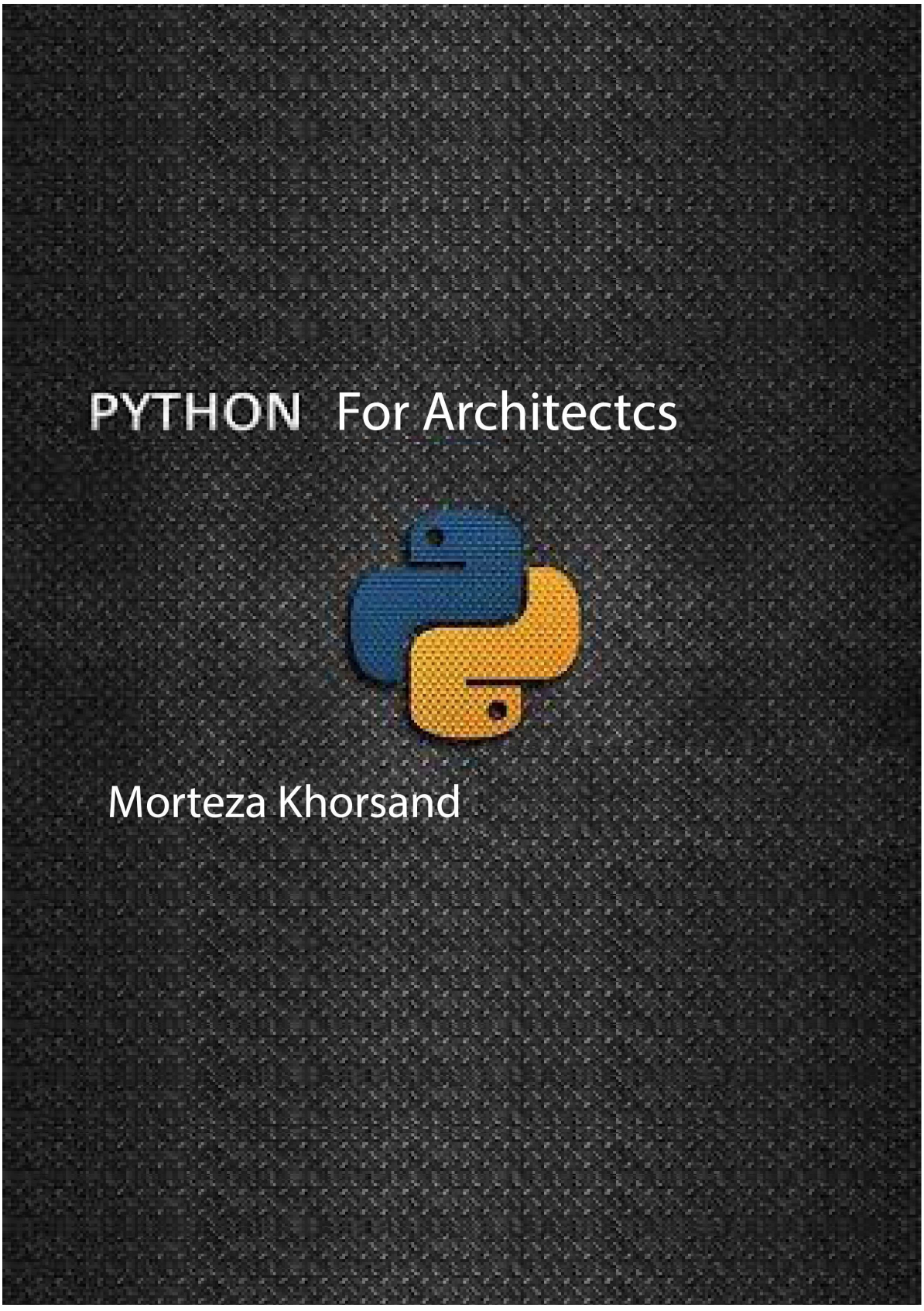


In [ ]:

In [ ]:

In [ ]:

variables

int

float

string

In [1]:
```
a= 12
```

In [2]:
```
b=12.3
```

In [3]:
```
c="morteza"
```

In [4]:
```
my_name = "morteza"       #snake case
```

In [5]:
```
myName = "ali"            # camel case
```

In [6]:
```
myname= "fereshteh"       #lower case
```

In [7]:
```
MyName= "reza"            #upper camel case
```

In [8]:
```
"""
1. start with alphabet

2. do not start with number

3. % , $ , * , & is not acceptable

4. keyword

""";
```

Python keywords

In [ ]:
```
and         A logical operator

as              To create an alias

assert      For debugging

break       To break out of a loop

class       To define a class

continue        To continue to the next iteration of a loop

def             To define a function
```

```
del             To delete an object
elif        Used in conditional statements, same as else if
else        Used in conditional statements
except      Used with exceptions, what to do when an exception occurs
False       Boolean value, result of comparison operations
finally     Used with exceptions, a block of code that will be executed no matter if t
for             To create a for loop
from        To import specific parts of a module
global      To declare a global variable
if              To make a conditional statement
import      To import a module
in              To check if a value is present in a list, tuple, etc.
is              To test if two variables are equal
lambda      To create an anonymous function
None        Represents a null value
nonlocal        To declare a non-local variable
not             A logical operator
or              A logical operator
pass        A null statement, a statement that will do nothing
raise       To raise an exception
return      To exit a function and return a value
True        Boolean value, result of comparison operations
try             To make a try...except statement
while       To create a while loop
with        Used to simplify exception handling
yield       To end a function, returns a generator
```

Arithmetic Operation

addition (+), subtraction (-), division (/), multiplication (*) , power , quotient , reminder

