**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]}.") |

Звіт до Теми №4

Виняткові ситуації

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

1. **Розширити програму калькулятор функцією запитів даних для виконання операцій від користувача, що обробляє виняткові ситуації введення некоректних даних для математичних операці та функціюю ділення обробкою виняткової ситуації ділення но нуль(завдання об’єднані для зрусності)**

Хід виконання завдання:

За основу була взята програма калькулятор з попередньої теми. Було зроблено дві варіації оновленної програми:

* Перша(Calc+) отримала: більший прийнятних чисел; перевірку чисел, що вводить користувач; більш інтуітивні відповіді на запит повторення циклу програми та перевірку значень у відповідь на цей запит; облегшенний варіант перевірки ділення на нуль.
* Друга(Calc++), поміж вище згаданих змін, отримала зручнішу, для роботи користувача, перевірку вибору інструменту обчислень.

Текст програми №1:

|  |
| --- |
| def inp():      while True:          print("Enter two numbers:")          a = input("a: ")          b = input("b: ")          try:              a = float(a)              b = float(b)              break          except ValueError:              print("It's not a number.\n")      return a, b  def calc(a, b):      tool = input("Select a tool: '+' , '-' , '\*' , '/' or '^'\n")      match tool:          case "+":              return a + b          case "-":              return a - b          case "\*":              return a \* b          case "/":              if b == 0:                  print("Cannot divide by 0")              else:                  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)      while True:          exreload = input("Again? Y/N: ")          if exreload.lower().strip() in ['y', 'n', 'yes', 'no']:              break          else:              print("Selection error.\n") |

Текст програми №2:

|  |
| --- |
| def inp():      while True:          print("Enter two numbers:")          a = input("a: ")          b = input("b: ")          try:              a = float(a)              b = float(b)              break          except ValueError:              print("It's not a number.\n")      return a, b  def calc(a, b):      while True:          tool = input("Select a tool: '+' , '-' , '\*' , '/' or '^'\n")          if tool in ['+', '-', '\*', '/', '^']:              break          else:              print("Invalid operator.\n")      match tool:          case "+":              return a + b          case "-":              return a - b          case "\*":              return a \* b          case "/":              if b == 0:                  print("Cannot divide by 0")              else:                  return a / b          case "^":              return a \*\* b    exreload = 1  while exreload != 0:      a, b = inp()      result = calc(a, b)      print("Result:", result)      while True:          exreload = input("Again? Y/N: ")          if exreload.lower().strip() in ['y', 'n', 'yes', 'no']:              break          else:              print("Selection error.\n") |

Звіт до Теми №5

Бібліотеки

Під час виконання практичного завдання до Теми №5 було надано варіанти рішення до наступних задач:

1. **Гра з комп’ютером: камінь, ножиці, папір**

Програма виконує запит від користувача на введення одного із значень ["rock", "scissor", "paper"]. Наступним кроком, використовуючи модуль random, програма у випадковому порядку вибирає одне із значень ["rock", "scissor", "paper"]. В залежності від умови, що камінь перемагає ножиці, ножиці перемагають папір, а папір перемагає камінь визначити переможця.

Хід виконання завдання:

Програма запитує у користувача вибір: камінь, ножиці або папір. Потім вона випадково обирає свій варіант і порівнює його з вибором користувача. В залежності від результату виводиться повідомлення: перемога, поразка або нічия.

Текст програми:

|  |
| --- |
| from random import choice  listChoice = ['stone', 'scissors', 'paper']  def inp():      while True:          per = input("stone, scissors, paper\nEnter your choice: ").lower().strip()          if per in listChoice:              return per          else:              print("Selection error.\n")  def game(per):      botChoice = choice(listChoice)      winList = ['stonescissors', 'scissorspaper', 'paperstone']      if per == botChoice:          print("Draw.")      else:          if (per+botChoice) in winList:              print("You win!")          else:              print("You lose.")  per = inp()  game(per) |

1. **Програма конвертування іноземної валюти в українську гривню**

Для отримання актуальних курсів валют необхідно використовувати API НБУ та модуль, що надає можливість виконувати запити до сторонніх сервісів requests. Достатня умова роботи – можливість конвертації для трьох іноземних валют EUR, USD, PLN. Користувачу надається можливість введення кількості та типу валюти, результат роботи програми – конвертоване значення в українських гривнях.

Хід виконання завдання:

Програма отримує з сайту НБУ список валют із деталями: курс до гривні, назва українською та код. Програма дозволяє: вивести коди та виконати конвертацію. Конвертація працює просто: користувач вводить код валюти й суму для перерахунку у гривні.

Текст програми:

|  |
| --- |
| import requests  listChoice = ['EUR', 'USD', 'PLN']  def inp():      while True:          per = input("EUR, USD, PLN or exit(EX)\nEnter your choice: ").upper()          if per == "EX":              exit(0)          if per in listChoice:              try:                  value = float(input("Enter amount: "))                  break              except ValueError:                  print("It's not a number.\n")          else:              print("Selection error.\n")      return per, value  def curr():      r = requests.get("https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?json")      for elem in r.json():          match elem['cc']:              case "EUR":                  rateEUR = elem['rate']              case "USD":                  rateUSD = elem['rate']              case "PLN":                  ratePLN = elem['rate']      return rateEUR, rateUSD, ratePLN  def con(rateEUR, rateUSD, ratePLN, per, value):      match per:          case "EUR":              print(f"Result of {value} in EUR to UAH = {rateEUR\*value}")          case "USD":              print(f"Result of {value} in USD to UAH = {rateUSD\*value}")          case "PLN":              print(f"Result of {value} in PLN to UAH = {ratePLN\*value}")    r1, r2, r3 = curr()  while True:      per, val = inp()      con(r1, r2, r3, per, val) |

1. **Використання модулів для програми калькулятор**

Функції додавання, віднімання, множення та ділення перенести в файл functions.py. Функції запиту на введення даних для операцій та самих операцій перемістити в файл operations.py. Програму калькулятор реалізувати в файлі calc.py, до якого підключають файл functions.py та operations.py.

Хід виконання завдання:

За основу було взято программу Calc++ з теми №4: дещо перероблено та розбито на декілька програм.

Текст програми functions:

|  |
| --- |
| def sum(a, b):      return a + b  def sub(a, b):      return a - b    def mult(a, b):      return a \* b  def div(a, b):      if b == 0:          print("Cannot divide by 0")      else:          return a / b    def deg(a, b):       return a \*\* b |

Текст програми operations:

|  |
| --- |
| def inp():      while True:          print("Enter two numbers:")          a = input("a: ")          b = input("b: ")          try:              a = float(a)              b = float(b)              break          except ValueError:              print("It's not a number.\n")      return a, b  def tool():      while True:              tool = input("Select a tool: '+' , '-' , '\*' , '/' or '^'\n")              if tool in ['+', '-', '\*', '/', '^']:                  return tool              else:                  print("Invalid operator.\n") |

Текст програми calc:

|  |
| --- |
| from operations import inp, tool  from functions import sum, sub, mult, div, deg  def calc(a, b):      match tool():          case "+":              res = sum(a, b)          case "-":              res = sub(a, b)          case "\*":              res = mult(a, b)          case "/":              res = div(a, b)          case "^":              res = deg(a, b)      print("Result:", res)  while True:      a, b = inp()      calc(a, b)      if (input("Enter exit to end the programm: ")) == "exit":          exit(0) |

Звіт до Теми №6

Робота з файлами

Під час виконання практичного завдання до Теми №6 було надано варіанти рішення до наступних задач:

1. **Розробити механізм логування всіх дій, що виконує програма**

Забезпечити зберігання інформації про введені данні, виконану операцію та результат виконання операції над даними.

**Попередні умови:** Реалізована програма калькулятор в файлі calc.py, до якого підключають файл functions.py та operations.py. Інструкції до оформлення вмісту файлів розміщенні в завданні 3 до теми 5.

Хід виконання завдання:

Як і було вказано, я взяв за основу програма калькулятор до теми №5. Зміни зачіпили лише файл calc. Додано: логування часу, точної дати, введених значень, обраних інструментів, результату обчислень і кількість витраченого часу на обчислення.

Текст програми calc:

|  |
| --- |
| from operations import inp, tool  from functions import sum, sub, mult, div, deg  from datetime import datetime  import time  import os  def calc(a, b):      act = tool()      start\_act\_time = time.perf\_counter()      match act:          case "+":              res = sum(a, b)          case "-":              res = sub(a, b)          case "\*":              res = mult(a, b)          case "/":              res = div(a, b)          case "^":              res = deg(a, b)      end\_act\_time = time.perf\_counter()      elapsed\_time = end\_act\_time - start\_act\_time      print("Result:", res)      file\_path = os.path.join(os.path.dirname(\_\_file\_\_), "log.log")      act\_time = datetime.now().strftime("%A %Y-%m-%d %H:%M:%S")      with open(file\_path, "a") as file:          file.write(f"{act\_time} | {a} {act} {b} = {res} | Elapsed: {elapsed\_time:.9f} seconds\n")  while True:      a, b = inp()      calc(a, b)      if (input("Enter exit to end the programm: ")) == "exit":          exit(0) |

Текст програм functions та operations залишились аналогічними цим же програмам до теми №5.

1. **Сортування функцію sorted()**

Маючи не відсортований список, елементами якого є словники з двома параметрами (ім’я та оцінка) виконати сортування списку, використовуючи стандартну функцію sorted(). Другим параметром для функції sorted() має бути lambda функція, що повертає ім’я або оцінку із елемента словника.

Хід виконання завдання:

Є не відсортований список словників студентів, нескінченний цикл що отримує від користувача вказівки та фінкція сортування, до якої звертається цикл з певним ключем, за яким проходить подальше сортування.

Текст програми:

|  |
| --- |
| students = [      {'name': "Richard", 'mark': 84},      {'name': "Gogi", 'mark': 65},      {'name': "James", 'mark': 98},      {'name': "Bob", 'mark': 23},      {'name': "Biden", 'mark': 87},      {'name': "Jon", 'mark': 90}  ]  def sort(key):      for elem in sorted(students, key=lambda x: x[key]):          print(f"Name — {elem['name']} | Mark — {elem['mark']}")  while True:      act = input("Select sorting: N — name, M — mark, or 'exit': ").lower()      match act:          case "exit":              break          case "n":              sort('name')          case "m":              sort('mark')          case \_:              print("Selection error.\n") |

Посилання на github:

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

Знімок екрану з посилання на github:

