In [1]:
```
# Lambda Expression
def add(x,y):
    return x+y
```

In [2]:
```
ans=add(4,9)
print(ans)
```

```
13
```

In [19]:
```
x=lambda x,y: x+y
```

In [20]:
```
print(type(x))
```

```
<class 'function'>
```

In [21]:
```
ans=x(8,9)
print("Addition is :",ans)
```

```
Addition is : 17
```

In [13]:
```
x=lambda a:a%2==0
```

In [15]:
```
x(4)
print(x)
```

```
<function <lambda> at 0x000001CD6E157550>
```

In [22]:
```
print("Addition is",x(8,7))
print("Full name is",x("Trupti","Gadkari"))
print("Total Amount is",x(78.85,65.67))
```

```
Addition is 15
Full name is TruptiGadkari
Total Amount is 144.51999999999998
```

In [26]:
```
def add(n1):
    return (lambda n2: n2+n1)    # Lambda n2:n2+5
ref=add(5)
print(ref(4))
```

```
9
```

In [36]:
```
# a^3
exp=lambda x,y :x**y
print(exp(8,3))
```

```
512
```

In [29]:
```
def findPow(b):
    return lambda p : b**p
```

In [33]:
```
ans=findPow(3)   # lambda p:3**p
print(ans(6))
```

```
729
```

In [47]:
```
l1=[4,5,-6,3,-7,8,-2]
positive=list(filter(lambda a: a>0 ,l1))
print(type(positive))
print(positive)
```

```
<class 'list'>
[4, 5, 3, 8]
```

In [57]:
```
# Take a list show all numbers divisible by 7
l1=[5,7,14,18,49,36]
print(l1)
exp=lambda x,y : x%7==0
print(exp)
```

```
<function <lambda> at 0x000001CD6E168880>
```

In [20]:
```
# Take a list of strings show all strings with minimum length 5
l1=["have","a","nice","day","sdfdre"]
l2=list(filter(lambda x:len(x)>5,l1))
print(l2)
```

```
['have', 'a', 'nice', 'day']
```

In [63]:
```
# Have tuple of string fetch all string with upper case
st=("trupti","Gadkari")
print(st)
l1=list(filter(lambda s:s[0:1].isupper(),st))
print(l1)
```

```
('trupti', 'Gadkari')
['Gadkari']
```

In [69]:
```
# Map
l1=[3,4,5,6,7,9]
print(list(map(lambda a : a*a,l1)))
```

```
[9, 16, 25, 36, 49, 81]
```

In [70]:
```
# Increase value of each element in list l1 by 2
l1=[3,4,5,6,7,8]
print(list(map(lambda a : a+2,l1)))
```

```
[5, 6, 7, 8, 9, 11]
```

In [79]:
```
l1=["fdhf","Xfh","erigd","Setrsf"]
print(list(map(lambda a : len(a),l1)))
```

```
[4, 3, 5, 7]
```

In [83]:
```
l1=["fdhf","Xfh","erigd","Setrsf"]
print(list(map(lambda a : a.isupper(),l1)))
```

```
[False, False, False, False]
```

In [80]:
```
l1=["fdhf","Xfh","erigd","Setrsf"]
print(list(map(lambda a : len(a),l1)))
l2=list(filter(lambda a:a[0:1].isupper(),l1))
print(l2)
l3=
```

```
[4, 3, 5, 7]
```

In [23]:
```
# Reduce
from functools import reduce
```

In [88]:
```
help(reduce)
```

```
Help on built-in function reduce in module _functools:

reduce(...)
    reduce(function, sequence[, initial]) -> value

    Apply a function of two arguments cumulatively to the items of a sequence,
    from left to right, so as to reduce the sequence to a single value.
    For example, reduce(lambda x, y: x+y, [1, 2, 3, 4, 5]) calculates
    ((((1+2)+3)+4)+5).  If initial is present, it is placed before the items
    of the sequence in the calculation, and serves as a default when the
    sequence is empty.
```

In [99]:
```
l1=[3,4,5,6]
total=reduce(lambda x,y: x+y ,l1)
print(total)
```

```
18
```

In [111…
```
# show sum of squares of each number in list
total=reduce(lambda x,y: x+y,list(map(lambda x:x**2,l1)))
print(total)
```

```
86
```

In [27]:
```
# Add all even numbers from given list
l1=[1,2,3,4,5,6,7,8,9]
total=reduce(lambda x,y:x+y,list(map(lambda x:x%2==0,l1)))
print(total)
```

```
4
```

In [29]:
```
# Concat all strings from given list which contains letter 'o'
string = "Concat all strings"
print("Given string :",string)
list1 = [i for i in string if i not in 'o ' ""]
print("All the consonants in the string :",list1)

Given string : Concat all strings
All the consonants in the string : ['C', 'n', 'c', 'a', 't', 'a', 'l', 'l', 's', 't', 'r', 'i', 'n', 'g', 's']
```

In [112…
```
l1
```

Out[112…
```
[3, 4, 5, 6]
```

In [113…
```
max(l1)
```

Out[113…
```
6
```

In [115…
```
# find max of l1 using reduce
max1=reduce(lambda x,y: x if x>y else y,l1)
print(max1)
```

```
6
```