In [9]:
```
num_1 = 2
num_2=5.
num_3=4.
```

In [10]:
```
add= num_1 + num_2
print(add)
```

```
7.0
```

In [11]:
```
mul=num_3 * num_1
print(mul)
```

```
8.0
```

In [12]:
```
quo=num_2 // num_1
print(quo)
```

```
2.0
```

In [13]:
```
re=num_3 % 1.5
print(re)
```

```
1.0
```

In [14]:
```
po=num_1**2
print(po)
```

```
4
```

In [15]:
```
var_1= (num_1 * num_2)**2+ num_3

var_2=num_1 * (num_2+ num_3)

print(var_1 , var_2)
```

```
104.0 18.0
```

In [16]:
```
str_1 = "morteza"
str_2 = "ali"
str_3= str_1+str_2
print(str_3)
```

```
mortezaali
```

In [17]:
```
str_1 *2
```

Out[17]:
```
'mortezamorteza'
```

In [18]:
```
x = 12
x=13
```

In [19]:
```
x=x+7
x+=7
print(x)
```

```
27
```

comparison operator

> , < , == , !=

In [20]:
```
number_1= 60
number_2= 30
```

In [21]:
```
number_1 >= number_2
```

Out[21]:
```
True
```

In [22]:
```
number_1<= number_2
```

Out[22]:
```
False
```

In [23]:
```
number_1==number_2
```

Out[23]:
```
False
```

In [24]:
```
number_1!= 10
```

Out[24]:
```
True
```

Data Structures

1. string
2. list
3. tuple
4. dictionary
5. set

In [25]:
```
seq_1= "python"
seq_2= "for architects"
```

In [26]:
```
seq_1 + seq_2
```

Out[26]:
```
'pythonfor architects'
```

In [27]:
```
seq_3 = "introduction to python for architets"
s_7=seq_3[:]
s_7
```

Out[27]:
```
'introduction to python for architets'
```

In [28]:
```
s_3= seq_3[16]
print(s_3)
```

p

In [29]:
```
s_4= seq_3[1:8]
print(s_4)
```

ntroduc

In [30]:
```
s_5= seq_3[1:15:2]
s_5
```

Out[30]: 'nrdcint'

In [ ]:
```
s_6=seq_3[-1 : 5: -1 ]
```

exercise

seq_4 = "introduction to java for architects"

methods for string

In [31]:
```
seq_3 = "introduction to python for architects"
```

In [32]:
```
seq_4=[len(seq_3)-1]
```

In [33]:
```
seq_3.upper()              #convert string to uppercase
print(a)
```

12

In [34]:
```
seq_3.lower()              #convert string to lowercase
```

Out[34]: 'introduction to python for architects'

In [35]:
```
seq_3.count("o")        #returns the number of times a specific value occurs in a stri
```

Out[35]: 5

In [36]:
```
seq_3.endswith("arcHitects")   #returns true if the string ends with especific value c
```

Out[36]: False

In [37]:
```
seq_3.find("t",4)      #searching the string for a specific value and returns the posi
```

Out[37]: 8

In [38]:
```
seq_3.index("t" ,5)
```

Out[38]: 8

In [39]:
```
seq_3.replace("t" , "H" )   #replace t with H
```

Out[39]: 'inHroducHion Ho pyHhon for archiHecHs'

In [40]:
```
seq_3.split(sep="o")
```

Out[40]:
```
['intr', 'ducti', 'n t', ' pyth', 'n f', 'r architects']
```

In [41]:
```
seq_3.split(sep="t")
```

Out[41]:
```
['in', 'roduc', 'ion ', 'o py', 'hon for archi', 'ec', 's']
```

In [42]:
```
seq_3.split(sep= "t" , maxsplit= 2)
```

Out[42]:
```
['in', 'roduc', 'ion to python for architects']
```

In [43]:
```
seq_3.title()    #converts the first character of each word to uppercase
```

Out[43]:
```
'Introduction To Python For Architects'
```

In [1]:
```
print(int(input("please enter your number: ")))
```

```
please enter your number: 10
10
```

In [2]:
```
#triangle squre

base= float(input("the base is: "))
hight= float(input("the hight is: "))
squre= base* hight /2
print("the squre is :", squre)
```

```
the base is: 3
the hight is: 5
the squre is : 7.5
```

In [3]:
```
#1)  sylinder volume
# v = pi r**2 h

#2) z= x**3 + 2x**2 + 3y - 5
```

list

In [4]:
```
number_list=[1,10,45,89,75,46]
name_list=["mahsa", "nika", "tomaj" , "sarina" ]
```

In [5]:
```
list_1= ["python" , 12. , 0 , "#" , "introduction to python for architects", "grasshop
```

In [6]:
```
list_1[0]
```

Out[6]:
```
'python'
```

In [7]:
```
list_1[: 3]
```

Out[7]:
```
['python', 12.0, 0]
```

In [8]:
```
list_1[:]
```

Out[8]:
```
['python',
 12.0,
 0,
 '#',
 'introduction to python for architects',
 'grasshopper']
```

In [9]:
```
list_1[: : 2]
```

Out[9]:
```
['python', 0, 'introduction to python for architects']
```

In [10]:
```
list_1[-1 :  : -1]
```

Out[10]:
```
['grasshopper',
 'introduction to python for architects',
 '#',
 0,
 12.0,
 'python']
```

In [11]:
```
list_1[0][1]
```

Out[11]:
```
'y'
```

In [12]:
```
list_1[len(list_1)-1]
```

Out[12]:
```
'grasshopper'
```

Nested list

In [13]:
```
buildings = ["taghbostan" ,["hafezieh", "takhtejamshid"] , ["azadi tower", "milad towe
```

In [14]:
```
buildings[1]
```

Out[14]:
```
['hafezieh', 'takhtejamshid']
```

In [15]:
```
buildings[1][1]
```

Out[15]:
```
'takhtejamshid'
```

In [16]:
```
buildings[1][1][1]
```

Out[16]:
```
'a'
```

updates in lists

In [17]:
```
scores= [1,20,19,18,14,10,5,6,19,20,17,15,14,12,13,16,14]
```

In [18]:
```
scores[0]= 18
scores
```

Out[18]:
```
[18, 20, 19, 18, 14, 10, 5, 6, 19, 20, 17, 15, 14, 12, 13, 16, 14]
```

In [19]:
```
scores[1:3] = 14,15
scores
```

Out[19]:
```
[18, 14, 15, 18, 14, 10, 5, 6, 19, 20, 17, 15, 14, 12, 13, 16, 14]
```

In [20]:
```
scores[1:3]=[]
scores
```

Out[20]:
```
[18, 18, 14, 10, 5, 6, 19, 20, 17, 15, 14, 12, 13, 16, 14]
```

In [21]:
```
scores[:] = []
scores
```

Out[21]:
```
[]
```

In [22]:
```
scores[1]=[]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_1580\3083942772.py in <module>
----> 1 scores[1]=[]

IndexError: list assignment index out of range
```

In [23]:
```
numbers_1 = [1,2,3]
numbers_2= [4,5,6]
numbers_1 + numbers_2
```

Out[23]:
```
[1, 2, 3, 4, 5, 6]
```

In [24]:
```
#numbers_1 + 2

print(type(numbers_1))
```

```
<class 'list'>
```

In [25]:
```
numbers_1 *2
```

Out[25]:
```
[1, 2, 3, 1, 2, 3]
```

list methods

In [26]:
```
student_scores=[12,15,18,17,19,15,14,16,20,8]
student_scores.reverse()
student_scores
```

Out[26]:
```
[8, 20, 16, 14, 15, 19, 17, 18, 15, 12]
```

In [27]:
```
student_scores.append(0)
student_scores
```

Out[27]:
```
[8, 20, 16, 14, 15, 19, 17, 18, 15, 12, 0]
```

In [28]:
```
student_scores.remove(8)
student_scores
```

Out[28]:
```
[20, 16, 14, 15, 19, 17, 18, 15, 12, 0]
```

In [29]:
```
a=student_scores.index(20)
a
```

Out[29]:
```
0
```

In [30]:
```
student_scores.extend([12 , 13])        #student_scores+=[12 , 13]
student_scores
```

Out[30]:
```
[20, 16, 14, 15, 19, 17, 18, 15, 12, 0, 12, 13]
```

In [31]:
```
student_scores.sort()
student_scores
```

Out[31]:
```
[0, 12, 12, 13, 14, 15, 15, 16, 17, 18, 19, 20]
```

In [32]:
```
student_scores.sort(reverse= True)         #key= len
student_scores
```

Out[32]:
```
[20, 19, 18, 17, 16, 15, 15, 14, 13, 12, 12, 0]
```

In [33]:
```
list_name= ["python" , "java", "kotlin", "csharp"]
list_name.sort(reverse= True)
list_name
```

Out[33]:
```
['python', 'kotlin', 'java', 'csharp']
```

In [34]:
```
a=sorted(student_scores, reverse= True)
a
```

Out[34]:
```
[20, 19, 18, 17, 16, 15, 15, 14, 13, 12, 12, 0]
```

In [35]:
```
student_scores.insert(0 , 14)    #insert 14 in 0 index
student_scores
```

Out[35]:
```
[14, 20, 19, 18, 17, 16, 15, 15, 14, 13, 12, 12, 0]
```

In [36]:
```
student_scores.pop()     #remove    #pop has return

student_scores
```

Out[36]:
```
[14, 20, 19, 18, 17, 16, 15, 15, 14, 13, 12, 12]
```

In [37]:
```
student_scores.copy()       #in python3
```

Out[37]:
```
[14, 20, 19, 18, 17, 16, 15, 15, 14, 13, 12, 12]
```

In [38]:
```
s= sum(student_scores)       #function

print(" sum of the scores are:",s)
```

```
 sum of the scores are: 185
```

In [39]:
```
ma= max(student_scores)
mi=min(student_scores)
mi
```

Out[39]:
```
12
```

In [40]:
```
#sort from biggest to bigest shortest
list_name= ["python" , "java","JavaScript", "c_sharp" , "C++", "SQL","MATLAB","Kotlin"

length=[]

length.append(len(list_name[0]))  , length.append(len(list_name[1]))
length.append(len(list_name[2]))  , length.append(len(list_name[3]))
length.append(len(list_name[4]))  , length.append(len(list_name[5]))
length.append(len(list_name[6]))  , length.append(len(list_name[7]))
length.sort(reverse= True)
length
```

Out[40]:
```
[10, 7, 6, 6, 6, 4, 3, 3]
```

exercise

specify a list of numbers and shows the reminder of the Summation to 7

specify a list of numbers and calculate the sum of maximum and minimum numbers and append it an empty list

specify a list and shows the second minimum of it

In [41]:
```
empty_list=[]
list_number=[15,10,2,48,97,63,25,10]
empty_list.append(max(list_number)+ min(list_number))
empty_list
```

Out[41]:
```
[99]
```

In [42]:
```
list_number=[15,10,2,48,97,63,25,10]
sum(list_number)%7
```

Out[42]:
```
4
```

for loop

for variable in iterator: statement

In [43]:
```
"""
sequ= "introduction to python for architects"
for i in sequ:
    print(i.upper())
"""
```

Out[43]:
```
'\nsequ= "introduction to python for architects"\nfor i in sequ:\n    print(i.upper
())\n'
```

In [44]:
```
list_num = [45,67,25,41,10,36,15,47,89,45,74,14,15,25]

for num in list_num:
    print(num*2)
```

```
90
134
50
82
20
72
30
94
178
90
148
28
30
50
```

In [45]:
```
empty_list=[]
for j in list_num:
    empty_list.append(j%2)

empty_list
```

Out[45]:
```
[1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1]
```

In [46]:
```
#sort from biggest to shortest
list_name= ["python" , "java","JavaScript", "c_sharp" , "C++", "SQL","MATLAB","Kotlin"

length=[]

for char in list_name:
    length.append(len(char))

length.sort(reverse= True)


print(length)
```

```
[10, 7, 6, 6, 6, 4, 3, 3]
```

range

range(start , end , step)

In [47]:
```
list_2=[]
for x in range(0,10):
    list_2.append(x**2)
list_2
```

Out[47]:
```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

In [48]:
```
#list_num = [45,67,25,41,10,36,15,47,89,45,74,14,15,25]
```

In [49]:
```
for i in range(10):
    a=i**2
a
```

Out[49]:
```
81
```

In [50]:
```
numbers=[]
for i in range(5):
    for j in range(5):
        numbers.append(i + j)
numbers
```

Out[50]:
```
[0, 1, 2, 3, 4, 1, 2, 3, 4, 5, 2, 3, 4, 5, 6, 3, 4, 5, 6, 7, 4, 5, 6, 7, 8]
```

In [51]:
```
list_total=[]
list_1 = [1,2,3]
list_2 = [4,5,6]

for x in list_1:
    for y in list_2:
        list_total.append(x+y)
list_total
```

Out[51]:
```
[5, 6, 7, 6, 7, 8, 7, 8, 9]
```

In [53]:
```
string= "grasshopper and rhino"

list_st=[]
for i in range(len(string)):
    list_st.append(string[i])
print(list_st)
```

```
['g', 'r', 'a', 's', 's', 'h', 'o', 'p', 'p', 'e', 'r', ' ', 'a', 'n', 'd', ' ', 'r',
'h', 'i', 'n', 'o']
```

In [ ]:
```
plain_text=input("enter the text: ")
encrypted_text = " "
for i in plain_text:
#The ord() function returns the number representing the unicode code of a specified ch
    x=ord(i)*2+5
#The chr() method converts an integer to its unicode character and returns it
    encrypted_text += chr(x)
print("encrypted_text: ", encrypted_text)
```

In [1]:
```
a = ['x','y','z']
a.clear()
#del a[1]

a
```

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

In [2]:
```
for i in range(10):
    pass
```

Conditional statement

if + condition : statement

In [3]:
```
num = 20
if num > 0 :
    print("the number is positive" , num)
```

```
the number is positive 20
```

In [ ]:
```
number= int(input("enter the number: "))
if number%2==0:
    display("the number is even" )           #display~ print
```

In [1]:
```
#f string
```

In [1]:
```
weight=float(input("your wight"))
if weight >= 90:
    print(f" your wight {weight} is overwight")
if weight <= 60:
    print(f" your wight {weight} is underwight")
if  60<weight<90:
    print(f" your wight {weight} is normal")
```

```
your wight80
 your wight 80.0 is normal
```

In [232…
```
weight=float(input("your wight"))
if weight >= 90:
    print("overwight")
elif weight <= 60:
    print("underwight")
elif  60<weight<90:
    print("normal")
```

```
your wight60
underwight
```

In [ ]:
```
score= float(input("enter your score"))
if score >=18 :
    print(f" your number {score} is A")
elif score >=16:
    print(f" your number {score} is B")
elif score >=14:
    print(f" your number {score} is C")
else:
                                  #else
    print(f" your number {score} is D")
```

In [ ]:
```
number= int(input("enter number: "))
if number %2 ==0:
    print("the number is even")
elif number %2!=0 :
    print("the number is odd")
```

In [ ]:
```
number= int(input("enter number: "))
if number %2 ==0:
    print("the number is even")
else:
    print("the number is odd")
```

In [ ]:
```
#even number smaller than 100

n= int(input("your number: "))
if n <= 100:
    if n%2==0:
        print(f" the number {n} is correct")
```

In [ ]:
```
#exercise
##even number smaller than 100 and coefficient of 3
```

In [ ]:
```
hight= float(input(" your hight(m) :  "))
weight= float(input(" your weight(kg) :  "))
BMI= weight / hight**2
if BMI < 18.5:
    print( f"your BMI is {BMI} and you are Underweight, \U0001F974 " )

elif 18.5<= BMI <=24.9:
    print( f"your BMI is {BMI} and you are normal, \U0001F44D ")
elif  25 <BMI<= 29.9:
    print( f"your BMI is {BMI} and you are Overweight, \U0001F627")
else:
    print( f"your BMI is {BMI} and you are obess, \U0001F631")
```

In [ ]:
```
r="rock"
p="paper"
s="scissors"
player_1= input(f"{r} or {p} or {s}:  ")
player_2= input(f"{r} or {p} or {s}:  ")
if player_1 == r and player_2==p:
    print("player_2 won")
elif player_1==r and player_2==s:
    print("player_1 won")
elif player_1 ==player_2:
    print("that is a tie")
elif player_1==p and player_2==r:
    print("player_1 won")
elif player_1==p and player_2==s:
    print("player_2 won")
elif player_1==s and player_2==r:
    print("player_2 won")
elif player_1==s and player_2==p:
    print("player_1 won")
else:
    print("something is wrong")
```

In [4]:
```
if 1:
    print(2)

if not False:
    print(5)
```

```
2
5
```

In [6]:
```
var= int(input("number"))

if var:
    print(var*2)
if not var:
    print("not acceptable")
```

```
number20
40
```

In [7]:
```
even_nembers=[]
odd_numbers=[]
for i in range(10):
    if i%2:
        odd_numbers.append(i)
    elif not i%2:
        even_nembers.append(i)


print(even_nembers)
print(odd_numbers)
```

```
[0, 2, 4, 6, 8]
[1, 3, 5, 7, 9]
```

In [8]:
```
#how many 10 exist in the list use for

list_number_1=[12,15,14,17,10,78,98,96,85,74,12,10,14,78,54,123
               ,45,10,41,12,36,98,74,10,15,14,17,89,
                12, 10,45,78,10,98,639,47,10,78,10,78,96,85]
counter=0
for num in list_number_1:
    if num==10:
        counter+=1
counter
```

Out[8]:
```
8
```

break and continue

In [9]:
```
for i in range(1,11):
    if i == 5:
        break
    print(i)
```

