**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:

Створив функцію що розвертає рядок.

Текст програми:

|  |
| --- |
| def stringReverse(string):      print(string[::-1])      return  string = "abcdefg123"  stringReverse(string) |

**Тестування методів для рядків**

Необхідно створити тести для методів рядків.

Хід виконання завдання:

Створив тести для методів рядків.

Текст програми:

|  |
| --- |
| string = "     Hello"  print (string.strip())  string1 = "hello"  print (string1.capitalize())  string2 = "hello world"  print (string2.title())  string3 = "hello"  print (string3.upper())  string4  = "HELLO"  print (string4.lower())  string5 = "Hello World"  print(string5.split(" "))  print(string5.count("l")) |

**Знаходження дискримінанта**

Необхідно створити функцію що знаходить дискримінант.

Хід виконання завдання:

Створив функцію для знаходження дискримінанта за формулою b\*b-4a\*c.

Текст програми:

|  |
| --- |
| def disk\_calculation(a, b, c):      D = b\*\*2 - 4\*a\*c      print("D is " + str(D))      return  a = int(input("Enter a: "))  b = int(input("Enter b: "))  c = int(input("Enter c: "))  disk\_calculation(a, b, c) |

Посилання на github:

Звіт до Теми №2

Умовні переходи

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

**Пошук коренів квадратного рівняння**

Необхідно дописати умови перевірки дискримінанта та знайти корені рівняння.

Хід виконання завдання:

Створив умови перевірки, функцію для знаходження коренів.

Код програми:

|  |
| --- |
| def disk\_calculation(a, b, c):      D = b\*\*2 - 4\*a\*c      return D  def calculation(a, b, D):      print("D is " + str(disk\_calculation(a, b, c)))      if D > 0:          x1 = (-b + D\*\*0.5) / (2\*a)          x2 = (-b - D\*\*0.5) / (2\*a)          return x1, x2      if D == 0:          x1 = (-b + D\*\*0.5) / (2\*a)          return x1      if D < 0:          return "Equation has no roots."  a = int(input("Enter a: "))  b = int(input("Enter b: "))  c = int(input("Enter c: "))  print(calculation(a, b, disk\_calculation(a, b, c))) |

**Калькулятор з конструкціями if else**

Потрібно створити калькулятор з умовними переходами.

Хід виконання завдання:

Створив калькулятор з умовними переходами.

Код програми:

|  |
| --- |
| def calculator(a, b, action):      if action == "+":          result = a + b          return result      elif action == "-":          result = a - b          return result      elif action == "\*":          result = a \* b          return result      elif action == "/":          if b == 0:              print ("Division by 0 is forbidden.")              return          else:              result = a / b              return result    def main():      a = float(input("Enter a: "))      b = float(input("Enter b: "))      action = (input("Enter ation (+, -, \* or /)"))      print("Result is " + str(calculator(a, b, action)))  main() |

**Калькулятор з конструкцією match case**

Потрібно створити калькулятор з конструкціями match case.

Хід виконання завдання:

Створив калькулятор з кейсами.

Код програми:

|  |
| --- |
| def calculator(a, b, action):      match action:          case "+":              result = a + b              return result          case "-":              result = a - b              return result          case "\*":              result = a \* b              return result          case "/":              if b == 0:                  print ("Division by 0 is forbidden.")                  return              else:                  result = a / b                  return result    def main():      a = float(input("Enter a: "))      b = float(input("Enter b: "))      action = (input("Enter ation (+, -, \* or /)"))      print("Result is " + str(calculator(a, b, action)))  main() |

Посилання на github:

\

Звіт до Теми №3

Цикли

**Калькулятор:**

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:

Доповнив калькулятор нескінченним циклом та запитом на початок/продовження виконання програми.

Код програми:

|  |
| --- |
| def calculator(a, b, action):      if action == "+":          result = a + b          return result      elif action == "-":          result = a - b          return result      elif action == "\*":          result = a \* b          return result      elif action == "/":          if b == 0:              print ("Division by 0 is forbidden.")              return          else:              result = a / b              return result    def main():      a = float(input("Enter a: "))      b = float(input("Enter b: "))      action = (input("Enter ation (+, -, \* or /): "))      print("Result is " + str(calculator(a, b, action)))  while True:      trigger = input("Do you want to continue?[Y/N]").upper()      if trigger == "N":          break      elif trigger == "Y":          main()      else:          print("Try again!") |

**Тестування методів для списків**:

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

Хід виконання завдання:

Створив тести для деяких методів списків.

Код програми:

|  |
| --- |
| list1 = [1, 2, 3]  list2 = [4, 5, 6]  list1.extend(list2)  print(list1)    list1 = ['Hello']  list1.append('World')  print(list1)    list1 = ['Hello', 'World']  list1.insert(1, 'to')  print(list1)  list1 = ['Hello', 'World', 'Pavlo']  list1.remove('Pavlo')    print(list1)  list1 = ['Hello', 'World']  list1.clear()  print(list1)  list1 = [5, 4, 3, 2, 1]  list1.sort()  print(list1)    list1 = ['World', 'Hello']  list1.reverse()  print(list1)  list1 = ['World', 'Hello']  print(list1.copy()) |

**Тестування методів для словників:**

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

Хід виконання завдання:

Створив тести для деяких методів словників.

Код програми:

|  |
| --- |
| car = {      'model': 'RAV4',      'engine': '2.0'  }  car.update({'VIN': '2165981981651816'})  print(car)  del car["engine"]  print(car)  car.clear()  print(car)  car.keys()  car["model"] = "Prius"  print(car)  car.values()  car["model"] = "CH-R"  print(car)  car.items()  car["engine"] = "1.6"  print(car) |

**Функція пошуку позиціі**

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

Хід виконання завдання:

Створив функцію пошуку позиції для додавання нового елементу в список.

Код програми:

|  |
| --- |
| ef search(sorted\_list, number):      list\_right = 0      list\_left = len(sorted\_list) - 1      while list\_right <= list\_left:          list\_mid = (list\_right + list\_left) // 2          if sorted\_list[list\_mid] == number:              print("Number exists. Position: " + str(list\_mid))              return          elif sorted\_list[list\_mid] < number:              list\_right = list\_mid + 1          else:              list\_left = list\_mid - 1      print ("Position to add is: " + str(list\_right))      return  sorted\_list = [1, 3, 5, 7, 9, 11]  while True:      number = input("Insert number to add: ")      number = int(number)      position = search(sorted\_list, number) |

Посилання на github: https://github.com/amaterasu809/LedoviyPavlo