- 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?

In [1]:
```
class Thing:
    pass
```

In [2]:
```
print(Thing)
```

```
<class '__main__.Thing'>
```

In [3]:
```
ex = Thing()
print(ex)
```

```
<__main__.Thing object at 0x000002ABC7AF77C0>
```

In [4]:
```
# Printed values are not same, it prints, class and object details.
```

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

In [5]:
```
class Things2:
    letter = 'abc'

Things2.letter
```

Out[5]:
```
'abc'
```

- 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?

In [7]:
```
class Things3:
    def __init__(self,letter):
        self.letter = letter
    def letters(self):
        print(self.letter)

Things3('xyz').letters()
```

```
xyz
```

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

In [8]:
```
class Element:
    def __init__(self,name,symbol,number):
        self.name = name
        self.symbol = symbol
        self.number = number

    def printThem(self):
        print(self.name, self.symbol, self.number)
```

In [9]:
```
obj = Element('hydrogen','h',1)
obj.printThem()
```

```
hydrogen h 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.

In [10]:
```
dict = {'name':'Hydrogen', 'symbol':'H','number': 1}
hydrogen = Element(**dict)
hydrogen.symbol
```

Out[10]:
```
'H'
```

- 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.

In [11]:
```
class Element:
    def __init__(self,name,symbol,number):
        self.name = name
        self.symbol = symbol
        self.number = number

    def dump(self):
        print(self.name, self.symbol, self.number)
```

In [12]:
```
hydrogen = Element('Hydrogen','H',1)
hydrogen.dump()
```

```
Hydrogen H 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.

In [20]:
```
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.num
```

In [21]:
```
hydrogen = Element('Hydrogen','H',1)
print(hydrogen)
```

```
name=Hydrogen, symbol=H, number=1
```

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

In [26]:
```
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)
```

In [27]:
```
h = Element('hydrogen','H',1)
h.name
```

Out[27]:
```
'hydrogen'
```

- 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.¶

In [ ]:
```

```