Higher Order Functions

A lecture

Rushikesh K. Joshi
Computer Science & Engineering
IIT Bombay

What is a higher order function?

A function that takes in a parameter which itself is a function, or the one that returns a value which is a function

- an input parameter is a function

  OR (logical OR!)

– its return result is a function

Example

```
def f(g,l):
    return g(l)

def func1(l):
    sum=0
    for x in l:
        sum = sum+x
    return sum

def func2(l):
    prod=1
    for x in l:
        prod= prod*x
    return prod

l = [1,2,3,4]
v1 = f (func1,l)
v2 = f (func2,l)
print (v1,v2),
```

```
def f(g,l):
    return g(l) → g is function

def func1(l):
    sum=0
    for x in l:
        sum = sum+x
    return sum

def func2(l):
    prod=1
    for x in l:
        prod= prod*x
    return prod

l = [1,2,3,4]
v1 = f (func1,l)   → func1 is function
v2 = f (func2,l)   → func2 is function
print (v1,v2),
```

Example

```
def f(g,l):
    return g(l)

def func1(l):
    sum=0
    for x in l:
        sum = sum+x
    return sum

def func2(l):
    prod=1
    for x in l:
        prod= prod*x
    return prod

l = [1,2,3,4]
v1 = f (func1,l)
v2 = f (func2,l)
print (v1,v2),
```

```
def f(g,l):
    return g(l) → g is function

def func1(l):
    sum=0
    for x in l:
        sum = sum+x
    return sum

def func2(l):
    prod=1
    for x in l:
        prod= prod*x
    return prod

l = [1,2,3,4]
v1 = f (func1,l)   → func1 is function
v2 = f (func2,l)   → func2 is function
print (v1,v2),
```

MAP

```
def f (x):
return x*x

s = {1,3,4}
l = [1,2,3,4,5,6,7]

r = map (f,s)
print (list(r))

r = map (f,l)
print (r)
```

Map invokes the given function passed in as parameter, on all elements in the given collection passed in as another parameter

It returns the collection of all the corresponding results

Reduce

```
import functools

l = [1,2,3,4,5,6,7]


def f (x,y):
print (x,y)
return x+y

r = functools.reduce (f,l)

print (r)
```

Reduce reduces the entire collection to a single value, using all the elements in that collection

To reduce, it uses the function passed in as parameter on the collection passed in as another parameter

Filter

```
import functools
l = [1,2,3,4,5,6,7]

def f (x,y):
    print (x,y)
    return x+y


r = functools.reduce (f,l)
print (r)
```

Filter filters out elements from a collection, and then returns the remaining which are not filtered out

The collection and the filtering function is passed in as parameters

Where can one use Higher Order functions

When you cannot solve it by non-function parameters

Generic function which depends on a changeable 'strategy', or in other words, a changeable 'algorithm'. That strategy or algorithm can be passed in as a function.

The body of this generic higher order function does not change. It just uses the function passed in

In OOP, an object can be passed into a **polymorphic** (pointer to base class) parameter, which carries its own virtual function implementation.