**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:

1. **Перетворення рядка**

Необхідно рядок, що вводить користувач написати зворотнім чином.

Хід виконання завдання:

Виводимо користувачу запит, питаючи в нього бажаний текст, і додаємо, в зрізі тексту, від’ємний параметр.

Текст програми:

|  |
| --- |
| print(input("Enter your text: ")[::-1]) |

1. **Виконати тестування функцій, що працюють з рядками: strip(), capitalize(), title(), upper(), lower()**

Необхідно протестувати функції роботи з текстом як: strip(), capitalize(), title(), upper(), lower().

Хід виконання завдання:

Вводжу дві зміні з текстом для тестування функцій strip(), capitalize(), title(), upper(), lower().

Текст програми:

|  |
| --- |
| text = "  yyYOUr tEXt"  print("original:", text, " ", "strip():", text.strip(), " ", "title():", text.title(), " ", "upper():", text.upper(), " ", "lower():", text.lower())  text1 = "specifically for capitalization"  print("original:", text1, " ","capitalize():", text1.capitalize()) |

1. **Написати функцію пошуку дискримінанту квадратного рівняння**

Необхіднофункцію пошуку дискримінанту квадратного рівняння.

Хід виконання завдання:

Оголошую змінну «exreload» надаючи їй значення «1», для того щоб надати можливість користувачу обчислювати дискримінант скільки завгодно. В циклі «while», з прив’язкую до змінної «exreload», питаю в користувача значення двох коефіціентів і вільного члена квадратного рівняння та рахується значення дискримінанту. Потім значення дискримінанту проходить перевірку, від якої залежить відповідь програми.

Текст програми:

|  |
| --- |
| print("Let's discriminate the numbers!")  exreload = 1  while exreload != 0:      b = int(input("b: "))      a = int(input("a: "))      c = int(input("c: "))      D = (b^2)-4\*a\*c      if D <= 0:          if D == 0:              print("Discriminant:", D)              exreload = int(input("Again? 1/0: "))          else:              print("Discriminant doesn`t exist") #If you don't use complex numbers              exreload = int(input("Again? 1/0: "))      else:          print("Discriminant:", D)          exreload = int(input("Again? 1/0: ")) |

Звіт до Теми №2

Умовний перехід

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

1. **Написати програми пошуку дискримінанту квадратного рівняння та його коренів**

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

Хід виконання завдання:

Так як програма з попередньої теми не повністю підходить під умови, тому я написав два коди: один з яких звичайна модифікація попереднього коду, другий же розбитий на функції та має функціонал першого коду.

Текст програми №1:

|  |
| --- |
| print("Let's discriminate the numbers!")  exreload = 1  while exreload != 0:      a = int(input("a: "))      b = int(input("b: "))      c = int(input("c: "))      D = (b\*\*2)-4\*a\*c      if D > 0:          print("Discriminant:", D)          print("First root: ", (-b+D\*\*0.5)/(2\*a), "Second root: ", (-b-D\*\*0.5)/(2\*a))          exreload = int(input("Again? 1/0: "))      elif D == 0:          print("Discriminant:", D)          print("Root: ", (-b)/(2\*a))          exreload = int(input("Again? 1/0: "))      else:          print("Discriminant doesn`t exist") #If you don't use complex numbers          exreload = int(input("Again? 1/0: ")) |

Текст програми №2:

|  |
| --- |
| def inp():      print("Let's discriminate the numbers!")      a = int(input("a: "))      b = int(input("b: "))      c = int(input("c: "))      return a,b,c  def discr(a,b,c):      D = (b\*\*2)-4\*a\*c      return D  def roots(D,a,b):      exreload = 1      if D > 0:          print("Discriminant:", D)          print("First root: ", (-b+D\*\*0.5)/(2\*a), "Second root: ", (-b-D\*\*0.5)/(2\*a))      elif D == 0:          print("Discriminant:", D)          print("Root: ", (-b)/(2\*a))      else:          print("Discriminant doesn`t exist") #If you don't use complex numbers      exreload = int(input("Again? 1/0: "))      return exreload  exreload = 1  while exreload != 0:      a,b,c = inp()      D = discr(a,b,c)      exreload = roots(D, a, b) |

1. **Написати програму калькулятор використовуючи if else конструкцію**

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

Хід виконання завдання:

Для оптимізації роботи, було взято дві функції(вводу та поновлення) з попередньої програми.

Текст програми:

|  |
| --- |
| def inp():      print("Enter two numbers:")      a = int(input("a: "))      b = int(input("b: "))      return a,b  def calc(a, b):      tool = input("Select a tool: '+' , '-' , '\*' , '/' or '^'\n")      if b == 0 and tool == "/":          print("Cannot divide by 0")      else:          if tool == "+":              return a + b          elif tool == "-":              return a - b          elif tool == "\*":              return a \* b          elif tool == "/":              return a / b          elif tool == "^":              return a \*\* b          else:              print("Input error.")    exreload = 1  while exreload != 0:      a, b = inp()      result = calc(a, b)      print("Result:", result)      exreload = int(input("Again? 1/0: ")) |

1. **Написати програму калькулятор використовуючи match конструкцію**

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

Хід виконання завдання:

Наступний код відрізняється лише тим, що конструкції if else замінені на match case.

Текст програми:

|  |
| --- |
| def inp():      print("Enter two numbers:")      a = int(input("a: "))      b = int(input("b: "))      return a,b  def calc(a, b):      tool = input("Select a tool: '+' , '-' , '\*' , '/' or '^'\n")      if b == 0 and tool == "/":          print("Cannot divide by 0")      else:          match tool:              case "+":                  return a + b              case "-":                  return a - b              case "\*":                  return a \* b              case "/":                  return a / b              case "^":                  return a \*\* b              case \_:                  print("Input error.")    exreload = 1  while exreload != 0:      a, b = inp()      result = calc(a, b)      print("Result:", result)      exreload = int(input("Again? 1/0: ")) |

Звіт до Теми №3

Цикли

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

1. **Розширити програму калькулятор, що була створена до Теми 2, постійними запитами на введення нових даних та операцій**

Розширити програму калькулятор, що була створена до Теми 2, постійними запитами на введення нових даних та операцій. Реалізувати механізм завершення програми після отримання відповідної команди.

Хід виконання завдання:

Під це завдання підпадає програма, що була написана мною до Теми №2, а саме до третього завдання.

Текст програми:

|  |
| --- |
| def inp():      print("Enter two numbers:")      a = int(input("a: "))      b = int(input("b: "))      return a,b  def calc(a, b):      tool = input("Select a tool: '+' , '-' , '\*' , '/' or '^'\n")      if b == 0 and tool == "/":          print("Cannot divide by 0")      else:          match tool:              case "+":                  return a + b              case "-":                  return a - b              case "\*":                  return a \* b              case "/":                  return a / b              case "^":                  return a \*\* b              case \_:                  print("Input error.")    exreload = 1  while exreload != 0:      a, b = inp()      result = calc(a, b)      print("Result:", result)      exreload = int(input("Again? 1/0: ")) |

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

Хід виконання завдання:

Для тестування функцій була написана програма з декількома функціями та невідсортованим списком.

Текст програми:

|  |
| --- |
| def extend(list):      list.extend([1, 11])      print("Function extend(): ", list)  def append(list):      list.append(99)      print("Function append(): ", list)  def insert(list):      list.insert(6, 200)      print("Function insert(): ", list)  def remove(list):      list.remove(2)      print("Function remove(): ", list)  def clear(list):      list.clear()      print("Function clear(): ", list)  def sort(list):      list.sort()      print("Function sort(): ", list)  def reverse(list):      list.reverse()      print("Function reverse(): ", list)  def copy(list):      copy\_list = list.copy()      print("Function copy(): ", copy\_list)  list = [0, 2, 4, 6, 22, 10, 11, 88, 100]  print("Original list: ", list)  extend(list.copy())  append(list.copy())  insert(list.copy())  remove(list.copy())  clear(list.copy())  sort(list.copy())  reverse(list.copy())  copy(list) |

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

Хід виконання завдання:

Для тестування функцій була написана програма з декількома функціями та невідсортованим словником.

Текст програми:

|  |
| --- |
| def update\_test(dictionary):      dictionary.update({'D': 4, 'E': 5, 'F': 6})      print("Function update(): ", dictionary)  def del\_test(dictionary):      del dictionary['A']      print("Function del(): ", dictionary)  def clear\_test(dictionary):      dictionary.clear()      print("Function clear(): ", dictionary)  def keys\_test(dictionary):      print("Function keys(): ", dictionary.keys())  def values\_test(dictionary):      print("Function values(): ", dictionary.values())  def items\_test(dictionary):      print("Function items(): ", dictionary.items())  dictionary = {'A': 1, 'B': 2, 'C': 3}  print("Original dictionary: ", dictionary)  update\_test(dictionary.copy())  del\_test(dictionary.copy())  clear\_test(dictionary.copy())  keys\_test(dictionary.copy())  values\_test(dictionary.copy())  items\_test(dictionary.copy()) |

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

Хід виконання завдання:

Була написана програма з функцією пошуку позиції по відсортованому списку, що складається з чисел.

Текст програми:

|  |
| --- |
| def sortPosition(list, numder):      position = 0      for elem in list:          if numder > elem:              position += 1          else:              break      return position  list = [1, 6, 9, 11, 22, 23, 200, 545]  print("List of numbers:", list)  numder = int(input("Enter a new number whose location you want to know: "))  position = sortPosition(list, numder)  if position == 0:      print("The number will be first")  elif position == len(list):      print("This numder will be last")  else:      print(f"This numder will be between {list[position - 1]} and {list[position]}.") |

Посилання на github:

https://github.com/GAGGAGX/TP-KB-231-Novyk-Maksym

Знімок екрану з посилання на github:

