 **Create a class Thing with no contents and print it, then create an object example:**

python

Copy code

class Thing:

pass

print(Thing) # Outputs: <class '\_\_main\_\_.Thing'>

example = Thing()

print(example) # Outputs: <\_\_main\_\_.Thing object at 0x...>

* The printed values are different. Thing refers to the class itself, while example refers to an instance of that class.

 **Create a class Thing2 and add a class attribute letters with the value 'abc':**

python

Copy code

class Thing2:

letters = 'abc'

print(Thing2.letters) # Outputs: 'abc'

 **Create a class Thing3 with an instance attribute letters and print it:**

python

Copy code

class Thing3:

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

self.letters = 'xyz'

# Create an object from the class

obj = Thing3()

print(obj.letters) # Outputs: 'xyz'

* Yes, you need to create an object from the class to access instance attributes.

 **Create an Element class and an object with attributes name, symbol, and number:**

python

Copy code

class Element:

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

self.name = name

self.symbol = symbol

self.number = number

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

 **Create a dictionary and an Element object from this dictionary:**

python

Copy code

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

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

 **Define a dump() method in Element to print the object's attributes:**

python

Copy code

class Element:

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

self.name = name

self.symbol = symbol

self.number = number

def dump(self):

print(f'Name: {self.name}, Symbol: {self.symbol}, Number: {self.number}')

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

hydrogen.dump() # Outputs: Name: Hydrogen, Symbol: H, Number: 1

 **Change the dump method to \_\_str\_\_ and print the hydrogen object:**

python

Copy code

class Element:

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

self.name = name

self.symbol = symbol

self.number = number

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

return f'Name: {self.name}, Symbol: {self.symbol}, Number: {self.number}'

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

print(hydrogen) # Outputs: Name: Hydrogen, Symbol: H, Number: 1

 **Make the attributes private and define getter properties:**

python

Copy code

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)

print(hydrogen.name) # Outputs: Hydrogen

print(hydrogen.symbol) # Outputs: H

print(hydrogen.number) # Outputs: 1

 **Define classes Bear, Rabbit, and Octothorpe with eats() methods:**

python

Copy code

class Bear:

def eats(self):

return 'berries'

class Rabbit:

def eats(self):

return 'clover'

class Octothorpe:

def eats(self):

return 'campers'

bear = Bear()

rabbit = Rabbit()

octothorpe = Octothorpe()

print(bear.eats()) # Outputs: berries

print(rabbit.eats()) # Outputs: clover

print(octothorpe.eats()) # Outputs: campers

 **Define classes Laser, Claw, SmartPhone, and Robot:**

python

Copy code

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):

print(f'Laser: {self.laser.does()}')

print(f'Claw: {self.claw.does()}')

print(f'SmartPhone: {self.smartphone.does()}')

robot = Robot()

robot.does()