29-4-23.

## List Comprehensions :-

~~CUP~~

-> In python, list comprehension is a simple and compact syntax for making a list out of a list.

-> It's a quick technique to make a new list by performing an operation on each item in the old one.

-> List comprehension is much faster than using a for loop to handle a list.

-> A list comprehension is made up of brackets that hold the expression that is run for each element, as well as for loop that iterates through each element.

Syntax :

① var = [returnvalue for var in coll]

② var = [returnvalue for var in coll if <condition>]

③ var = [returnvalue if <condition> else return-value for var in coll]

④ var = [returnvalue for var in coll for var in coll]

⑤ Var = [returnValue for var in collection for var in collection if <condition>]

⑥ Var = [returnvalue if <condition> else returnvalue for var in collection for var in collection].

program :-

```
l = [ ]
for i in 'hello':
    l += [i]
print (l)
```

By using list comprehension

```
l = [i for i in 'hello']
print (l)
```

program WAP to generate sequence of alphabets in given range by using list comprehension.

```
l = [chr(i) for i in range(int(input('enter:')), int(input('enter:'))+1)]
print (l).
```

program :-

I/P = 10 20 30

O/P = [10, 20, 30]

```
l = [ ]
for i in st.split(' '):
    l += [int(i)]
print(l)
```

```
l = [int(i) for i in st.split(' ')]
print(l).
```

program :-

```
st = "sai@gmail.com raghu@gmail.com
ratnana@gmail.com rani@outlook.com"
```

O/P : ['sai', 'raghu', 'ratnana', 'rani']

O/P : ['gmail.com', 'gmail.com', 'gmail.com', 'outlook.com']

```
l = [ ]
for i in st.split(' '):
    l += [i.split('@')[0]]
print(l)
```

```
l = [ ]
for i in st.split():
    l+ = [i.split('@')[1]]
print(l).
```

```
l = [i.split('@')[0] for i in st.split()]
print(l)
```

```
l = [i.split('@')[1] for i in st.split()]
print(l).
```

program :-

```
start = int(input('enter :'))
end = int(input('enter:'))
l = [ ]
for i in range(start, end+1):
    l+ = [i]
print(l)
```

o/p :-

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

```
l = [i for i in range (int (input ("enter:")),
                       int (input ("enter:")) + 1)]
print (l)
```

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

WAP to extract vowels and ascii numbers from given string by using list comprehension.

```
st = 'helloworld 123'
l = ''
for i in st:
    if i in 'aeiouAEIOU' or
    'o' <= i <= '9':
        l += i
print (l)
l = ''.join ([i for i in st if i
              in 'aeiouAEIOU' or
              'o' <= i <= '9'])
print (l)
```

2-5-23.

WAP to convert uppercase to lowercase.

```
st = "HELLO WORLD@123"
res = [ ]
for i in st:
    if 'A'<=i<='Z':
        res+=[chr(ord(i)+32)]
    else:
        res+=[i]
print(''.join(res))
```

```
res = ''.join([chr(ord(i)+32) if 'A'<=i
        <='Z' else i for i in
        st])
print(res)
```

o/p :-

hello world@123

hello world@123

WAP to eliminate the duplicate characters from given string.

```
st = "HELLO WORLD@123"
res = [ ]
for i in st:
    if i not in res:
        res+=[i]
```

```
print(res)
temp = ' '.join(set(list(st)))
print(temp)
```

o/p :-

```
['H', 'E', 'l', 'o', ' ', 'W', 'r', 'L',
 'D', '@', '1', '2', '3']
```

