**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 довільного вигляду перетворити таким чином, щоб він відтворився у зворотному напрямку.

Хід виконання завдання:

Я вирішив використати функціонал зрізу з кроком -1, щоб відтворити рядок не використовуючи циклів.

Текст програми:

|  |
| --- |
| input\_string = input("Write your line here:")  def reverse(input\_string):  return input\_string[::-1]  reversed\_string = reverse(input\_string)  print("Raw line:", input\_string)  print("Reversed line:", reversed\_string) |

**Виконати тестування функцій, що працюють з рядками**

Необхідно виконати тестування функцій, що працюють з рядками: strip(), capitalize(), title(), upper(), lower()..

Хід виконання завдання:

На прикладі одного й того самого рядка показав принцип роботи всіх функцій.

Текст програми:

|  |
| --- |
| string0 = " pYthoN is SimPLe!< "  print("strip() прибирає пробіли в рядку:", string0.strip())  print("strip() переводить перший символ рядка в верхній регістр:",string0.strip().capitalize()) # Прибрав пробіли для наглядності  print("strip() переводить кожну першу літеру слова в верхній регістр:", string0.title())  print("strip() переводить усі літери у верхній регістр:", string0.upper())  print("strip() переводить усі літери у нижній регістр:", string0.lower()) |

**Дискримінант**

Необхідно написати функцію пошуку дискримінанту квадратного рівняння.

Хід виконання завдання:

Використовуючи дві функції – одну для збору даних, іншу – для їх опрацювання дізнаємося дискримінант за заданими коефіцієнтами.

Текст програми:

|  |
| --- |
| def get\_coefficients():  print("Enter your coefficients (ax^2 + bx + c):")  a = float(input('Enter your a: '))  b = float(input('Enter your b: '))  c = float(input('Enter your c: '))  return a, b, c  def discriminant(a, b, c):  D = b\*\*2 - 4\*a\*c  return D  a, b, c = get\_coefficients()  print("The discriminant is:", discriminant(a, b, c)) |

Звіт до Теми №2

Умовні переходи

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

**Коренів квадратного рівняння**

Необхідно написати функцію пошуку коренів квадратного рівняння використовуючи функцію розрахунку дискримінанту з попередньої теми та умовні переходи.

Хід виконання завдання:

Використовуючи функцію розрахування дискримінанту з минулої теми, додав функцію обрахування коренів та перевірку дійсності вводу.

Текст програми:

|  |
| --- |
| import math  def get\_coefficients(): #input func  print("Enter your coefficients (ax^2 + bx + c):")  a = float(input('Enter your a: '))  b = float(input('Enter your b: '))  c = float(input('Enter your c: '))  return a, b, c  def discriminant(a, b, c): #disc calc func  return b\*\*2 - 4\*a\*c  def find\_roots(a, b, c): #root calc func  D = discriminant(a, b, c)  if D > 0:  x1 = (-b + math.sqrt(D)) / (2 \* a)  x2 = (-b - math.sqrt(D)) / (2 \* a)  return f"Two real roots: x1 = {x1}, x2 = {x2}"  elif D == 0:  x = -b / (2 \* a)  return f"One real root: x = {x}"  else:  return "No real roots exist."  a, b, c = get\_coefficients()  if a == 0: #is possible check  print("This is not a quadratic equation (a cannot be 0).")  else:  roots = find\_roots(a, b, c)  print("The roots are:", roots) |

**Калькулятор на if else конструкції**

Необхідно написати програму калькулятор використовуючи if else конструкцію. Кожна операція має бути виконана в окремій функції.

Хід виконання завдання:

Для початку створив всі необхідні функції для роботи операцій калькулятора, а саме додавання, віднімання, множення, ділення, остача від ділення, піднесення для степеня та цілочислене ділення.  
Наступна функція – перевірка вибору операції.

