

29-4-23.

## List Comprehensions :-

### What

- 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  $\text{Var} = [\text{returnValue for var in coll}]$
- ②  $\text{Var} = [\text{returnValue for var in coll}$   
if <condition>]
- ③  $\text{Var} = [\text{returnValue if < condition > else}$   
 $\text{return-value for var in coll}]$
- ④  $\text{Var} = [\text{returnValue for var in coll}$   
for var in coll]

⑤  $\text{Var} = [\text{return value for var in collection if } \text{condition}]$

⑥  $\text{Var} = [\text{return value if condition}] \text{ else return value for var in collection}$

Program :-

$\lambda = []$

$[\text{condition}] = +1$

$(\lambda) + \text{ring}$

for ;  $\text{if } \text{condition}; \text{ if } i \text{ not } (\text{condition}) = \lambda$   
 $i += [i]$   
print ( $\lambda$ )

- : Program

By using list comprehension

$\lambda = [i \text{ for } i \text{ in } \text{when}] \text{ by } \text{generator}$

print ( $\lambda$ )

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

$\lambda = [\text{chr}(i) \text{ for } i \text{ in } \text{range}(\text{int}(\text{input('center:'))}), \text{int}(\text{input('center:'))} + 1)]$

$[\text{ord}(\text{'A'}) + \text{ring}, i] = +1$

print ( $\lambda$ )

$(\lambda) + \text{ring}$

Program :- Ni. Yuv rot salavant [ ] = Yuv @

O/p : [10, 20, 30] ni Yuv

O/p : [40, 20, 30] fi salavant [ ] = Yuv @

l = [ ] ni Yuv rot salavant

for i in st.split(' ') :

l += [int(i)] - : step by

print(l)

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

print(l).

program :-

st = "sai@ghail.com raghu@ghail.com  
ratnana@ghail.com rani@outlook.com"

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

O/p : ['ghail.com', 'ghail.com', 'ghail.com', 'outlook.com']

l = [i for i in st.split(' ') if i.endswith('@')] = l

l += [i.split('@')[0]]

print(l)

$\{l \in [i] \mid l \text{ in } s \text{ and } l \neq i\} = l$

for  $i \in s$ :  $s[i] + l$

$l = [i, s[i] + l @ [i]]$  (1) string  
print(l).

Now we have working of HW  
 $l = [i, s[i] + l @ [i]]$  for  $i \in s$ :  $s[i] + l$   
print(l)

$l = [i, s[i] + l @ [i]]$  for  $i \in s$ :  $s[i] + l$   
print(l).

Program :-

start = int(input('enter:'))  
end = int(input('enter:'))

$l = [ ]$  (1) string  
 $i = + 1$

for  $i$  in range(start, end+1):

$l + [i]$  (1) string  
 $i = + 1$

print(l)

O/P :-

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

$\lambda = [i \text{ for } i \text{ in range}(int(input('Enter:')),$   
 $\text{int}(input('Enter:'))+1)] * of$

$\text{print}(\lambda)$

$[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$ . (1) + nisq

WAP to extract vowels and ascii  
numbers from a given string by using  
list comprehension. (1) string

$st = 'helloworld123'$   
 $\lambda = [i \text{ for } i \text{ in } st \text{ if } i \in 'aeiouAEIOU' \text{ or } i \in '0123456789']$

for i in st:

if i in 'aeiouAEIOU' or  
or '*i*' <= 'z' <= '9':  
 $\delta t = i$

$\text{print}(\lambda)$

: (1+nisq, nisq) + nisq  
 $\lambda = [i \text{ for } i \text{ in } st \text{ if } i \in 'aeiouAEIOU' \text{ or } '0' < i <= '9']$

$\text{print}(\lambda)$

[(1+nisq, nisq) + nisq]

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:

    rest += [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 = "HE110 WorLD@123"

res = []

for i in st:

    if i not in res:

        rest += [i]

```
print(res)
```

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

```
print(temp)
```

O/P :-

```
[‘H’, ‘E’, ‘l’, ‘o’, ‘ ‘, ‘w’, ‘r’, ‘l’,  
‘D’, ‘@’, ‘1’, ‘2’, ‘3’]
```

@01 123 DELH&W

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@DEH2Wl08
```

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:

        st += [i]

print(''.join(st))

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 = "helloworldhai"

O/P = "h1lll4w6yldh11213"

l = []

for i in range(0, len(st)):

    if st[i] in 'aeiouAEIOU':

        l += [st[i]]

    else:

        l += [st[i]]

print(''.join(l))

$st = "helloworldhavii"$

$\lambda = \lambda : join([st[i] if st[i] in 'AEIOU' or i+1 in range(0, len(st))])$

O/P:-

h1ll4 w681dhl1l213.

h1ll4 w681dhl1l213.

WAP to display combination of substring.

$\lambda = []$

for i in st:

    for j in st:

$lt = [i:j]$

print( $\lambda$ )

$st = "01"$

$\lambda = []$

for i in st:

    for j in st:

$lt = [i:j]$

print( $\lambda$ )

$\lambda = [lt; for i in "01" for j in "01"]$

print( $\lambda$ )

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 = '2165504213554521311551226'

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, 123421, 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 = '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])  
  
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

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

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)
```

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

$l = [j \text{ for } i \text{ in } l \text{ for } j \text{ in } i]$   
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 :  $[10, 3, 60, 25, 16, 18, 16, 49, 66, 81, 49, 12, 18, 12]$

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

if  $j \% 2 == 0:$

$u += [j]$   
else :

$u += [j ** 2]$   
print(u)

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

$l = [j \text{ if } j \% 2 == 0 \text{ else } j ** 2$   
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  $j \% 2 == 0:$

$u += j$

print(u)

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

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

$cold2 = coll2 + res * [0]$

if  $\text{len}(coll1) == \text{len}(coll2)$ :

$t = []$

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  
return 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 Comprehension:

If return value we should follow  
key and value format.

### Syntax:

`Var = {return key: returnvalue 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)`

$\lambda =$

$\lambda = (j \text{ for } i \text{ in } \lambda \text{ for } j \text{ in } i)$

print (tuple ( $\lambda$ ))

$\lambda =$

$\lambda = \{ j \text{ for } i \text{ in } \lambda \text{ for } j \text{ in } i\}$

print ( $\lambda$ )

$\lambda =$

$\lambda = \{ j : j \text{ for } i \text{ in } \lambda \text{ for } j \text{ in } i\}$

print ( $\lambda$ )

O/P :-

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

(10, 3, 60, 5, 16, 18, 16, 7, 66, 9, 12, 18, 12)

{66, 63, 5, 7, 9, 10, 12, 16, 18, 60}

{10 : 10, 3 : 3, 60 : 60, 5 : 5, 16 : 16, 18 : 18, 7 : 7}

"print( $\lambda$ )"

o/p

{ 3 = b }

Program :-

$s = \{1: 1, 2: 8, 3: 27\}$

$d = \{ \}$

for i in range(0, 10):

$d[i] = i**3$

Print(d)

$d = \{i: i**3 \text{ for } i \text{ in range}(0, 10)\}$

Print(d)

O/P :-

$\{0: 0, 1: 1, 2: 8, 3: 27, 4: 64, \dots\}$

$\{0: 0, 1: 1, 2: 8, 3: 27, 4: 64, \dots\}$

9-5-23.

WAP to find out the each word frequency.

St = "i like india i love indians ;  
like programming"

$d = \{ \}$

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

using dict. Compr. :-

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

WAP to find last occurrence of word, d  
st = "i like indian i love indians i like  
i programming" [st python]  
l = 0  
for j in set(st):  
 if l < st.count(j):  
 l = st.count(j)  
 s = j

```
print(s, l)
```

## Program :-

```
st = "i like india :: b::i::i".split()
```

$$d = \left[ \exists^{\epsilon: i} \right]_{\perp} + \left[ \exists^{\epsilon: i} \right]_b$$

```
for i in set(st):
```

$\text{df} = [\text{list}.\text{count}(i), i]$

```
print(max(d))
```

using list comprehension.

$d = \max([st.\text{Count}(i), i] : \text{for } i \text{ in set(st2)})$

print(d)

[6, like]

[6; like]

WAP to find out the max and min of six string.

WAP to find word each line of given string (file).

`st = " " . . . :60+no"';" +`

temp = st. splitlines('')[-1].split(' ')

Count = 1

for size in temp:

$m = \max([i.count(j), j] \text{ for } i \in \text{temp}, j \in \text{set}(i.split()))$

$fixo = \min([i[i].count(q), i] \text{ for } i \in \text{temp}, q \in \text{set}(i.split()))$

print("line", count, "max word", m[1])

"with count", m[0], "\n"

"min word", n[1], "with count" -

fixo = 21x0 0x0 0x0 n[1] 0x0

Count += 1.

10-5-23.

11512

function :-

```
def pys(s):  
    fixo = 0  
    return 0
```

Main Space

Execution area	Variable space
→ def pys():	
→ pys	pys = 0x7f1
0x7f1	APP = 0x7f1
→ pys()	bail = 0x48
0x7f1C	0x7f1
→ APP = pys	
0x7f1	
→ bail = pys()	(0x48) bba + 0x48
0x7f1C	0x48

method area  
variable

return 0

d+value space

(0x48) bba + 0x48

## Execution Area

```
def add(a, b):
    if a <= b:
```

$x = \text{a} + \text{b}$

$y = \text{a} * \text{b}$

$z = x - y$

$z = 3$

## Variable Space

$a = 10$

$b = 20$

$c = 30$

$d = 3$

$e = 15$

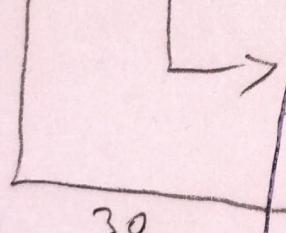
$f = 17$

## Value Space

10	20	30	3
0x12	0x14	0x15	0x17

## Shell

"Haiii"



Execution Area	Var Space
print("Haiii") return a+b $0x12 + 0x14$ $10 + 20$ 30	<del><math>a = 0x12</math></del> <del><math>b = 0x14</math></del> <del><math>c = 0x15</math></del> <del><math>d = 0x17</math></del>

0x61

was bottom

Lambda function :-

```
def add(a, b):
    return a+b
```

```
print(add(10, 20))
```

8450

1550 = 289

1550 = 99A

8450 = 114d

C) 289 ←

C) 114d ←

289 = 99A ←

114d ←

(289 = 114d) ←

(114d) ←

```

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

```

Lambda function  $\rightarrow$   $g1f$   $g2w$   $g1f$   $g2w$

- $\rightarrow$  Lambda function is a function programming  $\rightarrow$  function
- $\rightarrow$  Lambda functions, also known as 'anonymous functions', are similar to regular Python functions but these are functions without names.
- $\rightarrow$  The def keyword is used to define normal functions, while the lambda keyword is used to define anonymous functions.
- $\rightarrow$  Lambda function will return the address where the instructions are stored, and you can execute with the value of the variable where the address is stored and do the call from it. (row) string

→ 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: the syntax of lambda

Var = ~~lambda~~ ~~def~~ ~~print~~ ~~return~~ ~~if~~ ~~else~~ ~~value~~ ~~args~~ ~~arg1~~ ~~arg2~~ ... ~~return~~ ~~expression~~ # lambda function definition

print(Var(arg1, arg2, ...)) # lambda function call

Var = lambda arg1, arg2, ...  
return value if < condition > else return value

print(Var(args...)) # show variables  
lambda arg1, arg2, ...: return

Var = ~~lambda~~ ~~def~~ ~~print~~ ~~return~~ ~~value~~ ~~expression~~ (args...)

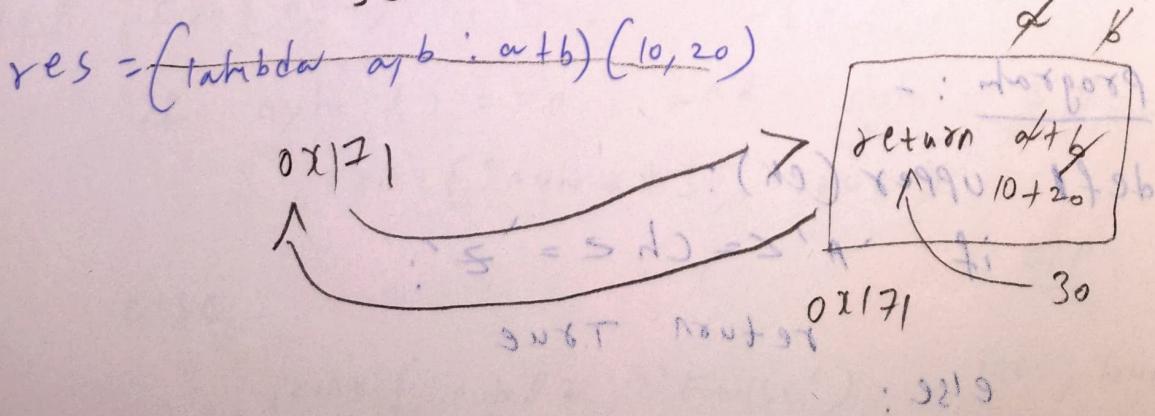
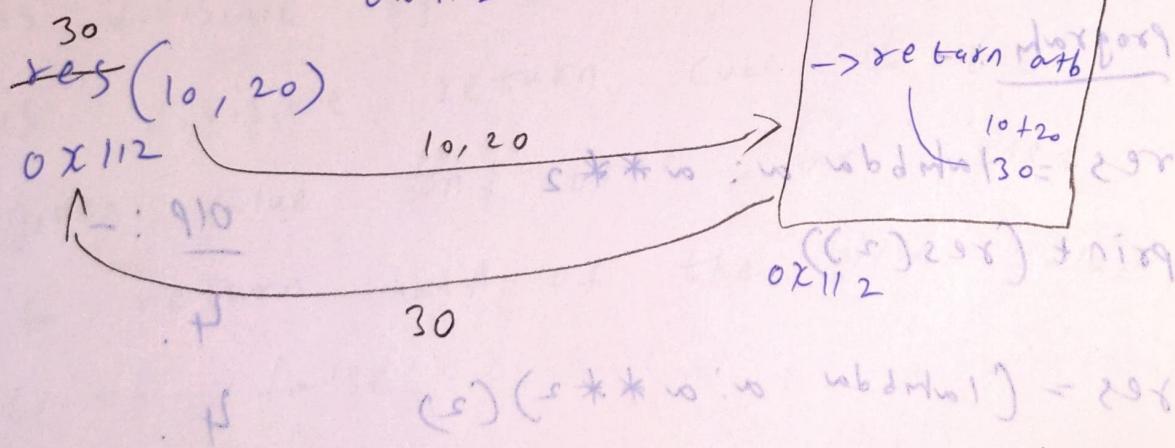
print(Var) . if start method say

~~Var = lambda arg1, arg2 : if condition else~~  
~~return value if condition else~~  
~~(return value) (args...)~~

Print(Var) number will be printed  
 (lambda position)

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

~~no if & else in lambda~~  
 res = lambda a, b : a + b res=0x112 20



using return

$\text{Var} = \lambda \text{lambda} \quad \begin{array}{l} \text{args1, args2: return value} \\ \text{Keyword args} \end{array}$   
 or  
 $\lambda \text{lambda} \quad \begin{array}{l} \text{Inputs} \\ \text{Condition/Expression} \end{array}$   
 $(\dots - \text{args}) (\text{lambda} - \text{condition})$

$\rightarrow \lambda \text{lambda}$  will return  $(\lambda \text{lambda})$   
 function address

$\rightarrow$  by the help of function address  
 we are calling the lambda function.  
 $\rightarrow$  it is used for small scale operation.

$11 - 5 = 23$        $\text{defn: dpm: dpm return value} = 23$   
 $11 \times 0$

Program :-

$\text{res} = \lambda \text{lambda} \ a: a^{**2}$   
 $\text{print(res(2))}$

$\text{print(res)}$

$\text{res} = (\lambda \text{lambda} \ a: a^{**2})(2)$   
 $\text{print(res)}$

Program :-

$\text{defn upper(ch):}$   
 $\text{if } 'A' \leq ch \leq 'Z':$   
 $\quad \text{return True}$   
 $\text{else:}$   
 $\quad \text{return False}$

$(23, 0) 23$

$11 \times 0$

O/P :-

4.

4.

$(23, 0) (dpm: dpm return value) = 23$

```

temp = upper('a') : fun wbdmho) = 238
print(temp)
3219 o == 2.1.1 bmo
res(1&h1b1d1(s2Ch1;stratn))if 'A' <= ch <= 'Z'
else false) ('a')
print(res) - : 210

```

(GNT, 000)

O/P :-

False. 3219 sat start of 9AY

False. 3219 sat next bmo

WAP to check whether given number  
is divisible by 2 and 5 if it  
is divisible return cube of the  
given value and true, 3219 else  
to return half of the given value  
with false((( )+091i)(w : w wbdmho) + n1x9) (7a)

```

def Ifun(num) lws : wbdmho) = 238
if num // 2 == 0 and num // 5 == 0:
    print('*num**3, 'True')

```

else :

O/P :

print( num//2, 'False') 1000, True.

fun(10)

```

res = (lambda num: ((num ** 3, True)) if n % 2 == 0
       and n * 1.5 == 0 else
       ((n // 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()  # to find number of
print((lambda a: a)(input()))  # o/p

```

```

res = (lambda : eval(input()))()
print(res)  # o/p

```

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

def fun(ch):

if '^a' <= ch <= '^z': print 'A' [1-110] = 208  
else: print 'B' [1-110] = 110  
return chr(ord(ch) + 32) [2-110] + 96

else:

return

(110) + 201

fun('b')

[1-110] : 110 : 110 word [1-110] = 208

print(chr(ord(ch) - 32)) [2-110] + 96

'a' < ch <= 'z'

else None) ('a') [2-110] + 96

B

[d, f, g, p, e, h, s, v] = 1

A

[d, f, g, p, e, h, s, v] = 1

WAP to check greatest of two numbers return greatest number.

def great(a, b):

O/P : [ ] = 8

if a > b:

[q + n] i yof

return a. o = s y i fi

else:

great(19, 10) return b [s \* i] = 11

WAP to return last value of the  
array.

Coll by using function :-  
 $\text{coll} = [1, 2, 3, 4]$  O/p :-  
def last(coll):  
 n = len(coll)  
 return coll[-1]

last (coll)  $\approx N + 3\ell$

programm) (noch 919 : 910

$$t = [1, 2, 4, 3, 9, 8, 7, 6]$$

$$O/P_{out} = [1, 4, 8, 3, 9, 16, 7, 12]$$

① 7901000 320-0978 080398 28984789

```

l = [ ]           : (d, o) resp   496
for i in t_pi    : d < o   si
    if i % 2 == 0.0  newj

```

$$\lambda^+ = [i * 2] \text{, } \lambda(0) = (0, 0) \text{, } \lambda(1) = (1, 0)$$

else :

$t = [i]$  : (100) 23812 + 36

print(t)

②  $[i * i] + [i * i * i * i]$  = 1

def check(coll): (1) (i 219)

$t = [ ]$

for i in coll:

if  $i \% 2 == 0$ :

$t + [i * i]$  : (100) 23812 + 37

$t + [i * 2]$

(1) (i 219) else: (100) 23812 + 37

(2) (E1)  $t + [i]$  + col

return t

(238) 23812 + 37

print(check(t))

③

def check(coll): whatal = 238

$t = [ ]$

for i in coll: ((238) 23812 + 37)

$t + [\lambda n: n * 2 \text{ if}$

$i \% 2 == 0 \text{ else } i)](i)$

print(check(t))

((238) 23812 + 37)

④

```
def check(coll): [i] = +1
    l = [lambda n: n * 2 if (l) + n == 0
          else i) (i). (for) in coll]
    return l
```

[i] = 1

print (check(t))

⑤

```
res = lambda coll: [lambda n: n * 2
                     if n * 1.2 == 0 else i) (i)
                     for i in coll] (t)
```

print (res)

⑥

```
res = map(lambda n: n * 2 if n * 1.2 == 0
          else n, t) [i] = 1
print (list(res))
```

⑦

```
res = [i * t for i in map(lambda n: n * 2
                           if (n % 2) == 0 else t)
                           for t in range(1, 10)]
```

print (res).

WAP to convert uppercase to lower case in given string.

