<https://github.com/jekaproc/TP-KB-241-Protsenko-Evgeniy/blob/main/topic_01/task02.py>

Знімок екрану з посилання на github:

**Зробити функцію для знаходження дискримінанта квадратного рівняння**

Підставити значення коефіцієнтів a, b, c у формулу дискримінанта D=b2-4ac.

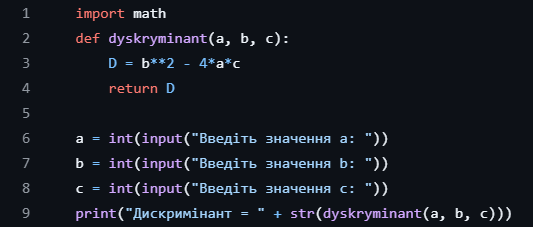
1. Запускаю Visual Studio Code, ознайомлююсь з поставленим завданням та модулем math.
2. Підключаю модуль math для математичних функцій та оголошую функцію dyskryminant (a, b, c) для обчислення дискимінанту.
3. Задаю змінну D з формулою для обчислення.
4. Далі надаю змінні коефіцієнтів (a, b, c) для введення та результат функції для виводу результату.

Текст програми:

|  |
| --- |
| import math  def dyskryminant(a, b, c):      D = b\*\*2 - 4\*a\*c      return D    a = int(input("Введіть значення a: "))  b = int(input("Введіть значення b: "))  c = int(input("Введіть значення c: "))  print("Дискримінант = " + str(dyskryminant(a, b, c))) |

Посилання на github:

<https://github.com/jekaproc/TP-KB-241-Protsenko-Evgeniy/blob/main/topic_01/task03.py>

Знімок екрану з посилання на github:

Звіт до Теми №2

Умовний перехід

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

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

Хід виконання завдання:

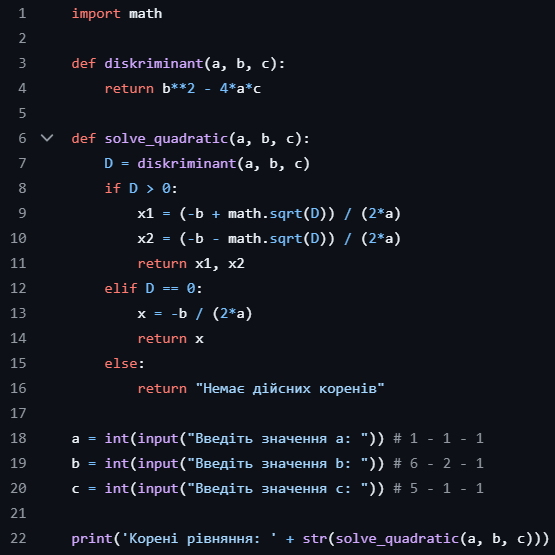
1. Копіюю код для знаходження дискримінанта з завдання попередньої теми.
2. Створюю функцію для пошуку коренів.
3. Додаю три різні сценарії для значення дискримінанта (D=0, D<0, D>0).
4. Надаю змінні коефіцієнтів (a, b, c) для введення їх значень та результат функції для виводу результату.

Текст програми:

|  |
| --- |
| import math  def diskriminant(a, b, c):  return b\*\*2 - 4\*a\*c  def solve\_quadratic(a, b, c):  D = diskriminant(a, b, c)  if D > 0:  x1 = (-b + math.sqrt(D)) / (2\*a)  x2 = (-b - math.sqrt(D)) / (2\*a)  return x1, x2  elif D == 0:  x = -b / (2\*a)  return x  else:  return "Немає дійсних коренів"  a = int(input("Введіть значення a: ")) # 1 - 1 - 1  b = int(input("Введіть значення b: ")) # 6 - 2 - 1  c = int(input("Введіть значення c: ")) # 5 - 1 - 1  print('Корені рівняння: ' + str(solve\_quadratic(a, b, c))) |

Посилання на github:

<https://github.com/jekaproc/TP-KB-241-Protsenko-Evgeniy/blob/main/topic_02/task01.py>

Знімок екрану з посилання на github:

**Написати програму калькулятор використовуючи if else конструкцію. Кожна операція має бути виконана в окремій функції.**

Хід виконання завдання:

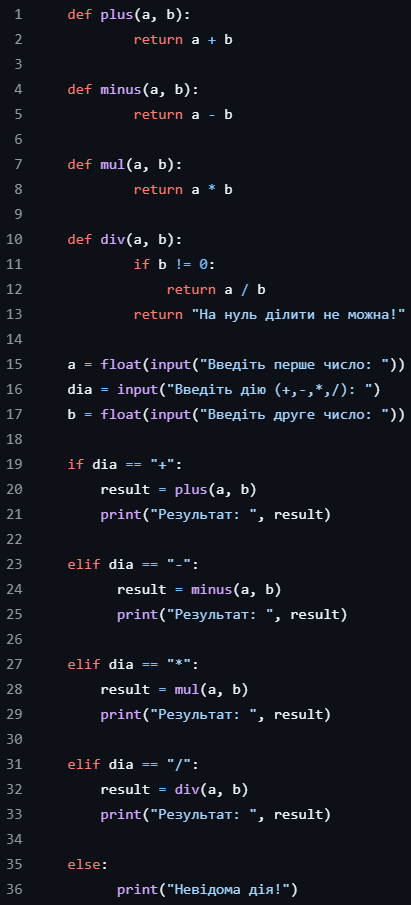
1. Створюю змінні a, b та dia для вводу чисел та операцію, яка повинна виконатись з цими числами (додавання, віднімання, множення або ділення).
2. Додаю функції під кожну дію.
3. Задаю умови під всі можливі дії.
4. Враховую правило ділення на нуль та введення невідомої дії

Текст програми:

|  |
| --- |
| def plus(a, b):  return a + b  def minus(a, b):  return a - b  def mul(a, b):  return a \* b  def div(a, b):  if b != 0:  return a / b  return "На нуль ділити не можна!"  a = float(input("Введіть перше число: "))  dia = input("Введіть дію (+,-,\*,/): ")  b = float(input("Введіть друге число: "))  if dia == "+":  result = plus(a, b)  print("Результат: ", result)  elif dia == "-":  result = minus(a, b)  print("Результат: ", result)  elif dia == "\*":  result = mul(a, b)  print("Результат: ", result)  elif dia == "/":  result = div(a, b)  print("Результат: ", result)  else:  print("Невідома дія!") |

Посилання на github:

<https://github.com/jekaproc/TP-KB-241-Protsenko-Evgeniy/blob/main/topic_02/task02.py>

Знімок екрану з посилання на github:

**Написати програму калькулятор використовуючи match конструкцію. Кожна операція має бути виконана в окремій функції.**

Хід виконання завдання:

1. Копіюю код з минулого завдання.
2. Замінюю умови if-else на match з кейсами на відповідну дію.

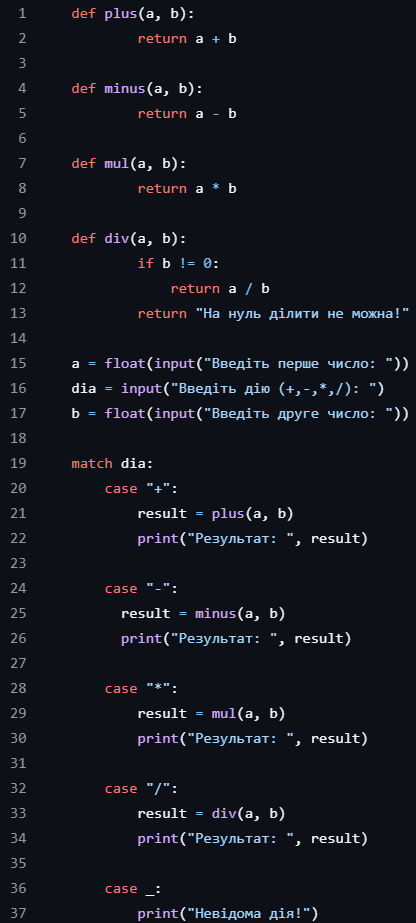
Текст програми:

|  |
| --- |
| def plus(a, b):  return a + b  def minus(a, b):  return a - b  def mul(a, b):  return a \* b  def div(a, b):  if b != 0:  return a / b  return "На нуль ділити не можна!"  a = float(input("Введіть перше число: "))  dia = input("Введіть дію (+,-,\*,/): ")  b = float(input("Введіть друге число: "))  match dia:  case "+":  result = plus(a, b)  print("Результат: ", result)  case "-":  result = minus(a, b)  print("Результат: ", result)  case "\*":  result = mul(a, b)  print("Результат: ", result)  case "/":  result = div(a, b)  print("Результат: ", result)  case \_:  print("Невідома дія!") |

