Технології програмування

Лабораторна робота №2

**Робота з файлами. Юніт тести.**

**Мета роботи**: Використовуючи теоретичне підґрунтя про роботу з файлами та тестування коду у мові Python розширити програму телефонного довідника студентів додавши функціонал, що буде вказано в завданні до лабораторної роботи.

**Теоретичні відомості**

**Параметри командного рядка**

Одним із механізмом визначення параметрів необхідних для виконання програми – використання аргументів командного рядка. Для можливості використання аргументів командного необхідно підключити модуль **argv**

Розглянемо приклад:

|  |
| --- |
| from sys import argv  print(f"Script name: {argv[0]}")  print(f"Input parameter: {argv[1]}") |

В самому початку підключаються модуль **argv**, зо забезпечує можливість використання параметрів командного рядка. Все, що було вказано користувачем під час запуску програми зберігається в список доступ до елементів якого здійснюється використовуючи індекси.

Запустивши програму на виконання наступним чином:

|  |
| --- |
| python lab2.py lab2.csv |

Отримаємо результат:

|  |
| --- |
| Script name: lab2.py  Input parameter: lab2.csv |

Під нульовим індексом завжди зберігається ім’я програми, яка запускається. Починаючи з індексу один розміщуються параметри які буди вказані під час запуску програми на виконання.

**Робота з файлами**

Для роботи з файлами існують набір стандартних функцій, тож для відкриття, закриття, читання та запису інформації достатньо використовувати готовий функціонал.

Для того, щоб відкрити файл необхідно виконати функцію open() ([https://docs.python.org/3/library/functions.html#open](https://docs.python.org/3/library/functions.html" \l "open)). Функція повертає посилання на файл, яке можна використовувати для наступних маніпуляцій. При закінченні використання файлу, його необхідно закрити.

|  |
| --- |
| file\_name = argv[1]  file = open(file\_name, "r")  file.close() |

Проте, використовуючи ключове слово **with** можна не використовувати **close()** в кінці використання файлу.

Після відкриття файлу існує можливість пройти всі рядки, що в ньому збережені. Розглянемо приклад:

|  |
| --- |
| file\_name = argv[1]  with open(file\_name, "r") as file:  for line in file:  print(f"line from file: {line}") |

Отримаємо результат

|  |
| --- |
| line from file: Name,Phone  line from file: Bob,1112233  line from file: Dilan,2223344  line from file: Zak,3334455 |

Для запису даних в файл використовується функція **write()**

|  |
| --- |
| with open(file\_name, "a") as file:  file.write("New student name,1231213") |

Для роботи з файлами формату **CSV** розроблено однойменний модуль, який одразу може сформувати словники з вхідних даних, що збережені в форматі CSV (<https://docs.python.org/3/library/csv.html>).

Функція **DictReader**() забезпечу читання файлу у вигляді словників

Розглянемо приклад:

|  |
| --- |
| file\_name = argv[1]  students = []  with open(file\_name) as file:  reader = csv.DictReader(file)  for row in reader:  students.append({"Name":row["Name"], "Phone":row["Phone"]})  print(students) |

Отримаємо результат

|  |
| --- |
| [{'Name': 'Bob', 'Phone': '1112233'}, {'Name': 'Dilan', 'Phone': '2223344'}, {'Name': 'Zak', 'Phone': '3334455'}] |

Для запису даних в CSV файл необхідно виконати дещо більше дій: відкрити файл для запису, створити обект класу **DictWriter**(), записати верхній рівень з назвою стовбців та безпосередньо записати данні в файл.

|  |
| --- |
| import csv  with open("lab2\_out.csv", "w", newline='') as csvfile:  fieldnames = ["Name", "Age"]  writer = csv.DictWriter(csvfile, fieldnames=fieldnames)  writer.writeheader()  writer.writerow({'Name': 'Ihor', 'Age': '37'}) |

З прикладами програм можна ознайомитись в файлі **lab2\_sample\_io.py**

Приклад CSV файлу **lab2.csv**

**Юніт тести**

Процес виконання завдання може включати розбиття великої задачі на маленькі під задачі, при цьому результатом виконання маленької частина має бути певна кількість функцій, сумарний результат виконання яких задовольняє критеріям виділеної під задачі. Для того, щоб впевнитись, що написаний код виконує саме те, що вказано в завданні, розробник формує набір тестів яким піддається написаний в рамках виконання завдання код. **Тестування коду – написання коду для тестування коду.** При цьому поняття Юніт тест означає тестування окремого функціоналу, наприклад однієї функції (юніту).

Для мові Python існує стороння бібліотека **pytest**,яка реалізує механізм написання тестів. Детальний опис бібліотеки доступний за посиланням <https://docs.pytest.org/en/stable/>

Для початку використання бібліотеки необхідно виконати інсталяцію

|  |
| --- |
| pip install pytest |

Розглянемо в якості прикладу завдання на написання програми калькулятор, що запитує у користувача два параметри над якими необхідно виконати дії. В даному прикладі операції додавання та множення винесені в окремі функції. Саме ці функціями будуть виступати окремими юнітами для тестування.

|  |
| --- |
| def add(a, b):  return a + b  def mul(a, b):  return a \* b  def main():  a = int(input("What is a: "))  b = int(input("What is b: "))  print(add(a, b))  print(mul(a, b))  if \_\_name\_\_ == "\_\_main\_\_":  main() |

Перед початком написання тестів необхідно відмітити існування ключового слова **assert**, що надає можливість перевірити на правдивість вказану умову. Також необхідно вказати, згідно конвенції назва юніт тесту має починатися зі слова **test\_**

Наведемо приклад файлу з тестами:

|  |
| --- |
| from lab2\_sample\_calc import add  from lab2\_sample\_calc import mul  def test\_add():  assert add(2, 2) == 4  assert add(3, 2) == 5  assert add(4, 2) == 6  def test\_mul\_positive\_both():  assert mul(2, 2) == 4  assert mul(3, 2) == 6  assert mul(4, 2) == 8  def test\_mul\_positive\_and\_negative():  assert mul(2, -2) == -4  assert mul(-3, 2) == -6  assert mul(4, -2) == -8  def test\_mul\_negative\_both():  assert mul(-2, -2) == 4  assert mul(-3, -2) == 6  assert mul(-4, -2) == 8 |

З самого початку необхідно підключити файл і функції які будуть піддаватись тестуванню. Далі відбувається тестування юніту додавання та юніту множення. Зверніть увагу, що для юніту множення розроблено три незалежних тести.

Приклади розміщені в файлах **lab2\_sample\_calc.py** та **lab2\_sample\_calc\_test.py**

Для запуску тестів використовується наступна команда:

|  |
| --- |
| pytest lab2\_sample\_calc\_test.py |

Приклад результату роботи

A screen shot of a computer

Description automatically generated

**Хід виконання завдання до лабораторної роботи**

1. Прочитав суть завдання.

2. Відкрив графічний редактор.

3. Написав код, який виконує завдання.

4. Написав звіт про виконання завдання.

5. Створив і виконав Юніт тести.

6. Завантажив всі зміни для розгляду в гілку Labs.

Код програми:

|  |
| --- |
| import csv  import sys  def printAllList():  for elem in list:  strForPrint = f"Student name is {elem['name']}, Phone is {elem['phone']}, Age is {elem['age']}, Email is {elem['email']}"  print(strForPrint)  return  def addNewElement():  name = input("Please enter student name: ")  phone = input("Please enter student phone: ")  age = int(input("Please enter student age: "))  email = input("Please enter student email: ")    newItem = {"name": name, "phone": phone, "age": age, "email": email}  insertPosition = 0  for item in list:  if name > item["name"]:  insertPosition += 1  else:  break  list.insert(insertPosition, newItem)  print("New element has been added")  return  def deleteElement():  name = input("Please enter name to be deleted: ")  deletePosition = -1  for item in list:  if name == item["name"]:  deletePosition = list.index(item)  break  if deletePosition == -1:  print("Element was not found")  else:  print(f"Delete position {str(deletePosition)}")  del list[deletePosition]  return  def updateElement():  name = input("Please enter name to be updated: ")  for index, student in enumerate(list):  if name == student["name"]:  print(f"Student {name} found.")  NewName = input("Please enter new name: ")  if NewName and NewName != name:  del list[index]  student["name"] = NewName  insertPosition = 0  for item in list:  if NewName > item["name"]:  insertPosition += 1  else:  break  list.insert(insertPosition, student)  phone = input("Please enter new phone: ")  age = input("Please enter new age: ")  email = input("Please enter new email: ")  student["phone"] = phone  student["age"] = int(age)  student["email"] = email  print("Element has been updated")  return  print("Student not found")  def LoadFile(file):  with open(file, encoding="utf-8") as file:  res = csv.DictReader(file)  return [row for row in res]  def SaveFile(file, list):  names = ["name", "phone", "age", "email"]  try:  with open(file, "w", newline="", encoding="utf-8") as file:  result = csv.DictWriter(file, fieldnames=names)  result.writeheader()  result.writerows(list)  print(f"The data is saved.")  except IOError as e:  print(f"Error saving")  list = [ ]  def main():  if len(sys.argv) > 1:  try:  file = sys.argv[1]  list.extend(LoadFile(file))  print("Data loaded successfully.")  except IOError as e:  print("File upload error.")  else:  print("No CSV file specified.")    while True:  choice = input("Please specify the action [ C create, U update, D delete, P print, S save, X exit ] ")  match choice.upper():  case "C":  print("New element will be created")  addNewElement()  case "U":  print("Existing element will be updated")  updateElement()  case "D":  print("Element will be deleted")  deleteElement()  case "P":  print("List will be printed")  printAllList()  case "S":  file = input("Enter a name for the CSV file: ")  SaveFile(file, list)  case "X":  print("Exit")  break  case \_:  print("Wrong choice")  if \_\_name\_\_ == "\_\_main\_\_":  main() |

Код Юніт тестів:

|  |
| --- |
| import pytest  from lab\_02 import LoadFile, SaveFile, addNewElement, deleteElement, updateElement, printAllList  from pathlib import Path  import csv  from io import StringIO  @pytest.fixture  def testing\_data():  return [  {"name": "Bob", "phone": "1112233", "age": "20", "email": "bob@example.com"},  {"name": "Dilan", "phone": "2223344", "age": "21", "email": "Dilan@example.com"},  {"name": "Zak", "phone": "3334455", "age": "23", "email": "zak@example.com"}  ]  def test\_load\_file(testing\_data, tmp\_path):  path\_file = tmp\_path / "lab2.csv"  with open(path\_file, "w", newline="", encoding="utf-8") as file:  result = csv.DictWriter(file, fieldnames=["name", "phone", "age", "email"])  result.writeheader()  result.writerows(testing\_data)  LoadedData = LoadFile(path\_file)  assert LoadedData == testing\_data  def test\_save\_file(testing\_data, tmp\_path):  path\_file = tmp\_path / "test\_save\_file.csv"  SaveFile(path\_file, testing\_data)  LoadedData = LoadFile(path\_file)  assert LoadedData == testing\_data  def test\_add\_new\_element(tmp\_path, monkeypatch):  lab\_02\_list = []  test\_input\_ans = iter(["Test", "123", "25", "test@example.com"])  def test\_input(prompt):  return next(test\_input\_ans)  monkeypatch.setattr('builtins.input', test\_input)  monkeypatch.setattr('lab\_02.list', lab\_02\_list)  addNewElement()  assert len(lab\_02\_list) == 1  assert lab\_02\_list[0] == {"name": "Test", "phone": "123", "age": 25, "email": "test@example.com"}  def test\_delete\_element(capsys, monkeypatch):  lab\_02\_list = [{"name": "John", "phone": "123", "age": 25, "email": "john@example.com"}]  test\_input\_ans = iter(["John"])  def test\_input(prompt):  return next(test\_input\_ans)  monkeypatch.setattr('builtins.input', test\_input)  monkeypatch.setattr('lab\_02.list', lab\_02\_list)  deleteElement()  captured = capsys.readouterr()  assert "Del position" in captured.out  def test\_update\_element(tmp\_path, capsys, monkeypatch):  lab\_02\_list = [{"name": "John", "phone": "123", "age": "27", "email": "john@example.com"}]  test\_input\_ans = iter(["John", "Jane", "123", "32", "john@example.com"])  def test\_input(prompt):  return next(test\_input\_ans)  monkeypatch.setattr('builtins.input', test\_input)  monkeypatch.setattr('lab\_02.list', lab\_02\_list)  updateElement()  captured = capsys.readouterr()  assert "Element has been updated" in captured.out  assert lab\_02\_list[0] == {"name": "Jane", "phone": "123", "age": 32, "email": "john@example.com"}  def test\_print\_all\_list(capsys, monkeypatch):  lab\_02\_list = [{"name": "John", "phone": "123", "age": 25, "email": "john@example.com"}]  monkeypatch.setattr('lab\_02.list', lab\_02\_list)  printAllList()  captured = capsys.readouterr()  assert "Student name is John, Phone is 123, Age is 25, Email is john@example.com" in captured.out |