Homework 1: Introduction to Python & Unit Testing (Revised)

1 Getting Started

In this assignment, you will set up a project in DevEdu, create projects in Python, explore some of the interesting properties of Python, and test your code. Read the *entire* assignment in its entirety before starting.

- Complete the following objectives:
  - ☐ Setup a container in DevEdu or run DevEdu's docker image locally
  - ☐ Install `pytest` in a virtual Python environment. The following commands will create the virtual Python environment and install `pytest`:

```
1 python3 -m venv your_custom_env_name_here  --system-site-packages
2 source your_custom_env_name_here/bin/activate
3 python3 -m pip install pytest
```

    Note: You will need to source your environment anytime you start your container *(line 2)*. It would be good to have three different environments for homework1, homework2, and the group project.
  - ☐ Create a new *cs4300* git repository on GitHub with your personal GitHub account.
  - ☐ Clone the new *cs4300* git repository to */home/student/*:
    - Complete the following objectives:
      - ☐ Create SSH keys in the environment and save them to GitHub - Docs.
      - ☐ Clone the repository using *ssh* - Docs.
  - ☐ Inside of the newly cloned directory *(/home/student/cs4300)* create two directories: homework1 and homework2.

Congratulations! You are done with getting started.

2 Homework 1 Tasks

Complete the following tasks and questions to explore your new environment. You will learn the key attributes of the Python programming language, package management using pip package manager, and advanced Python techniques. Additionally, you will use pytest to test your code.

Suggested project layout

```
cs4300/
|-- homework1/
|   |-- src/
|   |   |-- task1.py
|   |   |-- task2.py
|   |   |-- task3.py
|   |   |-- task4.py
|   |   |-- task5.py
|   |   |-- task6.py
|   |   \-- task7.py
|   |-- tests/
|   |   |-- test_task1.py
|   |   |-- test_task2.py
|   |   |-- test_task3.py
|   |   |-- test_task4.py
|   |   |-- test_task5.py
|   |   |-- test_task6.py
|   |   \-- test_task7.py
|   |-- task6_read_me.txt
|   |-- pyproject.toml   # pytest config (optional)
|   \-- README.md        # how to run your code and tests
\-- homework2/
```

Note: All work should be done within the *homework1* directory created previously in the Getting Started section.

2.1 Task 1: Introduction to Python and Testing

Create a new file named `task1.py`. Write a Python script that prints `"Hello, World!"` on the console. Set up a pytest test case that verifies the output of your script. - Docs.

2.2 Task 2: Variables and Data Types

Create a new file named `task2.py` demonstrating the use of various data types, including integers, floating-point numbers, strings, and boolean. Implement a Python using pytest to test case for each data type, ensuring that the script's behavior matches the expected outcomes.

2.3 Task 3: Control Structures

Create a new file named `task3.py`. Create an if statement to check if a given number is positive, negative, or zero. Implement a for loop to print the first 10 prime numbers (you may need to research how to calculate prime numbers). Create a while loop to find the sum of all numbers from 1 to 100. Write pytest test cases to verify the correctness of your code for each control structure.

2.4 Task 4: Functions and Duck Typing

Duck typing is the functionality of a language where "if it looks like a duck and quacks like a duck, you might as well treat it like a duck." This is quite common in interpreted languages. Create a new file named `task4.py` that calculates the final price of a product after applying a given discount percentage inside of a function named *calculate_discount*. The function should accept any numeric type for price and discount. Write pytest test cases to test the calculate_discount function with various types (integers, floats) for price and discount.

2.5 Task 5: Lists and Dictionaries

Create a new file named `task5.py` and inside create a list of your favorite books, including book titles and authors. Use list slicing to print the first three books in the list. Create a dictionary that represents a basic student database, including student names and their corresponding student IDs. Implement pytest test cases to verify the correctness of your code for each data structure.

2.6 Task 6: File Handling and Metaprogramming

Create two new files named `task6.py` and `task6_read_me.txt`. Inside of `task6_read_me.txt` add:

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis vitae feugiat
    tortor, quis tempus lacus. Maecenas sollicitudin rhoncus ultricies.
   Mauris neque metus, blandit sed sagittis aliquam, fringilla eget massa.
   Donec at luctus leo. Curabitur suscipit nulla aliquam sapien maximus,
   sit amet fermentum sem malesuada. Nulla suscipit, felis non consequat
   eleifend, sem quam pharetra turpis, vel efficitur tellus est placerat
   est. Nam metus orci, facilisis et ante sed, ultricies pulvinar lorem.
   Phasellus eu ipsum sit amet ex auctor volutpat. Suspendisse ac turpis et
    felis tristique facilisis vitae in diam. Donec maximus ex in lorem
   auctor vulputate. Nulla finibus sodales ante, convallis gravida metus
   iaculis id.
```

Then write a program inside `task6.py` of that reads `task6_read_me.txt` and counts the number of words in it. Include pytest test cases that verify the word count for each text file.

2.7 Task 7: Package Management in DevEdu

Use pip package manager to add a Python package of your choice to your project (e.g., requests, numpy, matplotlib). Create a new file named `task7.py` and write a Python script that demonstrates how to use the chosen package to perform a specific task or function. Implement pytest test cases to verify the correctness of your code when using the package.

3 Rubric (100 pts)

Area	Points	Notes
Environment & repo structure	10	Uses venv; clear layout; requirements recorded
Task 1	8	Working script + passing test
Task 2	12	Correct data types; parameterized tests
Task 3	18	Correct logic; edge cases tested
Task 4	14	Duck typing + input validation + tests
Task 5	12	Correct list/dict ops + tests
Task 6	14	Robust file I/O; parametrized tests
Task 7	8	Useful package example + tests (mocking if needed)
Code quality	4	Readability, docstrings, names

Grading Criteria: Your assignment will be evaluated based on the correctness of your Python code, the clarity of your comments and explanations, the effectiveness of pytest test cases, and your application of advanced Python techniques such as metaprogramming and duck typing.

4 Submission

1. Please submit your completed project along with explanations, comments, and pytest test cases for each task to Canvas. You can share your assignment by right clicking on the `cs4300` folder and downloading to your local computer.

2. Push all of your code to your github repo. On a successful push you should see on GitHub your `cs4300` repository with `homework1` with all the tasks in it and `homework2` that should be empty.