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?

class Thing:

pass

Thing

\_\_main\_\_.Thing

example= Thing()

<\_\_main\_\_.Thing at 0x2c45843ca00>

Here in object print output getting the additional memory location

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

class Thing2:

letter ='abc'

print(letter)

Thing2.letter

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?

No need to create an object we can access the Class attribute with Class name

class Thing3:

letter= 'xyz'

Thing3.letter

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:

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

self.name = name

self.symbol = symbol

self.number = number

E = 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.

class Element:

def \_\_init\_\_(self, data):

self.dd = data

for k, v in dd.items():

print(k,v)

dd ={'name': 'Hydrogen', 'symbol': 'H', 'number': 1

r = Element(dd)

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:

def \_\_init\_\_(self, data):

self.dd = data

for k, v in dd.items():

print(k,v)

def dump(self,name, symbol, number ):

print(name,symbol,number)

hydrogen = Element(dd)

hydrogen.dump('aj','hydra','123')

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.

print(hydrogen)

output: <\_\_main\_\_.Element object at 0x000002C4598D5490>

After overriding the \_\_str\_\_:

**---------------------------------------------------------------------------**

**TypeError** Traceback (most recent call last)

**<ipython-input-52-60c7bef336f2>** in <module>

**----> 1** print**(**hydrogen**)**

**TypeError**: \_\_str\_\_() missing 3 required positional arguments: 'name', 'symbol', and 'number'

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

class Element():

\_\_name='abc'

\_\_symbol='sym'

\_\_num =124

def get\_method(self):

self.name = Element.\_\_name

self.symbol = Element.\_\_symbol

self.number = Element.\_\_num

return self.name, self.symbol, self.number

e= Element()

e.get\_method()

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'

B = Bear()

B.eats()

R = Rabbit()

R.eats()

O = Octothorpe()

O.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.

class laser:

def does(self):

return 'disintegrate'

class Claw:

def does(self):

return 'crush'

class Smartphone:

def does(self):

return 'ring'

L = laser()

C = Claw()

S = Smartphone()

class Robot:

L,C, S

def does(self):

print(L.does())

print(C.does())

print(S.does())

Output:

R =Robot()

R.does()

disintegrate

crush

ring