Якщо вибір коректний – перехід до введення числових значень з вбудованою перевіркою їх коректності за допомогою функції try:

Остання частина – виконання обчислення за допомогою конструкцій if else

Текст програми:

|  |
| --- |
| # Operation func's  def add(a, b):  return a + b  def subtract(a, b):  return a - b  def multiply(a, b):  return a \* b  def divide(a, b):  if b != 0:  return a / b  else:  return "Division by zero is not allowed!"  def modulus(a, b):  return a % b  def exponent(a, b):  return a \*\* b  def floor\_division(a, b):  if b != 0:  return a // b  else:  return "Division by zero is not allowed!"  # Main function  def calculator():  print("Select an operation:")  print("1. Addition (+)")  print("2. Subtraction (-)")  print("3. Multiplication (\*)")  print("4. Division (/)")  print("5. Modulus (%)")  print("6. Exponentiation (\*\*)")  print("7. Floor Division (//)")  choice = input("Enter the number of the operation (1-7): ")  # Validate the choice of operation  if choice in ['1', '2', '3', '4', '5', '6', '7']:  try:  num1 = float(input("Enter the first number: "))  num2 = float(input("Enter the second number: "))  except ValueError:  print("Please enter valid numerical values!")  return    if choice == '1':  print(f"Result: {add(num1, num2)}")  elif choice == '2':  print(f"Result: {subtract(num1, num2)}")  elif choice == '3':  print(f"Result: {multiply(num1, num2)}")  elif choice == '4':  print(f"Result: {divide(num1, num2)}")  elif choice == '5':  print(f"Result: {modulus(num1, num2)}")  elif choice == '6':  print(f"Result: {exponent(num1, num2)}")  elif choice == '7':  print(f"Result: {floor\_division(num1, num2)}")  else:  print("Invalid operation choice!")  # Start  calculator() |

**Калькулятор на match конструкції**

Необхідно написати програму калькулятор використовуючи match конструкцію. Кожна операція має бути виконана в окремій функції.

Хід виконання завдання:

Використовуючи скелет калькулятора з минулого завдання, замінив конструкцію if else на конструкцію match

Текст програми:

|  |
| --- |
| # Operation func's  def add(a, b):  return a + b  def subtract(a, b):  return a - b  def multiply(a, b):  return a \* b  def divide(a, b):  if b != 0:  return a / b  else:  return "Division by zero is not allowed!"  def modulus(a, b):  return a % b  def exponent(a, b):  return a \*\* b  def floor\_division(a, b):  if b != 0:  return a // b  else:  return "Division by zero is not allowed!"  # Main function  def calculator():  print("Select an operation:")  print("1. Addition (+)")  print("2. Subtraction (-)")  print("3. Multiplication (\*)")  print("4. Division (/)")  print("5. Modulus (%)")  print("6. Exponentiation (\*\*)")  print("7. Floor Division (//)")  choice = input("Enter the number of the operation (1-7): ")  # Validate the choice of operation  if choice in ['1', '2', '3', '4', '5', '6', '7']:  try:  num1 = float(input("Enter the first number: "))  num2 = float(input("Enter the second number: "))  except ValueError:  print("Please enter valid numerical values!")  return    match choice:  case '1':  print(f"Result: {add(num1, num2)}")  case '2':  print(f"Result: {subtract(num1, num2)}")  case '3':  print(f"Result: {multiply(num1, num2)}")  case '4':  print(f"Result: {divide(num1, num2)}")  case '5':  print(f"Result: {modulus(num1, num2)}")  case '6':  print(f"Result: {exponent(num1, num2)}")  case '7':  print(f"Result: {floor\_division(num1, num2)}")  case \_:  print("Invalid operation choice!")  # Start  calculator() |

Звіт до Теми №3

Цикли

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

**Калькулятор з постійними запитами**

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:

Використовуючи код калькулятора з минулої теми я трохи модифікував його, використовуючи нескінченний цикл «while True:» та одну додаткову перевірку на введення команди exit, що припиняє роботу калькулятора.

