W02 Project Milestone: Sentences

Purpose

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

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, and verbs:

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

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

> Two birds fly.
> Some animals eat.
> Many cars drive.

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 named **`sentences.py`** that generates simple English sentences. During this prove milestone, you will write 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.

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

1. **`main`**
2. **`make_sentence`**
3. **`get_determiner`**
4. **`get_noun`**
5. **`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. All the functions that you must write for this milestone assignment are described in the Steps section below.

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 preparation content for the previous lesson explains how to call functions.
- The preparation content for this lesson explains how to write functions.
- The standard Python **random** module includes a **function named choice** that randomly chooses one element from a list and returns 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()
```

- In Python, it is easy to use an f-string to combine many strings into one large string like this:

```
given = "Michelle"
middle = "Aya"
surname = "Takechi"
full_name = f"{given} {middle} {surname}"
```

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", "some", "many".
    If quantity is 1, this function will return either "a",
    "one", or "the". Otherwise this function will return
    either "some", "many", or "the".
    Parameter
        quantity: an integer.
            If quantity is 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 = ["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 is 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 a function named **`make_sentence`** with the following header and documentation string. Your **`make_sentence`** function must call your **`get_determiner`**, **`get_noun`**, and **`get_verb`** function once each and build and return a sentence. Your **`make_sentence`** function must capitalize the first letter of the sentence and end it with a period (.).

```
def make_sentence(quantity, tense):
  """Build and return a sentence with three words:
  a determiner, a noun, and a verb. The grammatical
  quantity of the determiner and noun will match the
  number in the quantity parameter. The grammatical
  quantity and tense of the verb will match the number
  and tense in the quantity and tense parameters.
  """
```

6. Write the **`main`** function to call your **`make_sentence`** function six times 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

7. At the bottom of your **`sentences.py`** file, write a call to your **`main`** function as explained in this lesson's preparation content in the section titled The **`main`** User-Defined Function.

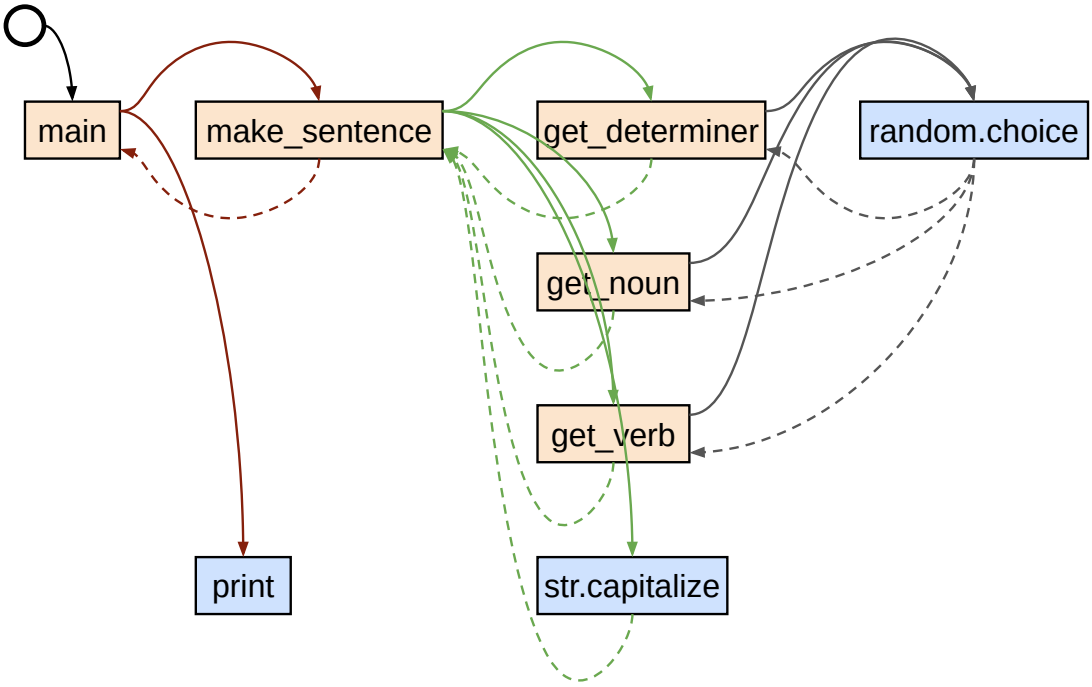
Call Graph

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`**, and **`get_verb`** functions.

4. While executing each of the **`get_determiner`**, **`get_noun`**, and **`get_verb`** functions, the computer calls the **`random.choice`** function.
5. Then, the computer executes the **`str.capitalize`** method.
6. Finally, the computer executes the **`print`** function.



The call graph for a program that builds and prints sentences.

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 outputs six sentences with the following characteristics:

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

	Quantity	Verb Tense
e.	plural	present
f.	plural	future

Your program's output should be similar to the sample run output shown here. However, because your program randomly chooses the determiners, nouns, and verbs, your program will generate different sentences than the six 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 five functions named **`main`**, **`make_sentence`**, **`get_determiner`**, **`get_noun`**, and **`get_verb`**. The **`main`** function is not easily reusable in another program because it prints to the terminal window. However, the **`make_sentence`**, **`get_determiner`**, **`get_noun`**, and **`get_verb`** functions are easily reusable in another program because each one gets input from its parameters and returns a value and does not get input from a user and does not print anything.

Submission

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

Useful Links:

- Return to: Week Overview | Course Home