

welcome

welcome

shift+enter=run the cell

dd=delete the cell

Keywords

In [1]:

```
import keyword  
print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

In [2]:

```
false=10
```

In [3]:

```
false
```

Out[3]:

```
10
```

identifier

name given to entity like class, functions, variables etc

In [4]:

```
1var=20
```

Input In [4]

```
1var=20
```

^

SyntaxError: invalid decimal literal

In []:

```
var1=20
```

In []:

```
var1
```

In []:

```
var@=30
```

In []:

```
var_=30
```

In []:

```
var_
```

In []:

```
finally=40
```

In []:

```
Finally=40
```

In []:

```
Finally
```

In []:

```
from=50
```

In []:

```
From=50
```

In []:

```
From
```

comments in python :used to explain the code for more readability

In []:

```
print('python')    #define python
```

In []:

```
#single line comment  
#multi line comment
```

In []:

```
"this  
is python  
session"
```

In []:

```
'''welcome  
to  
the  
india'''
```

statement

In []:

```
val5=10
```

In []:

```
p1=20+30  
p1
```

In []:

```
p2=20+30\  
+40+50\  
+70+80  
p2
```

In []:

```
p2=20+30\  
+40+50\  
+70+80  
p2
```

indentation

In []:

```
x=10  
if x==10:  
    print('x is equal to 10')
```

docstrings

In []:

```
def square(num):  
    """square function will return the square of a number"""  
    return num**2
```

In []:

```
square(2)
```

In []:

```
square.__doc__
```

variables

In []:

```
a=5
```

In []:

```
a
```

In []:

```
p=50  
q=25  
r=q
```

In []:

```
print(id(p))
```

In []:

```
print(id(q))
```

data type

int float bool/boolean complex

In []:

```
a=1254789631422
```

In []:

```
type(a)
```

In []:

```
b=10.2  
b
```

In []:

```
type(b)
```

In []:

```
bool1=True
```

In []:

```
bool1
```

In []:

```
print(bool(-2))
```

In []:

```
x=2+3j
```

In []:

```
type(x)
```

Strings

In []:

```
str1="hello"
```

In []:

```
type(str1)
```

In []:

```
len(str1)
```

In []:

```
str2=" hello python "
```

In []:

```
str2[0]
```

In []:

```
len(str2)
```

indexing

In []:

```
str3='welcome'
```

in python we always start indexing from 0

In []:

```
str3[0]
```

In []:

```
str3[-7]
```

Slicing

In []:

```
str3[3:6]
```

In []:

```
str3[3:7]
```

In []:

```
str3[0:]
```

In []:

```
str3[3:]
```

In []:

```
str4="bangalore"
```

In []:

```
str4[2:5]
```

In []:

```
str4[3:]
```

In []:

```
str5='data science'
```

In []:

```
str5[5:]
```

In []:

```
str5[2:8]
```

In []:

```
str6='heloo'
```

In []:

```
str[3]='l'
```

In []:

```
del str6
```

In []:

```
str6
```

string concatenation

In []:

```
s1='data'  
s2='science' #data science
```

In []:

```
print(s1,s2,"dvnsdvndkndvn")
```

In []:

```
print(s1+'-'+s2)
```

String Membership

In []:

```
mystr='hello everyone' #in,not in membership operator
```

In []:

```
print('hello' in mystr)
```

In []:

```
print('python' in mystr)
```

In []:

```
print(' eve' in mystr)
```

In []:

```
print('-eve' in mystr)
```

string partitioning

In []:

```
mystr1='natural language with python and R and java'  
mystr1
```

In []:

```
l=mystr1.partition('with')  
l
```

In []:

```
mystr1.capitalize()
```

In []:

```
mystr1.upper()
```

In []:

```
mystr1.lower()
```

In []:

```
mystr1.count('a')
```

In []:

```
mystr2="  hello  "  
mystr2
```

In []:

```
mystr2.strip()
```

In []:

```
mystr2.rstrip()
```


In []:

```
myst2.lstrip()
```

21 july 2023

In []:

```
str6="heloo"  
print(id(str6))
```

In []:

```
x=str6.replace('o','l')  
print(id(x))
```

In []:

```
str7='good morning'
```

In []:

```
str7.replace("good", "beautiful")
```

In []:

```
str8='one two three four five six seven'
```

In []:

```
str8
```

In []:

```
str8.split()
```

variable assignment

In []:

```
x=10
```

In [5]:

```
x,y,z=20,25.5,'hello'
```

In [6]:

```
print(x)
print(y)
print(z)
```

```
20
25.5
hello
```

Operators

Arithmetic

Assignment

comparition /relational

logical

membership

identity

bitwise

Arithmetic =====

Addition

Substraction

Multiplication

Division

Floor division

Modulos

Exponentional

In [7]:

```
print(10+2)
print(10-2)
print(10*2)
```

```
12
8
20
```

In [8]:

```
print(15/2)
print(15//2)
print(15%2)
```

7.5
7
1

In [9]:

```
print(2**2)
```

4

Assignment

In [10]:

```
x=5
```

In [11]:

```
y=10
y+=2    #y=y+2
```

In [12]:

```
print(y)
```

12

In [13]:

```
z=15
z-=10    #z=z-10
```

In [14]:

```
print(z)
```

5

comparision /relational

== Equal

!= not equal

#> greater than

< less than

<= less than equal to

#>= greater than equal to

In [15]:

```
x=10
y=20
print(x==y)
print(x<y)
print(x>y)
```

False
True
False

Logical operators

and

or

not

In [16]:

```
x=10
y=20
print((x<5) and (y<10))
print((x<5) or (y<10))
print((x==10) and (y==20))
print((x==10) or (y==20))
```

False
False
True
True

identity operators

In [17]:

```
print((10) is (10.0))
```

False

```
<>:1: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:1: SyntaxWarning: "is" with a literal. Did you mean "=="?
C:\Users\Mukund\AppData\Local\Temp\ipykernel_1312\436760226.py:1: SyntaxWarning: "is" with a literal. Did you mean "=="?
  print((10) is (10.0))
```

In [18]:

```
print('hello')
x=10
if (x>5):
    print('welcome')
print(hdsvvhsvn)
print('python')
```

hello
welcome

NameError Traceback (most recent call last)

Input In [18], in <cell line: 5>()
 3 if (x>5):
 4 print('welcome')
 ----> 5 print(hdsvvhsvn)
 6 print('python')

NameError: name 'hdsvvhsvn' is not defined

Type Casting

In []:

```
a='2'
b=2
```

In []:

```
type(a)
```

In []:

```
type(b)
```

Auto type casting

Forced Type casting

In []:

```
4+3.23+False    #4.00+3.23+0.00
```

In []:

```
3+2.3+False+True    #3.00+2.3+0.00+1.00
```

In []:

```
2+3+'data'+4.5
```

In []:

```
12+15+'10'+2
```

In []:

```
print(int(True))
```

In []:

```
print(int(bool("data")))
```

In []:

```
a='10'
```

In []:

```
print(int(a))
```

In []:

```
b='data123'
```

In []:

```
print(int(b))
```

In []:

```
z=3.2  
print(int(z))
```

In []:

```
x=3  
print(float(x))2
```

In []:

```
print(float(False))
```

Python KT session 24 July 23

List

In []:

```
list1=[10,20,30,40,50] #homogenous same kind of data  
print(list1)  
print('hello')
```

In []:

```
type(list1)
```

In []:

```
list2=[60,20.5,'happy'] #heterogenous multiple kind of data  
list2
```

In []:

```
type(list2)
```

In []:

```
list3=[[1,2,3],[4,5,6]] #nested list
```

In []:

```
type(list3)
```

Properties of List

In []:

```
#list store element in sequential order  
#list store heterogenous element  
#list allow duplicate value  
#list are mutable or changeable
```

In []:

```
list4=[1,1,20,30,40]  
id(list4)
```

In []:

```
list4
```

In []:

```
list4[1]=5
```

In []:

```
list4  
id(list4)
```

In []:

```
list5=[10,50,7,80,90,5,20,100]
```

In []:

```
list5[-4]
```

In []:

```
list5[-7]
```

In []:

```
list5[-5:-2]    #start index postion:
```

In []:

```
list6=[[1,2,3],[4,5,6]]
```

In []:

```
list6[1][1]
```

In []:

```
list7=[10,50,7,80,90,5,20,100]
```

In []:

```
list7.append(99)
```

In []:

```
list7
```

In []:

```
list7.append(105)
```

In []:

```
list7
```

In []:

```
list7.pop()
```

In []:

```
list7
```


In []:

```
list7.pop()
```

In []:

```
list7
```

In []:

```
list7.pop(2)
```

In []:

```
list7
```

In []:

```
list7.insert(0,5)
```

In []:

```
list7
```

In []:

```
list7.reverse()
```

In []:

```
list7
```

In []:

```
list7.sort()
```

In []:

```
list7
```

In []:

```
list7.sort(reverse=True)
```

In []:

```
list7
```

Tuple

In []:

```
tup1=(10,20,30,40)
tup1
```

In []:

```
type(tup1)
```

In []:

```
tup2=50,60,40,70
tup2
```

In []:

```
type(tup2)
```

Properties of tuple

In []:

```
#tuple store element in sequential order
#tuple allow duplicate values
#tuple is immutable or unchangeable
```

In []:

```
tup3=(1,1,5,7,8)
```

In []:

```
tup3
```

In []:

```
tup3[1]=9
```

In []:

```
tup4=(10,20,30,50,70,80,90)
```

In []:

```
tup4[3:5]
```

In []:

```
tup4.index(80)
```

In []:

```
tup4.count(50)
```

Set

In []:

```
set1={10,20,60}  
set1
```

In []:

```
type(set1)
```

In []:

```
set2={60,50,10,20,80,10,50}  
len(set2)
```

In []:

```
set2
```

Properties of Set

In []:

```
#set store element in random order  
#set didn't allow duplicate values  
#set is immutable or mutable  
#indexing and slicing is not possible in set
```

In []:

```
list10=[] #empty list  
type(list10)
```

In []:

```
tup10=() #empty tuple  
type(tup10)
```

In []:

```
set3=set() #empty set  
type(set3)
```

In []:

```
set4={10,20,50,70,80}
```

In []:

```
set4[1:3] #indexing /slicing is not possible
```

In []:

```
set4.remove(70)
```

In []:

```
set4
```

In []:

```
set4.discard(80)
```

In []:

```
set4
```

In []:

```
set5={10,20,50,70,80,90,100}
```

In []:

```
set5.remove(105)
```

In []:

```
set5.discard(109)
```

Dictionary

In []:

```
dict1={} #empty dictionary  
type(dict1)
```

In []:

```
dict2={'name':'rohit','age':25}
```

In []:

```
dict2
```

In []:

```
dict2.keys()
```

In []:

```
dict2.values()
```

In []:

```
dict2.items()
```

In []:

```
dict2['location']='chennai'
```

In []:

```
dict2
```

In []:

```
dict2['location']='mumbai'
```

In []:

```
dict2
```

In []:

```
dict3={1:'one',2:'two','A':['shankar','balaji','girish']}
```

In []:

```
dict3
```

In []:

```
dict3['A'][1][2]
```

In []:

```
dict4={0:{'one':1,'two':2},1:{'city':'banglore','area':'whitefield'},'three':['Mumbai','P
```

In []:

```
dict4
```

In []:

```
dict4[1]['area']
```

In []:

```
dict4[0]['two']
```

In []:

```
dict4['three'][0]
```

In []:

```
dict4['three'][1][2]
```

In []:

```
dict4[1]['city']
```

In []:

```
dict4.pop('three')
```

Conditional Statement

In []:

```
#if  
#if---else  
#if--elif--elif....  
#nested if
```

In []:

```
x=10  
if x==10:           #true  
    print('matched')  
print('hello')
```

In []:

```
x=20  
if x==10:           #false  
    print('matched')  
print('hello')
```

In []:

```
y=15  
if y>20:  
    print('welcome')  
print('python')
```

In []:

```
y=15  
if y<20:  
    print('welcome')  
print('python')
```

In []:

```
#write down the code as a output "logged in" when password is 12345
```

In []:

```
y=12345
if y==12345:
    print('logged in')
```

In []:

```
x=10
y=20
print('hello')
if(x+y==30 and x<y):    #true
    print('yes')
else:
    print('no')
print('welcome')
```

In []:

```
x=13
if x%2==0:                #false
    print('x is even')
else:
    print('x is odd')
```

In []:

```
y=12
if y%3==0:                #cond true
    print('divisible')
else:
    print('not divisible')
```

In []:

```
x=2
if x==1:                  #false
    print('python')
elif x==2:                #true
    print('jupyter')
elif x==3:
    print('ML')
else:
    print("AI")
```

In []:

```
x=10
if x==1:
    print('python')
elif x==2:
    print('jupyter')
elif x==3:
    print('ML')
else:
    print("AI")
```

In []:

```
light_color='pink'
if light_color=='red':
    print('stop')
elif light_color=='green':
    print('go')
elif light_color=='orange':
    print('slow')
else:
    print('invalid color')
```

In []:

```
light_color='green'
x=10
if light_color=='red':
    if x==10:
        print('hello')
    print('stop')
elif light_color=='green':
    print('go')
elif light_color=='orange':
    print('slow')
else:
    print('invalid color')
```

In []:

```
light_color='red'
x=11
if light_color=='red':
    if x==10:
        print('hello')
    print('stop')
elif light_color=='green':
    print('go')
elif light_color=='orange':
    print('slow')
else:
    print('invalid color')
```

input method

In []:

```
light_color=input("Enter the color of the traffic light(red/green/orange): ")

if light_color=='red':
    print('stop')
elif light_color=='green':
    print('go')
elif light_color=='orange':
    print('slow')
else:
    print('invalid color')
```

Python KT 25 July 23

In [20]:

```
name=input("enter your name : ") #input method always return string value
```

enter your name : mukund

In [21]:

```
age=input('enter the age :')
print(type(age))
```

enter the age :20
<class 'str'>

In [22]:

```
age=int(input('enter the age :')) #you need to convert the data type using type casting
print(type(age))
```

enter the age :30
<class 'int'>

Control Statement

Loop----while and for

In [23]:

```
i=1          #intializer
while i<=5:   #expression #true
    print(i)
    i=i+1
```

1
2
3
4
5

In [28]:

```
i=5
while i>=2:
    print(i)
    i=i-1
```

5
4
3
2

In [48]:

```
name='hello'
i=0
while i<=4:
    print(name[i],end=' ')
    i=i+2
```

h l o

In [46]:

```
name='umesh'
i=0
while i<=4:
    print(name[i],end=" ")
    i=i+1
```

u m e s h

In [35]:

```
print('hello','welcome',sep="*")
```

hello*welcome

In [40]:

```
print('hello')
print('welcome')
```

hello
welcome

In [45]:

```
print('hello',end=" ")
print('welcome')
```

hello welcome

For loop

In [50]:

```
x=[10,20,45,37,30,40,50]
for i in x:
    if i%2==0:
        print(i)
```

10
20
30
40
50

In [57]:

```
x=[10,40,21,35,32]
for i in x:
    if i%2!=0:
        print(i)
```

21
35

In [58]:

```
name='welocme'
for i in name:
    if i in "aeiou":
        print(i)
```

e
o
e

Nested For

In [61]:

```
a=['science','math','english']
b=[10,20,30]
for i in a:    #science
    for j in b:    #10
        print(j,i)
```

10 science
20 science
30 science
10 math
20 math
30 math
10 english
20 english
30 english

Range

In [60]:

```
#to generate range of numbers
```

In [63]:

```
x1=list(range(100))  
print(x1)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,  
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 3  
9, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57,  
58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 7  
6, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94,  
95, 96, 97, 98, 99]
```

In [64]:

```
x2=list(range(25)) #range will generate from 0 and end with n-1  
print(x2)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,  
21, 22, 23, 24]
```

In [65]:

```
x3=list(range(1,10))  
print(x3)
```

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

In [66]:

```
x3=list(range(5,20)) #it will start from intilizer and end with n-1(2 variable)  
print(x3)
```

```
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

In [67]:

```
x3=list(range(2,25,2))  
print(x3)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24]
```

In [73]:

```
x3=list(range(10,30,17))  
print(x3)
```

```
[10, 27]
```

In [76]:

```
#1-100 included
x3=list(range(1,101))
print(x3)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 2
1, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 5
8, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76,
77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 9
5, 96, 97, 98, 99, 100]
```

In [80]:

```
x3=list(range(101,5,-1))
print(x3)
```

```
[101, 100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 85, 84,
83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 6
5, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47,
46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 2
8, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10,
9, 8, 7, 6]
```

Packing and unpacking

In [86]:

```
marks=72,85,91,59           #packing
rohit,balaji,gayathri,gomathi=marks  #unpacking
print(rohit)
print(balaji)
print(gayathri)
print(gomathi)
print(type(marks))
```

```
72
85
91
59
<class 'tuple'>
```

In [88]:

```
x=10,20,30
a,b,c=x

print(a)
print(b)
print(c)
print(type(x))
```

```
10
20
30
<class 'tuple'>
```

Functions

In [89]:

```
#block of code which perform specific task for us
#once we will assign/create function we can call n number of times
#parameter
#argument
```

In []:

```
#build in
#user defined
#anonymous
```

In [90]:

```
x=[39,17,5,9,25]
print(len(x))    #no. of element
print(type(x))   #datatype
print(max(x))    #max no.
print(min(x))    #min no.
print(sum(x))    #sum no.
```

```
5
<class 'list'>
39
5
95
```

In [91]:

```
def myfun1(a,b):    #def defination to define user defined functions
    return a+b
res=myfun1(10,20)   #calling the function
print(res)
res*5
```

```
30
```

In [92]:

```
def myfun1(a,b):
    print(a+b)
myfun1(10,20)
```

```
30
```

In []:

```
#return to store the output .....
```

In [94]:

```
def square(n): #parameter variable define under the paranthesis
    n=n*n
    return n #store the output
res1=square(10) #argument assigning the value to the parameter
print(res1)
print(res1+5)
```

100

105

Types of Parameter/Argument

In [95]:

```
#positional
#keyword
#default
#Arbitrary / variable number of argument / args
#Keyword Arbitrary / keyword variable number of argument / kwargs
```

In [97]:

```
#positional
def add(x,y):
    print(x)
    print(y)

add(50,40)
```

50

40

In [98]:

```
#keyword
def add(x,y):
    print(x)
    print(y)

add(y=50,x=40)
```

40

50

In [107]:

```
#default
def add(x,y=20,z=50):
    print(x)
    print(y)
    print(z)

add(10)
```

10
20
50

In [109]:

```
#Arbitrary / variable number of argument / args
def fun2(x):
    print(x)

fun2()
```

```
-----
-
TypeError                                Traceback (most recent call las
t)
Input In [109], in <cell line: 5>()
      2 def fun2(x):
      3     print(x)
----> 5 fun2()
```

TypeError: fun2() missing 1 required positional argument: 'x'

In [110]:

```
def fun2(x):
    print(x)

fun2(10,20)
```

```
-----
-
TypeError                                Traceback (most recent call las
t)
Input In [110], in <cell line: 4>()
      1 def fun2(x):
      2     print(x)
----> 4 fun2(10,20)
```

TypeError: fun2() takes 1 positional argument but 2 were given

In [115]:

```
def fun2(*x):    #args
    print(x)

fun2()
fun2(10,20)
fun2(1,2,3)
fun2(99,80,70,60,70,50)

()
```

(10, 20)
(1, 2, 3)
(99, 80, 70, 60, 70, 50)

In [123]:

```
def fun3(x,*y,z):
    print(x)
    print(y)
    print(z)

#fun3(10,20,30,40,60)
#fun3(10,20,30,40)
fun3(10,20,30)
```

-

TypeError

Traceback (most recent call last)

t)

Input In [123], in <cell line: 8>()

```
4     print(z)
6 #fun3(10,20,30,40,60)
7 #fun3(10,20,30,40)
----> 8 fun3(10,20,30)
```

TypeError: fun3() missing 1 required keyword-only argument: 'z'

In [131]:

```
def fun3(x,*y,z=100):
    print(x)
    print(y)
    print(z)

fun3(10,20,30,z='hello')
```

10
(20, 30)
hello

In [130]:

```
x=10
print(x)
x=20
print(x)
```

```
10
20
```

In [133]:

```
#Keyword Arbitrary / keyword variable number of argument / kwargs
def fun5(**kwargs):
    print(kwargs)

fun5(name="rohit",location="mumbai",age=25)
```

```
{'name': 'rohit', 'location': 'mumbai', 'age': 25}
```

In [135]:

```
dict12={'name':"rohit",'location':"mumbai",'age':25}
def fun6(**kwargs):
    print(kwargs)

fun6()
```

```
{}
```

In []:

```
#anonymous function
# function without a name
# lambda is called inliner
#special function map and filter
```

In [136]:

```
def add(x,y):
    return x+y

res=add(15,25)
print(res)
```

```
40
```

In [137]:

```
add=lambda x,y:x+y #syntax Lambda input parameter : return value
res=add(15,25)
res
```

Out[137]:

```
40
```

In [138]:

```
fun6=lambda x:'even' if x%2==0 else "odd"  
res1=fun6(10)  
print(res1)
```

even

In [139]:

```
fun6=lambda x:'even' if x%2==0 else "odd"  
res1=fun6(13)  
print(res1)
```

odd

In [140]:

```
fun7=lambda y:'vowel' if y in "aeiou" else 'not vowel'  
res2=fun7('a')  
print(res2)
```

vowel

In [141]:

```
fun7=lambda y:'vowel' if y in "aeiou" else 'not vowel'  
res2=fun7('b')  
print(res2)
```

not vowel

In [145]:

```
str15='hello python'  
print('h'in str15)  
print(' 'in str15)
```

True

True

In [153]:

```
#map and filter  
num=[1,2,3,4,5,6,7,8,9,10] #print the output in such a way that it should be multiplicati  
mul_num=set(map(lambda x:x*2,num)) #map will take 2 parameter 1:function 2: iterable  
print(mul_num)
```

{2, 4, 6, 8, 10, 12, 14, 16, 18, 20}

In [155]:

```
num=[1,2,3,4,5,6,7,8,9,10]  
even_num=list(filter(lambda x:x%2==0,num)) #filter will take 2 parameter 1:function(cond  
print(even_num)
```

[2, 4, 6, 8, 10]

In [156]:

```
x=[1,2,3,4,5]
def fun6(x):
    return x+1

res=list(map(fun6,x))
print(res)
```

[2, 3, 4, 5, 6]

In [157]:

```
def fun7(x):
    return x**2

res=list(map(fun7,x))
print(res)
```

[1, 4, 9, 16, 25]

In [158]:

```
res=list(map(lambda x:x**2,x)) #
print(res)
```

[1, 4, 9, 16, 25]

In []:

```
addition=lambda *args:sum(args)
res=addition(10,20,30,40,50)
print(res)
```

Packages and module

In [161]:

```
#Import modulename
#import module name as alias name
#import module name with specific code
#import module with *
```

In [165]:

```
from datetime import date,datetime
#from one import add

today_date=date.today()
print(today_date)
now_time=datetime.now()
print(now_time)
```

2023-07-25

2023-07-25 16:23:49.942593

In [168]:

```
from datetime import date,datetime,timedelta
current_date = datetime.now()
yesterday = current_date - timedelta(days=1)
yesterday
```

Out[168]:

```
datetime.datetime(2023, 7, 24, 16, 28, 37, 514424)
```

break continue & pass

In [166]:

```
for i in range(10):
    if i==7:          #condtion is true it will break the Loop
        break
    print(i,end=" ")
```

0 1 2 3 4 5 6

In [167]:

```
for i in range(10):
    if i==7:
        continue      #condition will met it skip again loop will continue
    print(i,end=" ")
```

0 1 2 3 4 5 6 8 9

In [171]:

```
def my_fun():
    pass
    print('hello')
```

my_fun()

hello

In [172]:

```
def my_fun():
    pass
```

Local Variable & Global Variable

In [173]:

```
def m1(a):          #local variable which declared inside of the function
    p=50            #local variable

m1(10)
```

In [174]:

```
x=40      #global variable variable --outside of the function
y=35      #global variable
def m1(a): #local
    p=50   #local

m1(10)
```

In [175]:

```
#scope
#local variable: inside the function only
#global variable :scope is evreywhere
```

In [181]:

```
def f1():
    a=10
    print(a)

def f2():
    b=20
    print(b)

f1()
f2()
print(b)
```

```
10
20
20
```

In [182]:

```
x=10      #global
def f1():
    x=50   #local

print(x)
```

```
10
```

In [183]:

```
x=10      #global
def f1():
    x=50   #local
    print(x)

f1()
```

```
50
```

In []: