**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:

Звіт до Теми №4

Виняткові ситуації

**Калькулятор:**

**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:

Додав перевірку виняткових ситуацій до існуючего калькулятора.

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

Звіт до Теми №5

Бібліотеки

**Гра з комп’ютером:**

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

Хід виконання завдання:

Import бібліотеки random, створив фунцію запиту даних від користувача.

Код програми:

|  |
| --- |
| import random  def userChoice():      user\_choice = input("R for rock, S for scissors, P for paper: ")      if user\_choice == "R":          return "rock"      elif user\_choice == "S":          return "scissors"      elif user\_choice == "P":          return "paper"      else:          print("Wrong input!")  options = ["rock", "scissors", "paper"]  computer\_choice = random.choice(options)  def logic(userChoice, computerChoice):      if userChoice == computerChoice:          print("Draw!")      elif userChoice == "scissors" and computer\_choice == "paper" or userChoice == "paper" and computer\_choice == "rock" or userChoice == "rock" and computer\_choice == "scissors":          print("Computer chosen: " , computerChoice)          print("User wins!")      else:          print("Computer chosen: " , computerChoice)          print("Computer wins!")  def main():      while True:          trigger = input("Do you want to continue?[Y/N]").upper()          if trigger == "N":              break          elif trigger == "Y":              logic(userChoice(), computer\_choice)          else:              print("Try again!")  main() |

**Конвертер валют:**

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:

За допомогою запиту request, отримав дані від НБУ. Запит від користувача, порівняння даних та вивід конвертованої валюти.

Код програми:

|  |
| --- |
| import requests  request = requests.get("https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?json")  currency = input("\nEnter currency(1, 2 or 3):\n 1. USD \n 2. EUR \n 3. PLN \n")  if currency == "1":      currency = "USD"  elif currency == "2":      currency = "EUR"  elif currency == "3":      currency = "PLN"  else:      print("Wrong input!")    for elem in request.json():      if elem["cc"] == currency:          rate = elem["rate"]          break  amount = float(input("Enter amount of currency to convert to UAH: "))  print(amount, currency, " = ", rate\*amount, "UAH") |

**Модулі для калькулятора**

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:

Переніс функції в functions.py та operations.py. В головному файлі викликав їх.

Код functions.py:

|  |
| --- |
| def addition(a, b):      return a + b  def subtraction(a, b):      return a - b  def multiplication(a, b):      return a \* b  def division(a, b):      if b == 0:          raise ZeroDivisionError("Division by 0 is forbidden.")      return a / b |

Код operations.py:

|  |
| --- |
| def get\_numbers():      try:          a = float(input("Enter a: "))          b = float(input("Enter b: "))          return a, b      except ValueError:          print("Wrong input!")    def get\_operator():      return input("Enter action (+, -, \* or /): ")  def continue\_calculations():      trigger = input("Do you want to continue? [Y/N]").upper()      return trigger == "Y" |

Код calc.py:

|  |
| --- |
| from functions import addition, subtraction, multiplication, division  from operations import get\_numbers, get\_operator, continue\_calculations  def calculator(a, b, action):      if action == "+":          return addition(a, b)      elif action == "-":          return subtraction(a, b)      elif action == "\*":          return multiplication(a, b)      elif action == "/":          return division(a, b)      else:          print("Invalid operator.")    def main():      while True:          a, b = get\_numbers()          action = get\_operator()          result = calculator(a, b, action)          print(f"Result is {result}")          if not continue\_calculations():              break  if \_\_name\_\_ == "\_\_main\_\_":      main() |

Звіт до Теми №6

Робота з файлами

**Калькулятор з логуванням:**

Розробити механізм логування всіх дій, що виконує програма. Забезпечити зберігання інформації про введені данні, виконану операцію та результат виконання операції над даними.

Хід виконання завдання:

Створив запис повідомлень про помилки в файл calc.log. Програма записує повідослення про неправильний ввід чисел, оператора, помилку про ділення на 0 та про успішний результат виконання операції. Переніс помилку про ділення на 0 в calc.py.

Код програми calc.py:

|  |
| --- |
| from functions import addition, subtraction, multiplication, division  from operations import get\_numbers, get\_operator, continue\_calculations  import os  current\_directory = os.getcwd()  path\_to\_logs = os.path.join(current\_directory, 'topic\_06', 'calculator\_logging', 'calc.log')  try:      file = open(path\_to\_logs, 'a')  except IOError as e:      print(e)  def calculator(a, b, action):      try:          if action == "+":              return addition(a, b)          elif action == "-":              return subtraction(a, b)          elif action == "\*":              return multiplication(a, b)          elif action == "/":              return division(a, b)          else:              error\_message = "Invalid operator: " + action + "."              print(error\_message + '\n')              file.write(error\_message + '\n')      except ZeroDivisionError:          print("Division by 0 is forbidden.")          file.write('User tried to divide by zero.\n')    def main():      while True:          a, b = get\_numbers()          if a is None or b is None:              file.write('User entered wrong value when entering numbers.\n')              continue          action = get\_operator()          result = calculator(a, b, action)          if result is not None:              print(f"Result is {result}")              log\_message = f'User entered numbers {a} and {b}, action "{action}", received the result: {result}.'              file.write(log\_message + '\n')          if not continue\_calculations():              break  if \_\_name\_\_ == "\_\_main\_\_":      main()      file.close() |

**Сортування списку з lambda-функцією:**

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:

Створив програму що розбиває вміст текстового файлу. Після цього програма сортує список за ключем, у якому зберігається лямбда-функція, що вказує на оцінку студента.

Код програми:

|  |
| --- |
| students = []  file = open("topic\_06\students.txt")  for line in file:      name, mark = line.rstrip().split(",")        student = {}      student["name"] = name      student["mark"] = mark      students.append(student)  for student in sorted(students, key=lambda x: x["mark"]):      print(f"Name is {student['name']} mark is {student['mark']}") |

Звіт до Теми №7

ООП

**Методи класу:**

Ознайомитись з існуючими за замовченням методами класу по типу \_\_init\_\_(self) \_\_str\_\_(self)\_\_ та надати приклади використання.

Хід виконання завдання:

\_\_init\_\_ - це спеціальний метод в об'єктно-орієнтованому програмуванні, який викликається автоматично при створенні нового об'єкта класу. Цей метод використовується для ініціалізації атрибутів об'єкта і встановлення початкових значень. Він є обов'язковим для класів.

\_\_str\_\_ - це ще один спеціальний метод в Python, який використовується для визначення рядкового представлення об'єкта класу. Коли ви викликаєте функцію str() на об'єкті, який належить до класу, який має метод \_\_str\_\_, цей метод буде автоматично викликаний, і його результат буде повернути як рядкове представлення об'єкта.

Код програми:

|  |
| --- |
| class Student:      def \_\_init\_\_(self, name, age, groupID):          self.name = name          self.age = age          self.groupID = groupID      def \_\_str\_\_(self):          return f'{self.name}, {self.age}, {self.groupID}'    student1 = Student("Pasha", 18, "KB-221")  print(student1) |

**Класс student:**

Розробити клас **Student** атрибутами якого э два параметра **name** та **age**. Створити список елементами якого є об'єкти класу Student. Написати цикл який виводить на екран елементи списку у відсортованому порядку. Для сортування використати стандартну функцію sorted. Функція sorted має використовувати lambda функцію для визначення ключа сортування.

Хід виконання завдання:

Створив новий клас з назвою Student, список що скаладєтся з об’єктів класу.

Відсортував список за віком студентів використовуючи лямбда-функцію.

Код програми:

|  |
| --- |
| class Student:      def \_\_init\_\_(self, name, age) -> None:          self.name = name          self.age = age      def \_\_str\_\_(self) -> str:          return f'{self.name}, {self.age}'    students = [      Student("Andriy", 20),      Student("Pasha", 18),      Student("Denis", 19)      ]  students = sorted(students, key=lambda x: x.age)  for student in students:      print(f"Students name is {student.name}, age is {student.age}") |

**Калькулятор:**

Використовуючи принципи ООП переписати програму Калькулятор. Завдання має бути виконано використовуючи модульний підхід.

Хід виконання завдання:

Переписав модулі калькулятора з додаванням класів, змінив основний файл під зміни до модулів.

Код calc.py:

|  |
| --- |
| import functions  import operations  import os  current\_directory = os.getcwd()  path\_to\_logs = os.path.join(current\_directory, 'topic\_07', 'task3', 'calc.log')  try:      file = open(path\_to\_logs, 'a')  except IOError as e:      print(e)  def calculator(a, b, action):      try:          if action == "+":              return functions.Functions.addition(a, b)          elif action == "-":              return functions.Functions.subtraction(a, b)          elif action == "\*":              return functions.Functions.multiplication(a, b)          elif action == "/":              return functions.Functions.division(a, b)          else:              error\_message = "Invalid operator: " + action + "."              print(error\_message + '\n')              file.write(error\_message + '\n')      except ZeroDivisionError:          print("Division by 0 is forbidden.")          file.write('User tried to divide by zero.\n')  def main():      while True:          a, b = operations.Operations.get\_numbers()          if a is None or b is None:              file.write('User entered wrong value when entering numbers.\n')              continue          action = operations.Operations.get\_operator()          result = calculator(a, b, action)          if result is not None:              print(f"Result is {result}")              log\_message = f'User entered numbers {a} and {b}, action "{action}", received the result: {result}.'              file.write(log\_message + '\n')          if not operations.Operations.continue\_calculations():              break  if \_\_name\_\_ == "\_\_main\_\_":      main()      file.close()w |

Код operations.py:

|  |
| --- |
| class Operations:      @staticmethod      def get\_numbers():          try:              a = float(input("Enter a: "))              b = float(input("Enter b: "))              return a, b          except ValueError:              print("Wrong input!")              return None, None      @staticmethod      def get\_operator():          return input("Enter action (+, -, \* or /): ")      @staticmethod      def continue\_calculations():          trigger = input("Do you want to continue? [Y/N]").upper()          return trigger == "Y" |

Код functions.py:

|  |
| --- |
| class Functions:      @staticmethod      def addition(a, b):          return a + b        @staticmethod      def subtraction(a, b):          return a - b        @staticmethod      def multiplication(a, b):          return a \* b        @staticmethod      def division(a, b):          return a / b |