In [ ]:
```
#Object Orientation ... classes
#Class body
#data - variables - data members
#operations - methods - member functions
#create instances (use instances) objects

# the secret of NOT (name order type!!!!!!)
```

In [ ]:
```
class Bird():
    #this is the class body
    def __init__(self, name, age, breed):
        self.name=name
        self.age=age
        self.breed=breed
    def birdsit(self):
        print(self.name + ' is a bird that is sitting')
    def birdfly(self):
        print(self.name + ' is a bird that is flying')
    def birdage(self):
        print(str(self.age) + ' is a bird age')
    def birddata(self):
        print(str(self.name) + ' is a '+str(self.breed))

#modify this to use user input.
myBird=Bird('Pink',4)
myBird.birdsit()
```

In [ ]:
```
#define the class object
class triangle:
    #an attribute of the class
    base=9
    height=4

print(triangle.base)
print(triangle.height)
```

In [ ]:
```
#define the class object and use the built in functions
class triangle:
    #an attribute of the class
    base=9
    height=4
print(triangle.base)
print(triangle.height)
print('class name is ', triangle.__name__)
```

In [ ]:
```
#define the class object
#the function defined in a class is called a method
#data members: NB difference between class and instance variables .... test
class triangle:
    #an attribute of the class
    base=9
    height=4
    def calc(self):
        area=0.5*triangle.base*triangle.height
        return (area)

#create an instance
myTri=triangle()
print(myTri.base)
print(myTri.height)
#Python add self automatically!
myTri.calc()
```

In [ ]:
```python
#define the class object introducing the initialization ...
#the function defined in a class is called a method
#data members: NB difference between class and instance variables .... test
class triangle:
    def __init__(self):
        self.base=9
        self.height=4
    def calc(self):
        area=0.5*self.base*self.height
        return (area)

#create an instance
myTri=triangle()
print(myTri.base)
print(myTri.height)
#Python add self automatically!
myTri.calc()
```

In [ ]:
```python
#define the class object introducing the initialization ...
#the function defined in a class is called a method
#data members: NB difference between class and instance variables .... test
class triangle:
    def __init__(self, x, y):
        self.base=x
        self.height=y
    def calc(self):
        area=0.5*self.base*self.height
        return (area)

#create an instance

myTri=triangle(12,5)
print(myTri.base)
print(myTri.height)
#Python add self automatically!
myTri.calc()
```

```
In [ ]:
#define the class object introducing the initialization ...
#the function defined in a class is called a method
#data members: NB difference between class and instance variables .... test
class triangle:
    def __init__(self, x, y):
        self.base=x
        self.height=y
    def calc(self):
        area=0.5*self.base*self.height
        return (area)

#create an instance
myBase=int(input('Please enter the base length : '))
myHeight=int(input('Please enter the height : '))
myTri=triangle(myBase, myHeight)
print(myTri.base)
print(myTri.height)
#Python add self automatically!
myTri.calc()
```

In [ ]:
```
#define the class object introducing the initialization ...
#the function defined in a class is called a method
#data members: NB difference between class and instance variables .... test
#difference class method vs static method (has no cls parameter!!)
class triangle:
    #setting default values
    def __init__(self, x=10, y=10):
        self.base=x
        self.height=y
    ##using a str
    #def __str__
    #    return
    def calc(self):
        area=0.5*self.base*self.height
        return (area)

#create an instance
myBase=int(input('Please enter the base length : '))
myHeight=int(input('Please enter the height : '))
myTri=triangle(myBase, myHeight)
print(myTri.base)
print(myTri.height)
#Python add self automatically!
myTri.calc()
```

In [ ]:
```
#implementation of garbage collection .....
#work thorugh an example to show how garbage collection frees the memory
```

In [ ]:
```
#inheritance
#can inherit from an existing class base class or super-class
#if you are inheriting, then a derived class or sub-class
```

In [ ]:

```
#specifying the access control
#public member can be accessed from inside AND outside
class triangle:
    #setting default values
    def __init__(self, x=10, y=10):
        self.base=x
        self.height=y
    ##using a str
    #def __str__
    #    return
    def calc(self):
        area=0.5*self.base*self.height
        return (area)
#create an instance
myBase=int(input('Please enter the base length : '))
myHeight=int(input('Please enter the height : '))
myTri=triangle(myBase, myHeight)
print(myTri.base)
print(myTri.height)
#Python add self automatically!
myTri.calc()
```

In [ ]:

```
#private member can be accessed from inside ONLY
class triangle:
    #setting default values
    def __init__(self, x=10, y=10):
        self.__base=x
        self.__height=y
    ##using a str
    #def __str__
    #    return
    def calc(self):
        area=0.5*self.base*self.height
        return (area)
#create an instance
myBase=int(input('Please enter the base length : '))
myHeight=int(input('Please enter the height : '))
myTri=triangle(myBase, myHeight)
print(myTri.base)
print(myTri.height)
#Python add self automatically!
myTri.calc()
```

In [ ]:

```
#create a bird class
class Bird()
```

In [ ]:
```python
#allow the bird class to store the name of the bird and the age of the bird, include the init methods
#include two methods, the bird can sit or the bird can fly
class Bird():

    def __init__(self, name, age):
        self.name = name
        self.age=age
    def birdsit(self):
        print(self.name + ' is a bird that is sitting')
    def birdfly(self):
        print(self.name + ' is a bird that is flying')
    def birdwalk(self)
        print(self.name + ' is a bird that is walking')

#create the instance of the class
myBird=Bird('Blue',4)
print(myBird.name)
myBird.birdsit()
myBird.birdfly()
myBird.birdwalk()
```

In [ ]:
```python
#use the code and create three different instances of Bird()
class Bird():

    def __init__(self, name, age):
        self.name = name
        self.age=age
    def birdsit(self):
        print(self.name + ' is a bird that is sitting')
    def birdfly(self):
        print(self.name + ' is a bird that is flying')

#modify this to use user input.
myBird=Bird('Blue',4)
print(myBird.name)
myBird.birdsit()
myBird.birdfly()
```

In [ ]:
```python
#add the bird type to the attributes of bird
```

In [ ]:
```
#add the method walk to the class
```

In [ ]:
```
#create a class Book() and include title , author and publisher.
#Include the methods eBooAvailable and NoStock with a relevant message.  Suggest two other methods for the cl
ass Book()
```

In [ ]:
```
#create a class Student() Suggest two attributes and two methods for the class student()
```