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?

ANS: Printed values are not came, it prints, class and object details.

In [63]:

**class** Thing:

**pass**

In [64]:

print(Thing)

<class '\_\_main\_\_.Thing'>

In [65]:

example **=** Thing()

print(example)

<\_\_main\_\_.Thing object at 0x000001CCE2181190>

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

ANS: **class** Thing2:

letters **=** 'abc'

Thing2**.**letters

Out[66]:

'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?

ANS: **class** Thing3:

**def** \_\_init\_\_(self,letter):

self**.**letter **=** letter

**def** letters(self):

print(self**.**letter)

Thing3('xyz')**.**letters()

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.

ANS: **class** Element:

**def** \_\_init\_\_(self,name,symbol,number):

self**.**name **=** name

self**.**symbol **=** symbol

self**.**number **=** number

**def** printThem(self):

print(self**.**name,self**.**symbol,self**.**number)

In [69]:

obj **=** Element('Hydrogen','H',1)

obj**.**name

Out[69]:

'Hydrogen'

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.

ANS: dict **=** {'name':'Hydrogen', 'symbol':'H','number': 1}

hydrogen **=** Element(**\*\***dict) *# using dictionary unpacking \*\**

hydrogen**.**symbol

Out[70]:

'H'

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.

ANS: **class** Element:

**def** \_\_init\_\_(self,name,symbol,number):

self**.**name **=** name

self**.**symbol **=** symbol

self**.**number **=** number

**def** dump(self):

print(self**.**name,self**.**symbol,self**.**number)

In [72]:

hydrogen **=** Element('Hydrogen','H',1)

hydrogen**.**dump()

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.

ANS: **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) )

In [74]:

hydrogen **=** Element('Hydrogen','H',1)

print(hydrogen)

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.

ANS: **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)

In [76]:

hydrogen **=** Element('Hydrogen','H',1)

hydrogen**.**name

Out[76]:

'Hydrogen'

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.

ANS: **class** Bear:

**def** eats():

print('berries')

**class** Rabbit:

**def** eats():

print('clover')

**class** Octothorpe:

**def** eats():

print('campers')

In [78]:

Bear**.**eats()

berries

In [79]:

Rabbit**.**eats()

clover

In [80]:

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.

ANS: **class** Laser:

**def** does(self):

**return**('disitegrate')

**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** ('Laser is %s, Claw is %s, SmartPhone is %s' **%** (self**.**laser**.**does(),self**.**claw**.**does(),self**.**smartphone**.**does()))

In [82]:

robo **=** Robot()

In [83]:

robo**.**does()

Out[83]:

'Laser is disitegrate, Claw is crush, SmartPhone is ring'