МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ



Лабораторна робота №1

з дисципліни “Спеціалізовані мови програмування”

на тему

“Введення в Python.”

Виконав:

студент групи ІТ-31

Андрій КОЗУБЕНКО

Прийняв:

к.т.н.,

доц. кафедри ІСМ

Сергій ЩЕРБАК

Львів-2023

**Мета роботи:** Створення консольної програми-калькулятора за допомогою основних синтаксичних конструкцій Python, з іншим завданням на заміну тестуванню та валідації.

**План роботи**

*Завдання 1:* Введення користувача.

*Завдання 2:* Перевірка оператора.

*Завдання 3:* Обчислень.

*Завдання 4:* Повторення обчислень.

*Завдання 5:* Обробка помилок.

*Завдання 6:* Десяткові числа.

*Завдання 7:* Додаткові операції.

*Завдання 8:* Функція пам’яті.

*Завдання 9:* Історія обчислень.

*Завдання 10:* Налаштування користувача.

**Виконання роботи**

**Код програми:**

|  |
| --- |
| import os  import json  import uuid  logs\_path = 'logs.json'  logs = []  def add(augend, addend):      return augend + addend  def subtract(minuend, subtrahend):      return minuend - subtrahend  def multiply(multiplier, multiplicand):      return multiplier \* multiplicand  def divide(numerator, denominator):      if denominator != 0:          return numerator / denominator      else:          print('Division by zero is prohibited')  def power(base, exponent):      return pow(base, exponent)  def root(radicand, degree):      return radicand\*\*(1/degree)  def modulo(dividend, divisor):      return dividend % divisor  def get\_operator(valid\_operators = {'+', '-', '\*', '/'}):      operator = input('Select one operator ("+", or "-", or "\*", or "/", or "^", or "rt", or "%"): ')      if operator in valid\_operators:          return operator      else:          print('Invalid operator. Please try again')          return get\_operator(valid\_operators)  def get\_operand():      option = ''      operand\_retrieved = False      while operand\_retrieved == False:          option = input('\nSelect one option:\n[1] Enter a numeric value\n[2] Use the result from logs instead\n')          match option:              case '1':                  try:                      return float(input('Enter a numeric value: '))                  except:                      print('Entered value is not numeric', end='')                      continue              case '2':                  id = input('Enter the ID of the log: ')                  desired\_log = None                  for log in logs:                      if log['id'].startswith(id):                          desired\_log = log                          break                  if desired\_log == None:                      print('Log with such ID was not found', end='')                      continue                  else:                      print('Value retrieved:', desired\_log['result'])                      return desired\_log['result']                case '\_':                  print('Wrong option. Please try again')  def get\_operands(operator, operators\_and\_operand\_names):      print('\nInput ', operators\_and\_operand\_names[operator][0], '(', operators\_and\_operand\_names[operator][2], ')')      first\_operand = get\_operand()      print('\nInput ', operators\_and\_operand\_names[operator][1], '(', operators\_and\_operand\_names[operator][2], ')')      second\_operand = get\_operand()      return [first\_operand, second\_operand]  def perform\_operation(operators\_and\_functions, operator, operands):      return operators\_and\_functions[operator](operands[0], operands[1])  def determine\_continuation():      decision = input('Continue? Y/N: ')      if decision.lower() == 'y' or decision.lower() == 'yes':          return True      return False  def initialize\_logs\_file(logs\_path = 'logs.json'):      if os.path.isfile(logs\_path) == False:          logs\_file = open(logs\_path, 'w')          logs\_file.write('[]')          logs\_file.close()  def get\_logs(logs\_path = 'logs.json', ignore\_emptry = True):      logs\_file = open(logs\_path)      logs\_string = logs\_file.read()      logs\_file.close()      if len(logs\_string) != 0:           return json.loads(logs\_string)      else:          return []  def append\_logs(operands, operator, result, logs):      logs.append({"id": str(uuid.uuid4()), "operands": operands, "operator": operator, "result": result})  def save\_logs\_file(logs\_path, logs):      logs\_file = open(logs\_path, 'w')      logs\_file.write(json.dumps(logs))      logs\_file.close()  operators\_and\_functions = {      '+': add,      '-': subtract,      '\*': multiply,      '/': divide,      '^': power,      'rt': root,      '%': modulo  }  operators\_and\_operand\_names = {      '+':  ['augend', 'addend', 'augend + addend'],      '-':  ['minuend', 'subtrahend', 'minuend - subtrahend'],      '\*':  ['multiplier', 'multiplicand', 'multiplier \* multiplicand'],      '/':  ['numerator', 'denominator', 'numerator / denominator'],      '^':  ['base', 'exponent', 'base ^ exponent'],      'rt': ['radicand', 'degree', 'radicand ^ (1 / degree)'],      '%':  ['dividend', 'divisor', 'dividend mod divisor']  }  def perform\_calculation():      continue\_calculation = True      while continue\_calculation == True:          operator = get\_operator(list(operators\_and\_functions.keys()))          operands = get\_operands(operator, operators\_and\_operand\_names)          result = perform\_operation(operators\_and\_functions, operator, operands)          print('Result: ', result)          append\_logs(operands, operator, result, logs)          continue\_calculation = determine\_continuation()  def display\_logs():      print('\n\nLogs:')      for log in logs:          print('id:', log['id'])          print('expression:', log['operands'][0], log['operator'], log['operands'][1], '=', log['result'], end='\n\n')  def load\_logs():      path = input('Specify path to logs file: ')      if os.path.isdir(path) == True:          print('Specified file is a directory. Try again')          load\_logs()      if os.path.isfile(path) == True:          global logs          logs = get\_logs(path)          global logs\_path          logs\_path = path          print('Successfully retrieved the logs')      else:          print('Specified file does not exist')  def exit\_program():      save\_logs\_file(logs\_path, logs)  options\_and\_functions = {      '1': perform\_calculation,      '2': display\_logs,      '3': load\_logs,      '0': exit\_program  }  def main():      option = ''      initialize\_logs\_file(logs\_path)      global logs      logs = get\_logs(logs\_path)      while option != '0':          option = input('\nSelect one option:\n[1] - Perform calculation\n[2] - Display logs\n[3] - Load logs\n\n[0] - Exit\n')          options\_and\_functions[option]()  main() |

**Посилання на GitHub:** <https://github.com/Famezzs/uni-spec-lang-labs.git>

**Висновок:** Виконавши ці завдання, я створив простий консольний калькулятор Python, який може виконувати арифметичні операції, обробляти помилки та надавати користувачу зручний інтерфейс. Цей проект допоміг вивчити основний синтаксис Python і концепції, такі як введення користувача, умовні оператори, цикли та обробка помилок.