Name:_____ **Partner:** _____

Python Activity 34: Super Methods

Learning Objectives
Students will be able to:
Content:
- Explain how calling parent-class methods explicitly can reduce redundant code.

Process:
- Write sub-classes that invoke parent-class methods explicitly.

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

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

Critical Thinking Questions:

1. Examine the sample code below with its Terminal output in the dotted box.

purposerobot.py

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

4 class PurposefulRobot(Robot):
5     __slots__ = ['purpose']
6     def __init__(self, prp):
7         self.purpose = prp

9  if __name__ == '__main__':
10    pr1 = PurposefulRobot('save the world')
11    print(pr1.purpose)
12    print(pr1.name)
```

Output:
```
save the world
AttributeError:
'PurposefulRobot' has no
attribute 'name'
```

a. What type of object is `pr1`? _____

b. What slots does `PurposefulRobot` have? _____

c. Where does this slot appear on the left-hand side of an assignment operator? _____

d. What value is the slot assigned in the above code? _____

e. What does line 11 output? _____

f. What slots does `PurposefulRobot` inherit? _____

g. Where does this slot appear on the left-hand side of an assignment operator? _____

h. What value is the slot assigned in the above code? _____

i. What does line 12 output? _____

j. How does the difference in lines 11 & 12's output tell us which `__init__(self)` method is being called? _____

_____

k. When we instantiate a `PurposefulRobot` object which `__init__(self)` method is called? _____

l. How would you modify `PurposefulRobot` so that it could be initialized with a `name` attribute? What other line(s) would you have to modify? Does this solution introduce repeated code?

PurposefulRobot: _____

_____

_____

Other lines: _____

Repetitive code?: _____

2. Examine the sample code below which replaces the previous example.

purposerobot.py

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

4 class PurposefulRobot(Robot):
5     __slots__ = ['purpose']
6     def __init__(self, name, prp):
7         self.purpose = prp
8         super().__init__(name)

9  if __name__ == '__main__':
10   pr2 = PurposefulRobot('Spudnik','save the world')
11   print(pr2.purpose)
12   print(pr2.name)
```

Output:
```
save the world
Spudnik
```

a. What type of object is `pr2`? _____

b. What slots does `pr2` have? _____

c. Where are these slots assigned a value?

_____

d. How does the output of line 12 in this example differ from the previous example?

_____

e. How does line 10 in this example differ from line 10 in the previous example?

_____

f. How does `PurposefulRobot`'s initializer method header differ compared to the previous example?

_____

g. What line was added to `PurposefulRobot`'s initializer method?

_____

h. What might be the values being passed in this line for the above example?

_____

i. What method is likely being called on line 8?

_____

FYI: A ***parent-class'*** methods can be called from within a ***child-class*** by using the ***super()*** function, rather than an instance-name. This is useful for reducing redundant code.

j. If we added the following method to `Robot`:

```
def introduce(self):
    return 'I AM ' + self.name.upper()
```

…How might we add an `introduce(self)` method to `PurposefulRobot` so that it returns the robot's name introduction with its purpose in all uppercase lettering without repetitive code? _____

_____

_____