W02 Project: Sentences

Purpose

Prove that you can write functions with parameters and call those functions multiple times with arguments.

Problem Statement

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 a Python program that generates simple English sentences that you began in the prove milestone. As part of the previous lesson’s prove milestone, you wrote a program that generates English sentences with three parts: a determiner, a noun, and a verb. During this prove assignment, you will add functions so that your program generates 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.

To complete this prove assignment, your program must include at least these seven functions:

1. **`main`**
2. **`make_sentence`**
3. **`get_determiner`**
4. **`get_noun`**
5. **`get_verb`**
6. **`get_preposition`**
7. **`get_prepositional_phrase`**

You may add other functions if you find them helpful. 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 preparation content for week 01 explains how to call functions.
- The preparation 1 content for this lesson explains how to write functions.
- The preparation content 2 for this lesson explains variable scope and good function design.

Steps

Do the following:

1. Use the **`get_determiner`** function from the previous lesson's prove milestone 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.

```python
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 should
            be single or pluaral.
    Return: a prepositional phrase.
    """
```

3. Add code to the **`make_sentence`** 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:

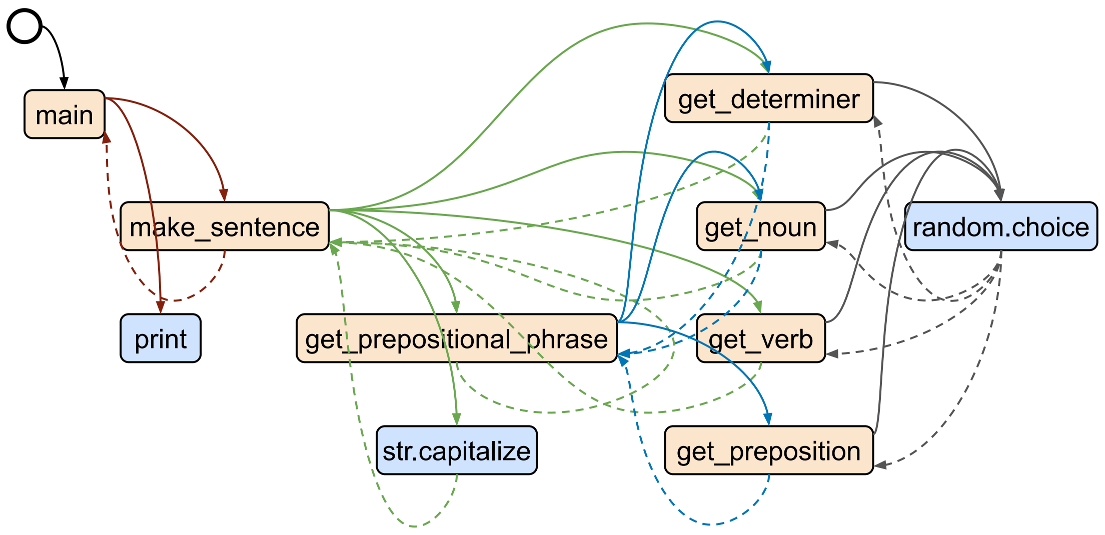
	Quantity	Verb Tense
a.	single	past
b.	single	present
c.	single	future
d.	plural	past
e.	plural	present
f.	plural	future

Call Graph

The following call graph shows the user-defined functions and function calls and returns as you should write them in your **`sentences.py`** program. From this call graph we see the following function calls:

1. The computer starts executing the **`sentences.py`** program by calling the **`main`** function.

2. While executing the **main** function, the computer calls the **make_sentence** function.
3. While executing the **make_sentence** function, the computer calls the **get_determiner**, **get_noun**, **get_verb**, and **get_prepositional_phrase** functions.
4. While executing the **get_prepositional_phrase** function, the computer calls the **get_preposition**, **get_determiner**, and **get_noun** functions.
5. While executing each of the **get_determiner**, **get_noun**, **get_verb**, and **get_preposition** functions, the computer calls the **random.choice** function.
6. Then, the computer executes the **str.capitalize** method.
7. Finally, the computer executes the **print** function.



The call graph for a program that builds and prints sentences. Notice in the call graph that the **get_prepositional_phrase** function calls the **get_preposition**, **get_determiner**, and **get_noun** functions.

Testing Procedure

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

1. Run your **sentences.py** program and ensure that your program's output is similar to the sample run output shown here. Because your program randomly chooses 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 your program fulfills the requirements for this assignment as described in the previous prove milestone and the Assignment section above, your program will earn 93% of the possible points. In order to earn the remaining 7% of points, you will need to add one or more features to your program so that it exceeds the requirements. Here are a few suggestions for additional features that you could add to your program if you wish.

- Within your **`make_sentence`** function add another call 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 **`make_sentence`** 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 **`make_sentence`** function to add an adverb to the sentences produced by your program.

Ponder

What changes would you have to make to your program so that it could produce sentences that fit the following form?

> {Determiner} {adjective} {noun} {prepositional_phrase} {adverb} {verb}
> {determiner} {adjective} {noun} {prepositional_phrase}.

Such as these sentences:

> The red birds in the air quickly ate some fast fish in the water.
> The busy bird with a car sweetly drank many smart rabbits by one boy.
> One dinky boy near the dog calmly grew some tall children across a cat.

Submission

To submit your program, return to Canvas and do these two things:

1. Upload your **`sentences.py`** file 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

NOTE: If an error prevents your program from running to completion the grader will score your assignment with a 0 and request that you fix and resubmit your program. In other words, rather than submitting a program that doesn't work, it's best to ask for help to understand how to fix the problem before submitting your program.

Useful Links:

- Return to: Week Overview | Course Home