```
1
2
3
4
```

In [10]:
```
for i in range(1,11):
    if i == 5:
        continue
    print(i)
```

```
"""
when the condition is met, that iteration is skipped.
But we do not exit the loop. Hence,
all the values except 5 are printed out.
"""
```

```
1
2
3
4
6
7
8
9
10
```

Out[10]:
```
'    \nwhen the condition is met, that iteration is skipped.\nBut we do not exit the
loop. Hence, \nall the values except 5 are printed out.\n'
```

Logical Operators

Operator	Example	Meaning
and	a and b	Logical AND: True only if both the operands are True
or	a or b	Logical OR: True if at least one of the operands is True
not	not a	Logical NOT: True if the operand is False and vice-versa.

In [12]:
```
number=int(input(" number"))

if  number >3  and number%2==0:
    print(True)
else:
    print(False)
```

```
 number20
True
```

In [13]:
```
x=10
y=15
if (x>7 and y >8):
    print("thats true")
if (x>14 or y>14):
    print("accepted")
```

```
thats true
accepted
```

In [14]:
```
number= int(input("enter your number"))
if (number%2==0 and number<100 and number%3==0):
    print(f"the number {number} has all the condition")
```

```
enter your number35
```

In [15]:
```
list_number=[12,45,78,100,25,13,10,78,95]
factor=[]
for num in list_number:
    if num%2==0 or num%3==0:
        factor.append(num)

factor
```

Out[15]:
```
[12, 45, 78, 100, 10, 78]
```

In [16]:
```
e_list=[]
for num in list_number:
    if not num%2==0:
        e_list.append(num)
e_list
```

Out[16]:
```
[45, 25, 13, 95]
```

In [17]:
```
scored1=[]
scored2=[]
x=float(input("enter the number"))
#x<0 or x>20
if x<10:
    scored1.append(x)
    print(f"you failed and your score is {scored1}")
else:
    scored2.append(x)
    print(f"you passed and your score is{scored2}")
```

```
enter the number5
you failed and your score is [5.0]
```

In [18]:
```
list1=[5,20,15,20,25,50,20]
for num in list1:
    if num==20:
        list1.remove(num)

list1
```

Out[18]:
```
[5, 15, 25, 50]
```

Membership operators

In [19]:
```
a = [1, 2, 3, 4, 5]
print(5 in a)

10 in a
```

Out[19]:
```
True
False
```

In [20]:
```
message = 'Hello world'
# check if 'H' is present in message string
```

```
print('H' in message)
# check if 'python' is present in message string
print('python' not in  message)
```

```
True
True
```

Identity operators

In [21]:
```
x1 = 5
y1 = 5
x2 = 'Hello'
y2 = 'Hello'
x3 = [1,2,3]
y3 = [1,2,3]

print(x1 is not y1)  # prints False

print(x2 is y2)  # prints True

print(x3 is y3)  # prints False
```

```
False
True
False
```

from, import

In [22]:
```
import math
import random

from math import sin, cos , pi
from random import randint
```

random numbers

In [23]:
```
from random import random ,seed    #real numbers  between 0 , 1

seed(5)
print(random()*10)
#seed(2)
#print(random())
```

```
6.229016948897019
```

In [24]:
```
random_numbers=[]

seed(3)
for _ in range(10):
    random_numbers.append(random())
print(random_numbers )
```

```
[0.23796462709189137, 0.5442292252959519, 0.36995516654807925, 0.6039200385961945, 0.
625720304108054, 0.06552885923981311, 0.013167991554874137, 0.83746908209646, 0.25935
401432800764, 0.23433096104669637]
```

In [25]:
```
from random import seed
from random import randint , seed    #interger numbers between (a, b)

random_integers=[]
seed(6)
for _ in range(10):
    random_integers.append(randint(5,9))

print(random_integers)
```

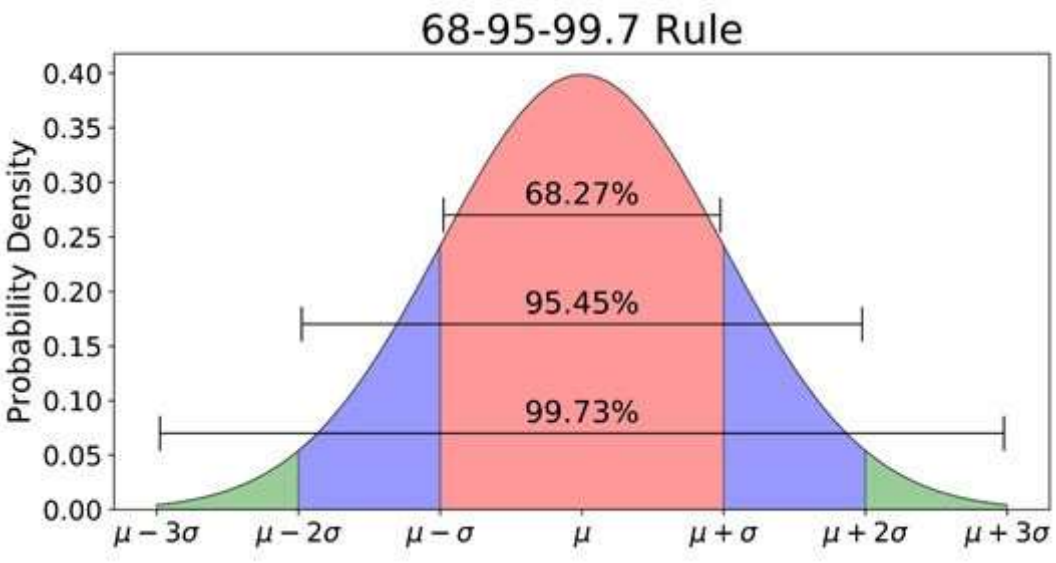
```
[9, 5, 8, 7, 5, 5, 6, 9, 8, 7]
```

In [26]:
```
from random import randrange
a=randrange(2,20,2)
a
```

Out[26]:
```
12
```

In [27]:
```
from random import uniform
uniform(2,6)
```

Out[27]:
```
5.080559343751961
```



#

In [28]:
```
import random                    #gussian randoms with mu and sigma

