Reto de lectura 3

Durante el reto, puedes recurrir a los diccionarios y glosarios igual como a tu material de clase.

Text Title: _____

Section 1

The usual syntax for defining a Python function is as follows:

```
def <function_name>([<parameters>]):
    <statement(s)>
```

Ilustration 1

The components of the definition are explained in the table below:

Component	Meaning
def	The keyword that informs Python that a function is being defined
<function_name>	A valid Python identifier that names the function
<parameters>	An optional, comma-separated list of parameters that may be passed to the function
:	Punctuation that denotes the end of the Python function header (the name and parameter list)
<statement(s)>	A block of valid Python statements

Table 1

The final item, <statement(s)>, is called the **body** of the function. The body is a block of statements that will be executed when the function is called. The body of a Python function is defined by indentation in accordance with the off-side rule. This is the same as code blocks associated with a control structure, like an if or while statement.

Section 2

The syntax for calling a Python function is:

```
<function_name>([<arguments>])
```

Illustration 2

<arguments> are the values passed into the function. They correspond to the **<parameters>** in the Python function definition. You can define a function that doesn't take any arguments, but the parentheses are still required. Both a function definition and a function call must always include parentheses, even if they're empty.

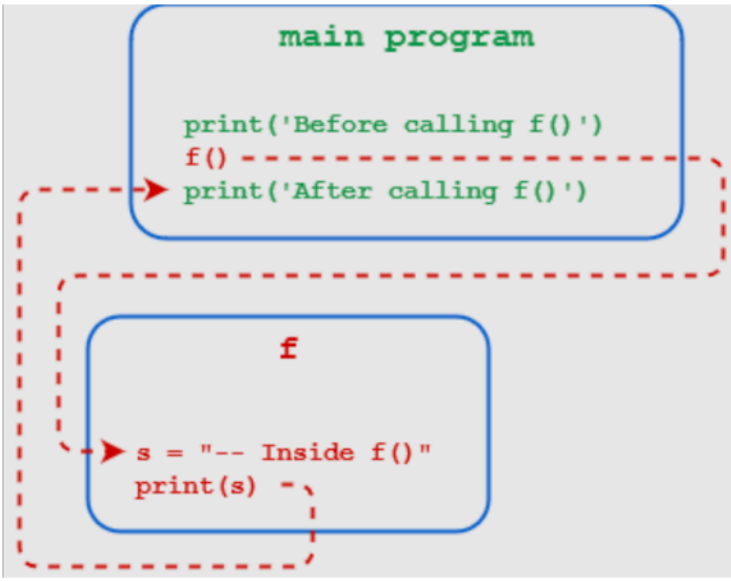
Section 3	As usual, you'll start with a small example and add complexity from there. Keep the time-honored mathematical tradition in mind. Call your first Python function ***f().*** Here's a script file, foo.py, that defines and calls ***f():***

```
1def f():
2    s = '-- Inside f()'
3    print(s)
4
5print('Before calling f()')
6f()
7print('After calling f()')
```

Illustration 3

Section 4	**Here's how this code works:**

1. **Line 1** uses the def keyword to indicate that a function is defined. Execution of the def statement only creates the definition of f(). All the following lines that are indented (lines 2 to 3) become part of the body of f(). They are stored as its definition, but they aren't executed yet.
2. **Line 4** is a bit of whitespace between the function definition and the first line of the main program. While it isn't syntactically necessary, it is nice to have. To learn more about whitespace around top-level Python function definitions, check out Writing Beautiful Pythonic Code With PEP 8.
3. **Line 5** is the first statement that isn't indented because it isn't a part of the definition of f(). It's the start of the main program. When the main program executes, this statement is executed first.
4. **Line 6** is a call to f(). Note that empty parentheses are always required in both a function definition and a function call, even when there are no parameters or arguments. Execution proceeds to f() and the statements in the body of f() are executed.
5. **Line 7** is the next line to execute once the body of f() has finished. Execution returns to this print() statement.

Section 5	The sequence of execution (or **control flow**) for foo.py is shown in the following diagram:  **Illustration 4**

Section 6	When you run **foo.py** from a Windows with command prompt a the result is as follows: C:\Users\john\Documents\Python\doc>python foo.py Before calling f() -- Inside f() After calling f() **Ilustration 5**