Текст програми:

|  |
| --- |
| # Операції  def add(a, b):  return a + b  def subtract(a, b):  return a - b  def multiply(a, b):  return a \* b  def divide(a, b):  if b != 0:  return a / b  else:  return "Division by zero is not allowed!"  def modulus(a, b):  return a % b  def exponent(a, b):  return a \*\* b  def floor\_division(a, b):  if b != 0:  return a // b  else:  return "Division by zero is not allowed!"  # Головна функція  def calculator():  print("Welcome to the calculator!")  print("Type 'exit' at any time to quit.")    while True:  print("\nSelect an operation:")  print("1. Addition (+)")  print("2. Subtraction (-)")  print("3. Multiplication (\*)")  print("4. Division (/)")  print("5. Modulus (%)")  print("6. Exponentiation (\*\*)")  print("7. Floor Division (//)")    choice = input("Enter the number of the operation (1-7) or 'exit' to quit: ").strip().lower()    if choice == 'exit':  print("Goodbye!")  break    # Перевірка вибору операції  if choice in ['1', '2', '3', '4', '5', '6', '7']:  try:  num1 = float(input("Enter the first number: "))  num2 = float(input("Enter the second number: "))  except ValueError:  print("Invalid input! Please enter numerical values.")  continue    match choice:  case '1':  print(f"Result: {add(num1, num2)}")  case '2':  print(f"Result: {subtract(num1, num2)}")  case '3':  print(f"Result: {multiply(num1, num2)}")  case '4':  print(f"Result: {divide(num1, num2)}")  case '5':  print(f"Result: {modulus(num1, num2)}")  case '6':  print(f"Result: {exponent(num1, num2)}")  case '7':  print(f"Result: {floor\_division(num1, num2)}")  else:  print("Invalid choice! Please select a valid operation.")  # Запуск  calculator() |

**Тестування функцій списків**

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

Хід виконання завдання:

Написав невеликий ланцю зміни одного й того самого списку з виведенням результату після кожної зміни.

Текст програми:

|  |
| --- |
| list0 = [5, 3, 8, 1, 2]  print("Початковий список:", list0)  list0.extend([6, 7, 9])  print("extend() додає всі елементи іншого списку до кінця:", list0)  list0.append(10)  print("append() додає один елемент до кінця:", list0)  list0.insert(2, 15)  print("insert(2, 15) вставляє 15 у позицію 2:", list0)  list0.remove(3)  print("remove(3) видаляє перше входження числа 3:", list0)  list0.clear()  print("clear() очищає список:", list0)  list0 = [5, 3, 8, 1, 2]  print("\nоновлений список для подальшого тестування:", list0)  list0.sort()  print("sort() сортує список за зростанням:", list0)  list0.reverse()  print("reverse() змінює порядок елементів на протилежний:", list0)  list\_copy = list0.copy()  print("copy() створює копію списку:", list\_copy) |

**Тестування функцій словників**

Необхідно написати програму тестування функцій словників таких як: update(), del(), clear(), keys(), values(), items()

Хід виконання завдання:

Аналогічно до завдання 2, ланцюг перетворення одного й того самого словника.

Текст програми:

|  |
| --- |
| dict0 = {"a": 1, "b": 2, "c": 3}  print("Початковий словник:", dict0)  dict0.update({"d": 4, "a": 10})  print("update({'d': 4, 'a': 10}) додає нові пари і оновлює існуючі ключі:", dict0)  del dict0["b"]  print("del dict0['b'] видаляє пару з ключем 'b':", dict0)  dict0.clear()  print("clear() очищає словник:", dict0)  dict0 = {"x": 5, "y": 10, "z": 15}  print("\nоновлений словник для подальшого тестування:", dict0)  print("keys() повертає всі ключі:", dict0.keys())  print("values() повертає всі значення:", dict0.values())  print("items() повертає всі пари ключ-значення:", dict0.items()) |

**Функція пошуку позиції для вставки**

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

Хід виконання завдання:

Використовуючи вбудований модуль bisect написав невеличку програму що виводить поточний список і запитує нове значення, після чого вставляє його.

Текст програми:

|  |
| --- |
| import bisect  sorted\_list = [1, 3, 5, 7, 9]  print("Поточний список:", sorted\_list)  new\_value = int(input("Введіть нове значення:"))  position = bisect.bisect\_left(sorted\_list, new\_value)  print("Список з новим елементом:", sorted\_list[:position] + [new\_value] + sorted\_list[position:]) |

Посилання на github:

https://github.com/MrMostliker/TP-KB-231-Dmytro-Potapenko

Знімок екрану з посилання на github:

