Example:

```
def calculator(a,b,f):
    res=f(a,b)
    return res

def main():
    res1=calculator(10,20,lambda x,y:x+y)
    res2=calculator(5,3,lambda x,y:x-y)
    res3=calculator(2,4,lambda x,y:x**y)
    print(res1,res2,res3)

main()
```

Output:

```
30 2 16
>>>
```

filter(function, iterable)

Construct an iterator from those elements of iterable for which function returns true. iterable may be either a sequence, a container which supports iteration, or an iterator. If function is None, the identity function is assumed, that is, all elements of iterable that are false are removed.

Example:

```
def main():
    list1=[1,7,9,2,4,8,45,76,89,23,55,67,98,23,45,54,76,67,89,22,33]
    list2=list(filter(lambda n:n%7==0,list1))
    list3=list(filter(lambda n:n%2==0,list1))
    print(list1)
    print(list2)
    print(list3)

main()
```

Output:

```
[1, 7, 9, 2, 4, 8, 45, 76, 89, 23, 55, 67, 98, 23, 45, 54, 76, 67, 89, 22, 33]
[7, 98]
[2, 4, 8, 76, 98, 54, 76, 22]
>>>
```

map(function, iterable, ...)
Return an iterator that applies function to every item of iterable, yielding the results. If additional iterable arguments are passed, function must take that many arguments and is applied to the items from all iterables in parallel. With multiple iterables, the iterator stops when the shortest iterable is exhausted.

Example:

```
def main():
    l1=[1,2,3,4,5]
    l2=[10,20,30,40,50]
    l3=list(map(lambda a,b:a+b,l1,l2))
    print(l1,l2,l3,sep="\n")
    l4=["45","56","65","76","89"]
    l5=list(map(int,l4))
    print(l4,l5,sep="\n")
    l6=list(map(int,input().split(" ")))   # "10 20 30 40 50".split(" ") ==>
["10","20","30","40","50"]
    print(l6)

main()
```

Output:

```
[1, 2, 3, 4, 5]
[10, 20, 30, 40, 50]
[11, 22, 33, 44, 55]
['45', '56', '65', '76', '89']
[45, 56, 65, 76, 89]
10 20 30 40 50
[10, 20, 30, 40, 50]
>>>
```

functools.reduce(function, iterable[, initializer])
Apply function of two arguments cumulatively to the items of iterable, from left to right, so as to reduce the iterable to a single value. For example, reduce(lambda x, y: x+y, [1, 2, 3, 4, 5]) calculates ((((1+2)+3)+4)+5). The left argument, x, is the accumulated value and the right argument, y, is the update value from the iterable. If the optional initializer is present, it is placed before the items of the iterable in the calculation, and serves as a

default when the iterable is empty. If initializer is not given and iterable contains only one item, the first item is returned.

Example:

```
import functools
def main():
    list1=list(range(10,110,10)) # 10 20 30 40 50 60 70 80 90 100
    print(list1)
    m=functools.reduce(lambda a,b:a if a>b else b,list1)
    print(m)
    n=functools.reduce(lambda a,b:a if a<b else b,list1)
    print(n)
    s=functools.reduce(lambda a,b:a+b,list1)
    print(s)

main()
```

Output:

```
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
100
10
550
>>>
```

Modules and Packages

What is module?

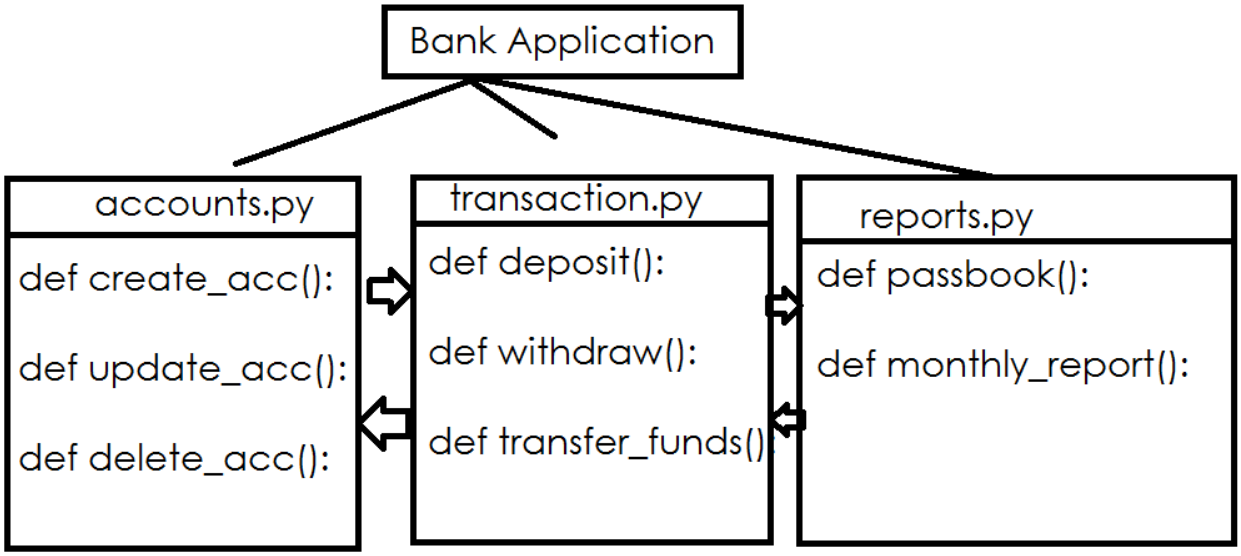
A module is python program, which is saved with extension .py
A module is collection of,

1. Variables/Objects
2. Functions
3. Classes

Python modules are two types

1. Predefined modules
2. User defined modules

The existing modules are called predefined modules
Eg: os,datetime,built-ins,sys,…
The modules which are developed by programmer are called user defined modules.



User defined modules two types

1. Reusable modules
2. Executable modules