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.

Problem Statement

The Turing test, named after Alan Turing, is a test of a computer's ability to make conversation that is indistinguishable from human conversation. A computer that could pass the Turing test would need to understand sentences typed by a human and respond with sentences that make sense.

In English and many other languages, grammatical quantity (also known as grammatical number) is an attribute of nouns, pronouns, adjectives, and verbs that expresses count distinctions, such as "one", "two", "some", or "many". The grammatical quantity of the words in a sentence must match. In English, there are two categories of grammatical quantity: single and plural. For example, here are three English sentences that contain single nouns, pronouns, adjectives, and verbs:

> The boy laughs.
> One dog eats.
> She drinks water.

Here are three English sentences that contain plural nouns, pronouns, adjectives, and verbs:

> Two birds fly.
> Some animals eat grass.
> Many cars are red.

Grammatical tense is an attribute of verbs that expresses when an action happened. Many languages include past, present, and future tenses. For example, here are three English sentences, the first with past tense, the second with present tense, and the third with future tense:

> The cat walked.
> The cat walks.
> The cat will walk.

Assignment

Write a Python program that generates simple English sentences. During this lesson, you will write and test functions that generate sentences with three parts:

1. a determiner (sometimes known as an article)
2. a noun
3. a verb

For example:

> A cat laughed.
> One man eats.
> The woman will think.
> Some girls thought.
> Many dogs run.
> Many men will write.

During the next lesson, you will write and test functions that generate English sentences with the three parts and a prepositional phrase, such as these sentences:

> One girl talked for the car.
> A bird drinks off one child.
> The child will run on the car.
> Some dogs drank above many rabbits.
> Some children laugh at many dogs.
> Some rabbits will talk about some cats.

For this milestone assignment, your program must include at least these four functions:

- `main`
- `get_determiner`
- `get_noun`
- `get_verb`

You may add other functions if you want. The functions `get_determiner`, `get_noun`, and `get_verb`, must randomly choose a word from a list of words and return the randomly chosen word.

Helpful Documentation

- In CSE 110, you studied Python lists. You should recall that we create a Python list with square brackets and commas like this list of strings:

```
# Create a list of strings and assign
# the list to a variable named words.
words = ["boy", "girl", "cat", "dog", "bird", "house"]
```

- The standard Python `random` module includes a function named `choice` that will randomly choose one element from a list and return that element. The choice function is easy to call like this:

```
import random

# Create a list of strings and assign
# the list to a variable named words.
words = ["boy", "girl", "cat", "dog", "bird", "house"]

# Call the random.choice function which will choose
# one string from the words list. Store the chosen
# string in a variable named word.
word = random.choice(words)
```

- The Python str.capitalize method will capitalize the first letter in a word. The capitalize method is easy to call like this:

```
# This could be any word from any source.
word = "horse"

# Call the capitalize method which will
# capitalize the first letter of the word.
cap_word = word.capitalize()

# Print the capitalized word.
print(cap_word)
```

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

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

Steps

Do the following:

1. Using VS Code, create a new file, import the `random` module at the top of the file, and save the file as `sentences.py`

2. Copy and paste the following `get_determiner` function into your program.

```
def get_determiner(quantity):
    """Return a randomly chosen determiner. A determiner is
    a word like "the", "a", "one", "two", "some", "many".
    If quantity == 1, this function will return
    either "the" or "one". Otherwise this function will
    return either "some" or "many".

    Parameter
        quantity: an integer.
            If quantity == 1, this function will return
            a determiner for a single noun. Otherwise this
            function will return a determiner for a plural noun.
    Return: a randomly chosen determiner.
    """
    if quantity == 1:
        words = ["a", "one", "the"]
    else:
        words = ["two", "some", "many", "the"]

    # Randomly choose and return a determiner.
    word = random.choice(words)
    return word
```

3. Use the `get_determiner` function as an example to help you write the `get_noun` function. The `get_noun` function must have the following header and fulfill the requirements of the following documentation string.

```
def get_noun(quantity):
    """Return a randomly chosen noun.
    If quantity == 1, this function will
    return one of these ten single nouns:
        "bird", "boy", "car", "cat", "child",
        "dog", "girl", "man", "rabbit", "woman"
    Otherwise, this function will return one of these
    ten plural nouns:
        "birds", "boys", "cars", "cats", "children",
        "dogs", "girls", "men", "rabbits", "women"

    Parameter
        quantity: an integer that determines
            if the returned noun is single or plural.
    Return: a randomly chosen noun.
    """
```

4. Use the `get_determiner` function as an example to help you write the `get_verb` function. The `get_verb` function must have the following header and fulfill the requirements of the following documentation string.

```
def get_verb(quantity, tense):
    """Return a randomly chosen verb. If tense is "past", this
    function will return one of these ten verbs:
        "drank", "ate", "grew", "laughed", "thought",
        "ran", "slept", "talked", "walked", "wrote"
    If tense is "present" and quantity is 1, this
    function will return one of these ten verbs:
        "drinks", "eats", "grows", "laughs", "thinks",
        "runs", "sleeps", "talks", "walks", "writes"
    If tense is "present" and quantity is NOT 1,
    this function will return one of these ten verbs:
        "drink", "eat", "grow", "laugh", "think",
        "run", "sleep", "talk", "walk", "write"
    If tense is "future", this function will return one of
    these ten verbs:
        "will drink", "will eat", "will grow", "will laugh",
        "will think", "will run", "will sleep", "will talk",
        "will walk", "will write"

    Parameters
        quantity: an integer that determines if the
            returned verb is single or plural.
        tense: a string that determines the verb conjugation,
            either "past", "present" or "future".
    Return: a randomly chosen verb.
    """
```

5. Write the `main` function and any other functions that you think are necessary for your program to generate and print six sentences with these characteristics:

	Quantity	Verb Tense
a.	single	past
b.	single	present
c.	single	future
d.	plural	past
e.	plural	present
f.	plural	future

6. In a new file named test_sentences.py, write three test functions named `test_get_determiner`, `test_get_noun`, and `test_get_verb`. Each of these three test functions must test one of the program functions: `get_determiner`, `get_noun`, and `get_verb`. The following example contains code for the `test_get_determiner` function. Copy and paste the code into your `test_sentences.py` file and use it as an example to help you write `test_get_noun` and `test_get_verb`.

```
from sentences import get_determiner, get_noun, get_verb
import pytest


def test_get_determiner():
    # 1. Test the single determiners.

    single_determiners = ["a", "one", "the"]

    # This loop will repeat the statements inside it 4 times.
    # If a loop's counting variable is not used inside the
    # body of the loop, many programmers will use underscore
    # (_) as the variable name for the counting variable.
    for _ in range(4):
        word = get_determiner(1)

        # Verify that the word returned from get_determiner is
        # one of the words in the single_determiners list.
        assert word in single_determiners

    # 2. Test the plural determiners.

    plural_determiners = ["two", "some", "many", "the"]

    # This loop will repeat the statements inside it 4 times.
    for _ in range(4):
        word = get_determiner(2)
        # Verify that the word returned from get_determiner
        # is one of the words in the plural_determiners list.
        assert word in plural_determiners
```

Testing Procedure

Verify that your test program works correctly by following each step in this procedure:

1. Run your `test_sentences.py` program and verify that all three of the test functions pass. If one or more of the tests don't pass, find and fix the mistakes in your program functions or test functions until the tests pass as shown in this output:

```
> python test_sentences.py
====================== test session starts =======================
platform win32--Python 3.8.6, pytest-6.1.2, py-1.9.0, pluggy-0.13.
rootdir: C:\Users\cse111\lesson05
collected 3 items

test_sentences.py::test_get_determiner PASSED               [ 33%]
test_sentences.py::test_get_noun PASSED                     [ 66%]
test_sentences.py::test_get_verb PASSED                     [100%]

======================= 3 passed in 0.10s ========================
```

2. Run your `sentences.py` program and ensure that your program outputs six sentences with the following characteristics:

	Noun Quantity	Verb Tense
a.	single	past
b.	plural	past
c.	single	present
d.	plural	present
e.	single	future
f.	plural	future

Your program's output should be similar to the sample run output shown here. However, because your program will randomly choose the determiners, nouns, and verbs, your program will generate different sentences than the ones shown here.

```
> python sentences.py
The cat laughed.
Some girls thought.
One man eats.
Many dogs run.
The woman will think.
Many men will write.
```

Ponder

During this assignment, you wrote four program functions named `main`, `get_determiner`, `get_noun`, and `get_verb`. Also, in a separate file, you wrote three test functions named `test_get_determiner`, `test_get_noun`, and `test_get_verb`. Each of the test functions called one of the program functions 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 five other software developers, how would test functions help you and the other developers write better code?

Submission

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