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:

The output is the same :-

```
class Thing:
    def __self__(self):
        self=self

    def something(self):
        return "this class"

objThing=Thing()

objThing.something()


OUTPUT >>> this class
```

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

Answer:

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

OUTPUT >>> 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?

Answer:

Yes an object has to be created to access the attribute letters within Thing3 class :-

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

objThing3=Thing3()
objThing3.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.

Answer:

```
class Element:
   name=''
   symbol=''
   number=0
   def __init__(self):
      self.name='Hydrogen'
      self.symbol='H'
      self.number=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.

Answer:

```
class Element:
   name=''
   symbol=''
   number=0
   def __init__(self,name,symbol,number):
      self.name=name
      self.symbol=symbol
      self.number=number


dict1={'name':'Hydrogen','symbol':'H','number':1}
objElement = Element(dict1['name'], dict1['symbol'], dict1['number'])

OUTPUT >>> objElement.name
'Hydrogen'
```

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:

```
class Element:
   name=''
   symbol=''
   number=0
   def __init__(self,name,symbol,number):
      self.name=name
      self.symbol=symbol
      self.number=number
   def dump(self):
      return self.name,self.symbol,self.number

dict1={'name':'Hydrogen','symbol':'H','number':1}
objElement = Element(**dict1)
objElement.dump()


OUTPUT >>> ('Hydrogen', 'H', 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.

Answer:

```
class Element:
   name=''
   symbol=''
   number=0
   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))



dict1={'name':'Hydrogen','symbol':'H','number':1}
hydrogen = Element(**dict1)

print(hydrogen)
```

```
OUTPUT >>> 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.

Answer:

```
class Element:

    def __init__(self,name,symbol,number):
        self._name=name
        self._symbol=symbol
        self._number=number

    @property
    def name(self):
        print('getting name value')
        return self._name

    @property
    def symbol(self):
        print('getting symbol value')
        return self._symbol

    @property
    def number(self):
        print('getting number value')
        return self._number

hydrogen=Element('Hydrogen','H',1)
print(hydrogen.name)
print(hydrogen.symbol)
print(hydrogen.number)

OUTPUT >>> getting name value
Hydrogen
getting symbol value
H
getting number value
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.

Answer:

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


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


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

print(Bear().eats())
print(Rabbit().eats())
print(Octothorpe().eats())


OUTPUT >>> berries
clover
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.

Answer:

```
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(self.laser.does())
        print(self.claw.does())
        print(self.smartphone.does())


robot=Robot()
robot.does()
```

OUTPUT >>> **disintegrate**
crush
ring