In [116…
```
min1=reduce(lambda x,y: x if x<y else y,l1)
print(min1)
```

```
3
```

In [3]:
```
import operator as o
from functools import reduce
```

In [4]:
```
l1=[3,5,6,7]
print(l1)
t=reduce(o.add,l1)
print(t)
t=reduce(o.mul,l1)
print(t)
```

```
[3, 5, 6, 7]
21
630
```

In [5]:
```
import itertools as it
```

In [6]:
```
l1
list(it.accumulate(l1,lambda x,y: x+y))
```

Out[6]:
```
[3, 8, 14, 21]
```

In [10]:
```
# Assignment
```

In [9]:
```
#1 Write a python to find palindromes in a given list of strings using Lambda
list1 = ["php", "madam", "Python", "abcd", "Java", "aaa"]
print("Given list of strings :",list1)
list2 = list(filter(lambda x: (x == "" .join(reversed(x))), list1))
print("List of palindromes :",list2)

Given list of strings : ['php', 'madam', 'Python', 'abcd', 'Java', 'aaa']
List of palindromes : ['php', 'madam', 'aaa']
```

In [28]:
```
#2 Write a python program that removes the positive numbers from a given list of numbers. Sum the negative numbers and
# print the absolute value using lambda function
list1 = [2, 4, -6, -9, 11, -12, 14, -5, 17, -3, 5, -8]
print("Original list:",list1)
list2 = list(filter(lambda x: x<0, list1))
print("Remove positive numbers : ",list2)
neglist = list(filter(lambda x:x<0,list1))
print("Sum of negative numbers:",sum(neglist))

Original list: [2, 4, -6, -9, 11, -12, 14, -5, 17, -3, 5, -8]
Remove positive numbers :  [-6, -9, -12, -5, -3, -8]
Sum of negative numbers:  -43
```

In [8]:
```
#3 Write a Python program to sort a list of tuples using Lambda.
# subject marks=[('English', 88), ('Science', 90), ('Maths', 97), ('Social sciences', 82)]
# output Sorting the list of tuples :[('Social sciences', 82), ('English', 88), ('Science', 90), ('Maths', 97)]
list1 = [('English', 88), ('Science', 90), ('Maths', 97), ('Social sciences', 82)]
print("Subject Marks :",list1)
list1.sort(key = lambda x: x[1])
print("Sorting the List of Tuples:",list1)

Subject Marks : [('English', 88), ('Science', 90), ('Maths', 97), ('Social sciences', 82)]
Sorting the List of Tuples: [('Social sciences', 82), ('English', 88), ('Science', 90), ('Maths', 97)]
```

In [13]:
```
#4 Write a Python program to square and cube every alternate number using list comprehension
list1 = [4,5,6,7,8,9,3,2]
print("Original list :",list1)
slist = list(map(lambda x: x ** 2, list1))
print("Square of every number :",slist)
clist = list(map(lambda x: x ** 3, list1))
print("Cube of every number :",clist)

Original list : [4, 5, 6, 7, 8, 9, 3, 2]
Square of every number : [16, 25, 36, 49, 64, 81, 9, 4]
Cube of every number : [64, 125, 216, 343, 512, 729, 27, 8]
```

In [19]:
```
#5 Write a program to find all prime number from given list using filter function
list1=[3,4,5,67,8,9,45,65,17,47,87]
nlist = list(filter(lambda x:all(x % y !=0 for y in range(2,x)),list1))
print(nlist)

[3, 5, 67, 17, 47]
```

In [21]:
```
#6 Write a python logic to multiply all the numbers in a list (use reduce function)
from functools import reduce
list1 = [1,3,4,5,6]
nlist = reduce(lambda x, y: x * y), list1)
print("Multiply all the numbers :",nlist)

Multiply all the numbers : 360
```

In [3]:
```
#7 WAP using dict comprehension to create dict from list of voters
# l1={{'pan_id':'ABPPS4344P','vote_casted':'BJP'},
#    {'pan_id':'ABPPS4342P','vote_casted':'Congress'},
#    {'pan_id':'ABPPS4378P','vote_casted':'BJP'}}
# Output : {'BJP':2,'Congress':1}
l1={{'pan_id':'ABPPS4344P','vote_casted':'BJP'},
    {'pan_id':'ABPPS4342P','vote_casted':'Congress'},
    {'pan_id':'ABPPS4378P','vote_casted':'BJP'}}
l2=list(do['vote_casted'] for do in l1)
print(l2)
d1={k:l2.count(k) for k in l2}
print(d1)

['BJP', 'Congress', 'BJP']
{'BJP': 2, 'Congress': 1}
```

In [3]:
```
#8 Write a program to find all armstrong number from given list of using filter function
def armstrong(a):
    sum=0
    t=a
    while t>0:
        d=t%10
        sum=sum+d**3
        t//=10
    if(a==sum):
        print(sum)
    else:
        pass

l1=[345,789,370,371,456,865]
l2=[]
l2=list(filter(armstrong,l1))
```

```
370
371
```

In [ ]:
```

```