Monkey Patching
In Python, the term monkey patch refers *to dynamic (or run-time) modifications of a class or module*.

Monkey patching is a technique used to dynamically update the behavior of a piece of code at run-time. A monkey patch is a way to extend or modify the runtime code of dynamic languages without altering the original source code.

Example:
```
def new_print(*data):
    f=open("file1.txt","a")
    l=list(map(str,data))
    str1="".join(l)
    f.write(str1)
    f.close()

print=new_print

print(10,20,30,40,50)
```

Output:
The output is saved inside a file named file1.txt

Example:
```
def patch(s):
    l=0
    for value in s:
        if value%2==0:
            l=l+1
    return l


len=patch
l1=list(range(1,11))
print(l1)
c=len(l1)
print(c)
```

Output:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
5
```

map,reduce,filter

map, reduce and filter are predefined functions in python. These functions receives input as one function to perform operations.

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

```
>>> list1=["10","20","30","40","50"]
>>> list2=list(map(int,list1))
>>> print(list1)
['10', '20', '30', '40', '50']
>>> print(list2)
[10, 20, 30, 40, 50]
>>> list3=["naresh","suresh","ramesh"]
>>> list4=list(map(str.upper,list3))
>>> print(list3)
['naresh', 'suresh', 'ramesh']
>>> print(list4)
['NARESH', 'SURESH', 'RAMESH']
>>> list5=list(map(float,list1))
>>> print(list5)
[10.0, 20.0, 30.0, 40.0, 50.0]
>>> l1=[1,2,3,4,5]
>>> l2=[6,7,8,9,10]
>>> l3=list(map(lambda x,y:x+y,l1,l2))
>>> print(l1)
[1, 2, 3, 4, 5]
>>> print(l2)
[6, 7, 8, 9, 10]
>>> print(l3)
[7, 9, 11, 13, 15]
```

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.

```
>>> import functools
>>> list1=[10,20,30,40,50]
>>> res1=functools.reduce(lambda x,y:x+y,list1)
>>> print(res1)
150
>>> res2=functools.reduce(lambda x,y:x if x>y else y,list1)
>>> print(res2)
50
>>> res3=functools.reduce(lambda x,y:x if x<y else y,list1)
>>> print(res3)
10
>>> l2=["a","b","c","d","e"]
>>> str1=functools.reduce(lambda x,y:x+y,l2)
>>> print(l2)
['a', 'b', 'c', 'd', 'e']
>>> print(str1)
abcde
```

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.

```
>>> list1=["a","b","d","e","f","g","i","j"]
>>> list2=list(filter(lambda x:x in "aeiouAEIOU",list1))
>>> print(list1)
['a', 'b', 'd', 'e', 'f', 'g', 'i', 'j']
>>> print(list2)
['a', 'e', 'i']
>>> list3=list(range(1,11))
>>> print(list3)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list4=list(filter(lambda x:x%2==0,list3))
>>> print(list4)
[2, 4, 6, 8, 10]
>>> list5=list(filter(lambda x:x%2!=0,list3))
```

```
>>> print(list5)
[1, 3, 5, 7, 9]
>>> list6=[1,2,3,4,5,0,0,0,6,7,8,0,9,0]
>>> list7=list(filter(None,list6))
>>> print(list7)
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Modules

What is module?

Module is a python program.