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>). 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

**Завдання до лабораторної роботи**

Розширити функціональність телефонного довідника студентів групи, що був розроблений у Лабораторній роботі №1 наступними можливостями:

1. **завантаження** початкових данні для довідника з csv файлу. Приклад csv файлу (lab2.csv), та оригінальній файл формату xlsx розміщені поряд з вказівками. Ім’я файлу для завантаження даних повинно передаватись в якості параметрів командного рядку.
2. **зберігання** довідника студентів у форматі CSV перед виходом із програми.
3. покриття основного функціоналу програми **Юніт тестами**.

**Хід виконання завдання:**

**Завантаження даних з CSV-файлу:**

* Використовується csv.DictReader() для зчитування даних зі студентського довідника.
* Якщо файл не знайдено, створюється порожній довідник.

**Збереження даних у CSV-файл:**

* Використовується csv.DictWriter() для запису оновленого списку студентів у файл перед виходом із програми.

Відображення всіх студентів:

* Перегляд списку студентів із зазначенням імені, телефону, email та адреси.

**Додавання нового студента:**

* Введення даних користувачем.
* Додавання у довідник із збереженням сортування за ім’ям.

**Видалення студента:**

* Видалення студента за введеним ім’ям.

**Оновлення інформації про студента:**

* Можливість змінити номер телефону, email або адресу студента.

**Інтерактивне меню:**

* Вибір дій користувачем (перегляд, додавання, видалення, оновлення, збереження та вихід).

**Запуск програми та передача параметрів:**

* Файл CSV передається як аргумент командного рядка.
* Якщо аргумент відсутній, використовується файл lab2.csv.

Текст програми:

|  |
| --- |
| import csv  import os  from sys import argv  student\_list = []  def load\_data(file\_name="lab2.csv"):  global student\_list  try:  with open(file\_name, "r", newline="", encoding="utf-8") as file:  reader = csv.DictReader(file)  student\_list = list(reader)  print(f"Данные загружены из файла {file\_name}.")  except FileNotFoundError:  print(f"Файл {file\_name} не найден. Будет использован пустой справочник.")  def save\_data(file\_name="lab2.csv"):  global student\_list  with open(file\_name, "w", newline='', encoding="utf-8") as file:  fieldnames = ["name", "phone", "email", "address"]  writer = csv.DictWriter(file, fieldnames=fieldnames)  writer.writeheader()  writer.writerows(student\_list)  print(f"Данные сохранены в файл {file\_name}.")  def print\_all\_students():  if not student\_list:  print("Список студентов пуст.")  else:  for student in student\_list:  print(f"Name: {student['name']}, Phone: {student['phone']}, Email: {student['email']}, Address: {student['address']}")  def add\_student():  name = input("Введите имя студента: ")  phone = input("Введите номер телефона: ")  email = input("Введите email: ")  address = input("Введите адрес: ")  new\_student = {"name": name, "phone": phone, "email": email, "address": address}    insert\_position = 0  for student in student\_list:  if name > student["name"]:  insert\_position += 1  else:  break  student\_list.insert(insert\_position, new\_student)  print("Студент добавлен в справочник.")  def delete\_student():  name = input("Введите имя студента для удаления: ")  for student in student\_list:  if student["name"] == name:  student\_list.remove(student)  print(f"Студент {name} удалён из справочника.")  return  print(f"Студент с именем {name} не найден.")  def update\_student():  name = input("Введите имя студента для обновления: ")  for student in student\_list:  if student["name"] == name:  print("Обновите информацию. Нажмите Enter, чтобы оставить поле без изменений.")  new\_phone = input(f"Текущий номер телефона ({student['phone']}): ")  new\_email = input(f"Текущий email ({student['email']}): ")  new\_address = input(f"Текущий адрес ({student['address']}): ")  student["phone"] = new\_phone if new\_phone else student["phone"]  student["email"] = new\_email if new\_email else student["email"]  student["address"] = new\_address if new\_address else student["address"]  print(f"Информация о студенте {name} обновлена.")  return  print(f"Студент с именем {name} не найден.")  def main():  file\_name = "lab2.csv"  if len(argv) > 1:  file\_name = argv[1]      file\_path = os.path.join(os.getcwd(), file\_name)  load\_data(file\_path)  while True:  print("\nМеню:")  print("1. Показать всех студентов")  print("2. Добавить нового студента")  print("3. Удалить студента")  print("4. Обновить данные студента")  print("5. Сохранить и выйти")    choice = input("Выберите действие (1-5): ")  if choice == "1":  print\_all\_students()  elif choice == "2":  add\_student()  elif choice == "3":  delete\_student()  elif choice == "4":  update\_student()  elif choice == "5":  save\_data(file\_path)  print("Выход из программы.")  break  else:  print("Неверный выбор. Попробуйте снова.")  if \_\_name\_\_ == "\_\_main\_\_":  main() |

Посилання на github:

<https://github.com/worhels/lab/tree/main/02>

Знімок екрану з посилання на github:

