P E S University
Department of Computer Science & Engineering
Session : Aug-Dec 2019

Introduction to Computing using Python Laboratory(UE19CS102)

Week 12– Programs on Functional Programming

1 Given a list of strings:

1)find all strings starting with a given prefix.
2) find all strings having a given substring.

Solution:

1)find all strings starting with a given prefix.

```
string = input("Enter a string : ").split(" ")
print(string)
prefix = input("Enter a prefix: ")
l = list(filter(lambda string:string.startswith(prefix), string))
print("Strings ending with prefix", prefix, " are ", l)
```

2)find all strings having a given substring.

```
string = input("Enter a string : ").split(" ")
print(string)
sub = input("Enter a substring: ")
l = list(filter(lambda s:sub in s, string))
print("Strings having a given substring ", sub, " are ", l)
```

2 Write a function mymap which takes a callback and an iterable, creates a list, applies the callback to each element of the iterable and puts the result into list and returns the list. mymap should mimic map.

Test this with the following calls.
a)Create a list of square of odd numbers from 1 to n.
b)Given a list of words, return a list of words with **ing** appended to it.
c)Given a list of words, return a list of tuples having the word and its length.

Solution:
1) Create a list of square of odd numbers from 1 to n.

```
def odd_square(a):
    return a ** 2


def mymap(my_function, my_list):
    result = []
    for i in my_list:
        result.append(my_function(i))
    return result

n = int(input("Enter value for upper range: "))
l= list( mymap( odd_square, range(1, n, 2)))
print(l)
```

2)Given a list of words, return a list of words with ‘ing’ appended to it.

```
def mymap(my_function, my_list):
    result = []
    for i in my_list:
        result.append(my_function(i))
    return result

my_list = ['morn', 'walk', 'eat', 'sleep']
sub = 'ing'
l = list( mymap( lambda string : string + sub, my_list))
print(l)
```

3)Given a list of words, return a list of tuples having the word and its length.

```
def tuple_str_length(s):
    return s, len(s)

def mymap(my_function, my_list):
    result = []
    for i in my_list:
        result.append(my_function(i))
    return result

my_list = ['morning', 'walk', 'eat', 'sleep']
l = list( mymap( tuple_str_length, my_list))
print(l)
```

3

Write a function to mimic filter - called myfilter.
Test this with the following calls.
1) Given a list of strings, remove all strings having first character as digit.

Solution:
1) Given a list of strings, remove all strings having first character as digit.

```
def myfilter(my_function, my_list):
    result = []
```

```
	for i in my_list:
		if my_function(i):
			result.append(i)
	return result

l = ["hi","1gff","h3445","6sds","dfdg","234234"]
print( list( myfilter( lambda x : x if x[0] . isdigit() == False else None, l)))
```

4

Write a function to mimic reduce - called myreduce.
Test this with the following calls.
a) Given a list of numbers, find maximum in the list.
b) Given a list of integers, combine all integers to form a single integer.

Solution:

a) Given a list of numbers, find maximum in the list.

```
def myreduce(my_function, my_list):
	r = my_list[0]
	for next in my_list[1:]:
		r = my_function(r, next)
	return r

l = [23,45,12,47,54]
print( myreduce( lambda x , y : x if x > y else y, l))
```

b) Given a list of integers, combine all integers to form a single integer.

```
def myreduce(my_function, my_list):
	r = my_list[0]
	for next in my_list[1:]:
		r = my_function(r, next)
	return r

l = [1,25,32,4]
print( int( myreduce( lambda x, y : str(x) + str(y), l)))
```