In [ ]:
```
Class:

field or data member
method
constructor

## Way to create object in java
Test ob = new Test()

## way to create object in python
## implicitly calls constructor - creates object
ob = Test()
```

In [1]:
```
class Test:

    def __init__(self):
        self.a = 10

    def setField(self,data):
        self.a = data

    def getField(self):
        print(self.a)

ob1 = Test()
ob2 = Test()

Test.setField(ob1, 20)
ob1.getField() #Test.getField(ob1)

ob2.setField(35)
ob2.getField()
```

```
20
35
```

Function

In [ ]:
```
Built in - id(), type(), len(), list(), max()
user defined
```

In [5]:
```
l = [int(i) for i in input("Enter values: ").split()]
print(max(l))
```

```
Enter values: 10 21 32 9 56
56
```

In [6]:
```
print(len(l))
```

```
5
```

In [2]:
```
def add():
    a = int(input("Enter val1: "))
    b = int(input("Enter val2: "))
    print(a + b)
    return


add()
```

```
Enter val1: 10
Enter val2: 20
```

In [3]:
```
def add(a, b):
    print(a + b)


add(10, 20)
```

```
30
```

In [4]:
```
def add(a, b):
    return a + b


res = add(10, 20)
print(res)
```

```
30
```

In [7]:
```
def fun():
    print("before pass")
    pass
    print("After pass")

fun()
```

```
before pass
After pass
```

Scope of variables

In [8]:
```
def fun():
    x = 20 #create local variable
    print("Inside function:", x)

x = 10
print("before function", x)
fun()
print("After function:", x)
```

```
before function 10
Inside function: 20
After function: 10
```

In [9]:
```
def fun():
    global x
    x = 20
    print("Inside function:", x)

x = 10
print("Before function: ", x)
fun()
print("After function:", x)
```

```
before function 10
Inside function: 20
After function: 20
```

In [ ]:
```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None


class LinkedList:
    def __init__(self):
        self.head = None

    def insertEnd(self, data):
        pass

    def display():
        pass
```