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?

printed values are different.

class Class1:

pass

print(Class1)

example = Class1()

print(example)

gives output as

<class '\_\_main\_\_.Class1'>

<\_\_main\_\_.Class1 object at 0x000001F615DB6260>

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

class Thing2:

letters = 'abc'

# printing letters directly from a class 'Thing2'

print(Thing2.letters)

# printing letters from an object created from a class 'Thing2'

object1 = Thing2()

print(object1.letters)

1. 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?

Yes. You need to make an object from the class to print letters.

If you are assigning attribute value to an object of the class, you need to print object of the class

class Thing3:

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

self.letters = letters

# create an object of the class

# assign the value 'xyz' to an instance (object) attribute called letters.

object1 = Thing3("xyz")

# print letters

print(object1.letters)

1. Create an Element class with the instance attributes name, symbol, and number. Create a class object with the values 'Hydrogen,' 'H,' and 1.

# 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

# Create a class object with the values 'Hydrogen,' 'H,' and 1.

object1 = Element("Hydrogen","H",1)

# print the object attributes

print(object1.name)

print(object1.symbol)

print(object1.number)

1. 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.

# 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.

dict1 = {'name': 'Hydrogen', 'symbol': 'H', 'number': 1}

# store dictionary values into the list object, l

l = list(dict1.values())

# check if list object, l contains all the dictionary values

print(l)

# Create a class Element

class Element:

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

self.name = name

self.symbol = symbol

self.number = number

# Create a class object with the values passed from list object, l

object1 = Element(l[0],l[1],l[2])

# print the object attributes

print(object1.name)

print(object1.symbol)

print(object1.number)

1. 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.

# 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.

# Declare a dictionary variable dict1

dict1 = {'name': 'Hydrogen', 'symbol': 'H', 'number': 1}

# store dictionary values into the list object, l

l = list(dict1.values())

# check if list object, l contains all the dictionary values

print(l)

# Create a class Element

class Element:

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

self.name = name

self.symbol = symbol

self.number = number

# define a dump method to print the attributes of an object

def dump(self):

print(self.name)

print(self.symbol)

print(self.number)

# Create a class object with the values passed from list object, l

object1 = Element(l[0],l[1],l[2])

# print the object attributes, using the dump() method

print(object1.dump())

1. 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.

# 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.

# Declare a dictionary variable dict1

dict1 = {'name': 'Hydrogen', 'symbol': 'H', 'number': 1}

# store dictionary values into the list object, l

l = list(dict1.values())

# Create a class Element

class Element:

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

self.name = name

self.symbol = symbol

self.number = number

# In the definition of Element, change the name of method dump to \_\_str\_\_,

def \_\_str\_\_(self):

print(self.name)

print(self.symbol)

print(self.number)

# Create a class object with the values passed from list object, l

object1 = Element(l[0],l[1],l[2])

# create a new hydrogen object, and call print(hydrogen) again.

obj1 = object1.name

print(obj1)

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

# Modify Element to make the attributes name, symbol, and number private.

# Define a getter property for each to return its value.

# Declare a dictionary variable dict1

dict1 = {'name': 'Hydrogen', 'symbol': 'H', 'number': 1}

# store dictionary values into the list object, l

l = list(dict1.values())

# check if list object, l contains all the dictionary values

print(l)

# Create a class Element

class Element:

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

self.\_\_name = name

self.\_\_symbol = symbol

self.\_\_number = number

# define a getter property for each to return its value.

def getnumber(self):

return self.\_\_number

def getname(self):

return self.\_\_name

def getsymbol(self):

return self.\_\_symbol

# Create a class object with the values passed from list object, l

object1 = Element(l[0],l[1],l[2])

# print the private attributes using the method each of the getter method of the attribute

print(object1.getnumber())

print(object1.getname())

print(object1.getsymbol())

1. 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.

# 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.

# creaet Bear, Rabbit and Octothorpe

class Bear:

def eats(self):

return 'berries'

class Rabbit:

def eats(self):

return 'clover'

class Octothorpe:

def eats(self):

return 'campers'

# create object for each class and print its eat() method

objBear = Bear()

print(objBear.eats())

objRabbit = Rabbit()

print(objRabbit.eats())

objOctorpe = Octothorpe()

print(objOctorpe.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.

# 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.

# create Laser, Claw, Smartphone classes

class Laser:

def does(self):

return 'disintegrate'

class Claw:

def does(self):

return 'crush'

class SmartPhone:

def does(self):

return 'ring'

# create Robot class

class Robot:

objLaser = Laser()

objClaw = Claw()

objSmartPhone = SmartPhone()

print(type(objLaser))

def does():

Laser.does()

Claw.does()

SmartPhone.does()