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:**

class thing:

pass

print(thing)

obj=thing()

print(obj)

yes, output of both objects are different:

1st print output: <class '\_\_main\_\_.thing'>

2nd print output: <\_\_main\_\_.thing object at 0x000001B03E673508>

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'

obj=thing2()

print(obj.letters)

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:

pass

thing3.letters='xyz'

print(thing3.letters)

- no we didn’t need to make an object from the class.

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

obj=Element('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.

**Ans:**

class Element:

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

self.name=name

self.symbol=symbol

self.number=number

d={'name': 'Hydrogen', 'symbol': 'H', 'number': 1}

obj=Element(d['name'], d['symbol'], d['number'])

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)

print(self.symbol)

print(self.number)

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

obj.dump()

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):

print(self.name)

print(self.symbol)

print(self.number)

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

print(hydrogen)

this is returning below error:

TypeError: \_\_str\_\_ returned non-string (type NoneType)

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

def name(self):

return self.\_\_name

def symbol(self):

return self.\_\_symbol

def number(self):

return self.\_\_number

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(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())

print(r.eats())

print(o.eats())

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 'disintegrate'

class claw:

def does(self):

return 'crush'

class smartphone:

def does(self):

return 'ring'

class robot:

def \_\_init\_\_(self, laser, claw, smartphone):

self.laser=laser

self.claw=claw

self.smartphone=smartphone

def does(self):

return "laser does - "+self.laser.does()+", claw does - "+self.claw.does()+", smartphone does - "+self.smartphone.does()

l=laser()

c=claw()

s=smartphone()

r=robot(l, c, s)

print(r.does())