PPL: Lab 5

Lambda , High order functions

What is Lambda?

- Python supports the creation of anonymous functions - that are not bound to a name
- Take a look at this comparison:

```
>>> def F (x): return x**2
>>> print (F(8))
64
>>> g = lambda x: x**2
>>> print (g(8))
64
```

How to use Lambda?

- We use the term “lambda” and specify the arguments it needs like this:

  (lambda x,y,z: (x+y)/z) (1,2,3)

- Common use:

```
>>> def make_incrementor (n): return lambda x: x + n
>>> f = make_incrementor(2)
>>> g = make_incrementor(6)
>>> print (f(42), g(42))
44 48
>>> print ( make_incrementor(22)(33))
55
```

Lists

- A list is a sequence of items of any type (like an array).
- For example:

  myList=[1,2,4,9]

- The for-in statement makes it easy to loop over the items in a list:

```
for item in list:
    statements
```

- For print values of list with the name myList :

```
for x in myList:
    print(x)
```

When to use Lambda?

- Lambda is a very powerful concept that's well integrated into Python and is often used in conjunction with typical functional concepts like filter(), map() and reduce(). A new list is returned! Note that they return an iterator and not list – convert first! list(…)

```
>>> foo = [2, 18, 9, 22, 17, 24, 8, 12, 27]
>>> print (list(filter(lambda x: x % 3 == 0, foo))) #remove unwanted values
[18, 9, 24, 12, 27]
>>> print(list(map(lambda x: x * 2 + 10, foo))) #apply calculation on each value
[14, 46, 28, 54, 44, 58, 26, 34, 64]
>>> from functools import reduce
>>> print(reduce(lambda x, y: x + y, foo)) #take 2 values, apply calc on all values
139
```

High order functions

Functions that manipulate functions are called higher-order functions.

They either:

- accept other functions as arguments

- return functions as values

accept other functions as arguments

```
def summation ( low , high , function , next ):
    total=0
    while low<=high:
        total+=function (low)
        low=next (low)
    return total
def nextTwo ( k ):
    return k+2
print(summation(0,5,lambda x: x**2,lambda x:x+1))
print(summation(0,5,lambda x: x**2,nextTwo))
```

;

return functions as values

```
def make_adder ( n ):
    """return a function that takes argument  k
    and returns  k+n"""
    def adder ( k ):
        return k+n
    return adder
three=make_adder(3)
three(4) #=>7
```