Python Exercises- Lambda

<https://www.w3resource.com/python-exercises/lambda/index.php>

'''

1. Write a Python program to create a lambda function that adds 15

to a given number passed in as an argument,

also create a lambda function that multiplies argument x with

argument y and prints the result.

Sample Output:

25

48

'''

l1=lambda x: x+15

l2=lambda x,y: x\*y

print(l1(10))

print(l2(6,8))

'''

2. Write a Python program to create a function that takes one argument,

and that argument will be multiplied with an unknown given number.

Sample Output:

Double the number of 15 = 30

Triple the number of 15 = 45

Quadruple the number of 15 = 60

Quintuple the number 15 = 75

'''

l1=lambda x,n: x\*n

print(l1(15,2))

print((l1(15,3)))

print(l1(15,4))

#Textbook

def func\_compute(n):

return lambda x : x \* n

result = func\_compute(2)

print("Double the number of 15 =", result(15))

result = func\_compute(3)

print("Triple the number of 15 =", result(15))

result = func\_compute(4)

print("Quadruple the number of 15 =", result(15))

result = func\_compute(5)

print("Quintuple the number 15 =", result(15))

'''

3.Write a Python program to sort a list of tuples using Lambda.

Original list of tuples:

[('English', 88), ('Science', 90), ('Maths', 97), ('Social sciences', 82)]

Sorting the List of Tuples:

[('Social sciences', 82), ('English', 88), ('Science', 90), ('Maths', 97)]

'''

lst1=[('English', 88), ('Science', 90), ('Maths', 97), ('Social sciences', 82)]

lst1.sort(key=lambda i: i[1])

print(lst1)

'''

4. Write a Python program to sort a list of dictionaries using Lambda.

Original list of dictionaries :

[{'make': 'Nokia', 'model': 216, 'color': 'Black'}, {'make': 'Mi Max', 'model': '2', 'color': 'Gold'}, {'make': 'Samsung', 'model': 7, 'color': 'Blue'}]

Sorting the List of dictionaries :

[{'make': 'Nokia', 'model': 216, 'color': 'Black'}, {'make': 'Samsung', 'model': 7, 'color': 'Blue'}, {'make': 'Mi Max', 'model': '2', 'color': 'Gold'}]

'''

lst1=[{'make': 'Nokia', 'model': 216, 'color': 'Black'}, {'make': 'Mi Max', 'model': '2', 'color': 'Gold'}, {'make': 'Samsung', 'model': 7, 'color': 'Blue'}]

lst1.sort(key=lambda d:d['color'])

print(lst1)

'''

5. Write a Python program to filter a list of integers using Lambda.

Original list of integers:

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

Even numbers from the said list:

[2, 4, 6, 8, 10]

Odd numbers from the said list:

[1, 3, 5, 7, 9]

'''

lst1=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print(list(filter(lambda x:not x%2,lst1)))

print(list(filter(lambda x:x%2,lst1)))

'''

6. Write a Python program to square and cube every number in

a given list of integers using Lambda.

Original list of integers:

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

Square every number of the said list:

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

Cube every number of the said list:

[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]

'''

lst1=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

sq=list(map(lambda x:x\*\*2,lst1))

print(sq)

cu=list(map(lambda x:x\*\*3,lst1))

print(cu)

'''

7. Write a Python program to find if a given string starts with

a given character using Lambda.

Sample Output:

True

False

'''

strs=input('input a string:\n')

char=input('input the char to check if it starts with:\n')

print(lambda strs:strs.startswith(char))

#just output as an object

print((lambda strs:strs.startswith(char))(strs)) #True

#even lambda needs argument, too !

#Textbook

starts\_with = lambda x: True if x.startswith('P') else False

print(starts\_with('Python'))

starts\_with = lambda x: True if x.startswith('P') else False

print(starts\_with('Java'))

'''

8. Write a Python program to extract year, month, date and time using Lambda.

Sample Output:

2020-01-15 09:03:32.744178

2020

1

15

09:03:32.744178

'''

#warmup

from datetime import datetime

str1='2020-01-15 09:03:32.744178'

time\_obj=datetime.strptime(str1,'%Y-%m-%d %H:%M:%S.%f')

#strptime is to convert the string into datetime object

# .%f is to handle the microseconds...

print(time\_obj)

print(time\_obj.second)

formatted\_time=datetime.strftime(time\_obj,'%H:%M:%S.%f')

print(formatted\_time)

print()

def extract\_dt(strs,attr):

time\_obj1=datetime.strptime(strs,'%Y-%m-%d %H:%M:%S.%f')

formatted\_time=datetime.strftime(time\_obj1,'%H:%M:%S.%f')

day = time\_obj1.strftime('%d')

func\_y=lambda time\_obj1:time\_obj1.year

func\_m=lambda time\_obj1:time\_obj1.month

func\_d=lambda time\_obj1:time\_obj1.date #it's 'day', not 'date'

if attr=='y':

return func\_y(time\_obj1)

elif attr=='m':

return func\_m(time\_obj1)

elif attr=='d':

return day

elif attr=='t':

return formatted\_time

else: pass

print(extract\_dt(str1,'y'))

print(extract\_dt(str1,'m'))

print(extract\_dt(str1,'d'))

print(extract\_dt(str1,'t'))

print()

#Textbook

import datetime

now = datetime.datetime.now()

print(now)

year = lambda x: x.year

month = lambda x: x.month

day = lambda x: x.day

t = lambda x: x.time()

print(year(now))

print(month(now))

print(day(now))

print(t(now))

'''

9. Write a Python program to check whether a given string is a number

or not using Lambda.

