

Name: _____ Partner: _____

Python Activity 24: Classes - Attributes

Learning Objectives

Students will be able to:

Content:

- Define **objects**, **attributes**, and **classes** in python
- Identify differences between attributes/variables
- Explain why classes are useful

Process:

- Write code that creates a new user-defined class with attributes
- Write code that uses user-defined types as function arguments, embedded objects, return values

Prior Knowledge

- Python concepts from Activities 1-23.

Folks, this is a brand new activity. If you encounter any issues/typos, please let Iris know!

Critical Thinking Questions:

1. Examine the following code from interactive python below using a Flower data structure.

Interactive Python	
<pre>3 >>> iris = Flower() 4 >>> iris.petal = 3 5 >>> iris.petal 6 3 7 >>> iris.color = 'purple' 8 >>> iris.color</pre>	<pre>10 flwrList = list() 11 flwrList = [iris]</pre>

- a. What type of object is `flwrList`? How do you know?

- b. What type of object is `iris`? How do you know?

- c. On which line do we place `flwrList` on the lefthand side of an assignment operator?
What value is assigned? _____
- d. On which line is `iris.petal` on the lefthand side of an assignment operator?
What value is assigned? _____
- e. What is displayed when we call `iris.petal`? _____
- f. What will be displayed when we call `iris.color`? _____

FYI: Creating a new object, such as `iris` or `flwrList`, is called **instantiation** and `flwrList` is an **instance** of a List class object.

2. Examine the following code below, that creates a new class in interactive python:

```
0 >>> class Flower:
1 ...     """ A new class representing flowers """

2 >>> iris = Flower()
3 >>> iris.petal = 3
4 >>> iris.sepal = 3
5 >>> print(iris.petal + iris.sepal)
```

a. What additional attribute are we giving to `iris` in this example?

b. What is likely to be the output after line 5? _____

FYI: We can assign values to named elements of objects. These named elements are called **attributes**.

c. What attributes does `iris` have in this example? _____

e. If we add `print(iris.bloomTime)` as our 7th line above, this code will generate the following error, "AttributeError: 'Flower' object has no attribute 'bloomTime'" why do you think that is?

f. Write a line of python to place before `print(iris.bloomTime)` so that the AttributeError won't occur:

3. Observe what happens when we enter the following lines, continuing from those above:

```
7 >>> def countPetals(flwr):
8 ...     return flwr.petal + flwr.sepal

9 >>> countPetals(iris)
10 6
```

a. What argument is being passed to `countPetals` on line 9? What is `countPetals`' parameter named? arg:_____ param:_____

b. Does `iris` or `flwr` appear on the lefthand side of an assignment operator in lines 7-10?

c. Is the `iris` object modified/changed in any way in lines 7-10?

FYI: User-defined object instances can be passed to functions just like built-in object instances.

4. Examine the following code below, that creates a new class in interactive python:

```
11 >>> class Garden:
12 ...     """ Represents a flower garden """

13 >>> myGarden = Garden()
14 >>> myGarden.flower = Flower()
15 >>> myGarden.flower.petals = 21
16 >>> myGarden.flower.petals
17 21
```

- What type of object is `myGarden`? How do you know?

- What type of object is `myGarden.flower`? How do you know?

- What type of object is `myGarden.flower.petals`? How do you know?

- What is new about the assignment of a value to `petals` in this example?

FYI: Embedded objects are used within other objects and can be referred to through dot notation.

5. The following code below continues from the previous example:

```
18 >>> iris.petals = 3
19 >>> myGarden.flower = iris
20 >>> myGarden.flower.petals = 6
21 >>> iris.petals
22 6
```

- What value is assigned to `iris.petals` on line 18? _____
- What value is assigned to `myGarden.flower` on line 19? _____
- What value is assigned to `myGarden.flower.petals` on line 20? _____
- What value is stored in `iris.petals`, according to line 22? _____
- On what line might `iris.petals`' value have been changed to this value? _____

FYI: Objects are mutable. Their attributes can be changed inside of functions or even when embedded in other objects.

6. The following code below continues from the previous example:

```
23 >>> def makeHybrid(flwr1, flwr2):
24 ...     hybrid = Flower()
25 ...     hybrid.petal = (flwr1.petal + flwr2.petal)/2
26 ...     return hybrid

27 >>> daisy = Flower()
28 >>> daisy.petal = 21
29 >>> iraisy = makeHybrid(daisy, iris)
30 >>> iraisy.petal
31 13.5
```

- a. What is the value stored in `iris.petal`? _____
- b. What is the value stored in `daisy.petal`? _____
- c. What is the value stored in `flwr1.petal` in this example? _____
- d. What is the value stored in `flwr2.petal` in this example? _____
- e. When line 25 is executed, what value is assigned to `hybrid.petal`? _____
- f. What type of object is `iraisy`? How do you know?

Application Questions: Use Python to check your work

- 1a. Create a class, `Dog`. Create an instance of `Dog` which has a `name` and an `age` as instance attributes.

- 1b. Write a function, `dogYears`, that takes a `Dog` object as a parameter and returns the dog's age in dog years (multiply age in years by 7).
`def dogYears(aDog):`

- 1c. Create a function, `addNickname` that accepts a `Dog` object as a parameter, and modifies that object by adding a `nickname` attribute to it. The nickname is 'schmoo' appended to the dog's name.

1d. Write a few lines of code for interactive python that uses all of the above functions you wrote:

```
>>> _____  
>>> _____  
>>> _____  
>>> _____  
>>> _____
```