@ol 123DELHrW

WAP to eliminate the duplicate Characters from given string output should be sorted order.

```
st = "HEllo WorLD@123123"
temp = ''.join(sorted(list(set(st))))
print(temp)
```

o/p :-

123@DEHrWlor

WAP to replace special characters and ascii numbers with '*'

```
St = "HELLO@ #$ WORLD 123"
s = [ ]
```

```
for i in st:
    if not('A'<=i<='Z' or 'a'<=i
            <='z'):
        st=['*']
    else:
        s+=[i]
print(''.join(s))
s=['*' if not('A'<=i<='Z' or 'a'<=i
    <='z') else i for i in st]
print(''.join(s))
```

o/p :-

HELLO****WORLD***
HELLO****WORLD***

Program :-

```
st = "helloworldhaii"
o/p = "h1l14w6r1dh11213"
l=[]
for i in range(0, len(st)):
        if st[i] in 'aeiouAEIOU':
            l+=[str(i)]
        else:
            l+=[st[i]]
print(''.join(l))
```

```
St = "helloworldhaii"
l = ''.join([str(i) if st[i] in 'AEIOU
            aeiou'
            else st[i] for i in
            range (0, len(st))])
Print(l)
```

O/P :-

h1l14w6r1d h11213.

h1114w6r1d h11213.

WAP to display combination of substring.

```
l = [ ]
for i in st:
    for j in st:
        l+ =[(i, j)]
print(l)
```

```
St = '01'
l = [ ]
for i in st:
    for j in st:
        l+ = [i+j]
print(l)
l= [i+j for i in '01' for j in '01']
print(l)
```

O/P :-

[('0', '0'), ('0', '1'), ('1', '0'), ('1', '1')]

['00', '01', '10', '11']

['00', '01', '10', '11']

WAP to display Palindromes in given String.

```
st = '21655042135545211311551226'
n = 2
l = []
for i in range(0, len(st) - (n-1)):
    if st[i:i+n] == st[i:i+n][::-1]:
        l += [st[i:i+n]]
print(l)
l = [st[i:i+n] for i in range(0, len(st)
    -(n-1)) if st[i:i+n] == st[i:i+n]
    [::-1]]
print(l)
```

['545', '454', '131']

['545', '454', '131'].

Program :-

```
St = '1234213'
for i in range (0, len(st)):
    for j in range (i+1, len(st)):
        print(st[i:j],
              end = ", ")
```

1, 12, 123, 1234, 1234 21, 2, 23, 234, 2342, 23421,
3, 34, 342, 3421, 4, 42, 421, 2, 21, 1.

Program :-

```
St = '12343213'
for i in range (0, len(st)):
    for j in range (i+1, len(st)):
        if len(st[i:j]) > 1 and
            st[i:j] == st[i:j][::-1]:
                print (st[i:j])
```

O/P :-

1234321
23432
343.

3-5-23

Program :-

```
st = '1234321'
for i in range (0, len(st)):
    for j in range (i+1, len(st)):
        if len(st[i:j]) > 1 and
            st[i:j] == st[i:j][::-1]:
            print (st[i:j])
l = [st[i:j] for i in range (0, len(st))
    if len(st[i:j]) > 1 and st[i:j]
    == st[i:j][::-1]]
print(l)
```

O/p :-

```
1234321
23432
343
```

['1234321', '23432', '343']

program :-

```
l = [[10, 3, 60], [5, 16, 18, 16], [7, 66, 9, 7],
     [12, 18, 12]]
u = [ ]
```

```
for i in l:
    for j in i:
        u+=[j]
print(u)
```

```
l = [[10, 3, 60], [5, 16, 18, 16], [7, 66, 9, 7],
     [12, 18, 12]]
l = [j for i in l for j in i]
print(l)
```

o/p :-

[10, 3, 60, 5, 16, 18, 16, 7, 66, 9, 7, 12, 18, 12]

program :-

```
l = [[10, 3, 60], [5, 16, 18, 16], [7, 66, 9, 7],
     [12, 18, 12]]
```

o/p : [10, 9, 60, 25, 16, 18, 16, 49, 66, 81, 49,
12, 18, 12]

```
u = [ ]
for i in l:
    for j in i:
```

```
        if j%2==0:
            ut=[j]
        else:
            ut=[j**2]
print(u)
```

```
l=[[10,3,60],[5,16,18,16],[7,66,9,7],
   [12,18,12]]
l=[ j if j%2==0 else j**2
   for i in l for j in i]
print(l)
```

Program :- sum of even numbers in a sub list.

```
l=[[10,3,60],[5,16,18,16],
   [7,66,9,7],[12,18,12]]
u=0
for i in l:
    for j in i:
        if j%2==0:
            ut=j
print(u)
```

```
l = sum([j for i in l for j in i if j%2==0])
print(l)
```

o/p :-

228

228

WAP add the values inside the two list. -> next page.

zip function : user defined

```
a = [1, 2, 3]
b = [4, 5, 6, 1, 5, 8]
def u-zip(coll1, coll2):
    if len(coll1,) < len(coll2):
        n = len(coll2)
        m = len(coll1)
        res = n-m
        coll1 = coll1 + res * [0]
    elif len(coll1) > len(coll2):
        n = len(coll2)
        m = len(coll1)
        res = m-n
```

```
        coll2 = coll2 + res * [0]
    if  len(coll1) == len(coll2):
        t = [ ]
        for i in range(0, len(coll1)):
            t += ((coll1[i], coll2[i]),)
    return t
res = u_zip(a,b)
print(res)

for i,j in res:
    print(i,j)
```

O/P :- [(1,4), (2,5), (3,6), (0,1), (0,5), (0,8)]

```
1 4
2 5
3 6
0 1
0 5
0 8
```

answer :-

```
a = [1, 2, 3, 0]        b = [4, 5, 6, 8]
l = [ ]
for i in range(0, len(b)):
    l += [a[i] + b[i]]
print(l)
```

4-5-23

## Tuple Comprehensive :

It returns the generator object.

Syntax :-

```
var = (returnvalue for var in coll)

var = (returnvalue for var in coll
       if <Condition>)

var = (returnvalue if <condition> else
       return value for var in coll)

var = (returnvalue for var in coll for var
       in coll)

var = (returnvalue for var in coll for
       var in coll if <condition>)

var = (returnvalue if <condition> else
       returns value for var in coll
       for var in coll)
```

## Set Comprehensive :

Syntax :-

```
var = {returnvalue for var in coll}
```

var = {returnvalue for var in coll
if <condition>}

var = {returnvalue if <condition> else
return value for var in coll}

var = {returnvalue for var in coll
for var in coll}

var = {returnvalue for var in
coll for var in coll if
<condition>}

var = {returnvalue if <condition> else
returnvalue for var in coll
for var in coll}

Dict Comprehensive:

It return value we should follow
key and value format.

Syntax:

var = {return key: return value for
var in coll}

var = {return key: returnvalue for
var in coll if <condition>}

var = {return key : returnvalue
if <condition> else return value
for var in coll }

var = {return key : returnvalue for
var in coll for var in coll
if <condition> }

var = {return key : returnvalue for var
in coll for var in coll }

var = {return key : returnvalue
if <condition> else return value
for var in coll for var
in coll }.

examples :-

l = [[10, 3, 60], [5, 16, 18, 16], [7, 66, 9, 7],
[12, 18, 12]]

l = [; for i in l for j in i]

print(l)

```
l =
l = (j for i in l for j in i)
print (tuple (l))

l =
l = {j for i in l for j in i}
print (l)

l =
l = {j:j for i in l for j in i}
print (l)
```

O/P :-

[10, 3, 60, 5, 16, 18, 16, 7, 66, 9, 7, 12, 18, 12]

(10, . . . . . .)

{66, 3, 5, 7, 9, 10, 12, 16, 18, 60}

{10: 10, 3: 3, 60: 60, 5: 5 . . . }

Program :-

```
S = {1:1, 2:8, 3:27}
d = { }
for i in range(0,10):
    d[i] = i**3
Print(d)

d = {i : i**3 for i in range(0,10)}
print(d)
```

O/P :-

{0:0, 1:1, 2:8, 3:27, 4:64 ....}

{0:0, 1:1, 2:8, 3:27, 4:64 ....}.

9-5-23.