Sample Output:

True

True

False

True

False

True

Print checking numbers:

True

True

'''

def is\_number(strs):

chk=lambda x: x.isnumeric()

return chk(strs)

print(is\_number('6')) #True

print(is\_number('134.23'))

#False 因為小數點不被認定為numeric,digit; 除非用re

print(is\_number('hello'))

#textbook

is\_num = lambda q: q.replace('.','',1).isdigit()

#replace '.' with '' and once

print(is\_num('26587'))

print(is\_num('4.2365')) #False

#even though replace '.' with '',all becomes a string! so it's false

print(is\_num('-12547'))

print(is\_num('00'))

print(is\_num('A001'))

print(is\_num('001'))

print("\nPrint checking numbers:")

is\_num1 = lambda r: is\_num(r[1:]) if r[0]=='-' else is\_num(r)

print(is\_num1('-16.4'))

print(is\_num1('-24587.11'))

'''

10. Write a Python program to create Fibonacci series up to n

using Lambda.

Fibonacci series upto 2:

[0, 1]

Fibonacci series upto 5:

[0, 1, 1, 2, 3]

Fibonacci series upto 6:

[0, 1, 1, 2, 3, 5]

Fibonacci series upto 9:

[0, 1, 1, 2, 3, 5, 8, 13, 21]

'''

def fib(n):

lst=[0,1]

func=lambda lst: lst.append(lst[-1]+lst[-2])

if n<2: return 'n need to be gt 2!'

else:

for i in range(2,n):

func(lst)

return lst

print(fib(2))

print(fib(5))

print(fib(6))

#Textbook

from functools import reduce

fib\_series = lambda n: reduce(lambda x, \_: x+[x[-1]+x[-2]],

range(n-2), [0, 1])

print("Fibonacci series upto 2:")

print(fib\_series(2))

print("\nFibonacci series upto 5:")

print(fib\_series(5))

print("\nFibonacci series upto 6:")

print(fib\_series(6))

print("\nFibonacci series upto 9:")

print(fib\_series(9))

print()

#syntax of reduce : functools.reduce(function, iterable[, initializer])

#warmup

#1. Summing up all the elements of a list:

lst1=[1,2,3,4,5,6]

res=reduce(lambda x,y:x+y,lst1) # reduce to single value

print(res)

#2.Finding the maximum element in a list:

res=reduce(lambda x,y :x if x>y else y, lst1)

print(res)

'''

11. Write a Python program to find the intersection of two given arrays

using Lambda.

Original arrays:

[1, 2, 3, 5, 7, 8, 9, 10]

[1, 2, 4, 8, 9]

Intersection of the said arrays: [1, 2, 8, 9]

'''

def inters(lst1,lst2):

func=lambda x,y: set(x).intersection(set(y))

return list(func(lst1,lst2))

print(inters([1, 2, 3, 5, 7, 8, 9, 10],[1,2,4,8,9]))

lst3=[3,6,8,1,14,7,2]

lst4=[5,8,12,14,7,9,1,2]

print(inters(lst3,lst4))

#Textbook

array\_nums1 = [1, 2, 3, 5, 7, 8, 9, 10]

array\_nums2 = [1, 2, 4, 8, 9]

print("Original arrays:")

print(array\_nums1)

print(array\_nums2)

result = list(filter(lambda x: x in array\_nums1, array\_nums2))

#filter的對象是 array\_nums2

print ("\nIntersection of the said arrays: ",result)

'''

12. Write a Python program to rearrange positive and negative numbers

in a given array using Lambda.

Original arrays:

[-1, 2, -3, 5, 7, 8, 9, -10]

Rearrange positive and negative numbers of the said array:

[2, 5, 7, 8, 9, -10, -3, -1]

'''

lst1=[-1, 2, -3, 5, 7, 8, 9, -10]

res=lambda \_: list(sorted(\_,reverse=True))

print(res(lst1))

#Textbook

array\_nums = [-1, 2, -3, 5, 7, 8, 9, -10]

print("Original arrays:")

print(array\_nums)

result = sorted(array\_nums, key = lambda i: 0 if i == 0 else -1 / i)

print("\nRearrange positive and negative numbers of the said array:")

print(result)

'''

13. Write a Python program to count the even and odd numbers in

a given array of integers using Lambda.

Original arrays:

[1, 2, 3, 5, 7, 8, 9, 10]

Number of even numbers in the above array: 3

Number of odd numbers in the above array: 5

'''

ce=lambda lst:len([i for i in lst if not i%2])

co=lambda lst:len([i for i in lst if i%2])

lst1=[1, 2, 3, 5, 7, 8, 9, 10]

print(ce(lst1),co(lst1))

#Textbook

array\_nums = [1, 2, 3, 5, 7, 8, 9, 10]

print("Original arrays:")

print(array\_nums)

odd\_ctr = len(list(filter(lambda x: (x%2 != 0) , array\_nums)))

even\_ctr = len(list(filter(lambda x: (x%2 == 0) , array\_nums)))

print("\nNumber of even numbers in the above array: ", even\_ctr)

print("\nNumber of odd numbers in the above array: ", odd\_ctr)

'''

14. Write a Python program to filter a given list to determine

if the values in the list have a length of 6 using Lambda.

Sample Output:

Monday

Friday

Sunday

'''

lst1=['Sunday','Monday','Tuesday','Wednesday','Thursday','Friday','Saturday']

print(list(filter(lambda x:len(x)==6, lst1)))

#Textbook

weekdays = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

days = filter(lambda day: day if len(day)==6 else '', weekdays)

for d in days:

print(d)

'''

15. Write a Python program to add two given lists using map and lambda.

Original list:

[1, 2, 3]

[4, 5, 6]

Result: after adding two list

[5, 7, 9]

'''

lst1=[1,2,3]

lst2=[4,5,6]

#warmup

print(\*map(lambda x:x+1, lst1))

zl=list(zip(lst1,lst2))

print(list(map(lambda x:x[0]+x[1],zl)))

print()

#Textbook

nums1 = [1, 2, 3]

nums2 = [4, 5, 6]

print("Original list:")

print(nums1)

print(nums2)

result = map(lambda x, y: x + y, nums1, nums2)

#if lambda has two independent variables, then map後面也有 2 iterables

print("\nResult: after adding two list")

print(list(result))

'''

16. Write a Python program to find the second lowest total marks

of any student(s) from the given names and marks of each student

using lists and lambda.

Input the number of students, the names and grades of each student.

Input number of students: 5

Name: S ROY

Grade: 1

Name: B BOSE

Grade: 3

Name: N KAR

Grade: 2

Name: C DUTTA

Grade: 1

Name: G GHOSH

Grade: 1

Names and Grades of all students:

[['S ROY', 1.0], ['B BOSE', 3.0], ['N KAR', 2.0], ['C DUTTA', 1.0], ['G GHOSH', 1.0]]

Second lowest grade: 2.0

Names:

N KAR

'''

stu=[['S ROY', 1.0], ['B BOSE', 3.0], ['N KAR', 2.0], ['C DUTTA', 1.0], ['G GHOSH', 1.0]]

stu.sort(key=lambda x:x[1])

print(stu)

for i in range(1,len(stu)):

lowest=stu[0][1]

if stu[i][1]>lowest:

print(stu[i][1],stu[i][0])

break

#my code will only output the first 'second-lowest' if there's more than 1 persons with 'second-lowest'

#Textbook

students = []

sec\_name = []

second\_low = 0

n = int(input("Input number of students: "))

for \_ in range(n):

s\_name = input("Name: ")

score = float(input("Grade: "))

students.append([s\_name,score])

print("\nNames and Grades of all students:")

print(students)

order =sorted(students, key = lambda x: int(x[1]))

for i in range(n):

if order[i][1] != order[0][1]:

second\_low = order[i][1]

break

print("\nSecond lowest grade: ",second\_low)

sec\_student\_name = [x[0] for x in order if x[1] == second\_low]

sec\_student\_name.sort()

print("\nNames:")

for s\_name in sec\_student\_name:

print(s\_name)

'''

17. Write a Python program to find numbers divisible by nineteen

or thirteen from a list of numbers using Lambda.

Orginal list:

[19, 65, 57, 39, 152, 639, 121, 44, 90, 190]

Numbers of the above list divisible by nineteen or thirteen:

[19, 65, 57, 39, 152, 190]

'''

lst1=[19, 65, 57, 39, 152, 639, 121, 44, 90, 190]

res=list(filter(lambda x: x%19==0 or x%13==0,lst1))

print(res)

'''

18. Write a Python program to find palindromes in a given list

of strings using Lambda.

Orginal list of strings:

['php', 'w3r', 'Python', 'abcd', 'Java', 'aaa']

List of palindromes:

['php', 'aaa']

'''

#warmup

str1='xia'

print(str1[::-1])

def fp(lsts):

revw=lambda w: w[::-1]

return list(word for word in lsts if revw(word)==word)

print(fp(['php', 'w3r', 'Python', 'abcd', 'Java', 'aaa']))

#Textbook

texts = ["php", "w3r", "Python", "abcd", "Java", "aaa"]

print("Orginal list of strings:")

print(texts)

result = list(filter(lambda x: (x == "".join(reversed(x))), texts))

print("\nList of palindromes:")

print(result)

'''

19. Write a Python program to find all anagrams of a string

in a given list of strings using Lambda.

Orginal list of strings:

['bcda', 'abce', 'cbda', 'cbea', 'adcb']

Anagrams of 'abcd' in the above string:

['bcda', 'cbda', 'adcb']

'''

def fa(lsts,kw):

chk=lambda x: sorted(x)==sorted(kw)

return list(filter(chk,lsts))

print(fa(['bcda', 'abce', 'cbda', 'cbea', 'adcb'],'abcd'))

#Textbook

from collections import Counter

texts = ["bcda", "abce", "cbda", "cbea", "adcb"]

str = "abcd"

print("Orginal list of strings:")

print(texts)

result = list(filter(lambda x: (Counter(str) == Counter(x)), texts))

print("\nAnagrams of 'abcd' in the above string: ")

print(result)

'''

20. Write a Python program to find the numbers in a given string

and store them in a list. Afterward, display the numbers

that are longer than the length of the list in sorted form.

Use the lambda function to solve the problem.

Original string: sdf 23 safs8 5 sdfsd8 sdfs 56 21sfs 20 5

Numbers in sorted form:

20 23 56

'''

def ltlol(strs):

res=sorted([i for i in strs.split(sep=' ') if i.isdecimal() and int(i)>len(strs.split(sep=' '))])

print([int(j) for j in res])

ltlol('sdf 23 safs8 5 sdfsd8 sdfs 56 21sfs 20 5')

#without using lambda

#Textbook

str1 = "sdf 23 safs8 5 sdfsd8 sdfs 56 21sfs 20 5"

print("Original string: ",str1)

str\_num=[i for i in str1.split(' ')]

lenght=len(str\_num)

numbers=sorted([int(x) for x in str\_num if x.isdigit()])

print('Numbers in sorted form:')

for i in ((filter(lambda x:x>lenght,numbers))):

print(i,end=' ')

'''

21. Write a Python program that multiplies each number in a list

with a given number using lambda functions. Print the results.

Original list: [2, 4, 6, 9, 11]

Given number: 2

Result:

4 8 12 18 22

'''

def mwn(lsts,n):

print(\*map(lambda x: x\*n, lsts))

mwn([2, 4, 6, 9, 11],2)

'''

22. Write a Python program that sums the length of a list of names

after removing those that start with lowercase letters.

Use the lambda function.

Result:

16

'''

def slon(lsts):

print(sum(list(map(lambda x: len(x) if x[0].isupper() else 0,lsts))))

slon(['Zoe','Jackson','jim','Mary','Ed','kim'])

#Textbook

sample\_names = ['sally', 'Dylan', 'rebecca', 'Diana', 'Joanne', 'keith']

sample\_names=list(filter(lambda el:el[0].isupper() and el[1:].islower(),sample\_names))

print("Result:")

print(len(''.join(sample\_names)))

'''

23. Write a Python program to calculate the sum of the positive and

negative numbers of a given list of numbers using the lambda function.

Original list: [2, 4, -6, -9, 11, -12, 14, -5, 17]

Sum of the negative numbers: -32

Sum of the positive numbers: 48

'''

def sopan(lsts):

print(sum(filter(lambda x:x<0, lsts)))

print(sum(filter(lambda x:x>0, lsts)))

sopan([2, 4, -6, -9, 11, -12, 14, -5, 17])

'''

24. Write a Python program to find numbers within a given range

where every number is divisible by every digit it contains.

Sample Output:

[1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 15, 22]

'''

def sep\_digit(num):

return list(filter(lambda x: '0' not in str(num), str(num)))

def dbed(range\_n):

res=[]

for i in range(1,range\_n+1):

chk=[]

for j in sep\_digit(i):

chk.append(i%int(j))

if not any(chk):

res.append(i)

return res

print(dbed(23))

#Textbook

def divisible\_by\_digits(start\_num, end\_num):

return [n for n in range(start\_num, end\_num+1) \

if not any(map(lambda x: int(x) == 0 or n%int(x) != 0, str(n)))]

print(divisible\_by\_digits(1,22))

'''

25. Write a Python program to create the next bigger number

by rearranging the digits of a given number.

Original number: 12

Next bigger number: 21

Original number: 10

Next bigger number: False

Original number: 201

Next bigger number: 210

Original number: 102

Next bigger number: 120

Original number: 445

Next bigger number: 454

'''

#warmup

from itertools import permutations as pm

str1='445'

print([''.join(i) for i in pm(str1,len(str1))])

def nbn(num):

chk=list(map(lambda x: int(''.join(x)), list(pm(str(num),len(str(num))))))

for i in chk:

if i>num: return i

return False

print(nbn(12))

print(nbn(10))

print(nbn(201))

print(nbn(102))

print(nbn(445))

#Textbook

def rearrange\_bigger(n):

#Break the number into digits and store in a list

nums = list(str(n))

for i in range(len(nums)-2,-1,-1):

if nums[i] < nums[i+1]:

z = nums[i:]

y = min(filter(lambda x: x > z[0], z))

z.remove(y)

z.sort()

nums[i:] = [y] + z

return int("".join(nums))

return False

n = 12

print("Original number:",n)

print("Next bigger number:",rearrange\_bigger(n))

n = 10

print("\nOriginal number:",n)

print("Next bigger number:",rearrange\_bigger(n))

n = 201

print("\nOriginal number:",n)

print("Next bigger number:",rearrange\_bigger(n))

n = 102

print("\nOriginal number:",n)

print("Next bigger number:",rearrange\_bigger(n))

n = 445

print("\nOriginal number:",n)

print("Next bigger number:",rearrange\_bigger(n))

'''

26. Write a Python program to find a list with

maximum and minimum length using lambda.

Original list:

[[0], [1, 3], [5, 7], [9, 11], [13, 15, 17]]

List with maximum length of lists:

(3, [13, 15, 17])

List with minimum length of lists:

(1, [0])

'''

#warmup

lst1=[[0], [1, 3], [5, 7], [9, 11], [13, 15, 17]]

lst1.sort(key=lambda x:len(x),reverse=True)

print(lst1)

def maml(lsts):

lsts.sort(key=lambda x: len(x), reverse=True)

print('max lgth of lists:')

print(len(lsts[0]),lsts[0])

print('min lgth of lists:')

print(len(lsts[-1]),lsts[-1])

maml([[0], [1, 3], [5, 7], [9, 11], [13, 15, 17]])

#Textbook

def max\_length\_list(input\_list):

max\_length = max(len(x) for x in input\_list )

max\_list = max(input\_list, key = lambda i: len(i))

return(max\_length, max\_list)

def min\_length\_list(input\_list):

min\_length = min(len(x) for x in input\_list )

min\_list = min(input\_list, key = lambda i: len(i))

return(min\_length, min\_list)

list1 = [[0], [1, 3], [5, 7], [9, 11], [13, 15, 17]]

print("Original list:")

print(list1)

print("\nList with maximum length of lists:")

print(max\_length\_list(list1))

print("\nList with minimum length of lists:")

print(min\_length\_list(list1))

'''

27. Write a Python program to sort each sublist of strings

in a given list of lists using lambda.

Original list:

[['green', 'orange'], ['black', 'white'], ['white', 'black', 'orange']]

After sorting each sublist of the said list of lists:

[['green', 'orange'], ['black', 'white'], ['black', 'orange', 'white']]

'''

#chatGPT

lst1=[['green', 'orange'], ['black', 'white'], ['white', 'black', 'orange']]

lst1.sort(key=lambda x: x[1])

print(lst1)

# but the output is [['white', 'black', 'orange'], ['green', 'orange'], ['black', 'white']]

#Textbook

def sort\_sublists(input\_list):

result = [sorted(x, key = lambda x:x[0]) for x in input\_list]

return result

color1 = [["green", "orange"], ["black", "white"], ["white", "black", "orange"]]

print("\nOriginal list:")

print(color1)

print("\nAfter sorting each sublist of the said list of lists:")

print(sort\_sublists(color1))

'''

28. Write a Python program to sort a given list of lists

by length and value using lambda.

