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?

A:

class thing:

pass

print(thing)

o/p

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

o/p

example=thing()

example

<\_\_main\_\_.thing at 0x7fde8c265d50>

2. Create a new class called Thing2 and add the value &#39;abc&#39; to the letters class attribute. Letters

should be printed.

A:

class thing2:

letters="abc"

a=thing2()

print(a.letters)

3. Make yet another class called, of course, Thing3. This time, assign the value &#39;xyz&#39; to an instance

(object) attribute called letters. Print letters. Do you need to make an object from the class to do

this?

A:

class Thing3:

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

self.letters = 'xyz'

a=Thing3()

print(a.letters)

4. Create an Element class with the instance attributes name, symbol, and number. Create a class

object with the values &#39;Hydrogen,&#39; &#39;H,&#39; and 1.

A:

class Element:

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

self.name ='Hydrogen'

self.symbol ='H'

self.number=1

x=Element()

x.name

x.number

5. Make a dictionary with these keys and values: &#39;name&#39;: &#39;Hydrogen&#39;, &#39;symbol&#39;: &#39;H&#39;, &#39;number&#39;: 1. Then,

create an object called hydrogen from class Element using this dictionary.

A:

class Element:

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

self.name ='name'

self.symbol ='symbol'

self.number='number'

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

el\_dict = {'name': 'Hydrogen', 'symbol': 'H', 'number': 1}

hydrogen = Element(el\_dict['name'], el\_dict['symbol'], el\_dict['number'])

hydrogen.name

hydrogen.symbol

or

hydrogen = Element(\*\*el\_dict)

hydrogen.name

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.

A:

class Element:

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

self.name = name

self.symbol = symbol

self.number = number

def dump(self):

print('name=%s, symbol=%s, number=%s' %(self.name, self.symbol, self.number))

hydrogen = Element(\*\*el\_dict)

hydrogen.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.

A:

<\_\_main\_\_.Element object at 0x7fde8c31b280>

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))

hydrogen = Element(\*\*el\_dict)

print(hydrogen)

8. Modify Element to make the attributes name, symbol, and number private. Define a getter

property for each to return its value.

A:

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

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

9. Define three classes: Bear, Rabbit, and Octothorpe. For each, define only one method: eats(). This

should return &#39;berries&#39; (Bear), &#39;clover&#39; (Rabbit), or &#39;campers&#39; (Octothorpe). Create one object from

each and print what it eats.

A:

class Bear:

def eat(self):

return "barries"

class rabbit:

def eat(self):

return "clover"

class Octothrope:

def eat(self):

return "campers"

a=Bear()

print(a.eat())

b=rabbit()

print(b.eat())

c=Octothrope()

print(c.eat())

10. Define these classes: Laser, Claw, and SmartPhone. Each has only one method: does(). This

returns &#39;disintegrate&#39; (Laser), &#39;crush&#39; (Claw), or &#39;ring&#39; (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.

A:

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):

return '''I have many attachments:

My laser, to %s.

My claw, to %s.

My smartphone, to %s.''' % (self.laser.does(),self.claw.does(),self.smartphone.does() )

robbie = Robot()

print( robbie.does())