**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:

|  |
| --- |
| string = "abcdefg123"  def reverse(var):  var = var[::-1]  return var  reversedString = reverse(string)  print(reversedString) |

**Тестування методів для рядків**

Необхідно протестувати деякі методи для рдяків.

Хід виконання завдання:

Протестував такі методи:

Strip() – видаляє пробіли і табуляції спереду та позаду рядку.

Capitalize() – робить 1 букву рядку вищого регістру, інші нижчого.

Title() – всі 1 букви кожного слова в рядку робить вищого регістру.

Upper() – всі букви до верхнього регістру.

Lower() – всі букви до нижчого регістру.

Find() – шукає позицію символа в рядку.

Split() – розбиває рядок по заданому символу та записує частини в масив.

Count() – рахує кількість входжень символу до рядку.

Swapcase() – міняє місцями регістри літер.

Текст програми:

|  |
| --- |
| #strip() method  stringWithSpaces = " I am Denis "  print (stringWithSpaces.strip())  #capitalize() method  notCapitalizedString = "i am a student"  print (notCapitalizedString.capitalize())  #title() method  string = "Hello i am denis"  print (string.title())  #upper() and lower() methods  weirdString = "hElLo i aM dEnIs aNd i aM a sTuDeNt"  print (weirdString.upper())  print (weirdString.lower())  #some other  #find() method  print(string.find("i"))  #split() method  print(weirdString.upper().split(" "))  #count() method  print(string.count("l"))  #swapcase() method  print (weirdString.swapcase()) |

**Знаходження дискримінанта**

Необхідно створити функцію що знаходить дискримінант.

Хід виконання завдання:

Формула дискримінанта b\*\*2 – 4\*a\*c. Запрошуємо ввід 3-х коефіцієнтів від користувача та передаємо їх функції на обчислення.

Текст програми:

|  |
| --- |
| def dSearching(a, b, c):  res = b\*\*2-4\*a\*c  return res  a = float(input("Enter a: "))  b = float(input("Enter b: "))  c = float(input("Enter c: "))  D = dSearching(a, b, c)  print(D) |

Посилання на github:

<https://github.com/Aatheeriss/TP-KB-221-Denis-Sazonenko.git>

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 чи дискримінант більший, дорівнює, або інше (тобто менший) за нуль. Створив функцію для знаходження коренів.

Код програми:

|  |
| --- |
| #calculating D  def dSearching(a, b, c):      res = b\*\*2 - 4\*a\*c      return res  #calculating D  def dSearching(a, b, c):      res = b\*\*2 - 4\*a\*c      return res  #variables input  a = float(input("Enter a: "))  b = float(input("Enter b: "))  c = float(input("Enter c: "))  #D output  D = dSearching(a, b, c)  print("\nD =", D)  #topic\_02  #roots calculating function  def calculatingRoots(b, D, a):      x1 = (-b + D\*\*0.5) / (2\*a)      x2 = (-b - D\*\*0.5) / (2\*a)      if x1 == x2:          return x1      else:          return x1, x2    #D conditions check and roots output  if D > 0:      print("\nEquation has 2 roots: ")      print("x =", calculatingRoots(b, D, a))  elif D == 0:      x = -b / (2\*a)      print("\nEquation has 1 root: ")      print("x =", calculatingRoots(b, D, a))  else:      print("\nEquation has no roots.") |

**Калькулятор з конструкціями if else**

Потрібно створити калькулятор з умовними переходами.

Хід виконання завдання:

Створив калькулятор з 4 діями. Ділення на 0 – виключення.

Код програми:

|  |
| --- |
| a = float(input("Enter the first number: "))  b = float(input("Enter the second number: "))  action = input("Enter action (+, -, \*, /): ").strip()  if action == "+":      print(a + b)  elif action == "-":      print(a - b)  elif action == "\*":      print(a \* b)  elif action == "/":      try:          result = a / b          print(result)      except ZeroDivisionError:          print("Division by zero is not allowed.")  else:      print("Invalid action. Please enter +, -, \*, or /.") |

**Калькулятор з конструкцією match case**

Потрібно створити калькулятор з конструкцією match case.

Хід виконання завдання:

Створив калькулятор з 4 діями. Ділення на 0 – виключення.

Код програми:

|  |
| --- |
| a = float(input("Enter the first number: "))  b = float(input("Enter the second number: "))  action = input("Enter action (+, -, \*, /): ").strip()  match action:      case "+":          print(a + b)      case "-":          print(a - b)      case "\*":          print(a \* b)      case "/":          try:              result = a / b              print(result)          except ZeroDivisionError:              print("Division by zero is not allowed.")      case \_:          print("Invalid action. Please enter +, -, \*, or /.") |

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:

Додав нескінченний цикл.

Код програми:

|  |
| --- |
| while True:  a = float(input("Enter the first number: "))  b = float(input("Enter the second number: "))  action = input("Enter action (+, -, \*, /): ").strip()  if action == "+":  print(a + b)  elif action == "-":  print(a - b)  elif action == "\*":  print(a \* b)  elif action == "/":  try:  result = a / b  print(result)  except ZeroDivisionError:  print("Division by zero is not allowed.")  else:  print("Invalid action. Please enter +, -, \*, or /.") |

**Тестування функцій**:

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

Хід виконання завдання:

Написав програму тестування даних функцій.