Original list:

[[2], [0], [1, 3], [0, 7], [9, 11], [13, 15, 17]]

Sort the list of lists by length and value:

[[0], [2], [0, 7], [1, 3], [9, 11], [13, 15, 17]]

'''

lst1=[[2], [0], [1, 3], [13, 15, 17], [0, 7], [9, 11] ]

lst1.sort(key=lambda x:len(x))

max\_len=len(lst1[-1])

res=[]

for i in range(1,max\_len+1):

ele=list(j for j in lst1 if len(j)==i)

ele.sort(key=lambda x:x[0])

res.append(ele)

print(res)

#Textbook

def sort\_sublists(input\_list):

result = sorted(input\_list, key=lambda l: (len(l), l))

return result

list1 = [[2], [0], [1, 3], [0, 7], [9, 11], [13, 15, 17]]

print("Original list:")

print(list1)

print("\nSort the list of lists by length and value:")

print(sort\_sublists(list1))

#(len(l),l): for sorting length, then lexicographically

#warmup

lst2=['hello','what','are','you','doing','today']

lst2.sort(key=lambda x:(len(x),x[1]))

print(lst2)

#using tuple in lambda can sort by multiple factors

lst = [('Alice', 25, 80), ('Bob', 30, 90), ('Charlie', 20, 70), ('David', 25, 85)]

result = sorted(lst, key=lambda x: (-x[1], x[2], x[0]))

#sort by age(descending), score and name

print(result)

'''

29. Write a Python program to find the maximum value in a

given heterogeneous list using lambda.

Original list:

['Python', 3, 2, 4, 5, 'version']

Maximum values in the said list using lambda:

5

'''

def mvihl(hl):

return max(hl,key=lambda x: str(x).isdigit() and int(x) or 0)

#the lambda function first checks if the element is a digit by converting it to a string and calling the isdigit() method. If the element is a digit, it is converted to an integer using the int() function. If the element is not a digit, it is converted to 0

print(mvihl(['Python', 3, 2, 4, 5, 'version']))

#warmup- select all words start with 'a' and grab the longest one

def a\_longest(lsts):

return max(lsts, key=lambda x: str(x).startswith('a') and len(x) or 0)

lst1=['tim','sucks','abc','alphabet','Alternative',345,29.8]

print(a\_longest(lst1)) #alphabet

#Textbook

def max\_val(list\_val):

max\_val = max(list\_val, key = lambda i: (isinstance(i, int), i))

return(max\_val)

list\_val = ['Python', 3, 2, 4, 5, 'version']

print("Original list:")

print(list\_val)

print("\nMaximum values in the said list using lambda:")

print(max\_val(list\_val))

'''

30. Write a Python program to sort a given matrix in ascending order

according to the sum of its rows using lambda.

Original Matrix:

[[1, 2, 3], [2, 4, 5], [1, 1, 1]]

Sort the said matrix in ascending order according to the sum of its rows

[[1, 1, 1], [1, 2, 3], [2, 4, 5]]

Original Matrix:

[[1, 2, 3], [-2, 4, -5], [1, -1, 1]]

Sort the said matrix in ascending order according to the sum of its rows

[[-2, 4, -5], [1, -1, 1], [1, 2, 3]]

'''

def sbs(lsts):

lsts.sort(key=lambda x:sum(x))

return lsts

print(sbs([[1, 2, 3], [2, 4, 5], [1, 1, 1]]))

print(sbs([[1, 2, 3], [-2, 4, -5], [1, -1, 1]]))

'''

31. Write a Python program to extract a specified size of strings

from a given list of string values using lambda.

Original list:

['Python', 'list', 'exercises', 'practice', 'solution']

length of the string to extract:

8

After extracting strings of specified length from the said list:

['practice', 'solution']

'''

lst1=['Python', 'list', 'exercises', 'practice', 'solution']

print(list(filter(lambda x:len(x)==8, lst1)))

#Textbook

def extract\_string(str\_list1, l):

result = list(filter(lambda e: len(e) == l, str\_list1))

return result

'''

32. Write a Python program to count float values in a mixed list

using lambda.

Original list:

[1, 'abcd', 3.12, 1.2, 4, 'xyz', 5, 'pqr', 7, -5, -12.22]

Number of floats in the said mixed list:

3

'''

def cfn(lsts):

return len(list(filter(lambda x: isinstance(x,float),lsts)))

print(cfn([1, 'abcd', 3.12, 1.2, 4, 'xyz', 5, 'pqr', 7, -5, -12.22]))

#Textbook

def count\_integer(list1):

ert = list(map(lambda i: isinstance(i, float), list1))

result = len([e for e in ert if e])

return result

'''

33. Write a Python program to check whether a given string contains

a capital letter, a lower case letter, a number and

a minimum length using lambda.

Input the string: W3resource

['Valid string.']

