Міністерство освіти і науки України
Львівський національний університет імені Івана Франка

Звіт
про виконання лабораторної роботи №2
на тему:

«Обробка файлів CSV та обчислення геометричних характеристик фігур у процедурному стилі»

Виконала:
Студентка групи ФеС-32
Філь Дарина

Львів – 2025

Мета:

Ознайомитися з основами обробки текстових файлів у форматі CSV, навчитися використовувати процедурне програмування для розрахунку площі та периметра геометричних фігур, а також написати юніт-тести для перевірки коректності роботи програми.

Теоретичні відомості:

1. Формат CSV (Comma - Separated Values)

CSV - це текстовий формат, у якому дані зберігаються у вигляді таблиці, де значення розділені комами або іншими роздільниками (наприклад, ; або \t). Кожен рядок файлу представляє один запис, а стовпці відповідають полям цього запису.

2. Обчислення геометричних характеристик

- Коло:

- Площа: S = π * r^2

- Периметр (довжина кола): P = 2 * π * r

- Прямокутник:

- Площа: S = a * b

- Периметр: P = 2 * (a + b)

- Трикутник (рівносторонній):

- Площа: S = (sqrt(3) / 4) * a^2

- Периметр: P = 3 * a

3. Процедурне програмування

Процедурне програмування - це стиль програмування, у якому програма організована у вигляді набору функцій або процедур, які виконують конкретні завдання. Основні принципи:

● Використання функцій для розбиття програми на логічні частини.

● Використання змінних для збереження даних.

● Чітке розмежування вхідних і вихідних параметрів функцій.

У Python немає окремого поняття "процедура", як у деяких інших мовах програмування (наприклад, Pascal). Але можна умовно розрізнити функції та процедури так:

● Функція – це блок коду, який повертає значення за допомогою return.

● Процедура – це блок коду, який не повертає значення (або повертає None).

4. Юніт-тестування

Юніт-тестування - це процес перевірки окремих функцій програми на коректність роботи. У Python для цього зазвичай використовується модуль unittest.

Більше теоретичного матеріалу наведено в підготовці до роботи 1.

Правильне оформлення коду:

1. Дотримуйтесь PEP 8 (Python Style Guide):

- Використовуйте 4 пробіли для відступів.
- Використовуйте зрозумілі назви змінних та функцій (snake_case).
- Розділяйте логічні блоки пустими рядками.

2. Лінтери та форматери:

- Використовуйте flake8, black або pylint для автоматичної перевірки стилю.

Style Guide — це набір правил і рекомендацій щодо написання чистого, зрозумілого та узгодженого коду. У Python основним стилем коду є PEP 8 (Python Enhancement Proposal 8).

Для чого використовується Style Guide?

✅ Покращує читабельність коду – легше розуміти код іншим розробникам (і вам у майбутньому).

✅Забезпечує узгодженість – якщо всі пишуть код однаково, він виглядає організовано.

✅ Допомагає уникнути помилок – правила стилю часто запобігають прихованим проблемам.

✅Сприяє командній роботі – в команді код має виглядати так, ніби його писала одна людина.

3. Обробка помилок:

- Використовуйте try-except для уникнення збоїв під час обчислень.

Код повинен відповідати функціональному підходу, уникаючи глобальних змінних, використовуючи чисті функції та обробляючи дані через композицію функцій.

Чисті функції (pure functions) в Python — це функції, які:

1. Не змінюють зовнішній стан (відсутність побічних ефектів)

- Вони не змінюють глобальні змінні, файли, базу даних тощо.
- Всі обчислення відбуваються всередині функції, а результати повертаються через return.

2. Завжди повертають однаковий результат для однакових вхідних даних

- Якщо викликати функцію з тими ж аргументами кілька разів, результат завжди буде однаковим.

Завдання до роботи:

1. Реалізувати програму на мові python або іншій, використовуючи функціональний підхід, яка:
   - Читає дані з CSV-файлу.
   - Визначає тип фігури та її параметри.
   - Обчислює площу та периметр кожної фігури.
   - Виводить результати у зручному форматі.
2. Написати юніт-тести для перевірки коректності роботи функцій.
3. Переконатися, що всі тести проходять успішно.
4. Продемонструйте роботу лінтера коду, наведіть скріншоти результатів роботи у звіті.

Хід роботи

1. Реалізувати програму на мові python або іншій, використовуючи функціональний підхід, яка:

- Читає дані з CSV-файлу.

```
def parse_csv(file_path):  1 usage
    with open(file_path, newline='') as f:
        reader = csv.reader(f)
        return [row[:1] + list(map(int, row[1:])) for row in reader]
```

-Визначає тип та її параметри

```
    shape_type = shape[0]
    params = shape[1:]

    if shape_type == 'triangle':
        area = get_triangle_area(params[0])
        perim = get_triangle_perimeter(params[0])
    elif shape_type == 'square':
        area = get_rectangle_area(params[0], params[1])
        perim = get_rectangle_perimeter(params[0], params[1])
    elif shape_type == 'circle':
        area = get_circle_area(params[0])
        perim = get_circle_circumference(params[0])
    else:
        print(f'[WARN] Unknown figure: {shape_type}')
        continue
```

-Обчислює площу та периметр кожної фігури

```python
def get_circle_area(radius):  2 usages
    return 3.14 * radius ** 2

def get_circle_circumference(radius):  2 usage
    return 2 * 3.14 * radius

def get_triangle_area(side):  2 usages
    return (sqrt(3) / 4) * side ** 2

def get_triangle_perimeter(side):  2 usages
    return side * 3

def get_rectangle_area(a, b):  2 usages
    return a * b

def get_rectangle_perimeter(a, b):  2 usages
    return 2 * (a + b)
```

-Виводить результати у зручному форматі

```python
print(f'{shape_type.title()} ({", ".join(map(str, params))}): '
      f'Area = {area:.2f}, Perimeter = {perim:.2f}')
```

2. Написати юніт-тести для перевірки коректності роботи функцій.

```python
    def test_area_circle(self):
        self.assertAlmostEqual(get_circle_area(2),  second: 12.56)

    def test_circumference_circle(self):
        self.assertAlmostEqual(get_circle_circumference(2),  second: 12.56)

    def test_area_triangle(self):
        self.assertAlmostEqual(get_triangle_area(4), (sqrt(3) / 4) * 16)

    def test_perimeter_triangle(self):
        self.assertEqual(get_triangle_perimeter(5),  second: 15)

    def test_area_rectangle(self):
        self.assertEqual(get_rectangle_area( a: 6,  b: 3),  second: 18)

    def test_perimeter_rectangle(self):
        self.assertEqual(get_rectangle_perimeter( a: 6,  b: 3),  second: 18)

    def test_csv_parsing(self):
        raw = [['circle', '5'], ['triangle', '3'], ['square', '2', '4']]
        expected = [['circle', 5], ['triangle', 3], ['square', 2, 4]]
        self.assertEqual(parse_csv_data(raw), expected)
```

3. Переконатися, що всі тести проходять успішно.

```
PS E:\СПП\Lab2> pytest main.py
============================================================================= test session starts ==
platform win32 -- Python 3.12.8, pytest-8.3.5, pluggy-1.5.0
rootdir: E:\СПП\Lab2
collected 7 items

main.py .......

============================================================================== 7 passed in 0.03s ===
```

4. Продемонструйте роботу лінтера коду, наведіть скріншоти результатів роботи у звіті.

```
main.py:7:9: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
```

Висновок:

У цій лабораторній роботі я ознайомилася з роботою з текстовими файлами у форматі CSV та закріпила навички процедурного програмування на мові Python. У рамках завдання була реалізована програма, яка зчитує дані з CSV-файлу, визначає тип геометричної фігури та обчислює її площу і периметр. Також було написано юніт-тести для перевірки правильності роботи функцій. Роботу лінтера Pylint було перевірено на прикладі відкриття файлу без параметра encoding, що дозволило побачити відповідне попередження. Усі тести виконано успішно, код працює правильно та відповідає поставленим вимогам.

Додаток

Код програми:

```
import csv
import unittest
from math import sqrt

def parse_csv(file_path):
    with open(file_path, newline='') as f:
        reader = csv.reader(f)
        return [row[:1] + list(map(int, row[1:])) for row in reader]

def get_circle_area(radius):
    return 3.14 * radius ** 2
```

```
def get_circle_circumference(radius):
    return 2 * 3.14 * radius

def get_triangle_area(side):
    return (sqrt(3) / 4) * side ** 2

def get_triangle_perimeter(side):
    return side * 3

def get_rectangle_area(a, b):
    return a * b

def get_rectangle_perimeter(a, b):
    return 2 * (a + b)

def process_figures(data):
    for shape in data:
        shape_type = shape[0]
        params = shape[1:]

        if shape_type == 'triangle':
            area = get_triangle_area(params[0])
            perim = get_triangle_perimeter(params[0])
        elif shape_type == 'square':
            area = get_rectangle_area(params[0], params[1])
            perim = get_rectangle_perimeter(params[0], params[1])
        elif shape_type == 'circle':
            area = get_circle_area(params[0])
            perim = get_circle_circumference(params[0])
        else:
            print(f'[WARN] Unknown figure: {shape_type}')
            continue

        print(f'{shape_type.title()} ({", ".join(map(str, params))}): '
              f'Area = {area:.2f}, Perimeter = {perim:.2f}')

class GeometryTests(unittest.TestCase):

    def test_area_circle(self):
        self.assertAlmostEqual(get_circle_area(2), 12.56)
```

```
    def test_circumference_circle(self):
        self.assertAlmostEqual(get_circle_circumference(2), 12.56)

    def test_area_triangle(self):
        self.assertAlmostEqual(get_triangle_area(4), (sqrt(3) / 4) * 16)

    def test_perimeter_triangle(self):
        self.assertEqual(get_triangle_perimeter(5), 15)

    def test_area_rectangle(self):
        self.assertEqual(get_rectangle_area(6, 3), 18)

    def test_perimeter_rectangle(self):
        self.assertEqual(get_rectangle_perimeter(6, 3), 18)

    def test_csv_parsing(self):
        raw = [['circle', '5'], ['triangle', '3'], ['square', '2', '4']]
        expected = [['circle', 5], ['triangle', 3], ['square', 2, 4]]
        self.assertEqual(parse_csv_data(raw), expected)

def parse_csv_data(fake_csv):
    return [row[:1] + list(map(int, row[1:])) for row in fake_csv]

if __name__ == '__main__':
    dataset = parse_csv('lab2/csv.csv')
    process_figures(dataset)
```

Зміст csv файлу:

```
circle,10
circle,3
circle,7
triangle,5
triangle,8
triangle,4
triangle,6
square,4,6
square,2,5
square,10,10
```