Посилання на github:

<https://github.com/jekaproc/TP-KB-241-Protsenko-Evgeniy/blob/main/topic_02/task03.py>

Знімок екрану з посилання на github:



Звіт до Теми №3

Цикли

Під час виконання практичного завдання до Теми №3 було надано варіанти рішення до наступних задач:

**Написати програму калькулятор з постійними запитами на введення нових даних та операцій. Реалізувати механізм завершення програми після отримання відповідної команди.**

Хід виконання завдання:

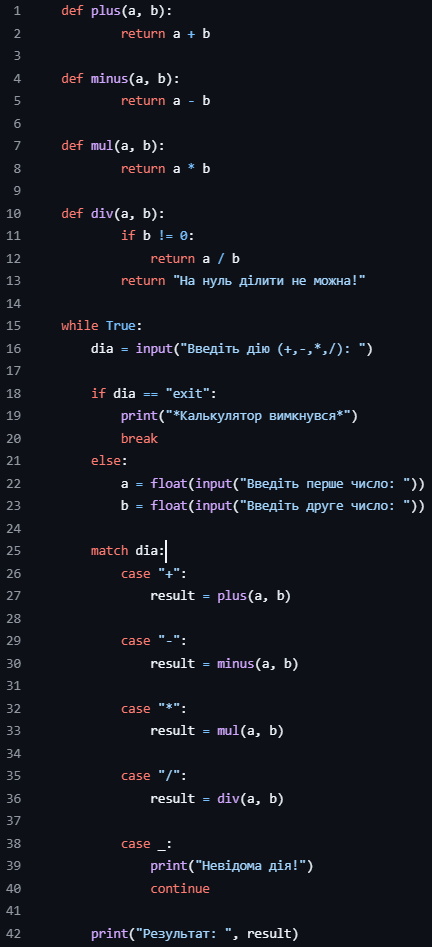
1. Копіюю код програми калькулятора із завдання попередньої теми.
2. Охоплюю другу частину коду без функцій циклом “while True”. Програма буде працювати доки користувач не напише “exit” на запит дії з числами. Якщо користувач введе будь-які інші символи під час запиту дії окрім +, -, \*, /, тоді програма запитає дію знову, показавши повідомлення “Невідома дія!”.

Текст програми:

|  |
| --- |
| def plus(a, b):  return a + b  def minus(a, b):  return a - b  def mul(a, b):  return a \* b  def div(a, b):  if b != 0:  return a / b  return "На нуль ділити не можна!"  while True:  dia = input("Введіть дію (+,-,\*,/): ")  if dia == "exit":  print("\*Калькулятор вимкнувся\*")  break  else:  a = float(input("Введіть перше число: "))  b = float(input("Введіть друге число: "))  match dia:  case "+":  result = plus(a, b)  case "-":  result = minus(a, b)  case "\*":  result = mul(a, b)  case "/":  result = div(a, b)  case \_:  print("Невідома дія!")  continue  print("Результат: ", result) |

Посилання на github:

<https://github.com/jekaproc/TP-KB-241-Protsenko-Evgeniy/blob/main/topic_03/task01.py>

Знімок екрану з посилання на github:

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

Хід виконання завдання:

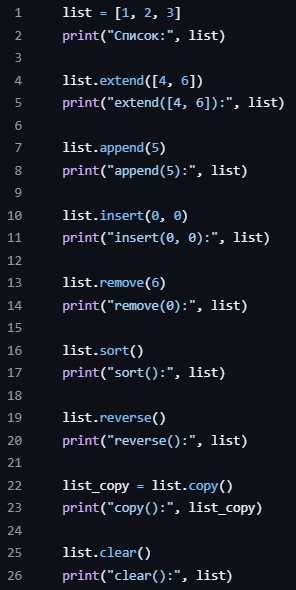
1. Задаю змінну для списку list, який складається з трьох чисел: 1, 2, 3.
2. Далі застосовую різні функції для цього списку. Результати функцій виводяться після вказання використаної функції.

