Name: Md Sajid Ansari
Full Stack Data Science- iNeuron 2021
Python Basic Assignment: Assignment 19

Question 1.
Make a class called Thing with no contents and print it. Then, create an object called example from this class and also print it. Are the printed values the same or different?

Answer 1:

```
class Thing:
    pass
print(Thing)

Output: <class '__main__.Thing'>
```

```
example = Thing()
print(example)

Output: <__main__.Thing object at 0x000002B5D9C00D00>
```

Question 2.
Create a new class called Thing2 and add the value 'abc' to the letters class attribute. Letters should be printed.

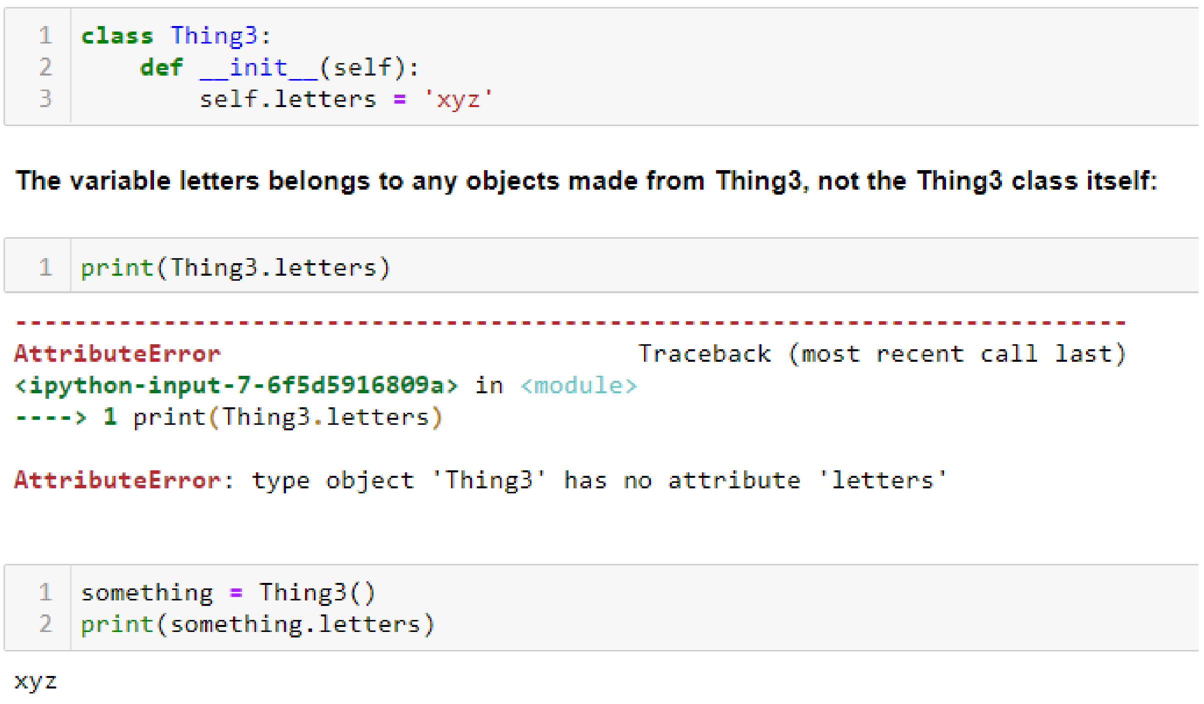
Answer 2:

```
class Thing2:
    letters = 'abc'
print(Thing2.letters)

Output: abc
```

Question 3.
Make yet another class called, of course, Thing3. This time, assign the value 'xyz' to an instance (object) attribute called letters. Print letters. Do you need to make an object from the class to do this?

Answer 3:



```
class Thing3:
    def __init__(self):
        self.letters = 'xyz'
```

The variable letters belongs to any objects made from Thing3, not the Thing3 class itself:

```
print(Thing3.letters)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-7-6f5d5916809a> in <module>
----> 1 print(Thing3.letters)

AttributeError: type object 'Thing3' has no attribute 'letters'
```

```
something = Thing3()
print(something.letters)
```

```
xyz
```

Question 4.

Create an Element class with the instance attributes name, symbol, and number. Create a class object with the values 'Hydrogen,' 'H,' and 1.

Answer 4:

```
class element:
    def __init__(self,name,symbol,number):
        self.name = name
        self.symbol = symbol
        self.number = number
        print("\n name: {} \n symbol: {} \n number: {}".format(self.name,self.symbol,self.number))


ele = element('hydrogen', 'H', 1)
print(ele)
```

```
Output: name: hydrogen
 symbol: H
 number: 1
<__main__.element object at 0x000001EFC189D820>
```

Question 5.

Make a dictionary with these keys and values: 'name': 'Hydrogen', 'symbol': 'H', 'number': 1. Then, create an object called hydrogen from class Element using this dictionary.

Answer 5:

```
#Starting with the dictionary:
el_dict = {'name': 'Hydrogen', 'symbol': 'H', 'number': 1}

#Creating object called hydrogen from class Element using el_dict
hydrogen = Element(el_dict['name'], el_dict['symbol'], el_dict['number'])

hydrogen.name
```

```
Output: 'Hydrogen'
```

We can also initialize the object directly from the dictionary, because its key names match the arguments to "init"

```
hydrogen = Element(**el_dict)
hydrogen.name
```

```
Output: 'Hydrogen'
```

Name: Md Sajid Ansari
Full Stack Data Science- iNeuron 2021
Python Basic Assignment: Assignment 19

Question 6.

For the Element class, define a method called dump() that prints the values of the object’s attributes (name, symbol, and number). Create the hydrogen object from this new definition and use dump() to print its attributes.

Answer 6:

```
class Element:
    def __init__(self, name, symbol, number):
        self.name = name
        self.symbol = symbol
        self.number = number
    def dump(self):
        print('name=%s, symbol=%s, number=%s' %(self.name, self.symbol, self.number))
hydrogen = Element(**el_dict)
hydrogen.dump()
```

```
Output: name=Hydrogen, symbol=H, number=1
```

Question 7.

Call print(hydrogen). In the definition of Element, change the name of method dump to __str__, create a new hydrogen object, and call print(hydrogen) again.

Answer 7:

```
print(hydrogen)
```

```
Output: <__main__.Element object at 0x000002B5DB25E760>
```

```
class Element:
    def __init__(self, name, symbol, number):
        self.name = name
        self.symbol = symbol
        self.number = number
    def __str__(self):
        return ('name=%s, symbol=%s, number=%s' %(self.name, self.symbol, self.number))

hydrogen = Element(**el_dict)
print(hydrogen)
```

```
Output: name=Hydrogen, symbol=H, number=1
```

Question 8.

Modify Element to make the attributes name, symbol, and number private. Define a getter property for each to return its value.

Answer 8:

```
class Element:
   def __init__(self, name, symbol, number):
      self.__name = name
      self.__symbol = symbol
      self.__number = number
   @property
   def name(self):
      return self.__name
   @property
   def symbol(self):
      return self.__symbol
   @property
   def number(self):
      return self.__number
hydrogen = Element('Hydrogen', 'H', 1)

hydrogen.name
output: 'Hydrogen'

hydrogen.symbol
Output: 'H'

hydrogen.number
Output: 1
```

Question 9.

Define three classes: Bear, Rabbit, and Octothorpe. For each, define only one method: eats(). This should return 'berries' (Bear), 'clover' (Rabbit), or 'campers' (Octothorpe). Create one object from each and print what it eats.

Answer 9:

```
class Bear:
   def eats(self):
      return 'berries'

class Rabbit:
   def eats(self):
      return 'clover'

class Octothorpe:
   def eats(self):
      return 'campers'

b = Bear()
r = Rabbit()
o = Octothorpe()
```

```
print(b.eats())
```
Output: berries

```
print(r.eats())
```
Output: clover

```
print(o.eats())
```
Output: campers

Question 10.

Define these classes: Laser, Claw, and SmartPhone. Each has only one method: does(). This returns 'disintegrate' (Laser), 'crush' (Claw), or 'ring' (SmartPhone). Then, define the class Robot that has one instance (object) of each of these. Define a does() method for the Robot that prints what its component objects do.

Answer 10:

```
class Laser:
    def does(self):
        return 'disintegrate'

class Claw:
    def does(self):
        return 'crush'

class SmartPhone:
    def does(self):
        return 'ring'

class Robot:
    def __init__(self):
        self.laser = Laser()
        self.claw = Claw()
        self.smartphone = SmartPhone()

    def does(self):
        return '''I have many attachments:
        My laser, to %s.
        My claw, to %s.
        My smartphone, to %s.''' % (self.laser.does(),self.claw.does(),self.smartphone.does() )
robbie = Robot()
print( robbie.does())
```

```
Output: I have many attachments:
                    My laser, to disintegrate.
                    My claw, to crush.
                    My smartphone, to ring.
```