'''

print('Input the string:')

str1=input()

cap\_set=set('ABCDEFGHIJKLMNOPQRSTUVWXYZ')

low\_set=set('abcdefghijklmnopqrstuvwxyz')

num\_set=set('1234567890')

sp\_set=set(' ')

if all((len(set(str1).intersection(cap\_set)),

len(set(str1).intersection(low\_set)),

len(set(str1).intersection(num\_set)),

not len(set(str1).intersection(sp\_set)))):

print('Valid string')

else: print('Invalid string')

#Textbook

def check\_string(str1):

messg = [

lambda str1: any(x.isupper() for x in str1) or 'String must have 1 upper case character.',

lambda str1: any(x.islower() for x in str1) or 'String must have 1 lower case character.',

lambda str1: any(x.isdigit() for x in str1) or 'String must have 1 number.',

lambda str1: len(str1) >= 7 or 'String length should be atleast 8.',]

result = [x for x in [i(str1) for i in messg] if x != True]

if not result:

result.append('Valid string.')

return result

s = input("Input the string: ")

print(check\_string(s))

'''

34. Write a Python program to filter the height and weight of students,

which are stored in a dictionary using lambda.

Original Dictionary:

{'Cierra Vega': (6.2, 70), 'Alden Cantrell': (5.9, 65), 'Kierra Gentry': (6.0, 68), 'Pierre Cox': (5.8, 66)}

Height> 6ft and Weight> 70kg:

{'Cierra Vega': (6.2, 70)}

'''

dic1 = {'Cierra Vega': (6.2, 70), 'Alden Cantrell': (5.9, 65), 'Kierra Gentry': (6.0, 68), 'Pierre Cox': (5.8, 66)}

result = list(filter(lambda x: x[1][0] > 6 and x[1][1] > 69, dic1.items()))

print(result)

'''

35. Write a Python program to check whether a specified list

is sorted or not using lambda.

Original list:

[1, 2, 4, 6, 8, 10, 12, 14, 16, 17]

Is the said list is sorted!

True

Original list:

[1, 2, 4, 6, 8, 10, 12, 14, 16, 17]

Is the said list is sorted!

False

'''

#warmup

lst1=[1,2,3,4,5]

lst2=[3,2,4,1,5]

print(lst1==lst2) #False

print(sorted(lst2)==lst1) #True

lst3=[1, 2, 4, 6, 8, 10, 12, 14, 16, 17]

lst4=[1, 2, 4, 6, 8, 12, 10, 14, 16, 17]

chk=lambda x:sorted(x)==x

print(chk(lst3))

print(chk(lst4))

#Textbook

def is\_sort\_list(nums, key=lambda x: x):

for i, e in enumerate(nums[1:]):

if key(e) < key(nums[i]):

return False

return True

nums1 = [1,2,4,6,8,10,12,14,16,17]

print ("Original list:")

print(nums1)

print("\nIs the said list is sorted!")

print(is\_sort\_list(nums1))

nums2 = [2,3,8,4,7,9,8,2,6,5,1,6,1,2,3,4,6,9,1,2]

print ("\nOriginal list:")

print(nums1)

print("\nIs the said list is sorted!")

print(is\_sort\_list(nums2))

'''

36. Write a Python program to extract the nth element from a given list

of tuples using lambda.

Original list:

[('Greyson Fulton', 98, 99), ('Brady Kent', 97, 96), ('Wyatt Knott', 91, 94), ('Beau Turnbull', 94, 98)]

Extract nth element ( n = 0 ) from the said list of tuples:

['Greyson Fulton', 'Brady Kent', 'Wyatt Knott', 'Beau Turnbull']

Extract nth element ( n = 2 ) from the said list of tuples:

[99, 96, 94, 98]

'''

lst1=[('Greyson Fulton', 98, 99), ('Brady Kent', 97, 96), ('Wyatt Knott', 91, 94), ('Beau Turnbull', 94, 98)]

def ene(lsts,n):

return list(map(lambda x:x[n], lst1))

print(ene(lst1,0))

print(ene(lst1,2))

'''

37. Write a Python program to sort a list of lists by

a given index of the inner list using lambda.

Original list:

[('Greyson Fulton', 98, 99), ('Brady Kent', 97, 96), ('Wyatt Knott', 91, 94), ('Beau Turnbull', 94, 98)]

Sort the said list of lists by a given index ( Index = 0 ) of the inner list

[('Beau Turnbull', 94, 98), ('Brady Kent', 97, 96), ('Greyson Fulton', 98, 99), ('Wyatt Knott', 91, 94)]

Sort the said list of lists by a given index ( Index = 2 ) of the inner list

[('Wyatt Knott', 91, 94), ('Brady Kent', 97, 96), ('Beau Turnbull', 94, 98), ('Greyson Fulton', 98, 99)]

'''

def sbi(lsts,idx=0):

return sorted(lsts,key=lambda x:x[idx])

lst1=[('Greyson Fulton', 98, 99), ('Brady Kent', 97, 96), ('Wyatt Knott', 91, 94), ('Beau Turnbull', 94, 98)]

print(sbi(lst1))

print(sbi(lst1,2))

'''

38. Write a Python program to remove all elements from a given list

present in another list using lambda.

Original lists:

list1: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

list2: [2, 4, 6, 8]

Remove all elements from 'list1' present in 'list2:

[1, 3, 5, 7, 9, 10]

'''

#warmup

lst1=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

lst2=[2, 4, 6, 8]

lst1f=list(filter(lambda x:x>3, lst1))

print(lst1f) # [4, 5, 6, 7, 8, 9, 10]

def repial(lsts1,lsts2):

return list(filter(lambda x:x not in lsts2, lsts1))

print(repial(lst1,lst2))

'''

39. Write a Python program to find the elements of a given list

of strings that contain a specific substring using lambda.

Original list:

['red', 'black', 'white', 'green', 'orange']

Substring to search:

ack

Elements of the said list that contain specific substring:

['black']

Substring to search:

abc

Elements of the said list that contain specific substring:

[]

'''

#warmup

sub1='ack'

print(sub1 in 'black') #True

def css(strs,subs):

return list(filter(lambda x: subs in x ,strs))

print(css(['red', 'black', 'white', 'green', 'orange'],'ack'))

'''

40. Write a Python program to find the nested list elements,

which are present in another list using lambda.

