Name:_____ **Partner:** _____

Python Activity 33: Inheritance

Learning Objectives
Students will be able to:
Content:
- Describe the difference between **instance attributes** and **class attributes**.
- Explain how **parent** and **child classes** are relevant to **class inheritance.**

Process:
- Write code that uses both instance and class attributes.
- Build sub-classes that inherit attributes and methods from the parent class.

Prior Knowledge
- Python concepts, including creating user-defined classes with attributes and [special] methods.

If you encounter any issues/typos, please let Iris know! Questions? Ask Iris or the POGILing forum

Critical Thinking Questions:

1. Examine the sample code below.

robot.py

```
0 class EvilRobot:
1     morality = 'evil'
2     __slots__ = ['name']
3     def __init__(self, nm):
4         self.name = nm
```

Interactive python

```
a >>> from robot import EvilRobot
b >>> er1 = EvilRobot('Herbert')
c >>> er1.name
d 'Herbert'
e >>> er2 = EvilRobot('Pearl')
f >>> er1.morality
g 'evil'
h >>> er2.morality
i 'evil'
j >>> EvilRobot.morality
k 'evil'
```

a. What attributes does `EvilRobot` have?

_____

b. How do you know those are its attributes?

_____

c. What type of object is `er1`? _____

d. On what line is `er1's name` attribute on the left-hand side of an assignment operator?

_____

e. On what line is `morality` on the left-hand side of an assignment operator? _____

f. How do the location of the assignments differ between `name` and `morality`?:

_____

g. What are lines c & f & h doing?:

_____

h. Why is the output on lines g & i the same?:

_____

i. How does line j differ from lines f & h?:

_____

FYI: ***Class Attributes*** are attributes that have shared values for all instances of that class. Class attributes are definedinside a class definition but outside any method. ***Instance Attributes*** are attributes that can have different values for different instances.

j. Is `name` an instance or class attribute? How do you know?:

_____

k. Add a line of python to add `mission` as a class attribute of Evil Robot:

_____

2. Examine the sample code below which continues from the above example.

robot.py

```
0 class EvilRobot:
1     morality = 'evil'
2     __slots__ = ['name']
3     def __init__(self, nm):
4         self.name = nm
```

Interactive python

```
# ...continues from above ^^
m >>> er1.name = 'Potato'
n >>> er1.name
o 'Potato'
p >>> er2.name
q 'Pearl'
r >>> EvilRobot.morality = 'bad'
s >>> er1.morality
t 'bad'
u >>> er2.morality
v 'bad'
```

a. What value is stored in `er1.name` after line m?: _____

b. What value is stored in `er2.name` after line m?: _____

c. What value is stored in `er1.morality` after line r?: _____

d. What value is stored in `er2.morality` after line r?: _____

e. What might be displayed if we printed `EvilRobot.morality`?: _____

f. How do lines m & r differ?

_____

e. Why do `er1` and `er2` have the same `morality` after line r, but they do not have the same `name` after line m?

_____

f. Write a line of python to change all `EvilRobots`' `morality` to `'no good'`.:

_____

FYI: The values stored in ***Class Attributes*** can be referenced by placing the class name prior to the dot notation followed by the variable name. This is in contrast to instance attributes, which are referenced by placing the instance prior to the dot notation.

3. Examine the sample code below which is from a different script.

goodbot.py

```
0 class Robot:
1     __slots__ = ['name']
4     def introduce(self):
5           return 'I AM ' + self.name.upper()

6 class GoodRobot(Robot):
7     morality = 'good'
8     __slots__ = []

9  if __name__ == '__main__':
10    gr1 = GoodRobot()
11    gr1.name = 'Fifi'
12    print(gr1.name)
```

a. What type of object is `gr1`? _____

b. What slots does `GoodRobot` have? _____

c. If we added the line `gr1.serialNum=20200410` to line 13, what would happen?

_____

d. What attribute is being modified on line 11? _____

e. What slots does `Robot` have? _____

f. What is different about the class declaration on line 6 that we haven't seen before?

_____

g. Line 12 will print `'Fifi'`, without an error. Why might that be?

_____

FYI: A ***child-class*** or ***sub-class*** is a new class created by inheriting from a ***parent-class*** or ***super-class***. ***Inheritance*** is the ability to define a new class that is a modified version of a previously defined class.

h. If we added the line `gr1.introduce()` to line 13 `'I AM FIFI'` would be displayed. What method was likely called, and what class does it belong to?

4. Examine the sample code below which is a slightly modified version of the previous script.

goodbot.py

```
0 class Robot:
1     __slots__ = ['name']
2     def __init__(self, nm):
3          self.name = nm
4     def introduce(self):
5          return 'I AM ' + self.name.upper()

6 class GoodRobot(Robot):
7     morality = 'good'
8     __slots__ = []

9  if __name__ == '__main__':
14    gr2 = GoodRobot('Stan')
15    print(gr2.name)
```

a. What type of object is `gr2`? _____

b. What class does `GoodRobot` inherit from? _____

c. What is the parameter we are attempting to initialize `gr2` with? _____

d. Does `GoodRobot` have an initializer method with a name parameter? _____

e. Does `Robot` have an initializer method with a name parameter? _____

g. Does `GoodRobot` have `name` slot? _____

h. Does `Robot` have `name` slot? _____

f. Line 15 will display `'Stan'`. What is stored in `gr2.name`? _____

g. In line 14, what special method is being called? Which class does it belong to?

_____

FYI: Child-classes will inherit the attributes, methods, and special methods of their parent-class.

Application Questions: Use the Python Interpreter to check your work

1. a. Create a class, `NeutralRobot`, that inherits from the `Robot` class above. This class should have `morality` as a class attribute, and a `purpose` (i.e., `'sit and watch the grass grow'`, `'long walks on the beach'`, etc.) instance attribute:

_____

_____

_____

b. Write a `__str__(self)` function to display a `NeutralRobot`'s `name, morality`, and `purpose`:

c. Write an `if __name__=='__main__'` function to instantiate a new `NeutralRobot`, assign it `name, morality`, and `purpose` and then print the `NeutralRobot`.