WAP to find out the each word frequency.

```
St = "i like india i love indians i
      like programming"
d = { }
```

```
for i in st.split():
    if i in d:
        d[i] += 1
    else:
        d[i] = 1
print(d)
```

O/P :-

```
{'i': 3, 'like': 2, 'india': 1, 'love': 1, 'indians': 1, 'programming': 1}
```

using dict. comp. :-

```
st = st.split()
d = {i : st.count(i) for i in st}
print(d)
```

WAP to find max occurance word.

```
st = "i like india i love indians i like python"
m = 0
s = ""
for j in set(st):
    if m < st.count(j):
        m = st.count(j)
        s = j
print(s,m)
```

O/P

```
i 4
```

Program :-

```
st = " i like india ".split()
d = []
for i in set(st):
    d += [[st.count(i), i]]
print(max(d))
```

using list comp.

```
d = max([[st.count(i), i] for i in set(st)])
print(d)
```

o/p :-

[6, like]

[6, like]

WAP to find out the max and min word each line of given string (file).

```
st = """ """
temp = st.splitlines()
count = 1
```

```
for i in temp:
    m = max([[i.count(j), j] for
        j in set(i.split())])
    n = min([[i.count(j), j] for
        j in set(i.split())])
print("line", count, "maxword", m[1],
    "with count", m[0], \n
    "min word", n[1], "with count",
    n[0])
count += 1
```

10-5-23.

function :-

```
def pys():
    return 0
```

mainspace

| execution area | variable space |
|---|---|
| -> def pys():<br>-> ~~pys~~<br>0x71<br>-> ~~pys()~~<br>0x71()<br>-> app = ~~pys~~<br>0x71<br>-> ball = ~~pys()~~<br>0x71() | pys = 0x71<br>app = 0x71<br>ball = 0x48 |

method area

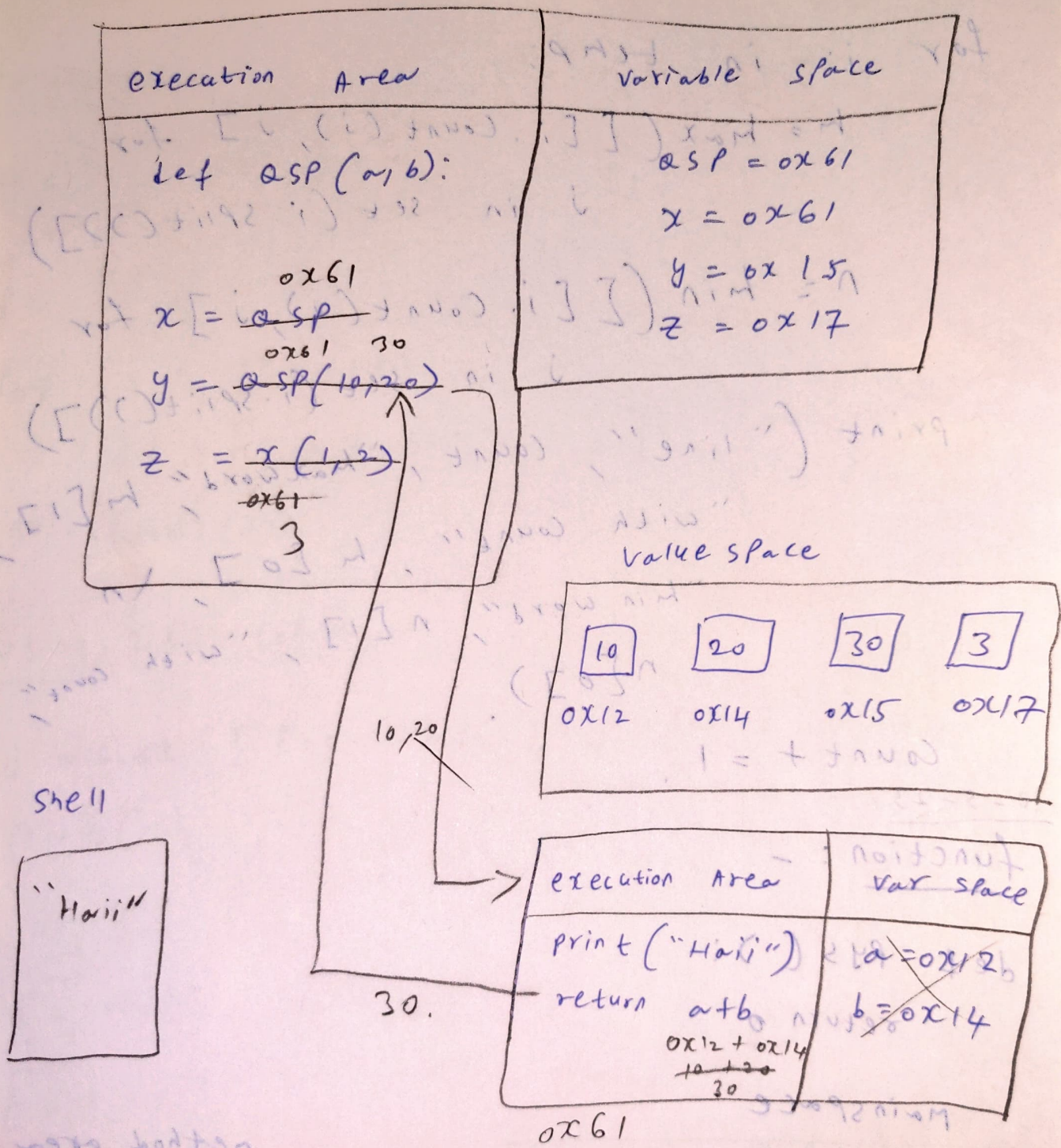
return 0

value space.

0

0x48

0x71




Lambda function :-

```
def add(a,b):
    return a+b
print(add(10,20))
```

```
res = lambda a,b : a+b
print(res(10,20))
```

O/P :-

30

```
res = (lambda a,b : a+b)(10,20)
print(res)
```

30

## Lambda

-> lambda function is a function programming

-> lambda functions, also known as 'anonymous functions', are similar to regular python functions; these are functions without names.

-> The def keyword is used to define normal functions, while the lambda keyword is used to define anonymous function.

-> Lambda function will return the address where the instructions are stored and you can execute with the name of the variable, where the address is stored and do the call from it.

-> To reduce the size of the code we use the lambda function. When we reduce the lines of code the program efficiency is more.

-> It is contained in one line of code.

-> when we use this function can eliminate the repetitive task.

-> whenever we use lambda function the program time complexity is very less compared to normal functions.

Syntax:

```
var = lambda arg1, arg2 . . . :
    returnvalue <expression> # lambda function definition
print(var(arg1, args2 . . .)) # lambda function call
```

```
Var = lambda arg1, arg2 . . . :
    returnvalue if <condition> else return value
print (var (args . . .))
```

```
var = lambda arg1, arg2 . . . :(return value expression)(args . . .)
print (var)
```

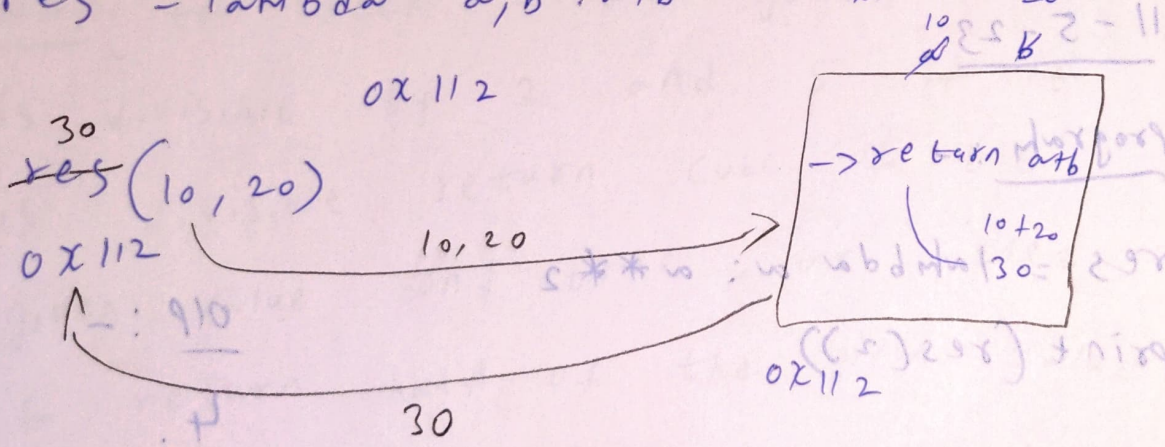
Var = lambda arg1, arg2 . . . :

(return_value if <condition> else return_value) (args . . .)

