ASSIGNMENT - 19 (PYTHON)

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?

```
# Define the Thing class
class Thing:
    pass

# Print the Thing class
print(Thing)

# Create an object called example from the Thing class
example = Thing()

# Print the example object
print(example)
```

Output will be:
<class '__main__.Thing'>
<__main__.Thing object at 0x7fe3cf63e940>

The printed values are different. The first one is the representation of the class Thing, indicating it's a class object, while the second one is the representation of the instance example created from the Thing class, showing it's an object of that class with its memory address.

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

```
class Thing2:
    # Define the class attribute 'letters'
    letters = 'abc'

# Print the value of the class attribute 'letters'
print(Thing2.letters)
```

Output will be:

abc

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?

```
class Thing3:
    # Define the class constructor
    def __init__(self):
        # Assign the value 'xyz' to the instance attribute 'letters'
        self.letters = 'xyz'

# Create an object (instance) of the Thing3 class
thing3_obj = Thing3()

# Print the value of the instance attribute 'letters'
print(thing3_obj.letters)
```

Output: xyz

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

```
class Element:
    # Define the class constructor with instance attributes
    def __init__(self, name, symbol, number):
        self.name = name
        self.symbol = symbol
        self.number = number

# Create a class object with the specified values
hydrogen = Element('Hydrogen', 'H', 1)

# Print the values of the instance attributes of the class object
print(hydrogen.name)
print(hydrogen.symbol)
print(hydrogen.number)
```

Output will be:
Hydrogen
H
1

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.

```
class Element:
    # Define the class constructor with instance attributes
    def __init__(self, name, symbol, number):
        self.name = name
        self.symbol = symbol
        self.number = number

# Create a dictionary with the specified keys and values
hydrogen_data = {'name': 'Hydrogen', 'symbol': 'H', 'number': 1}

# Create an object called hydrogen from the Element class using the dictionary
hydrogen = Element(**hydrogen_data)

# Print the values of the instance attributes of the hydrogen object
print(hydrogen.name)
print(hydrogen.symbol)
print(hydrogen.number)
```

Output will be:

```
Hydrogen
H
1
```

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.

```
class Element:
    # Define the class constructor with instance attributes
    def __init__(self, name, symbol, number):
        self.name = name
        self.symbol = symbol
        self.number = number

    # Define the dump method to print the attributes
    def dump(self):
        print(f"Name: {self.name}")
        print(f"Symbol: {self.symbol}")
        print(f"Number: {self.number}")
```

```
# Create the hydrogen object from the Element class
hydrogen = Element('Hydrogen', 'H', 1)

# Use the dump method to print the attributes of the hydrogen object
hydrogen.dump()
```

Output will be:
Name: Hydrogen
Symbol: H
Number: 1

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.

```
class Element:
    # Define the class constructor with instance attributes
    def __init__(self, name, symbol, number):
        self.name = name
        self.symbol = symbol
        self.number = number

    # Define the __str__ method to return a string representation of the object
    def __str__(self):
        return f"Name: {self.name}, Symbol: {self.symbol}, Number: {self.number}"

# Create the hydrogen object from the Element class
hydrogen = Element('Hydrogen', 'H', 1)

# Call print(hydrogen) to print the object using its string representation
print(hydrogen)
```

Output will be:
Name: Hydrogen, Symbol: H, Number: 1

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

```
class Element:
    # Define the class constructor with private instance attributes
    def __init__(self, name, symbol, number):
        self.__name = name
```

```
        self.__symbol = symbol
        self.__number = number

    # Define getter properties for the private attributes
    @property
    def name(self):
        return self.__name

    @property
    def symbol(self):
        return self.__symbol

    @property
    def number(self):
        return self.__number

# Create the hydrogen object from the Element class
hydrogen = Element('Hydrogen', 'H', 1)

# Access the attributes using the getter properties
print(hydrogen.name)
print(hydrogen.symbol)
print(hydrogen.number)
```

Output will be:
Hydrogen
H
1

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.

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

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

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

```
# Create objects from each class
bear = Bear()
rabbit = Rabbit()
octothorpe = Octothorpe()

# Print what each object eats
print("Bear eats:", bear.eats())
print("Rabbit eats:", rabbit.eats())
print("Octothorpe eats:", octothorpe.eats())
```

Output will be:

```
Bear eats: berries
Rabbit eats: clover
Octothorpe eats: campers
```

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.

```
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):
        print("Laser:", self.laser.does())
        print("Claw:", self.claw.does())
```

```
        print("SmartPhone:", self.smartphone.does())

# Create an object of the Robot class
robot = Robot()

# Call the does() method of the Robot object
robot.does()
```

Output will be:

```
Laser: disintegrate
Claw: crush
SmartPhone: ring
```