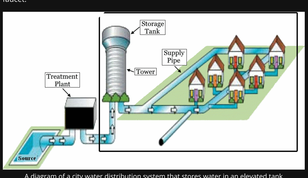
05 Prove Milestone: Testing and Fixing Functions

Purpose

Prove that you can write a Python program and write and run test functions to help you find and fix mistakes in your program.

Problem Statement

Getting clean water to all buildings in a city is a large and expensive task. Many cities will build an elevated water tank, and install a pump that pushes water up to the tank where the water is stored. In the city, when a thirsty person opens a water faucet, water runs from the tank through pipes to the faucet. Earth's gravity pulling on the water in the elevated tank pressurizes the water and causes it to spray from the faucet.



A diagram of a city water distribution system that stores water in an elevated tank

Before a city builds a water distribution system, an engineer must design the system and ensure water will flow to all buildings in the city. An engineer must choose the tower height, pipe type, pipe diameter, and pipe path. Engineers use software to help them make these choices and design a working water distribtuion system.

Assignment

Write a Python program that could help an engineer design a water distribution system. During this prove milestone, you will write three program functions and three test functions as described in the Steps section below.

Helpful Documentation

The preparation content for this lesson explains how to use `pytest`, `assert`, and `approx` to automatically verify that functions are correct. It also contains an example test function and links to additional documentation about `pytest`.

The `pytest.approx` function accepts optional named arguments. One of those named arguments is *abs*. The *abs* named argument causes the `approx` function to compare the actual and expected values up to a specified digit after the decimal point and to ignore the following digits. For example, the following two lines of code cause pytest to compare the actual number returned from `pressure_loss_from_fittings` to -0.306 to only the third digit after the decimal point and to ignore all digits in the actual number after the 6.

```
    assert pressure_loss_from_fittings(1.75, 5) \
        == approx(-0.306, abs=0.001)
```

Notice in the previous example, that the value for *abs* is 0.001, which causes the approx function to ignore all digits after the third digit after the decimal point.

This video about test functions (20 minutes) shows a BYU-Idaho faculty member writing two test functions and using `pytest` to run them.

Help from a Tutor

As a BYU-Idaho campus or online student you can get help from a tutor to complete your CSE 111 assignments. Each tutor is a current BYU-Idaho student employed by BYU-Idaho. Meeting with a tutor is free. It will not cost you any money to meet with a tutor. To get help from a tutor, you simply make an appointment and then meet with the tutor. Campus students meet with tutors in the tutoring center. Online students meet with tutors in Zoom. To make an appointment, follow the instructions in the course tutoring guide.

Steps

Do the following:

1. Using VS Code, create a new file and save it as `water_flow.py`
2. Create another new file, save it as `test_water_flow.py`, and copy and paste the following import statements into the file.

```
from pytest import approx
import pytest
```

3. In your `water_flow.py` program, write a function named `water_column_height` that calculates and returns the height of a column of water from a tower height and a tank wall height. The function must have this header:

```
def water_column_height(tower_height, tank_height):
```

In your function, use the following formula for calculating the water column height.

$$h = t + \frac{3w}{4}$$

where
   - *h* is height of the water column
   - *t* is the height of the tower
   - *w* is the height of the walls of the tank that is on top of the tower

4. In your `test_water_flow.py` file, write a test function named `test_water_column_height`. This test function must call `water_column_height` at least four times to verify that it is working correctly. Use the following numbers in your test function.

Tower Height	Tank Wall Height	Expected Water Column Height
0	0	0
0	10	7.5
25	0	25
48.3	12.8	57.9

5. In your `water_flow.py` program, write a function named `pressure_gain_from_water_height` that calculates and returns the pressure caused by Earth's gravity pulling on the water stored in an elevated tank. The function must have this header:

```
def pressure_gain_from_water_height(height):
```

In your function, use the following formula for calculating pressure caused by Earth's gravity.

$$P = \frac{\rho g h}{1000}$$

where
   - *P* is the pressure in kilopascals
   - ρ is the density of water (998.2 kilogram / meter^3)
   - *g* is the acceleration from Earths gravity (9.80665 meter / second^2)
   - *h* is the height of the water column in meters

6. In your `test_water_flow.py` file, write a test function named `test_pressure_gain_from_water_height`. This test function must call `pressure_gain_from_water_height` at least three times to verify that it is working correctly. Use the following numbers in your test function.

Height	Expected Pressure	approx Absolute Tolerance
0	0	0.001
30.2	295.628	0.001
50	489.450	0.001

7. In your `water_flow.py` program, write a function named `pressure_loss_from_pipe` that calculates and returns the water pressure lost because of the friction between the water and the walls of a pipe that it flows through. The function must have this header:

```
def pressure_loss_from_pipe(pipe_diameter,
        pipe_length, friction_factor, fluid_velocity):
```

In your function, use the following formula for calculating pressure loss from friction within a pipe.

$$P = \frac{-f L \rho v^2}{2000\, d}$$

where
   - *P* is the lost pressure in kilopascals
   - *f* is the pipe's friction factor
   - *L* is the length of the pipe in meters
   - ρ is the density of water (998.2 kilogram / meter^3)
   - *v* is the velocity of the water flowing through the pipe in meters / second
   - *d* is the diameter of the pipe in meters

8. In your `test_water_flow.py` file, write a test function named `test_pressure_loss_from_pipe`. This test function must call `pressure_loss_from_pipe` at least seven times to verify that it is working correctly. Use the following numbers in your test function.

Pipe Diameter	Pipe Length	Friction Factor	Fluid Velocity	Expected Pressure Loss	approx Absolute Tolerance
0.048692	0	0.018	1.75	0	0.001
0.048692	200	0	1.75	0	0.001
0.048692	200	0.018	0	0	0.001
0.048692	200	0.018	1.75	-113.008	0.001
0.048692	200	0.018	1.65	-100.462	0.001
0.28687	1000	0.013	1.65	-61.576	0.001
0.28687	1800.75	0.013	1.65	-110.884	0.001

9. Copy and paste the following code at the bottom of your `test_water_flow.py` file.

```
# Call the main function that is part of pytest so that the
# computer will execute the test functions in this file.
pytest.main(["-v", "--tb=line", "-rN", __file__])
```

Testing Procedure

Verify that your functions work correctly by following each step in this procedure:

1. Run your `test_water_flow.py` file and ensure that all three of the test functions pass. If any of the test functions don't pass, there is a mistake in either the program function that you wrote or the test function that you wrote. Read the output from `pytest`, fix the mistake, and run the `test_water_flow.py` file again until all the test functions pass.

```
> python test_water_flow.py
====================== test session starts ======================
platform win32 -- Python 3.11.1, pytest-7.2.1, pluggy-1.0.0 --
rootdir: C:\Users\cse111\lesson05
collected 3 items

test_water_flow.py::test_water_column_height PASSED
test_water_flow.py::test_pressure_gain_from_water_height PASSED
test_water_flow.py::test_pressure_loss_from_pipe PASSED

======================= 3 passed in 0.07s =======================
```

Ponder

During this assignment, you wrote three program functions named `water_column_height`, `pressure_gain_from_water_height`, and `pressure_loss_from_pipe`. Also, in a separate file, you wrote three test functions. Each of the test functions called one of the program functions multiple times and automatically verified that the value returned from the program function was correct. If you worked as a software developer on a large project with four other software developers, how would test functions help you and the other developers ensure that your program worked correctly?

Submission

On or before the due date, return to I-Learn and report your progress on this milestone.

Last modified 2023-03-24