Original lists: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]

[[12, 18, 23, 25, 45], [7, 11, 19, 24, 28], [1, 5, 8, 18, 15, 16]]

Intersection of said nested lists:

[[12], [7, 11], [1, 5, 8]]

'''

#chatGPT

original\_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]

nested\_list = [[12, 18, 23, 25, 45], [7, 11, 19, 24, 28], [1, 5, 8, 18, 15, 16]]

intersection = list(map(lambda x: list(filter(lambda y: y in original\_list, x)), nested\_list))

print("Original lists:", original\_list, nested\_list)

print("Intersection of said nested lists:", intersection)

#Textbook

def intersection\_nested\_lists(l1, l2):

result = [list(filter(lambda x: x in l1, sublist)) for sublist in l2]

return result

nums1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]

nums2 = [[12, 18, 23, 25, 45], [7, 11, 19, 24, 28], [1, 5, 8, 18, 15, 16]]

print("\nOriginal lists:")

print(nums1)

print(nums2)

print("\nIntersection of said nested lists:")

print(intersection\_nested\_lists(nums1, nums2))

'''41. Write a Python program to reverse strings in a given list

of string values using lambda.

Original lists:

['Red', 'Green', 'Blue', 'White', 'Black']

Reverse strings of the said given list:

['deR', 'neerG', 'eulB', 'etihW', 'kcalB']

'''

def rs(lsts):

return list(map(lambda x:''.join(x[-1::-1]), lsts))

print(rs(['Red', 'Green', 'Blue', 'White', 'Black']))

#Textbook

def reverse\_strings\_list(string\_list):

result = list(map(lambda x: "".join(reversed(x)), string\_list))

return result

#warmup

str1='helloworld'

print(reversed(str1)) #an object

print(''.join(reversed(str1)))

'''

42. Write a Python program to calculate the product of a given list

of numbers using lambda.

list1: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Product of the said list numbers:

3628800

list2: [2.2, 4.12, 6.6, 8.1, 8.3]

Product of the said list numbers:

4021.8599520000007

'''

from functools import reduce

#warmup

lst1=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print(reduce(lambda x,y:x\*y, lst1))

def pol(lsts):

return reduce(lambda x,y:x\*y, lsts)

print(pol([2.2, 4.12, 6.6, 8.1, 8.3]))

'''

43. Write a Python program to multiply all the numbers in a given list

using lambda.

Original list:

[4, 3, 2, 2, -1, 18]

Mmultiply all the numbers of the said list: -864

Original list:

[2, 4, 8, 8, 3, 2, 9]

Mmultiply all the numbers of the said list: 27648

'''

from functools import reduce

def mil(lsts):

return reduce(lambda x,y:x\*y,lsts)

print(mil([4, 3, 2, 2, -1, 18]))

print(mil([2, 4, 8, 8, 3, 2, 9]))

'''

44. Write a Python program to calculate the average value of the numbers

in a given tuple of tuples using lambda.

Original Tuple:

((10, 10, 10), (30, 45, 56), (81, 80, 39), (1, 2, 3))

Average value of the numbers of the said tuple of tuples:

(30.5, 34.25, 27.0)

Original Tuple:

((1, 1, -5), (30, -15, 56), (81, -60, -39), (-10, 2, 3))

Average value of the numbers of the said tuple of tuples:

(25.5, -18.0, 3.75)

'''

#warmup

tup1=((10, 10, 10), (30, 45, 56), (81, 80, 39), (1, 2, 3))

print(tuple(zip(\*tup1)))

#((10, 30, 81, 1), (10, 45, 80, 2), (10, 56, 39, 3))

def avon(tups):

zt=tuple(zip(\*tups))

return tuple(map(lambda x: sum(x)/len(x), zt))

print(avon(((10, 10, 10), (30, 45, 56), (81, 80, 39), (1, 2, 3))))

print(avon(((1, 1, -5), (30, -15, 56), (81, -60, -39), (-10, 2, 3))))

#Textbook

def average\_tuple(nums):

result = tuple(map(lambda x: sum(x) / float(len(x)), zip(\*nums)))

return result

'''

45. Write a Python program to convert string elements to integers

inside a given tuple using lambda.

Original tuple values:

(('233', 'ABCD', '33'), ('1416', 'EFGH', '55'), ('2345', 'WERT', '34'))

New tuple values:

((233, 33), (1416, 55), (2345, 34))

'''

#warmup

str1='1234'

str2='hello'

soi=lambda x: int(x) if x.isdecimal() else None

print(soi(str1))

print(soi(str2))

soi = lambda x: int(x) if x.isdecimal() else None

def csti(tups):

res = tuple()

for i in tups:

filtered\_tup=tuple(filter(lambda y: y.isdecimal(),i))

res += tuple(map(soi,filtered\_tup))

return res

print(csti((('233', 'ABCD', '33'), ('1416', 'EFGH', '55'), ('2345', 'WERT', '34'))))

# (233, 33, 1416, 55, 2345, 34) not the desired output

#Textbook

def tuple\_int\_str(tuple\_str):

result = tuple(map(lambda x: (int(x[0]), int(x[2])), tuple\_str))

return result

tuple\_str = (('233', 'ABCD', '33'), ('1416', 'EFGH', '55'), ('2345', 'WERT', '34'))

print("Original tuple values:")

print(tuple\_str)

print("\nNew tuple values:")

print(tuple\_int\_str(tuple\_str))

'''

46. Write a Python program to find the index position

and value of the maximum and minimum values in a given list

of numbers using lambda.

Original list:

[12, 33, 23, 10.11, 67, 89, 45, 66.7, 23, 12, 11, 10.25, 54]

Index position and value of the maximum value of the said list:

(5, 89)

Index position and value of the minimum value of the said list:

(3, 10.11)

'''

def ipomm(lsts):

# print([i for i,j in enumerate(lsts) if j==max(lsts)])

print(list(map(lambda x:(x.index(max(x)),max(x)),[lsts]))[0])

print(list(map(lambda x:(x.index(min(x)),min(x)),[lsts]))[0])

ipomm([12, 33, 23, 10.11, 67, 89, 45, 66.7, 23, 12, 11, 10.25, 54])

#Textbook

def position\_max\_min(nums):

max\_result = max(enumerate(nums), key=(lambda x: x[1]))

min\_result = min(enumerate(nums), key=(lambda x: x[1]))

return max\_result,min\_result

nums = [12,33,23,10.11,67,89,45,66.7,23,12,11,10.25,54]

print("Original list:")

print(nums)

result = position\_max\_min(nums)

print("\nIndex position and value of the maximum value of the said list:")

print(result[0])

print("\nIndex position and value of the minimum value of the said list:")

print(result[1])

'''

47. Write a Python program to sort a given mixed list of integers

and strings using lambda. Numbers must be sorted before strings.

Original list:

[19, 'red', 12, 'green', 'blue', 10, 'white', 'green', 1]

Sort the said mixed list of integers and strings:

[1, 10, 12, 19, 'blue', 'green', 'green', 'red', 'white']

'''

lst1=[19, 'red', 12, 'green', 'blue', 10, 'white', 'green', 1]

lsti=sorted(list(filter(lambda x:str(x).isdecimal(),lst1)))

lstst=sorted(list(filter(lambda x:str(x).isalpha(),lst1)))

print(lsti+lstst)

#Textbook

def sort\_mixed\_list(mixed\_list):

mixed\_list.sort(key=lambda e: (isinstance(e, str), e))

return mixed\_list

mixed\_list = [19,'red',12,'green','blue', 10,'white','green',1]

print("Original list:")

print(mixed\_list)

print("\nSort the said mixed list of integers and strings:")

print(sort\_mixed\_list(mixed\_list))

'''

48. Write a Python program to sort a given list of strings

(numbers) numerically using lambda.

Original list:

['4', '12', '45', '7', '0', '100', '200', '-12', '-500']

Sort the said list of strings(numbers) numerically:

['-500', '-12', '0', '4', '7', '12', '45', '100', '200']

'''

lst1=['4', '12', '45', '7', '0', '100', '200', '-12', '-500']

print(sorted(list(map(lambda x:int(x), lst1))))

#Textbook

def sort\_numeric\_strings(nums\_str):

result = sorted(nums\_str, key=lambda el: int(el))

return result

nums\_str = ['4','12','45','7','0','100','200','-12','-500']

print("Original list:")

print(nums\_str)

print("\nSort the said list of strings(numbers) numerically:")

print(sort\_numeric\_strings(nums\_str))

'''

49. Write a Python program to count the occurrences of items

in a given list using lambda.

Original list:

[3, 4, 5, 8, 0, 3, 8, 5, 0, 3, 1, 5, 2, 3, 4, 2]

Count the occurrences of the items in the said list:

{3: 4, 4: 2, 5: 3, 8: 2, 0: 2, 1: 1, 2: 2}

'''

from collections import Counter

lst1=[3, 4, 5, 8, 0, 3, 8, 5, 0, 3, 1, 5, 2, 3, 4, 2]

print(dict(Counter(lst1)))

func=lambda x: Counter(x)

print(dict(func(lst1)))

#Textbook

def count\_occurrences(nums):

result = dict(map(lambda el : (el, list(nums).count(el)), nums))

return result

#warmup-dict

lst3=[1,2,3,4]

lst4=[11,12,13,14]

print(dict((zip(lst3,lst4))))

'''

50. Write a Python program to remove specific words from

a given list using lambda.

Original list:

['orange', 'red', 'green', 'blue', 'white', 'black']

Remove words:

['orange', 'black']

After removing the specified words from the said list:

['red', 'green', 'blue', 'white']

'''

def rsw(lsts,rl):

return list(filter(lambda x:x not in rl, lsts))

print(rsw(['orange', 'red', 'green', 'blue', 'white', 'black'],['orange','black']))

'''

51. Write a Python program to find the maximum and minimum values

in a given list of tuples using the lambda function.

Original list with tuples:

[('V', 62), ('VI', 68), ('VII', 72), ('VIII', 70), ('IX', 74), ('X', 65)]

Maximum and minimum values of the said list of tuples:

(74, 62)

'''

lst1=[('V', 62), ('VI', 68), ('VII', 72), ('VIII', 70), ('IX', 74), ('X', 65)]

print(tuple((max(map(lambda x:x[1], lst1)),min(map(lambda x:x[1], lst1)))))

#Textbook

def max\_min\_list\_tuples(class\_students):

return\_max = max(class\_students,key=lambda item:item[1])[1]

return\_min = min(class\_students,key=lambda item:item[1])[1]

return return\_max, return\_min

'''

52. Write a Python program to remove None values from a given list

using the lambda function.

Original list:

[12, 0, None, 23, None, -55, 234, 89, None, 0, 6, -12]

Remove None value from the said list:

[12, 0, 23, -55, 234, 89, 0, 6, -12]

'''

def rn(lsts):

return list(filter(lambda x:x!=None, lsts))

print(rn([12, 0, None, 23, None, -55, 234, 89, None, 0, 6, -12]))

#Textbook

def remove\_none(nums):

result = filter(lambda v: v is not None, nums)

return list(result)