guss_list=[]                     #normalvariate
random.seed(5)
for _ in range(12):
    guss_list.append(random.gauss(1 , 1))
guss_list
```

Out[28]:
```
[-0.1788417512306717,
 -0.14816068079080158,
 1.6694689143859696,
 -1.293910094631911,
 0.8566161616995311,
 -1.2560772673958667,
 2.1009715963243587,
 1.2028981117525226,
 2.3563159867990544,
 0.49581743002085077,
 1.3981949121429393,
 0.7141226372189547]
```

In [29]:
```
from random import seed            #choose a sublist  from a list
from random import choice

point_list=[[0,0,0], [2,3,6], [5,4,8], [1,2,-5], [8,7,4],[1,5,6]]

seed(5)
r_list=[]
for _ in range(3):
    r_list.append(choice(point_list))

r_list
```

Out[29]:
```
[[8, 7, 4], [5, 4, 8], [1, 5, 6]]
```

In [30]:
```
from random import seed            #choose a sublist  from a list
from random import sample

point_list=[[0,0,0], [2,3,6], [5,4,8], [1,2,-5], [8,7,4],[1,5,6]]
seed(1)
print(sample(point_list,2))
```

```
[[2, 3, 6], [8, 7, 4]]
```

In [31]:
```
import random

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

random.shuffle(list_number)

list_number
```

Out[31]:
```
[8, 3, 1, 9, 6, 7, 4, 10, 5, 2]
```

In [1]:
```
import random
coumputer_choice = random.randint(0,2)
if coumputer_choice==0:
    computer_choice="rock"
elif coumputer_choice==1:
    computer_choice="paper"
elif computer_choice==2:
    computer_choice="scissors"
r="rock"
p="paper"
s="scissors"
player_name=input("enter your name to start the game:")
player_1=input(f"{player_name}  {r} or {p} or {s}: ")
```

```
if player_1==r and computer_choice=="paper" :
    print(f"computer choice is {computer_choice} and computer won")
elif player_1==r and computer_choice =="scissors":
    print(f"computer choice is {computer_choice} and {player_name} won")
elif player_1 ==computer_choice:
    print(f"computer choice is {computer_choice} and thats a tie")
elif player_1==p and computer_choice==r:
    print(f"computer choice is {computer_choice} and {player_name} won")
elif player_1==p and computer_choice==s:
    print(f"computer choice is {computer_choice} and computer won")
elif player_1==s and computer_choice==r:
    print(f"computer choice is {computer_choice} and computer won")
elif player_1==s and computer_choice==p:
    print(f"computer choice is{computer_choice} and {player_name} won")
else:
    print("something is wrong")
```

```
enter your name to start the game:morteza
morteza  rock or paper or scissors: rock
computer choice is paper and computer won
```

In [2]:
```
#1000 random numbers and specify how many 6 in it

from random import randrange
counter=0
for i in range(10000):
    a=randrange(1,7)
    if a%6==0:
        counter+=1
print(counter)
```

```
1567
```

In [3]:
```
#1M random numbers and specify how many tails a and how many head
heads=0
tails=0
from random import randint

for i in range(100):
    num= randint(1,2)
    if num%2==0:
        heads+=1
    else:
        tails+=1
print(heads,"is: " , tails, "is: " , end=" ")
```

```
65 is:  35 is:
```

Dictionaries

In [4]:
```
dic={0: "python" , 1: "grasshopper" , 2: "rhino" , 3: "machine learning"}

#key ..... value
```

In [5]:
```
list_1= ["python" , "grasshopper","rhino","machine learning" ]
```

In [6]:
```
dic[1]
```

Out[6]:
```
'grasshopper'
```

In [7]:
```
dic_2= {"car": "bmw" , "salaery":12000, "education": "architecture" }
```

In [8]:
```
dic_2["salaery"]
```

Out[8]:
```
12000
```

In [9]:
```
dic_2[1]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_12228\627571172.py in <module>
----> 1 dic_2[1]

KeyError: 1
```

In [10]:
```
print(dir(dict))
```

```
['__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__d
ir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem_
_', '__gt__', '__hash__', '__init__', '__init_subclass__', '__ior__', '__iter__', '__
le__', '__len__', '__lt__', '__ne__', '__new__', '__or__', '__reduce__', '__reduce_ex
__', '__repr__', '__reversed__', '__ror__', '__setattr__', '__setitem__', '__sizeof_
_', '__str__', '__subclasshook__', 'clear', 'copy', 'fromkeys', 'get', 'items', 'key
s', 'pop', 'popitem', 'setdefault', 'update', 'values']
```

In [11]:
```
dic_2["age"]= 30
dic_2
```

Out[11]:
```
{'car': 'bmw', 'salaery': 12000, 'education': 'architecture', 'age': 30}
```

In [12]:
```
dic_2["car"]= "pride"
dic_2
```

Out[12]:
```
{'car': 'pride', 'salaery': 12000, 'education': 'architecture', 'age': 30}
```

In [13]:
```
#update method

dic_2.update({"car":"samand"})

dic_2.update({"hight": 176})

print(dic_2)
```

```
{'car': 'samand', 'salaery': 12000, 'education': 'architecture', 'age': 30, 'hight':
176}
```

In [14]:
```
car_class = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
```

