build in functions

In [1]:
```
#function is a block of code written one and can use multiple time
#by calling it

max(12,11,1,65)
```

Out[1]:
```
65
```

In [6]:
```
l=(22,34,23,54,55,66,4,312,3,45,6)
```

In [7]:
```
len(l)
```

Out[7]:
```
11
```

In [ ]:
```

```

user defined functions

In [2]:
```
# defined or created by user
#ypu use it multiple time after writing code one time
```

In [3]:
```
#def keyword (define)
```

In [5]:
```
def add():
    a=int(input("enter 1st num:"))
    b=int(input("enter 2nd num:"))
    print("addition is: ",a+b)

add()
```

```
enter 1st num:2
enter 2nd num:3
addition is:  5
```

In [8]:
```
add()
```

```
enter 1st num:4
enter 2nd num:34
addition is:  38
```

In [ ]:
```
def evenodd():
    a=int(input("enter number"))
    if a%2==0:
        print("Even")
    else:
        print("odd")

evenodd()
```

Function even name

In [2]:
```
def name():
    a=(input("enter first name:"))
    b=(input("enter last name:"))
    print("name: ",a," ",b)

name()
```

```
enter first name:gayu
enter last name:pawar
name:  gayu   pawar
```

with arguments and without return

In [6]:
```
#fixed lenght argument

def addition(x,y):
    print(x+y)

addition(5,6)
```

```
11
```

In [7]:
```
addition(y=4,x=29)
```

```
33
```

In [11]:
```
def flist(l):
    print(len(l))
flist([2,34,5,3,2])
```

```
5
```

In [17]:
```
#waf to find length of tuple and max of list
def lenmax(tup,maxlist):
    print(len(tup),max(maxlist))

lenmax((12,2,1,4),[23,55,666,753,43])
```

```
4 753
```

In [18]:
```
#variable length argument
```

In [19]:
```
def add(*x):
    for i in x:
        print(i)
add(12,3,42,34)
```

```
12
3
42
34
```

In [23]:
```
def add(*x):
        print(x)
add(12,3,42,34)
```

```
(12, 3, 42, 34)
```

In [35]:
```
# "*" is variable length tuple
# "**"
def summation(*x):

    sum=0
    for i in x:
        sum=sum+i
    print(sum)
summation(12,3,42,34)
```

```
91
```

In [36]:
```
#default argument
```

In [38]:
```
def ad(x,y=19):
    print(x+y)
ad(3)
ad(y=23,x=4)
```

```
22
27
```

In [39]:
```
#variable length key value pair


```

In [45]:
```
def addd(**x):
    print("hello")
addd()
addd(x=12,y=13,z=44)
```

```
hello
hello
```

In [25]:
```
def key(**x):
    print(x.keys())
    print(x.values())
    print(x.items())
    print(type(x))
key(a=12,n=3,b=13)
```

```
dict_keys(['a', 'n', 'b'])
dict_values([12, 3, 13])
dict_items([('a', 12), ('n', 3), ('b', 13)])
<class 'dict'>
```

In [41]:
```
def fact(n):
    mul=1
    for i in range(1,n+1):
        mul=mul*i
    print("factorial of" , n,"is",mul)
fact(34)
fact(4)
```

```
factorial of 34 is 295232799039604140847618609643520000000
factorial of 4 is 24
```

return statement

In [43]:
```
#after return statement rest of program does not get executed
#return returns only single value(number)
#return is only use in function


def add(x,y):
    return x+y
add(3,4)
```

Out[43]: 7

In [45]:
```
def add(x,y):
    print(x)
    return x+y
    return x

add(3,4)
```

```
3
```

Out[45]: 7

In [62]:
```
def add(x,y):
    return x+y
def sub(x,y):
    return x-y
def mul(x,y):
    return x*y
def div(x,y):
    return x/y
```

In [58]:
```
'''
    print("select the operation u want . 1.addition,"
              "2.substraction,3.multiplication ,4. division")

    if type(a,b)==int:
        if choice in ("1","2","3","4"):
            a=


            print("Addition is: "add(a,b))

    '''
```

Out[58]:
```
'\n    print("select the operation u want . 1.addition,"\n
"2.substraction,3.multiplication ,4. division")\n    \n    if type(a,b)==i
nt:\n        if choice in ("1","2","3","4"):\n            a=\n        \n
\n            print("Addition is: "add(a,b))\n        \n    '
```

