06 Prove Assignment: Troubleshooting 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, a preposition is a word used to express spatial or temporal relations, such as "in", "over", and "before". A prepositional phrase is group of words that begins with a preposition and includes a noun. For example:

> above the water
> in the kitchen
> after the meeting

Assignment

Write the second half of the Python program that you began in the previous lesson's prove milestone, a program that generates simple English sentences. During this lesson, you will write and test functions that generate sentences with four parts:

1. a determiner
2. a noun
3. a verb
4. a prepositional phrase

For example:

> 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.

After this lesson, your program must include at least these six functions:

- `main`
- `get_determiner`
- `get_noun`
- `get_verb`
- `get_preposition`
- `get_prepositional_phrase`

You may add other functions if you want. The `get_preposition` function must randomly choose a preposition from a list and return the randomly chosen preposition. The `get_prepositional_phrase` function must make a prepositional phrase by calling the `get_preposition`, `get_determiner`, and `get_noun` functions.

Helpful Documentation

The prepare content for this lesson explains how to troubleshoot functions and entire programs that are not working correctly.

Steps

Do the following:

1. Use the `get_determiner` function from the previous lesson as an example to help you write the `get_preposition` function. The `get_preposition` function must have the following header and fulfill the requirements of the following documentation string.

```
def get_preposition():
    """Return a randomly chosen preposition
    from this list of prepositions:
        "about", "above", "across", "after", "along",
        "around", "at", "before", "behind", "below",
        "beyond", "by", "despite", "except", "for",
        "from", "in", "into", "near", "of",
        "off", "on", "onto", "out", "over",
        "past", "to", "under", "with", "without"

    Return: a randomly chosen preposition.
    """
```

2. Write the `get_prepositional_phrase` function to have the following header and fulfill the requirements of the following documentation string.

```
def get_prepositional_phrase(quantity):
    """Build and return a prepositional phrase composed of three
    words: a preposition, a determiner, and a noun by calling the
    get_preposition, get_determiner, and get_noun functions.

    Parameter
        quantity: an integer that determines if the determiner
            and noun in the prepositional phrase returned from
            this function are singular or pluaral.
    Return: a prepositional phrase.
    """
```

3. Add code to the `main` function and write any other functions that you think are necessary for your program to generate and print six sentences, each with a determiner, a noun, a verb, and a prepositional phrase. The six sentences must have the following characteristics:

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

4. In the `test_sentences.py` file write two functions named `test_get_preposition` and `test_get_prepositional_phrase` that test the `get_preposition` and `get_prepositional_phrase` functions.

   Perhaps you are wondering what code you should write in the `test_get_prepositional_phrase` function. To answer that question, ask yourself, "What do we know about the

`get_prepositional_phrase` function?" From its description, we know the `get_prepositional_phrase` function returns a phrase made of three words: a preposition, a determiner, and a noun. So you could write code in the `test_get_prepositional_phrase` function that calls the `get_prepositional_phrase` function and then asserts that the string returned from `get_prepositional_phrase` contains three words separated by spaces. In addition, you could write code that calls the Python string `split` method to split the returned phrase into its three words and checks each of the three words.

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 five 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\lesson06
collected 5 items

test_sentences.py::test_get_determiner PASSED               [ 20%]
test_sentences.py::test_get_noun PASSED                     [ 40%]
test_sentences.py::test_get_verb PASSED                     [ 60%]
test_sentences.py::test_get_preposition PASSED              [ 80%]
test_sentences.py::test_get_prepositional_phrase PASSED     [100%]

====================== 5 passed in 0.10s ======================
```

2. Run your sentences.py program and ensure that your program's output is similar to the sample run output shown here. Because your program will randomly choose the determiners, nouns, verbs, and prepositions, your program will generate different sentences than the ones shown here.

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

Exceeding the Requirements

If you wish to exceed the requirements of this assignment, here are a few suggestions for additional features that you could add to your program. If you wish, you can add different features to your program. However, you don't have to add any additional features to your program because exceeding the requirements of this assignment is optional.

- Within your `main` function add one or more calls to `get_prepositional_phrase` so that each sentence includes two prepositional phrases like this:

> One girl across one cat talked for the car.
> A bird near the rabbit drinks off one child.
> The child under the cat will run on the car.
> Some dogs without a cat drank above many rabbits.
> Some children from a bird laugh at many dogs.
> Some rabbits behind one man will talk about some cats.

- Write a function named `get_adjective` and call it in your `main` function to add an adjective to the sentences produced by your program. Does it make sense to call `get_adjective` in your `get_prepositional_phrase` function?

- Write a function named `get_adverb` and call it in your `main` function to add an adverb to the sentences produced by your program.

Ponder

How hard would it be to modify your program to pass the Turing test?

Submission

To submit your program, return to I-Learn and do these two things:

1. Upload your `sentences.py` and `test_sentences.py` files for feedback.
2. Add a submission comment that specifies the grading category that best describes your program along with a one or two sentence justification for your choice. The grading criteria are:
    - 1. Some attempt made
    - 2. Developing but significantly deficient
    - 3. Slightly deficient
    - 4. Meets requirements
    - 5. Exceeds requirements