①  $s = \text{'Hello WORLD@123'}$

st = ''

for i in s: if ord(i) >= 65 and ord(i) <= 90:

if 'A' <= i <= 'Z': st += chr(ord(i) + 32)

else:

st += i

print(st)

②

def islower(coll):

st = ''

for i in coll: if i <= 'z' and i >= 'a':

if 'A' <= i <= 'Z': st += chr(ord(i) - 32)

else:

st += i

return st

res = islower(s) print(res)

print(res) if res == 'Hello world@123':

(52) ((Hello ni i rof (i) (A) 9218

(298) + nisq

③ of 32-37991 9AM

pair2 navip n 322

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)+32)  
 if 'A' <= ch <= 'Z': ch  
 else ch])  
 res = islower(st)  
 print(res)

⑤

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

⑥

- :  $x = \text{chr}$

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

⑦ This program is written in C++ using C++

using `vector` and `for` loop.

```
res = '' + join(list(map(lambda ch:
    [ch, ch + ord(ch)+32]) if 'A' <= ch <= 'Z'
    + s + ))
```

print(res);

12-5-23. So now explain what exactly

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.

$$[e^{**i}] = +1$$

Syntax:-

$\lambda(x) : x \mapsto \text{what}(x)$   $x = 2 \rightarrow \text{what}(2) = 2^2 = 4$   
Var. = map (address - of - function  
 $\Rightarrow s \mapsto x \mapsto A$  if  $(\{x\} \text{ bro})$   
Collection)  $\{x + (A)\}$   
print(list(var))  $((x, A))$

→ Map() will return a MAP object  
address that can be either given  
to the start, (x), iterator, for loop  
[so  $\rightarrow$  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:-  
if no input then it is from :- num

Map to print the series of squares  
Consider only odd values in a list  
even values are present so directly stores  
in list. so not even to

we have got to find out what or  
def odd(a): not just got to do not  
as  $i = [1, 3, 5]$  and we have to do  
not just got to do 2nd position to 2nd  
position got to do 3rd position to 3rd  
if  $i \% 2 != 0:$   
 $i += [i ** 2]$

```

else: Inside condition block if not found
    S[i] = 2 * S[i] + 1 if condition is not found set new
    value inside + 1 = [i]
else: Inside condition if found return
return 1 present say to find S[i] so
print(odd([10, 15, 18, 19, 17, 16, 13, 12])) or <-
    below same S[i] not found & set it odd
var = map(lambda a: a**2 if: a%2 != 0
        else a, [10, 15, 18, 19, 17, 16, 13, 12])
print(list(var))
, ((var) + 1) string

```

### 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:  $filter(function\_address, collection)$

$([1, 2, 3, 4, 5, 6, 7, 8, 9], \lambda i : i \text{ is odd})$

print(list(var)).

Program = no filter benitsberg w  $i + i^2$

$\lambda = [1, 2, 3, 4, 5, 6, 7, 8, 9]$   $\lambda i : i \text{ is odd}$

def odd(coll):  $\lambda i : i \text{ is odd}$  print( $i^2$ )

$\lambda = []$

for i in coll:  $\lambda i : i \text{ is odd}$   $\lambda i : i^2$

if  $i \% 2 != 0$ :  $\lambda i : i^2$

$\lambda i : i^2$

else:  $\lambda i : i^2$

$\lambda i : i^2$

return  $\lambda i : i^2$

odd( $\lambda$ )

$\lambda = [1, 2, 3, 4, 5, 6, 7, 8, 9]$   $\lambda i : i^2$

def check(i):  $\lambda i : i^2$

if  $i \% 2 != 0$ :  $\lambda i : i^2$

return  $\lambda i : i^2$

if else: if  $i \neq * i$  is odd) result =  $\lambda w$   
( $\lambda P. \lambda x. x$  return  $i$ )  $i$  odd

def map(func, coll):  $\lambda wv$  fair  
 $\lambda = []$   $\underline{E \in 2 - 2}$

for i in coll:

$\lambda += [func(i)]$

return  $\lambda$

print(map(check, l))

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

res = [lambda i:  $i * i^2$  if  $i \% 2 == 0$   
else i]

def map(func, coll):

$\lambda = []$

for i in coll:

$\lambda += [func(i)]$

return  $\lambda$

print(map(res, l))

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

res = [lambda i:  $i * i^2$  if  $i \% 2 == 0$  else i]

var = map(res, l)

print(var)

`var1 = map(lambda i: i**2 if i%2 == 0  
else i, [1, 2, 3, 4, 5, 6, 7, 8, 9])  
print(var1).`

15-5-23.

Program :-

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

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

```

return s
print(u-lower(st))
"ABCDEF123456789" = st

def check(ch):
    if '^A' <= ch <= '^Z':
        return chr(ord(ch)+32)
    else:
        return ch

def l-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 :-
[ '^e', '^t', '^P', '^o', '^w', '^o', '^r', '^t', '^d', '^@',
  '^1', '^2', '^3' ]

```

Program :-

$((\text{st}) \times \text{word\_v}) \text{ string}$   
 $\text{st} = \text{"ABCDE * 123kgf st ABCDE * 123kgf"}$

$\text{def duplicate(coll):}$        $((\text{st})) \times \text{coll} \rightarrow \text{list}$   
     $\lambda = [\ ]$        $\because s \Rightarrow \lambda = \lambda \cup s;$   
     $\text{for i in } (\text{coll})$        $\text{return copy}(\text{coll}, \lambda)$   
         $\text{if i not in } \lambda:$        $\therefore \text{coll}$   
             $\lambda = [\ i; \ ]$        $\text{add } i \text{ to } \lambda$   
         $\text{else:}$        $((\text{coll}), \lambda) \text{ dont\_i} \rightarrow \lambda$   
             $\lambda = [\ \text{None} \ ]$        $[\ ] = \lambda$   
     $\text{return } \lambda$        $((\text{st})) \text{ if i in st}$

$\text{print(duplicate(st))} \Rightarrow [\ ] = +1$

$\text{def check(ch, l)}$        $((\text{st}), \text{coll}) \text{ string}$   
     $\text{if ch not in l:}$   
         $\text{return ch}$        $((\text{st}), \text{coll}) \text{ result = 20}$   
     $\text{else:}$        $((\text{st}), \text{coll}) \text{ string}$   
         $\text{return None}$

$\text{def u-map(func, coll):}$        $((\text{st}), \text{coll}) \text{ result = 20}$   
     $\lambda = [\ ]$        $((\text{st}), \text{coll}) \text{ string}$   
     $\text{for i in coll:}$        $\therefore \text{20}$   
         $\lambda = [\ \text{func}(i, \lambda) \ ]$        $[\ ] = \lambda$

$\therefore \text{20} \times (\text{st}) \times \lambda = [\ \text{func}(\text{st}, \lambda) \ ]$

return null; } n = width) first = 238

Print (u-map (checky st))  
 $((2 \times 4) + 2 \times 1) + 4 \times 89$

res = set(st)

```
res = set(st)
print(res) # {A, B, C, D} (order may vary)
```

filter function :-  $((x \geq 1) + \text{airy})$

→ If condition is satisfied it stores the actual value or if it stored none value. But, (in final result) it avoid the none value. ((233) + (1)) + 19

program :-

```
def b_odd(coll):
```

$$\lambda = \lfloor \quad \rceil$$

for i in coll:

[Show if  $i^{100} \cdot 2 = 2^{100}$ : 188, 100, 02, 105, 101]

$$\lambda^+ = [i]$$

else :

$i+ = \text{None}$  [ 8, 10, 12, 15, 17 ]

return 1

```
return l  
print(u-odd(1))
```

```

res = map(lambda n: n if n // 2 == 0
          else None, x)
print(list(res))

```

```

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

```

return  $\max(\text{lambda}, \frac{n}{n+2})$  if  $n \geq 2$   
else None, d).  $\max(\text{lambda}, \frac{n}{n+2})$   
print (list (res))

```
res = filter(lambda n: n**2 if n % 2 == 0 else None, [1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
print(list(res))
```

O/P :-

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

$$\text{If } \int_{\gamma} f(z) dz = 0, \text{ then } f(z) \text{ is analytic.}$$

$$[10, 20, 50, 60, 88]_{\text{non}} = +1$$

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

$$[10, 20, 50, 60, 88]$$