Код програми:

|  |
| --- |
| #1. extend()  def extendTesting():  list1 = [1, 2, 3, 4, 5]  list2 = [6, 7, 8, 9, 0]  list1.extend(list2)  print(list1)  return  #2. append()  def appendTesting():  list1 = ['Hello', 'I', 'am']  list1.append('Denis')  print(list1)  return  #3. insert(id, val)  def insertTesting():  list1 = ['Hello', 'I', 'Denis']  list1.insert(2, 'am')  print(list1)  return  #4. remove()  def removeTesting():  list1 = ['Hello', 'I', 'removeMEpLeAsE', 'am', 'Denis']  list1.remove('removeMEpLeAsE')    print(list1)  return  #5. clear()  def clearTesting():  list1 = [1, 2, 3, 4, 5]  list1.clear()  list1.append('list is empty but this message')  print(list1)  return  #6. sort()  def sortTesting():  list1 = [2, 5, 3, 1, 4]  list1.sort()  print(list1)  return  #7. reverse()  def reverseTesting():  list1 =[5, 4, 3, 2, 1]  list1.reverse()  print(list1)  return  #8. copy()  def copyTesting():  list1 = [1, 2, 3, 4, 5]  print(list1.copy())  return  def main():  a = input("Enter number 1-8:\n 1. extend()\n 2. append()\n 3. insert()\n 4. remove()\n 5. clear()\n 6. sort()\n 7. reverse()\n 8. copy()\n")  match a:  case "1":  extendTesting()  case "2":  appendTesting()  case "3":  insertTesting()  case "4":  removeTesting()  case "5":  clearTesting()  case "6":  sortTesting()  case "7":  reverseTesting()  case "8":  copyTesting()  case \_:  print("Try again!")    print("\n\n-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\n\n")    return  while True:  main() |

**Тестування функцій:**

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

Хід виконання завдання:

Написав програму тестування даних функцій.

Код програми:

|  |
| --- |
| laptop = {  "brand": "lenovo",  "series": "legion"  }  #1. update()  def updateTesting():  laptop.update({"serial number": "LL252433"})  print(laptop)  return  #2. del()  def delTesting():  del laptop["brand"]  print(laptop)  return  #3. clear()  def clearTesting():  laptop.clear()  print(laptop)  return  #4. keys()  def keysTesting():  laptop.keys()  laptop["brand"] = "asus"  print(laptop)  return  #5. values()  def valuesTesting():    laptop.values()  laptop["brand"] = "asus"  print(laptop)  return  #6. items()  def itemsTesting():  laptop.items()  laptop["brand"] = "asus"  print(laptop)  return  def main():  a = input("Enter number 1-6:\n 1. update()\n 2. del()\n 3. clear()\n 4. keys()\n 5. values()\n 6. items()\n")  match a:  case "1":  updateTesting()  case "2":  delTesting()  case "3":  clearTesting()  case "4":  keysTesting()  case "5":  valuesTesting()  case "6":  itemsTesting()  case \_:  print("Try again!")    print("\n\n-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\n\n")    return  while True:  main() |

**Функція пошуку позиціі**

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

Хід виконання завдання:

Створив функцію бінарного пошуку елементу в масиві, та позиціі для додавання.

Код програми:

|  |
| --- |
| def positionFind(list1, element):  list\_start = 0  list\_end = len(list1) - 1  while list\_start <= list\_end:  list\_mid = (list\_start + list\_end) // 2  if list1[list\_mid] == element:  print("Element is already in list. Position is: " + str(list\_mid))  return  elif list1[list\_mid] < element:  list\_start = list\_mid + 1  else:  list\_end = list\_mid - 1  print ("Position to add is: " + str(list\_start))  return  list1 = [1, 3, 5, 7, 9]  while True:  print(list1)  element = input("Enter the element to find position to add: ")  element = int(element)  position = positionFind(list1, element) |

Звіт до Теми №4

Виняткові ситуації

**Розширення калькулятора:**

**Попередні умови**: реалізована програма калькулятор, що використовує метод нескінченного введення даних для обробки. Всі дії (додавання, віднімання, множення, ділення) реалізовані як окремі функції та використовуються у відповідних місцях.

1. Розширити програму калькулятор функцією запитів даних для виконання операцій від користувача, що обробляє виняткові ситуації.
2. Розширити функцію ділення обробкою виняткової ситуації ділення но нуль

Хід виконання завдання:

Розклав дії програми по окремих функціях. В функції insertion реалізував захоплення винятку ValueError. В фукнції calculation – ZeroDivisionError.

Загалом по програмі – keyboardInterrupt exeption, який виводить повідомлення перед закінченням програми у випадку, якщо програма переривається за допомогою клавіатури.

Функція insertion буде запрошувати дані на введення до тих пір доки не отримає валідні дані для обробки. Функція calculation завершиться або виведенням підрахованого значення, або зайде в виняток ділення на нуль и закінчиться, програма почне роботу спочатку завдяки нескінченному циклу визову функції main, яка в свою чергу визиває попередні функції.

Код програми:

|  |
| --- |
| try:      #insertion function which repeats until getting valid values of required functions      def insertion():          while True:              try:                  a = float(input("Enter the first number: "))                  b = float(input("Enter the second number: "))                  action = input("Enter action (+, -, \*, /): ").strip()                  if action not in ('+', '-', '\*', '/'):                      print("Invalid action. Please enter +, -, \*, or /.")                      continue                  return a, b, action              except ValueError:                  print('Invalid input. Please enter valid numbers.')      #calculation function (catches division by zero exeption)      def calculation(a, b, action):          if action == "+":              print(a + b)          elif action == "-":              print(a - b)          elif action == "\*":              print(a \* b)          elif action == "/":              try:                  result = a / b                  print(result)              except ZeroDivisionError:                  print("Division by zero is not allowed.")      #main      def main():          a, b, action = insertion()          calculation(a, b, action)      while True:          main()  except KeyboardInterrupt:      print('\n\nBye!')      exit |

Посилання на github:

<https://github.com/Aatheeriss/TP-KB-221-Denis-Sazonenko.git>