Print (Var)

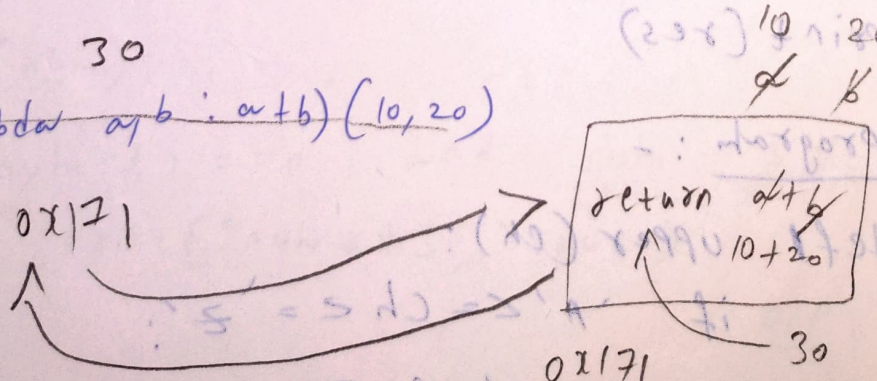
Note: inside the lambda function we can write only if else condition, if try to give any condition it return syntax error.

res = lambda a,b : a+b  res=0x112




30

res = (lambda a,b : a+b) (10, 20)



Var = lambda args1, args2 : return value

keyword — Inputs — output condition/expression

—> lambda will return (lambda conditional)

function address

—> by the help of function address we are call the lambda function.

—> it is ~~using~~ used for only small scale operation.

11-5-23

Program

```
res = lambda a: a**2
print(res(2))
```

O/P :-

4.

```
res = (lambda a: a**2)(2)
print(res)
```

4.

Program :-

```
def upper(ch):
    if 'A'<=ch<='Z':
        return True
    else:
        return False
```

```
temp = upper('a')
print (temp)

res (lambda ch : True if 'A' <= ch <= 'Z'
        else False) ('a')
print (res)
```

o/p :-

False
False.

WAP to check whether given number is divisible by 2 and 5 if it is divisible return cube of the given value and true, or else to return half of the given value with false

```
def fun(num):
    if num % 2 == 0 and num % 5 == 0:
        print(num**3, 'True')
    else:
        print ( num//2, 'False')
fun(10)
```

o/p:

1000, True.

```
res
=(lambda num : (num**3, True) if n%2==0
                and n%5==0 else
                (num//2, False))(10)
print(res)
```

o/p :-

(1000, True)

WAP to take input from the user and return the input.

```
def demo():
    s=input()
    print(s)
demo()
print((lambda a: a)(input()))
```

(or)

```
res = (lambda : eval(input()))()
print(res)
```

WAP to check given character is lowercase or not, if lowercase return uppercase character, or else return none.

```
def fun(ch):
    if 'a'<=ch<='z':
        return chr(ord(ch)-32)
    else:
        return
fun('b')
```

```
print((lambda ch: chr(ord(ch)-32) if 'a'<=ch<='z' else None)('a'))
```

O/P:

B

A

WAP to check greatest of two numbers return greatest number.

```
def great(a,b):
    if a>b:
        return a
    else:
        return b
great(19,10)
```

res = (lambda a, b : a if a > b else b)
(10, 20)
print (res).

o/p :
20

WAP to return last value of the
Coll by using function

coll = [1, 2, 3, 4]
def last (coll):
    return coll[-1]

o/p :-
4.

last (Coll)

res = (lambda Coll : Coll[-1]) ('hello')
print (res)

o/p
o

program :-

t = [1, 2, 4, 3, 9, 8, 7, 6]
o/p = [1, 4, 8, 3, 9, 16, 7, 12]

①
l = [ ]
for i in t:
    if i % 2 == 0:
        l += [i * 2]

```
    else:
        l+=[i]
print(l)
```

②

```
def check(coll):
    l=[]
    for i in coll:
        if i%2==0:
            l+=[i*2]
        else:
            l+=[i]
    return l
print(check(t))
```

③

```
def check(coll):
    l=[]
    for i in coll:
        l+=[(lambda n:n*2 if n%2==0 else i)(i)]
    return l
print(check(t))
```

④

```
def check(coll):
    l = [(lambda n: n*2 if n%2==0 else i)(i) for i in coll]
    return l
print(check(t))
```

⑤

```
res = (lambda coll: [(lambda n: n*2 if n%2==0 else i)(i) for i in coll])(t)
print(res)
```

⑥

```
res = map(lambda n: n*2 if n%2==0 else n, t)
print(list(res))
```

⑦

```
res = list(map(lambda n: [n, n*2][n%2==0], t))
print(res).
```

WAP to convert uppercase to lower case in given string.

S = 'Hello WORLD @123'

①

```
St = ''
for i in S:
    if 'A'<=i<='Z':
        St += chr(ord(i)+32)
    else:
        St += i
print(St)
```

②

```
def islower(Coll):
    St = ''
    for i in Coll:
        if 'A'<=i<='Z':
            St += chr(ord(i)+32)
        else:
            St += i
    return St
res = islower(S)
print(res)
```

③

```
def islower(coll):
    st=""
    for i in coll:
        st+=(lambda ch: chr(ord(ch)+32)
             if 'A'<=ch<='Z' else ch)
    return st
res = islower(st)
print(res)
```

④

```
def islower(coll):
    return ''.join([(lambda ch: chr(ord(ch)
        if 'A'<=ch<='Z' else ch)(i)
        for i in coll])
res = islower(st)
print(res)
```

⑤

```
res=(lambda coll: ''.join([(lambda
    ch: chr(ord(ch)+32) if 'A'<=ch<='Z'
    else ch)(i) for i in coll]))(st)
print(res)
```

(6)

```
res = ''.join(list(map(lambda ch: chr
(ord(ch)+32) if 'A'<=ch<='Z' else
ch, st)))
```

(7)

```
res = ''.join(list(map(lambda ch:
[ch, chr(ord(ch)+32)] ['A'<=ch<='Z']
[st)))
print(res)
```

12-5-23

**Map** :- Map is a predefined function. It will perform mapping between collection of values and function.

-> whenever we want to perform the lambda operation or functionality on the collection of values then we use map function.

-> To reduce the size of the code we can use the map function.

-> we will create a link between the list of values and the function by providing them in the collection one by one.

Syntax :-

```
var = map(address_of_function, collection)
print(list(var))
```

-> MaP( ) will return a MAP object address that can be either given to the static iterator, for loop or and typecasting.

-> which stores the address returned by map function is called map(object)

-> map function returns the address where the values will be stored.

Example:

Wap to print the series of squares consider only odd values in a list if even values are present directly stored in a list.

```
def odd(a):
    l = []
    for i in a:
        if i % 2 != 0:
            l += [i ** 2]
```

else:

    l += [i]

return l

print(odd([10,15,18,19,17,16,13,12]))

```
var = map(lambda a: a**2 if a%2!=0
          else a, [10,15,18,19,17,16,13,12])
print(list(var))
```

filter:

-> It is a predefined function used to filter the values from the given collection and that condition is satisfied or specifying those results are stored inside the filter object.

-> Inside the filter function we provide the address of the function and collection.

-> Address of function is a function based on which we decide whether to keep the values or not by using condition statements.

-> filter will also return the memory address called filter object, where the values will be stored.

-> filter() will return a filter object that can be either given to an static iterator or for loop or typecasting.

-> which stores the address returned by filter

function is called filter object.

-> when the condition is satisfied it stores the satisfied value inside the filter object based on the result of the function.

-> to reduce the size of the code by using the filter function.

-> filter function will not accept the none value. It will avoid the none type value.

Syntax :

```
var = filter (function_address, collection)
print (list (var)).
```

Program :-

```
l = [1, 2, 3, 4, 5, 6, 7, 8, 9]
def odd (coll):
    l = [ ]
    for i in coll:
        if i % 2 != 0:
            l += [i**2]
        else:
            l += [i]
    return l
odd(l)
```

```
l = [1, 2, 3, 4, 5, 6, 7, 8, 9]
def check(i):
    if i % 2 != 0:
        return i**2
```

```
    else:
        return i
def map(func, coll):
    l = [ ]
    for i in coll:
        l += [func(i)]
    return l
print(map(chec*, l))

l = [1, 2, 3, 4, 5, 6, 7, 8, 9]
res = lambda i: i**2 if i%2!=0
    else i
def map(func, coll):
    l = [ ]
    for i in coll:
        l += [func(i)]
    return l
print(map(res, l))

l = [1, 2, 3, 4, 5, 6, 7, 8, 9]
res = lambda i: i**2 if i%2!=0 else i
var = map(res, l)
print(var)
```

```
val = map(lambda i: i**2 if i%2!=0
          else i, [1,2,3,4,5,6,7,8,9])
print(val)
```

15-5-23.

Program :-

```
coll = 'helloWORLD@123'
s = []
for i in coll:
    if 'A'<=i<='Z':
        st = [chr(ord(i)+32)]
    else:
        st = [i]
print(s)
```

```
St = 'helloWORLD@123'
def u-lower(coll):
    s = []
    for i in coll:
        if 'A'<=i<='Z':
            st = [chr(ord(i)+32)]
        else:
            st = [i]
```

```
    return s
print(u_lower(st))

def check(ch):
    if 'A'<=ch<='Z':
        return chr(ord(ch)+32)
    else:
        return ch
def u_map(func, coll):
    l=[]
    for i in coll:
        l+=[func(i)]
    return l
print(u_map(check, st))

res = map(check, st)
print(list(res))

res = map(lambda ch: chr(ord(ch)+32)
          if 'A'<=ch<='Z' else ch, st)
print(list(res))
```

O/P :-

['h', 'e', 'l', 'l', 'o', 'w', 'o', 'r', 'l', 'd', '@', '1', '2', '3'].

Program :-

```
St = "ABCDE $123%gf$st   ABCDE $123%gf"

def duplicate(coll):
    l = [ ]
    for i in coll:
        if i not in l:
            l+=[i]
        else:
            l+=[None]
    return l
print(duplicate(st))

def check(ch, l):
    if ch not in l:
        return ch
    else:
        return None
def u_map(func, coll):
    l = [ ]
    for i in coll:
        l+=[func(i, l)]
```

return l

Print (u_map(check, st))

res = set(st)

print(res)

## filter function :-

→ If condition is satisfied it stores the actual value or it stores none value. But, in final result it avoid the none value.

## program :-

```
l = [10, 20, 50, 60, 88, 129, 131, 89]

def u_odd(coll):
    l = [ ]
    for i in coll:
        if i % 2 == 0:
            l += [i]
        else:
            l += [None]
    return l
print(u_odd(l))
```

```
res = map(lambda n: n if n%2==0
          else None, l)
print(list(res))

res = filter(lambda n: n if n%2==0
             else None, l)
print(list(res))

res = map(lambda n: n**2 if n%2==0
          else None, l)
print(list(res))

res = filter(lambda n: n**2 if n%2
             ==0 else None, l)
print(list(res))
```

o/p :-

[10, 20, 50, 60, 88, None, None, None]

" " "

[10, 20, 50, 60, 88]

[100, 400, 2500, 3600, 7744, None, None, None]

[10, 20, 50, 60, 88]