In [73]:
```
def calculator(x,y):
    print("Welcome")

    if type(x)==int and type(y)==int:
        print("select the operation you want to perform ")
        print('''
        1.add
        2.sub
        3.mul
        4.div
        ''')
        choice = int(input("enter a choice: "))

        if choice in range(1,5):
                if choice==1:
                    print("addition is",x+y)
                elif choice==2:
                    print("Substraction is" ,x-y)
                elif choice ==3:
                    print("multiplication is" ,x*y)
                else:
                    print("division is",x/y)
        else:
            print("enter valid choice between(1 to 4)")

    else:
        print("invalid type")


calculator(4,3)
```

```
Welcome
select the operation you want to perform

        1.add
        2.sub
        3.mul
        4.div

enter a choice: 2
Substraction is 1
```

In [60]:
```
calculator(4,2)
```

```
9
```

In [83]:
```


a=input("Enter string: ")
if a[::1]==a[::-1]:
    print(a, "is a palindrome string")
else:
    print(a, "is not a palindrome string")
```

```
Enter string: nitin
nitin is a palindrome string
```

In [84]:
```
#using function pallidrom program

def rev():
    a=input("Enter string: ")
    if a[::1]==a[::-1]:
        print(a, "is a palindrome string")
    else:
        print(a, "is not a palindrome string")
rev()
```

```
Enter string: nitin
nitin is a palindrome string
```

In [87]:
```
a=input("Enter string 1: ")
b=input("Enter string 2: ")
if a[0]==b[0]:
    print("True")
else:
    print("false")
```

```
Enter string 1: mon
Enter string 2: mon
True
```

In [97]:
```
def ab():
    a=input("Enter string of two name: ")
    l=a.split(" ")
    if l[0][0]==l[1][0]:
        print("True")
    else:
        print("false")
ab()
```

```
Enter string of two name: hay bay
false
```

In [ ]:
```

```

In [93]:
```
a=input("Enter string of two name: ")
l=a.split(" ")
print(l)
```

```
Enter string of two name: hay jay
['hay', 'jay']
```

In [ ]:
```
a= input("enter a to z :")
```

```
# 1.anonymous or lamba function -- no name-- : when u have to give
    function as an argumant to another function. it is single line
    function .execute fast
2.Recurssive function : which calls itself
```

lambda function

In [1]:
```
x=lambda i:i**2
print(x(2))
```

```
4
```

In [2]:
```
x(4)
```

Out[2]:
```
16
```

In [7]:
```
add=lambda x,y:x+y
print(add(3,4))
```

```
7
```

In [8]:
```
mul=lambda a,b,c:a*b*c
print(mul(2,3,4))
```

```
24
```

In [13]:
```
cube=lambda c:c**3
print(cube(3))
```

```
27
```

Type *Markdown* and LaTeX: α^2

In [15]:
```
g=lambda a:a>100
print(g(102))
```

```
True
```

In [16]:
```
leng=lambda i:len(i)
print(leng("gayu"))
```

```
4
```

In [17]:
```
even=lambda a:a%2==0
print(even(4))
```

```
True
```

```
lambda function use for
1.filter
2.map
```

In [18]:
```
l=[-1,2,3,-44,2]
positive=list(filter(lambda x:x>0,l))
print(positive)
```

```
[2, 3, 2]
```

In [19]:
```
a=[12,34,21,11,22]
b=["a","b",5,6,4,"c"]
```

In [21]:
```
eo=list(filter(lambda x:x%2==0,a))
odd=list(filter(lambda x:x%2!=0,a))
print(eo)
print(odd)
```

```
[12, 34, 22]
[21, 11]
```

In [27]:
```
string=list(filter(lambda x:type(x)==str,b))
print(string)
integer=list(filter(lambda x:type(x)==int,b))
print(integer)
```

```
['a', 'b', 'c']
[5, 6, 4]
```

In [9]:
```
l=[10,10,10,20,20,30]
#o/p {10:3,20:2,30:1}
d={}
output={element: l.count(element) for element in set(l)}
print(output)
```

```
{10: 3, 20: 2, 30: 1}
```

Exception handling

In [10]:
```
a=int(input("enter first number : "))
b=int(input("enter second number : "))
c=a/b
print(c)
```

```
enter first number : 6
enter second number : 7
0.8571428571428571
```

```
value error:
zerodivisionerror:
```

```
execption is only use in runtime error
When you try to handle error i runtime is known as exception handling
```

In [15]:
```
try:
    a=int(input("enter first number : "))
    b=int(input("enter second number : "))
    c=a/b
    print(c)
except ValueError:
    print("please enter integer number!!!")
except ZeroDivisionError:
    print("Please enter another number except 0 !!")
```

```
enter first number : sux
please enter integer number!!!
```

scope of variable

global scope

In [17]:
```
x=20
def abc():
    print(x)
abc()
```

```
20
```

local scope

In [19]:
```
def xyz():
    y=20
    print(y)
xyz()
print(y)
```

```
20

---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
Cell In[19], line 5
      3     print(y)
      4 xyz()
----> 5 print(y)

NameError: name 'y' is not defined
```

In [ ]:
```

```