Текст програми:

|  |
| --- |
| list = [1, 2, 3]  print("Список:", list)  list.extend([4, 6])  print("extend([4, 6]):", list)  list.append(5)  print("append(5):", list)  list.insert(0, 0)  print("insert(0, 0):", list)  list.remove(6)  print("remove(0):", list)  list.sort()  print("sort():", list)  list.reverse()  print("reverse():", list)  list\_copy = list.copy()  print("copy():", list\_copy)  list.clear()  print("clear():", list) |

Посилання на github:

<https://github.com/jekaproc/TP-KB-241-Protsenko-Evgeniy/blob/main/topic_03/task02.py>

Знімок екрану з посилання на github:

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

Хід виконання завдання:

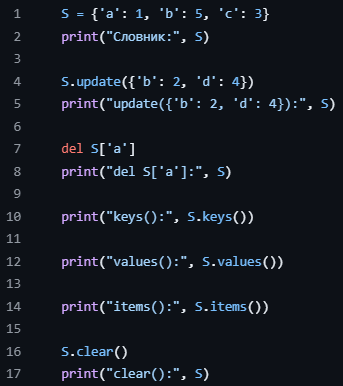
1. Задаю змінну для словника S, який складається з трьох елементів з ключами: a: 1, b: 5, c: 3.
2. Далі застосовую різні функції для цього словника. Результати функцій виводяться після вказання використаної функції.

Текст програми:

|  |
| --- |
| S = {'a': 1, 'b': 5, 'c': 3}  print("Словник:", S)  S.update({'b': 2, 'd': 4})  print("update({'b': 2, 'd': 4}):", S)  del S['a']  print("del S['a']:", S)  print("keys():", S.keys())  print("values():", S.values())  print("items():", S.items())  S.clear()  print("clear():", S) |

Посилання на github:

<https://github.com/jekaproc/TP-KB-241-Protsenko-Evgeniy/blob/main/topic_03/task03.py>

****Знімок екрану з посилання на github:

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

Хід виконання завдання:

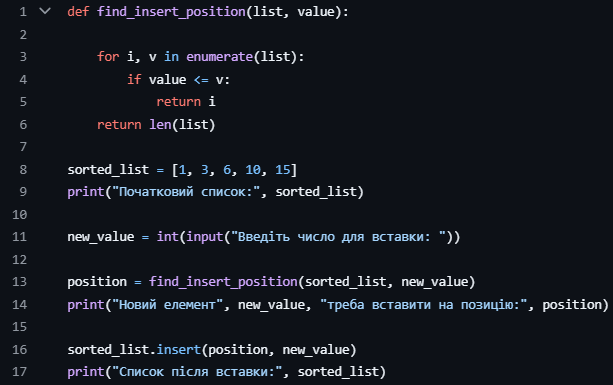
1. Створюю функцію для знаходження позиції для вставки заданого користувачем числа шляхом підбору комірки, число в якій буде більшим за нове значення. На випадок введення числа, більшого за найбільше число у списку, буде взята наступна позиція після останньої за допомогою значення довжини списку len().
2. Задаю відсортований список та залишаю число для вставки на власний вибір користувача.
3. Задаю повідомлення для виведення комірки та числа яке можна туди вставити не порушуючи сортування списку.

Текст програми:

|  |
| --- |
| def find\_insert\_position(list, value):  for i, v in enumerate(list):  if value <= v:  return i  return len(list)  sorted\_list = [1, 3, 6, 10, 15]  print("Початковий список:", sorted\_list)  new\_value = int(input("Введіть число для вставки: "))  position = find\_insert\_position(sorted\_list, new\_value)  print("Новий елемент", new\_value, "треба вставити на позицію:", position)  sorted\_list.insert(position, new\_value)  print("Список після вставки:", sorted\_list) |

Посилання на github:

<https://github.com/jekaproc/TP-KB-241-Protsenko-Evgeniy/blob/main/topic_03/task04.py>

Знімок екрану з посилання на github:

Звіт до Теми №4

Виняткові ситуації

Під час виконання практичного завдання до Теми №4 було надано варіанти рішення до наступних задач:

**Розширити програму калькулятор функцією запитів даних для виконання операцій від користувача, що обробляє виняткові ситуації.**

Хід виконання завдання:

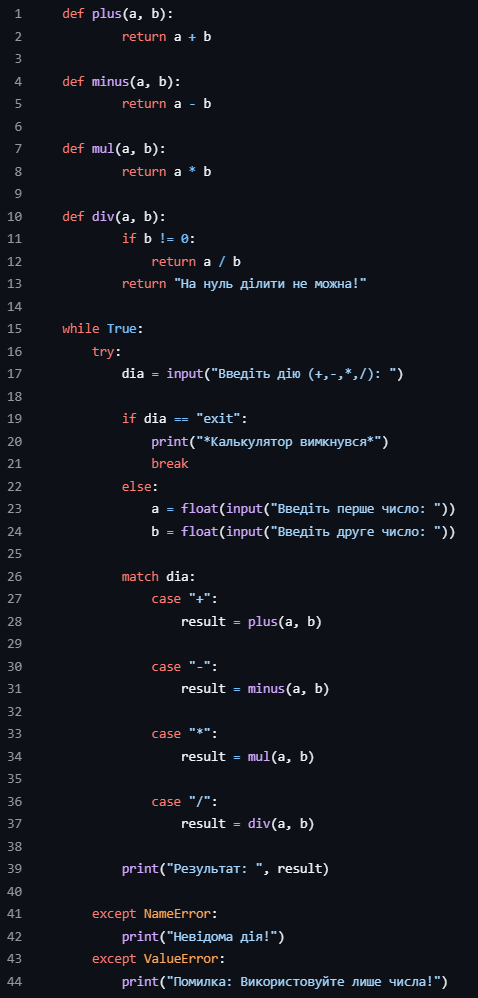
1. Копіюю код програми калькулятора із завдання попередньої теми.
2. Досліджую помилки, які можуть виникнути під час введення літер замість цифр або при введенні будь-якого символу для зазначення дії крім “+”, “-”, “\*”, “/”.
3. Охоплюю другу частину коду винятковою конструкцією “try…except” та роблю помилки NameError і ValueError винятковими. NameError виникає коли програма хоче вивести result, але не може цього зробити, бо жоден case не виводить його значення. ValueError виникає, коли користувач на запит ввести перше або друге число вводить літеру чи інший символ замість числа. Тепер можна прибрати “case \_” для виведення повідомлення "Невідома дія!", оскільки воно буде показуватись при виконанні блоку except NameError.

Текст програми:

|  |
| --- |
| def plus(a, b):  return a + b  def minus(a, b):  return a - b  def mul(a, b):  return a \* b  def div(a, b):  if b != 0:  return a / b  return "На нуль ділити не можна!"  while True:  try:  dia = input("Введіть дію (+,-,\*,/): ")  if dia == "exit":  print("\*Калькулятор вимкнувся\*")  break  else:  a = float(input("Введіть перше число: "))  b = float(input("Введіть друге число: "))  match dia:  case "+":  result = plus(a, b)  case "-":  result = minus(a, b)  case "\*":  result = mul(a, b)  case "/":  result = div(a, b)  print("Результат: ", result)    except NameError:  print("Невідома дія!")  except ValueError:  print("Помилка: Використовуйте лише числа!") |

Посилання на github:

<https://github.com/jekaproc/TP-KB-241-Protsenko-Evgeniy/blob/main/topic_04/task01.py>

Знімок екрану з посилання на github:

**Розширити функцію ділення обробкою виняткової ситуації ділення на нуль**

Хід виконання завдання:

1. Дізнаюся, що помилка, яка виникає при ділені на нуль, називається ZeroDivisionError.
2. Вставляю помилку в except блок та залишаю у функції ділення лише “return a / b”.
3. Також ознайомлююсь зі списком виняткових ситуацій за наданим посиланням для ширшого розуміння теми.

Текст програми:

|  |
| --- |
| def plus(a, b):  return a + b  def minus(a, b):  return a - b  def mul(a, b):  return a \* b  def div(a, b):  return a / b  while True:  try:  dia = input("Введіть дію (+,-,\*,/): ")  if dia == "exit":  print("\*Калькулятор вимкнувся\*")  break  else:  a = float(input("Введіть перше число: "))  b = float(input("Введіть друге число: "))  match dia:  case "+":  result = plus(a, b)  case "-":  result = minus(a, b)  case "\*":  result = mul(a, b)  case "/":  result = div(a, b)  print("Результат: ", result)    except NameError:  print("Помилка: Невідома дія!")  except ValueError:  print("Помилка: Використовуйте лише числа!")  except ZeroDivisionError:  print("Помилка: На нуль ділити не можна!") |

Посилання на github:

<https://github.com/jekaproc/TP-KB-241-Protsenko-Evgeniy/blob/main/topic_04/task02.py>

Знімок екрану з посилання на github:

