Міністерство освіти і науки України

Національний університет «Чернігівська політехніка»

Навчально-науковий інститут електронних та інформаційних технологій

Кафедра кібербезпеки та математичного моделювання

**Звіт про виконання практичних завдань**

**до лекцій з курсу:**

**«Технології програмування на мові Python»**

виконав

студент групи КБ-232

Тітенко Д. А.

перевірив

Дюба І.М\_\_\_\_\_\_\_\_\_

2024 р

**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".

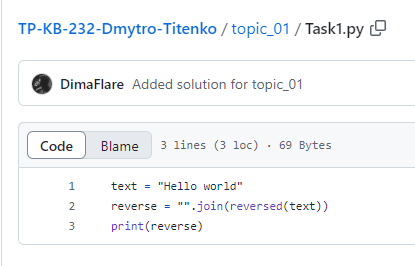
Хід виконання завдання:

Текст програми:

|  |
| --- |
| text = "Hello world" reverse = "".join(reversed(text)) print(reverse) |

Посилання на github: <https://github.com/DimaFlare/TP-KB-232-Dmytro-Titenko/blob/main/topic_01/Task1.py>

Знімок екрану з посилання на github:



**Тестування функцій**

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

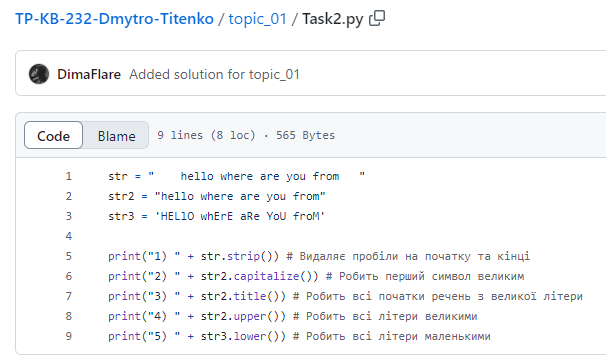
Хід виконання завдання:

Текст програми:

|  |
| --- |
| str = " hello where are you from " str2 = "hello where are you from" str3 = 'HELlO whErE aRe YoU froM'  print("1) " + str.strip()) # Видаляє пробіли на початку та кінці print("2) " + str2.capitalize()) # Робить перший символ великим print("3) " + str2.title()) # Робить всі початки речень з великої літери print("4) " + str2.upper()) # Робить всі літери великими print("5) " + str3.lower()) # Робить всі літери маленькими |

Посилання на github: <https://github.com/DimaFlare/TP-KB-232-Dmytro-Titenko/blob/main/topic_01/Task2.py>

Знімок екрану з посилання на github:



**Написання функцій**

Написати функцію пошуку дискримінанту квадратного рівняння

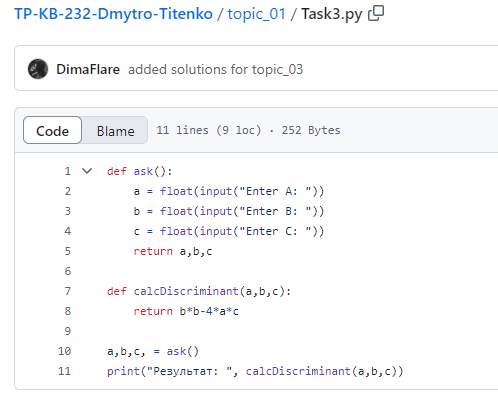
Хід виконання завдання:

Текст програми:

|  |
| --- |
| def ask():  a = float(input("Enter A: "))  b = float(input("Enter B: "))  c = float(input("Enter C: "))  return a,b,c  def calcDiscriminant(a,b,c):  return b\*b-4\*a\*c  a,b,c, = ask() print("Результат: ", calcDiscriminant(a,b,c)) |

Посилання на github: <https://github.com/DimaFlare/TP-KB-232-Dmytro-Titenko/blob/main/topic_01/Task3.py>

Знімок екрану з посилання на github:



**Звіт до Теми №2**

**Умовний перехід**

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

**Написання функцій**

Написати функцію пошуку коренів квадратного рівняння використовуючи функцію розрахунку дискримінанту з попередньої теми та умовні переходи

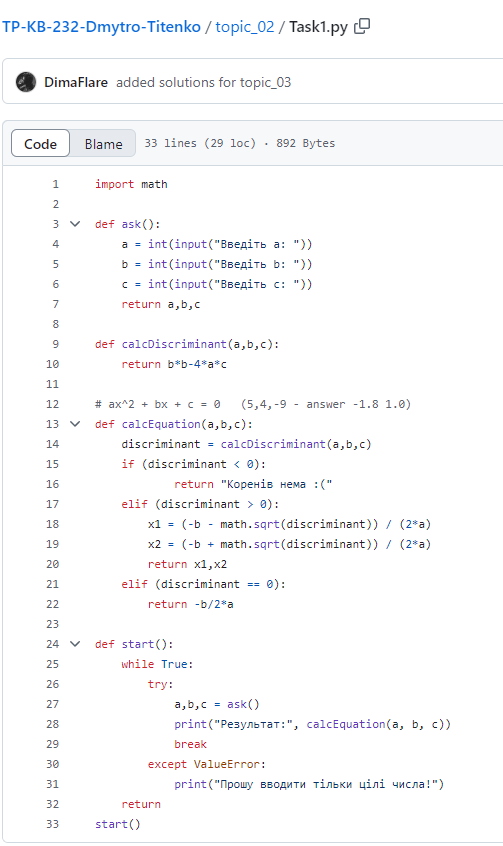
Хід виконання завдання:

Текст програми:

|  |
| --- |
| import math  def ask():  a = int(input("Введіть a: "))  b = int(input("Введіть b: "))  c = int(input("Введіть c: "))  return a,b,c  def calcDiscriminant(a,b,c):  return b\*b-4\*a\*c  # ax^2 + bx + c = 0 (5,4,-9 - answer -1.8 1.0) def calcEquation(a,b,c):  discriminant = calcDiscriminant(a,b,c)  if (discriminant < 0):  return "Коренів нема :("  elif (discriminant > 0):  x1 = (-b - math.sqrt(discriminant)) / (2\*a)  x2 = (-b + math.sqrt(discriminant)) / (2\*a)  return x1,x2  elif (discriminant == 0):  return -b/2\*a  def start():  while True:  try:  a,b,c = ask()  print("Результат:", calcEquation(a, b, c))  break  except ValueError:  print("Прошу вводити тільки цілі числа!")  return start() |

Посилання на github: <https://github.com/DimaFlare/TP-KB-232-Dmytro-Titenko/blob/main/topic_02/Task1.py>

Знімок екрану з посилання на github:



**Написання програми калькулятор**

Написати програму калькулятор використовуючи **if else match** конструкції. Кожна операція має бути виконана в окремій функції.

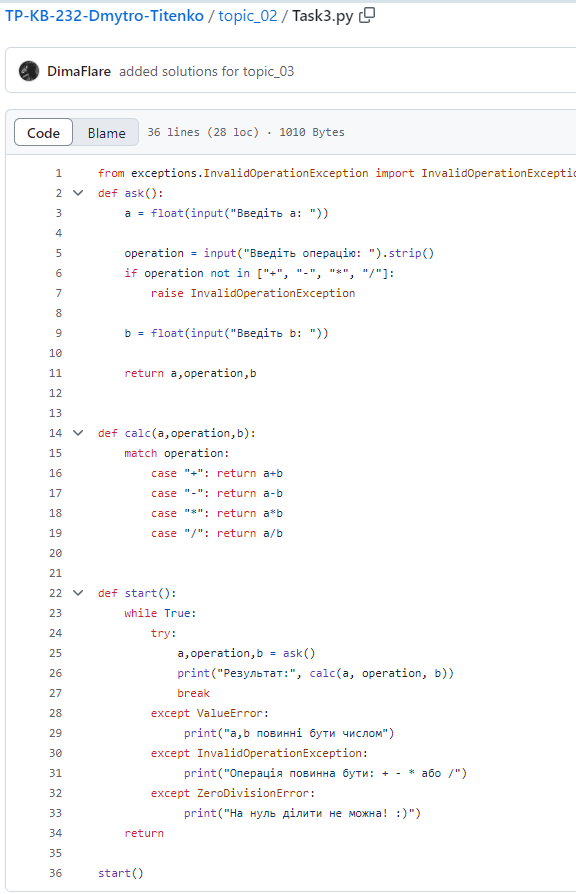
Хід виконання завдання:

Текст програми:

|  |
| --- |
| from exceptions.InvalidOperationException import InvalidOperationException def ask():  a = float(input("Введіть a: "))   operation = input("Введіть операцію: ").strip()  if operation not in ["+", "-", "\*", "/"]:  raise InvalidOperationException   b = float(input("Введіть b: "))   return a,operation,b   def calc(a,operation,b):  if (operation == "+"):  return a+b  elif (operation == "-"):  return a-b  elif (operation == "\*"):  return a\*b  elif (operation == "/"):  return a/b  def start():  while True:  try:  a,operation,b = ask()  print("Результат:", calc(a, operation, b))  break  except ValueError:  print("a,b повинні бути числом")  except InvalidOperationException:  print("Операція повинна бути: + - \* або /")  except ZeroDivisionError:  print("На нуль ділити не можна! :)")  return  start() |

Посилання на github: <https://github.com/DimaFlare/TP-KB-232-Dmytro-Titenko/blob/main/topic_02/Task3.py>

Знімок екрану з посилання на github:



**Звіт до Теми №3**

**Цикли**

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

**Написання програми**

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

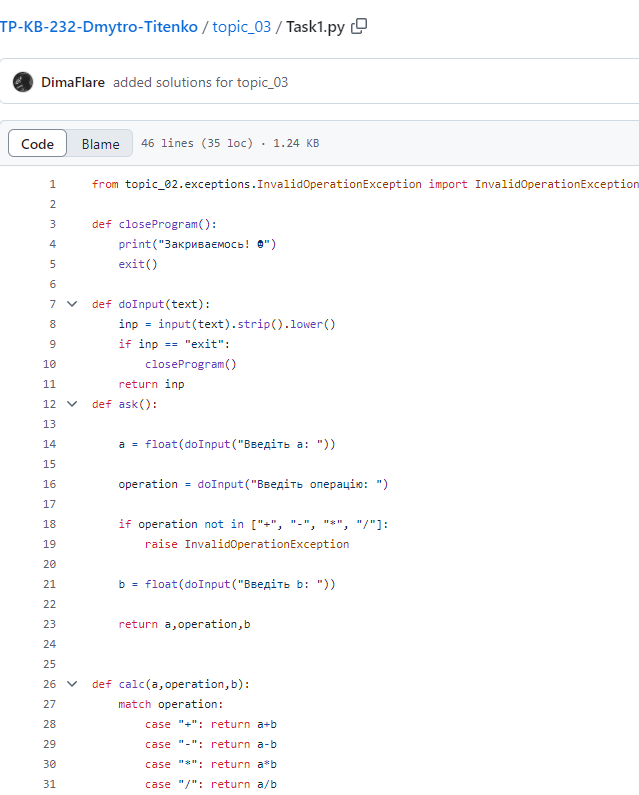
Хід виконання завдання:

Текст програми:

|  |
| --- |
| from topic\_02.exceptions.InvalidOperationException import InvalidOperationException  def closeProgram():  print("Закриваємось! ☻")  exit()  def doInput(text):  inp = input(text).strip().lower()  if inp == "exit":  closeProgram()  return inp def ask():   a = float(doInput("Введіть a: "))   operation = doInput("Введіть операцію: ")   if operation not in ["+", "-", "\*", "/"]:  raise InvalidOperationException   b = float(doInput("Введіть b: "))   return a,operation,b   def calc(a,operation,b):  match operation:  case "+": return a+b  case "-": return a-b  case "\*": return a\*b  case "/": return a/b  def start():  while True:  print("\nНапишіть exit щоб завершити програму!")  try:  a,operation,b = ask()  print("Результат:", calc(a, operation, b))  except ValueError:  print("a,b повинні бути числом")  except InvalidOperationException:  print("Операція повинна бути: + - \* або /")  except ZeroDivisionError:  print("На нуль ділити не можна! :)")  start() |

Посилання на github: <https://github.com/DimaFlare/TP-KB-232-Dmytro-Titenko/blob/main/topic_03/Task1.py>

Знімок екрану з посилання на github:



**Тестування функцій**

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

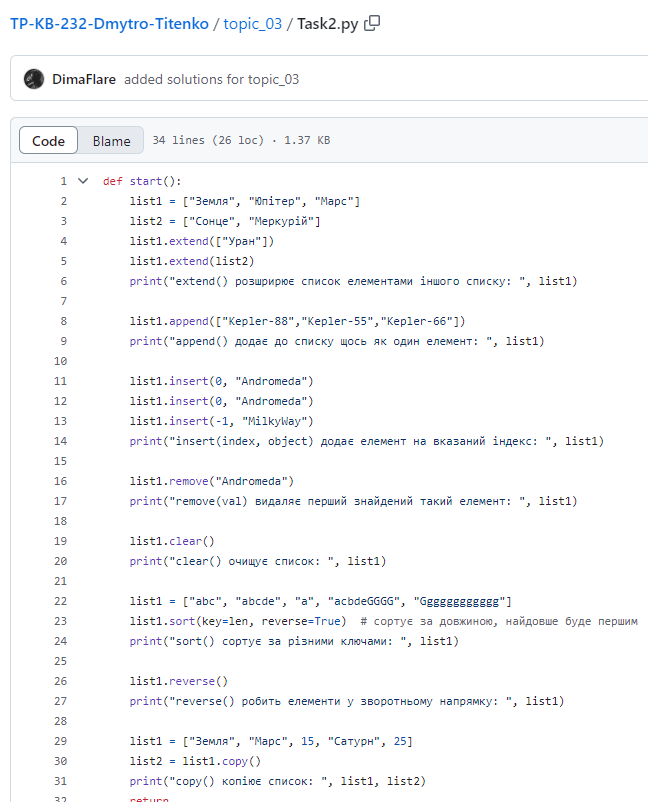
Хід виконання завдання:

Текст програми:

|  |
| --- |
| def start():  list1 = ["Земля", "Юпітер", "Марс"]  list2 = ["Сонце", "Меркурій"]  list1.extend(["Уран"])  list1.extend(list2)  print("extend() розшрирює список елементами іншого списку: ", list1)   list1.append(["Kepler-88","Kepler-55","Kepler-66"])  print("append() додає до списку щось як один елемент: ", list1)   list1.insert(0, "Andromeda")  list1.insert(0, "Andromeda")  list1.insert(-1, "MilkyWay")  print("insert(index, object) додає елемент на вказаний індекс: ", list1)   list1.remove("Andromeda")  print("remove(val) видаляє перший знайдений такий елемент: ", list1)   list1.clear()  print("clear() очищує список: ", list1)   list1 = ["abc", "abcde", "a", "acbdeGGGG", "Gggggggggggg"]  list1.sort(key=len, reverse=True) # сортує за довжиною, найдовше буде першим  print("sort() сортує за різними ключами: ", list1)   list1.reverse()  print("reverse() робить елементи у зворотньому напрямку: ", list1)   list1 = ["Земля", "Марс", 15, "Сатурн", 25]  list2 = list1.copy()  print("copy() копіює список: ", list1, list2)  return  start() |

Посилання на github: <https://github.com/DimaFlare/TP-KB-232-Dmytro-Titenko/blob/main/topic_03/Task2.py>

Знімок екрану з посилання на github:



**Тестування функцій**

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

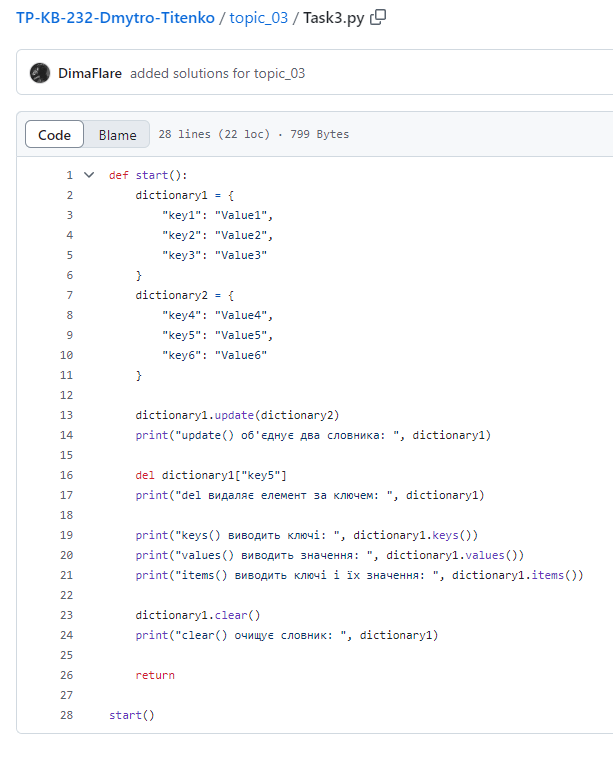
Хід виконання завдання:

Текст програми:

|  |
| --- |
| def start():  dictionary1 = {  "key1": "Value1",  "key2": "Value2",  "key3": "Value3"  }  dictionary2 = {  "key4": "Value4",  "key5": "Value5",  "key6": "Value6"  }   dictionary1.update(dictionary2)  print("update() об'єднує два словника: ", dictionary1)   del dictionary1["key5"]  print("del видаляє елемент за ключем: ", dictionary1)   print("keys() виводить ключі: ", dictionary1.keys())  print("values() виводить значення: ", dictionary1.values())  print("items() виводить ключі і їх значення: ", dictionary1.items())   dictionary1.clear()  print("clear() очищує словник: ", dictionary1)   return  start() |

Посилання на github: <https://github.com/DimaFlare/TP-KB-232-Dmytro-Titenko/blob/main/topic_03/Task3.py>

Знімок екрану з посилання на github:



**Робота зі списками та написання функцій**

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

Хід виконання завдання:

Текст програми:

|  |
| --- |
| def insert(list, a):  if not list:  list.extend([a])  else:  for e in list:  if a >= list[len(list)-1]:  list.insert(len(list), a)  break  elif a <= e:  list.insert(list.index(e), a)  break  return  def start():  list = []  print("Список:", list)   while True:  try:  a = int(input("Введіть а: "))  except ValueError:  print("a повинно бути цілим числом")  continue   insert(list,a)  print("Список:",list)   return  start() |

Посилання на github: <https://github.com/DimaFlare/TP-KB-232-Dmytro-Titenko/blob/main/topic_03/Task4.py>

Знімок екрану з посилання на github:



**Звіт до Теми №4**

**Виняткові ситуації**

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

**Удосконалення коду**

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

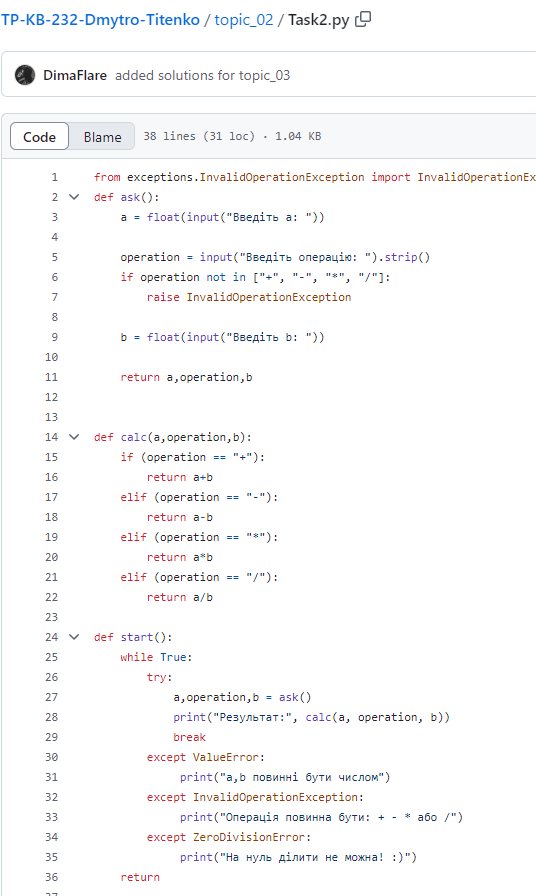
Хід виконання завдання:

Текст програми:

|  |
| --- |
| from exceptions.InvalidOperationException import InvalidOperationException def ask():  a = float(input("Введіть a: "))   operation = input("Введіть операцію: ").strip()  if operation not in ["+", "-", "\*", "/"]:  raise InvalidOperationException   b = float(input("Введіть b: "))   return a,operation,b   def calc(a,operation,b):  if (operation == "+"):  return a+b  elif (operation == "-"):  return a-b  elif (operation == "\*"):  return a\*b  elif (operation == "/"):  return a/b  def start():  while True:  try:  a,operation,b = ask()  print("Результат:", calc(a, operation, b))  break  except ValueError:  print("a,b повинні бути числом")  except InvalidOperationException:  print("Операція повинна бути: + - \* або /")  except ZeroDivisionError:  print("На нуль ділити не можна! :)")  return  start() |

Посилання на github: <https://github.com/DimaFlare/TP-KB-232-Dmytro-Titenko/blob/main/topic_02/Task2.py>

Знімок екрану з посилання на github:



**Звіт до Теми №5**

**Бібліотеки**

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

**Написання гри**

Гра з комп’ютером: камінь, ножиці, папір. Програма 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:

Текст програми:

|  |
| --- |
| import random  choices = ("rock", "paper", "scissors")  beat = {  "rock":"scissors",  "scissors":"paper",  "paper":"rock"  }  def ask():  while True:  choice = input("\nОберіть хід: ").strip().lower()  if choice == "exit":  print("Завершую гру")  exit()   if choice in choices:  break  else:  print("Хід повинен бути одним із:", choices)  continue   return choice  def play(choice):  randomChoice = choices[random.randint(0, len(choices)-1)]  if choice == randomChoice:  return "Нічия. Суперник вибрав: " + randomChoice  elif randomChoice == beat[choice]:  return "Ви виграли. Суперник вибрав: " + randomChoice  else:  return "Ви програли. Суперник вибрав: " + randomChoice   return  def start():  while True:  print(play(ask()))  return  start() |

Посилання на github: <https://github.com/DimaFlare/TP-KB-232-Dmytro-Titenko/blob/main/topic_05/RockPaperScissors.py>

Знімок екрану з посилання на github:



**Написання програми**

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.

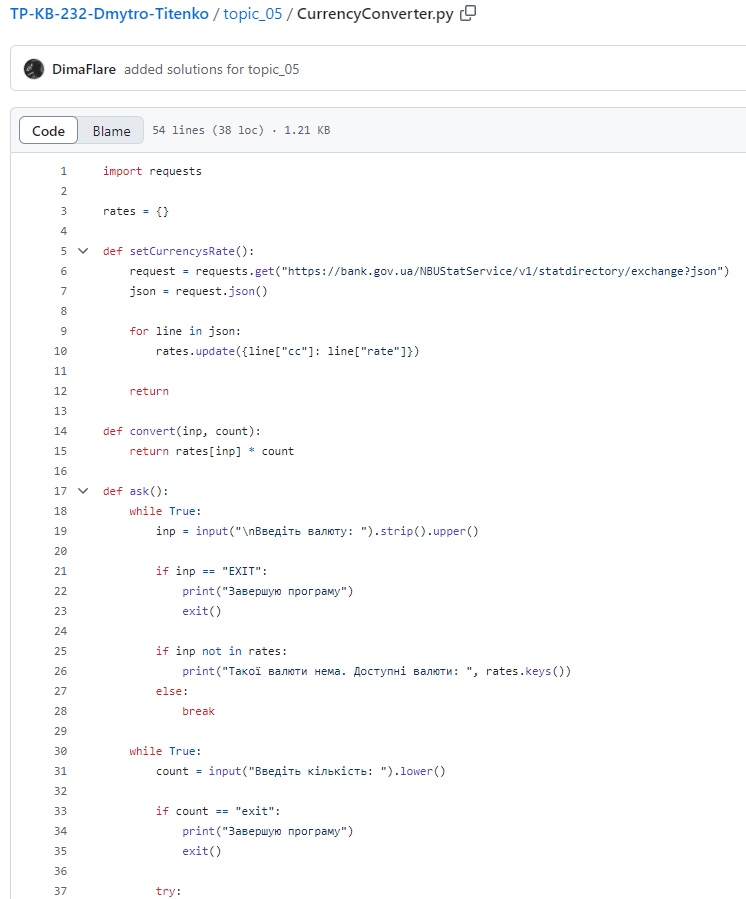
Хід виконання завдання:

Текст програми:

|  |
| --- |
| import requests  rates = {}  def setCurrencysRate():  request = requests.get("https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?json")  json = request.json()   for line in json:  rates.update({line["cc"]: line["rate"]})   return  def convert(inp, count):  return rates[inp] \* count  def ask():  while True:  inp = input("\nВведіть валюту: ").strip().upper()   if inp == "EXIT":  print("Завершую програму")  exit()   if inp not in rates:  print("Такої валюти нема. Доступні валюти: ", rates.keys())  else:  break   while True:  count = input("Введіть кількість: ").lower()   if count == "exit":  print("Завершую програму")  exit()   try:  count = float(count)  break  except ValueError:  print("Кількість повинна бути числом!")   return inp, count  def start():  setCurrencysRate()   while True:  inp,count = ask()  print(str(count) + " " + inp + " в гривнях:", convert(inp, count))   return  start() |

Посилання на github: <https://github.com/DimaFlare/TP-KB-232-Dmytro-Titenko/blob/main/topic_05/CurrencyConverter.py>

Знімок екрану з посилання на github:



**Удосконалення коду**

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:

Текст calc.py:

|  |
| --- |
| from calcmodules.Functions import calc from calcmodules.Operations import ask from topic\_02.exceptions.InvalidOperationException import InvalidOperationException from topic\_05.calcmodules.Log import makeLog  def start():  while True:  print("\nНапишіть exit щоб завершити програму!")  try:  a,operation,b = ask()  result = calc(a, operation, b)  print("Результат:", result)  makeLog(a,operation,b,result)  except ValueError:  print("a,b повинні бути числом")  except InvalidOperationException:  print("Операція повинна бути: + - \* або /")  except ZeroDivisionError:  print("На нуль ділити не можна! :)")  start() |

Текст functions.py:

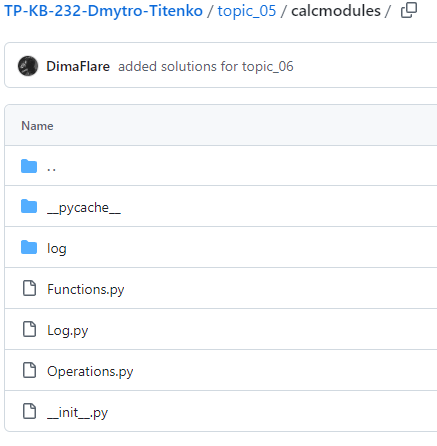
|  |
| --- |
| def calc(a,operation,b):  match operation:  case "+": return a+b  case "-": return a-b  case "\*": return a\*b  case "/": return a/b |

Текст operations.py:

|  |
| --- |
| from topic\_02.exceptions.InvalidOperationException import InvalidOperationException  def closeProgram():  print("Закриваємось! ☻")  exit()  def doInput(text):  inp = input(text).strip().lower()  if inp == "exit":  closeProgram()  return inp def ask():   a = float(doInput("Введіть a: "))   operation = doInput("Введіть операцію: ")   if operation not in ["+", "-", "\*", "/"]:  raise InvalidOperationException   b = float(doInput("Введіть b: "))   return a,operation,b |

Посилання на github: <https://github.com/DimaFlare/TP-KB-232-Dmytro-Titenko/tree/main/topic_05/calcmodules>

Знімок екрану з посилання на github:



**Звіт до Теми №6**

**Робота з файлами**

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

**Удосконалення коду**

Попередні умови: Реалізована програма калькулятор в файлі calc.py, до якого підключають файл functions.py та operations.py.

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

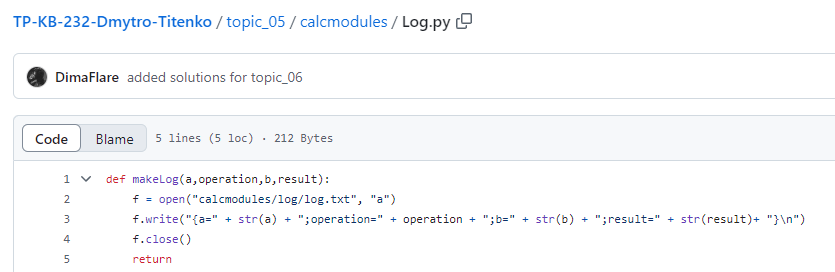
Хід виконання завдання:

Текст програми:

|  |
| --- |
| def makeLog(a,operation,b,result):  f = open("calcmodules/log/log.txt", "a")  f.write("{a=" + str(a) + ";operation=" + operation + ";b=" + str(b) + ";result=" + str(result)+ "}\n")  f.close()  return |

Посилання на github: <https://github.com/DimaFlare/TP-KB-232-Dmytro-Titenko/blob/main/topic_05/calcmodules/Log.py>

Знімок екрану з посилання на github:



**Сортування списків**

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.

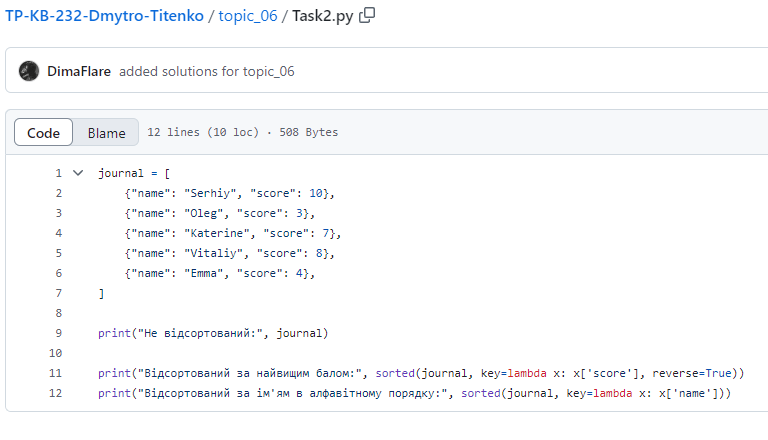
Хід виконання завдання:

Текст програми:

|  |
| --- |
| journal = [  {"name": "Serhiy", "score": 10},  {"name": "Oleg", "score": 3},  {"name": "Katerine", "score": 7},  {"name": "Vitaliy", "score": 8},  {"name": "Emma", "score": 4}, ]  print("Не відсортований:", journal)  print("Відсортований за найвищим балом:", sorted(journal, key=lambda x: x['score'], reverse=True)) print("Відсортований за ім'ям в алфавітному порядку:", sorted(journal, key=lambda x: x['name'])) |

Посилання на github: <https://github.com/DimaFlare/TP-KB-232-Dmytro-Titenko/blob/main/topic_06/Task2.py>

Знімок екрану з посилання на github:



**Звіт до Теми №7**

**Об’єктно-орієнтоване програмування**

Під час виконання практичного завдання до Теми №7 було надано варіанти рішення до наступних задач:

**Ознайомитись з деякими особливостями класів**

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

Хід виконання завдання:

Текст програми:

|  |
| --- |
| class User():   def \_\_init\_\_(self, login, password):  self.login = login  self.password = password   def \_\_str\_\_(self): # Строкове уявлення об'єкту неформального плану, може містити будь яку зручну для розуміння інформацію  return f"Користувач з логіном {self.login} та паролем {self.password}"   def \_\_repr\_\_(self): # Строкове уявлення об'єкту формального плану, зазвичай повинен бути таким, щоб з цього можно було створити об'єкт  return f"User(login='{self.login}', password={self.password})"  Ivan = User("ivanqwerty", "12345")  print(Ivan) print(repr(Ivan)) |

Посилання на github: <https://github.com/DimaFlare/TP-KB-232-Dmytro-Titenko/blob/main/topic_07/Task2.py>

Знімок екрану з посилання на github:



**Розробка класів**

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.

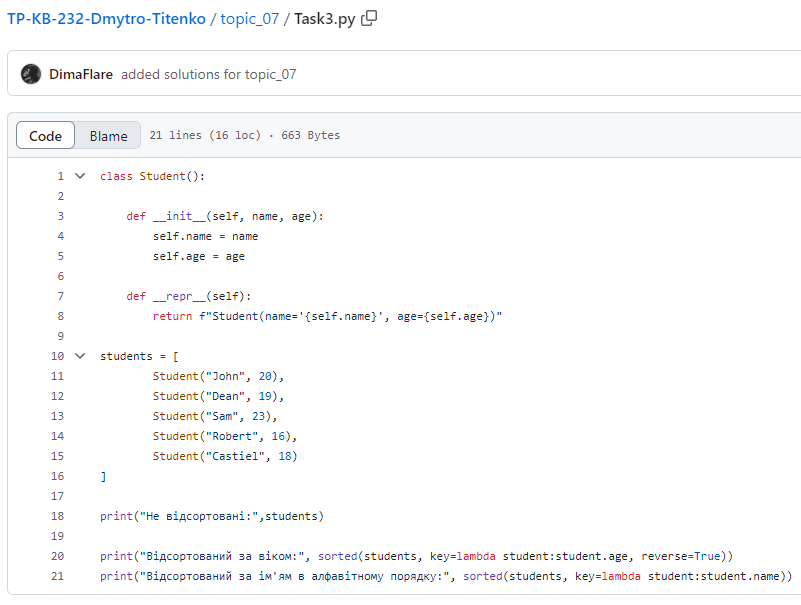
Хід виконання завдання:

Текст програми:

|  |
| --- |
| class Student():   def \_\_init\_\_(self, name, age):  self.name = name  self.age = age   def \_\_repr\_\_(self):  return f"Student(name='{self.name}', age={self.age})"  students = [  Student("John", 20),  Student("Dean", 19),  Student("Sam", 23),  Student("Robert", 16),  Student("Castiel", 18) ]  print("Не відсортовані:",students)  print("Відсортований за віком:", sorted(students, key=lambda student:student.age, reverse=True)) print("Відсортований за ім'ям в алфавітному порядку:", sorted(students, key=lambda student:student.name)) |

Посилання на github: <https://github.com/DimaFlare/TP-KB-232-Dmytro-Titenko/blob/main/topic_07/Task3.py>

Знімок екрану з посилання на github:



**Удосконалення коду**

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

Хід виконання завдання:

Сalculator.py:

|  |
| --- |
| from topic\_02.exceptions.InvalidOperationException import InvalidOperationException from topic\_07.calculator.Handler import Handler from topic\_07.calculator.Interact import Interact  class Calculator:   def start(self):  while True:  print("\nНапишіть exit щоб завершити програму!")  try:  interact = Interact()  a, operation, b = interact.ask()   handler = Handler(a,operation,b)  print("Результат:", handler.calc())   except ValueError:  print("a,b повинні бути числом")  except InvalidOperationException:  print("Операція повинна бути: + - \* або /")  except ZeroDivisionError:  print("На нуль ділити не можна! :)") |

Handler.py:

|  |
| --- |
| class Handler:  def \_\_init\_\_(self, a, operation, b):  self.a = a  self.operation = operation  self.b = b   def calc(self):  match self.operation:  case "+":  return self.a + self.b  case "-":  return self.a - self.b  case "\*":  return self.a \* self.b  case "/":  return self.a / self.b |

Interact.py:

|  |
| --- |
| from topic\_02.exceptions.InvalidOperationException import InvalidOperationException  class Interact:   def closeProgram(self):  print("Закриваємось! ☻")  exit()   def doInput(self, text):  inp = input(text).strip().lower()  if inp == "exit":  self.closeProgram()  return inp   def ask(self):   a = float(self.doInput("Введіть a: "))   operation = self.doInput("Введіть операцію: ")   if operation not in ["+", "-", "\*", "/"]:  raise InvalidOperationException   b = float(self.doInput("Введіть b: "))   return a, operation, b |

Посилання на github: <https://github.com/DimaFlare/TP-KB-232-Dmytro-Titenko/tree/main/topic_07/calculator>

Знімок екрану з посилання на github:

