Programming Workshop: GitHub, Python Practice, and Classes vs Functions

Intermediate Workshop #3
26 September 2016

Outline

1. Speed Exercises
2. “Homework” - Completion
3. GitHub - Fork Our Repository
4. Python:
    a. Functions
    b. Classes
    c. When to Use Either
    d. Practice

Speed Exercises

1:

Make a function that counts to 15 and prints each odd number.

60 seconds

Speed Exercises

2:

Make a function that prints all prime numbers between 1 and 100.

180 seconds

Speed Exercises

3:

Make a script that takes a string from the user and prints how many characters it contains.

120 seconds

Speed Exercises

4:

Make a function that accepts a number and prints that many numbers in the fibonacci sequence. Limit the input to >=200.

300 seconds

“Homework”

1. **Get Linux and Python running!**
2. Make a GitHub account
3. Go through the Codecademy Python lesson

Forking our GitHub Repository

What is a clone?

- **Copy** of repository stored locally

What is a fork?

- **Bifurcation** of repository
- Can be cloned to local machine

Forking our GitHub Repository

Forking our repository:

1. Open our repo in a web browser
2. In the upper right side of the interface, click “fork this repository”
3. In the list of your repositories, select the fork of our repository
4. Clone your fork of our repository to your computer

Python - Classes

- Meant to allow for reusable, abstractable, extendable code
  - It’s the core principle behind OO paradigm
- More complex
  - Less easy to read
  - Less easy to debug
  - Absolutely must have thorough documentation, thorough application testing, and thorough unit testing
  - Harder to make meanings or operations clear

Python - Class Example

Let’s make a kiln and pottery system.

Python - Class Example (cont'd)

```
class Kiln():
    def __init__(self, input_temperature=78):
        self._contents = []
        self._temperature = input_temperature

    def add_pottery(self, added_pottery):
        for pottery in added_pottery:
            self._contents.append(pottery)
            if pottery.get_bake_temperature() >= self._temperature:
                pottery.mark_as_baked()

    def change_temperature(self, new_temperature):
        self._temperature = new_temperature
        for pottery in self._contents:
            if pottery.get_bake_temperature() >= self._temperature:
                pottery.mark_as_baked()

    def get_temperature(self):
        return self._temperature

```

Python - Class Example (cont'd)

```
21      def get_contents(self, state=None):
22          if state:
23              pottery_of_specified_state = [x for x in self._contents if x.get_bake_status() == state]
24              return pottery_of_specified_state
25          return self._contents
26
27      def list_contents(self, state=None):
28          if state:
29              pottery_of_specified_state = [x for x in self._contents if x.get_bake_status() == state]
30              for pottery in pottery_of_specified_state:
31                  if pottery.get_description():
32                      print("%s: bake temperature=%i, description=%s" %(pottery.get_name(),
                            pottery.get_bake_temperature(), pottery.get_description()))
33                  else:
34                      print("%s: bake temperature=%i" %(pottery.get_name(), pottery.get_bake_temperature()))
35          else:
36              for pottery in self._contents:
37                  if pottery.get_description():
38                      print("%s: bake temperature=%i, description=%s" %(pottery.get_name(),
                            pottery.get_bake_temperature(), pottery.get_description()))
39                  else:
40                      print("%s: bake temperature=%i" %(pottery.get_name(), pottery.get_bake_temperature()))
```

Python - Class Example (cont'd)

```
class Pottery():
    def __init__(self, input_name, input_bake_temperature, input_description=None):
        self._name = input_name
        self._bake_temperature = input_bake_temperature
        self._description = input_description
        self._bake_status = "unbaked"

    def mark_as_baked(self):
        self._bake_status = "baked"

    def get_name(self):
        return self._name

    def get_bake_temperature(self):
        return self._bake_temperature

    def get_description(self):
        return self._description

    def get_bake_status(self):
        return self._bake_status

```

Python - Class Example (cont'd)

(Example of Use)

Python - Functions

- Meant to perform a generic operation on some specific input format
  - Data processing? Use a function.
  - Noticing a repeated operation? Use a function.
- Simple
  - Easy to read
  - Easy to debug
  - Should be documented, but not absolutely required

Functions vs Classes

- Functions can be used to solve any issue. Classes have a specific purpose.
- Functions can call functions - but if a function has a subfunction, use a class.
- If a class has only one method, consider using a function.
- **If you do not have time to thoroughly build documentation, application tests, and unit tests for a class, *do not use classes.***

Python - Function Examples

```
def sort_items_by_color(input_list, color_order = [“red”, “orange”, “yellow”, “green”, “blue”, “violet”]):
	return_list = []
	for color in color_order:
		for item in input_list:
			if item.get_color() == color:
				return_list.append(item)
	return return_list
```

Python - Function Examples

```
def sort_items_by_color(input_list, color_order = [“red”, “orange”, “yellow”, “green”, “blue”, “violet”]):
	“””
	Function to sort items by color
	Takes:
		List of items
		[Optional] Ordered list of color terms to sort by
	Returns:
		List of items sorted by colors listed in ordered list
	“””

	return_list = []
	for color in color_order:
		for item in input_list:
			if item.get_color() == color:
				return_list.append(item)
	return return_list
```

Python - Function Examples

```
def sort_items_by_color(input_list, color_order = [“red”, “orange”, “yellow”, “green”, “blue”, “violet”]):
      “””
      Function to sort items by color
      Takes:
            List of items
            [Optional] Ordered list of color terms to sort by
      Returns:
            List of items sorted by colors listed in ordered list
      “””

      return [item for color in color_order for item in input_list if item.get_color == color]
```

Python - Function Examples

```
def sort_items_by_color(input_list, color_order = [“red”, “orange”, “yellow”, “green”, “blue”, “violet”]):
	“””
	Function to sort items by color
	Takes:
		List of items
		[Optional] Ordered list of color terms to sort by
	Returns:
		List of items sorted by colors listed in ordered list
	“””

	return_list = []
	items_by_color = {}
	for color in color_order:
		items_by_color[color] = []

	for item in input_list:
		items_by_color[item.get_color()].append(item)

	for color in color_order:
		return_list.extend(items_by_color[color])

	return return_list
```