```
car_class.update({"color": "White"})
car_class
```

Out[14]:
```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'White'}
```

In [15]:
```
car_class * 2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_12228\4220569741.py in <module>
----> 1 car_class * 2

TypeError: unsupported operand type(s) for *: 'dict' and 'int'
```

In [16]:
```
"brand" in car_class
```

Out[16]:
```
True
```

In [17]:
```
"factory" in car_class
```

Out[17]:
```
False
```

In [18]:
```
"Mustang" in car_class
```

Out[18]:
```
False
```

Nested Dictionary

In [19]:
```
season={
    "Spring":["March", "April" , "May"] ,
    "Summer":["Jun","July","Augest"],
      "Fall":["September","October","November"],
    "Winter":["December","January","February"]
}

season["Fall"][1]
```

Out[19]:
```
'October'
```

Dictionary methods

In [20]:
```
print(dir(dict))
```

```
['__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__d
ir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem_
_', '__gt__', '__hash__', '__init__', '__init_subclass__', '__ior__', '__iter__', '__
le__', '__len__', '__lt__', '__ne__', '__new__', '__or__', '__reduce__', '__reduce_ex
__', '__repr__', '__reversed__', '__ror__', '__setattr__', '__setitem__', '__sizeof_
_', '__str__', '__subclasshook__', 'clear', 'copy', 'fromkeys', 'get', 'items', 'key
s', 'pop', 'popitem', 'setdefault', 'update', 'values']
```

In [21]:
```
Math_commands={0:"rs.Addpoint([0,0,0])", 1:"rs.Addcircle([0,0,0], 10)" , 2:"rs.AddPlar
```

In [22]:
```
Math_commands[0]
```

Out[22]:
```
'rs.Addpoint([0,0,0])'
```

In [23]:
```
Math_commands.pop(1)
```

Out[23]:
```
'rs.Addcircle([0,0,0], 10)'
```

In [24]:
```
Math_commands.clear()           #empty dictionary
Math_commands
```

Out[24]:
```
{}
```

In [25]:
```
Math_commands.values()
```

Out[25]:
```
dict_values([])
```

In [26]:
```
Math_commands.keys()
```

Out[26]:
```
dict_keys([])
```

In [27]:
```
Math_commands.items()
```

Out[27]:
```
dict_items([])
```

for in dictionaries

In [28]:
```
dic_3= {"name": "sara" , "age": 27, "education": "architecture" , "job":"manager"}
```

In [29]:
```
for i in dic_3:
    print(i)
```

```
name
age
education
job
```

In [30]:
```
for i in dic_3:
    print(dic_3[i])
```

```
sara
27
architecture
manager
```

In [31]:
```
for values in dic_3.values():
    print(values)
```

```
sara
27
architecture
manager
```

In [32]:
```
for keys in dic_3.keys():
    print(keys)
```

```
name
age
education
job
```

In [33]:
```
for item in dic_3.items():
    print(item)
```

```
('name', 'sara')
('age', 27)
('education', 'architecture')
('job', 'manager')
```

In [34]:
```
sample_dict={
    "name": "kelly",
    "age":25,
    "salary": 8000,
    "city": "New york"
}

#keys to remove
#keys=["name", "salary"]



y = {}
# eliminate the unrequired element
for key, value in sample_dict.items():
    if key == 'age' or key=="city":
        y[key] = value


print(y)
```

```
{'age': 25, 'city': 'New york'}
```

while

initial value while + condition: statement

In [35]:
```
num = 10
while num>0:
    print(num)
    num-=1
```

```
10
9
8
7
6
5
4
3
2
1
```

In [36]:
```
number=10
while number<=20:
    print(number*2)
    number+=1
```

```
20
22
24
26
28
30
32
34
36
38
40
```

#danger infinitive loop n=10 while n>0: print(n)

In [37]:
```
#even numbers between 1 to 50

even_numbers=[]

number=1
while number<=50:
    if number%2==0:
        even_numbers.append(number)
    number+=1

print(even_numbers)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44,
46, 48, 50]
```

In [38]:
```
#home exercise: factors of 5 between 1 to 1000
```

In [39]:
```
password= int (input ("enter your password:  "))

while password != 1234:
    print(f"the password {password} you entered is wrong")
    password=int(input ("enter your password: "))
if password==1234:
    print("successful")
```

```
enter your password:  1235
the password 1235 you entered is wrong
enter your password: 1234
successful
```

In [40]:
```
#create 20 random numbers between 0 to 100

import random

a=[]
random.seed(10)
while len(a)<20:
    rand=random.randrange(0,100)
    a.append(rand)                                          #use if

a
```

Out[40]:
```
[73, 4, 54, 61, 73, 1, 26, 59, 62, 35, 83, 20, 4, 66, 62, 41, 9, 31, 95, 46]
```

In [41]:
```
b=[]
c=0
while True:
    c+=1
    if c%6==0 and c%14==0:                                  #k.m.m
        break
    b.append(c)

print(b)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 2
4, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41]
```

In [42]:
```
import random
counter=0
while True:
    ran=random.random()+1
    counter+=ran
    if counter>=20:
        break

print(counter)
```

```
20.505253406323266
```

In [43]:
```
my_name= "morteza khorsand \n form tehran"
print(my_name)
```

```
morteza khorsand
 form tehran
```

List comprehension

In [44]:
```
number_list=[]
for i in range(10):
    number_list.append(i*2)

number_list
```

Out[44]:
```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

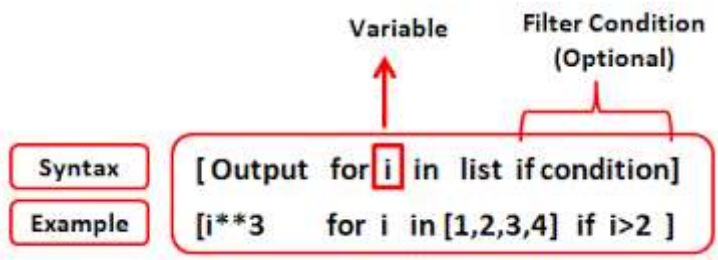
In [45]:
```
number_list2=[i*2 for i in range(10)]                       #just for append
```

```
number_list2
```

Out[45]: [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]

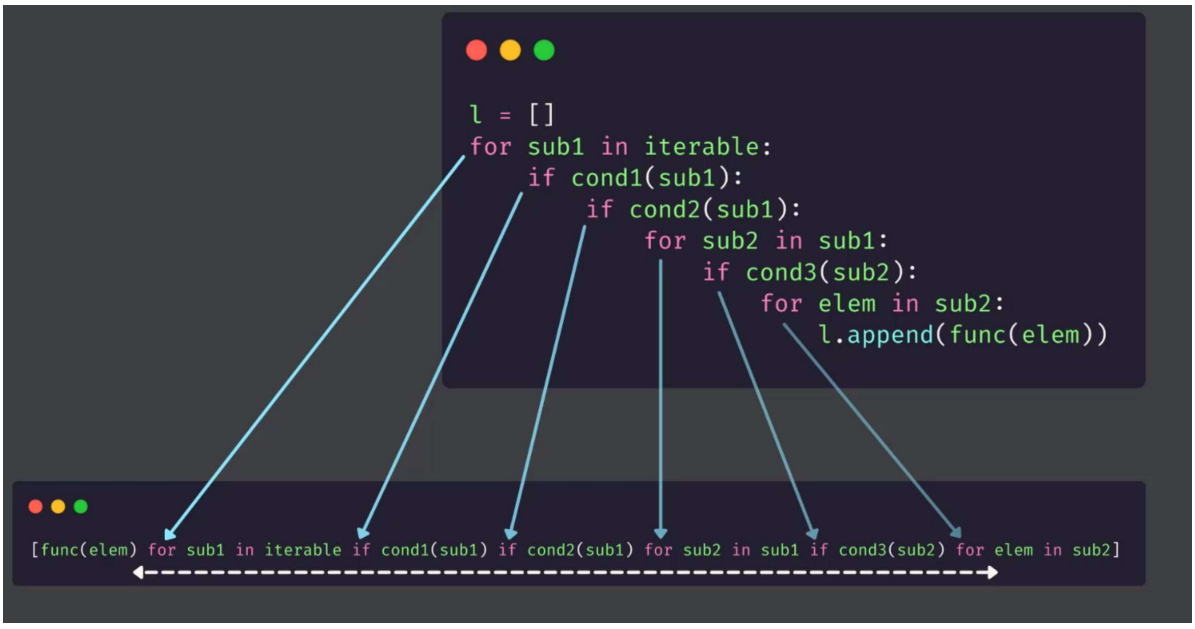
In [46]:
```
num=[]
for i in range(10):
    if i%2==0:
        num.append(i)
num
```

Out[46]: [0, 2, 4, 6, 8]



In [47]:
```
num1=[i for i in range(10) if i%2==0]

num1
```

Out[47]: [0, 2, 4, 6, 8]



In [48]:
```
a=[]
for i in range(20):
    if i%2==0:
        a.append(i)
    else:
        a.append(-i)
print(a)
```

[0, -1, 2, -3, 4, -5, 6, -7, 8, -9, 10, -11, 12, -13, 14, -15, 16, -17, 18, -19]

In [49]:
```
num3=[i if i%2==0 else -i  for i in range(20)]
print(num3)
```

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

In [50]:
```
import random
#exercise : change to list comprehension
random_numbers=[]
for i in range(10):
    random_numbers.append(random.randint(1,20))


a=[random.randint(1,20) for i in range(10)]
a
```

Out[50]:
```
[12, 8, 11, 18, 15, 14, 16, 3, 19, 11]
```

In [ ]:
```

```

EXercise

In [51]:
```
numbers = []
for i in range(1,1001):
    numbers.append(i)
```

In [52]:
```
list_8 = []
nums=[1,18,56,9,8,74,12,16,24,80,50,23,14,78,95]

for num in nums:
    if num%8==0:
        list_8.append(num)
```

In [53]:
```
list_6=[]
for j in range(1,1000):
    if "6" in str(j):
        list_6.append(j)

print(list_6)
```

```
[6, 16, 26, 36, 46, 56, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 76, 86, 96, 106, 116,
126, 136, 146, 156, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 176, 186, 196,
206, 216, 226, 236, 246, 256, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 276,
286, 296, 306, 316, 326, 336, 346, 356, 360, 361, 362, 363, 364, 365, 366, 367, 368,
369, 376, 386, 396, 406, 416, 426, 436, 446, 456, 460, 461, 462, 463, 464, 465, 466,
467, 468, 469, 476, 486, 496, 506, 516, 526, 536, 546, 556, 560, 561, 562, 563, 564,
565, 566, 567, 568, 569, 576, 586, 596, 600, 601, 602, 603, 604, 605, 606, 607, 608,
609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625,
626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642,
643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659,
660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676,
677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693,
694, 695, 696, 697, 698, 699, 706, 716, 726, 736, 746, 756, 760, 761, 762, 763, 764,
765, 766, 767, 768, 769, 776, 786, 796, 806, 816, 826, 836, 846, 856, 860, 861, 862,
863, 864, 865, 866, 867, 868, 869, 876, 886, 896, 906, 916, 926, 936, 946, 956, 960,
961, 962, 963, 964, 965, 966, 967, 968, 969, 976, 986, 996]
```

In [54]:
```
string= "python in for student of architecture is important"

counter=[]
for char in string:
    if char== " " :
        counter.append(char)

print(len(counter))

a=len([char for char in string if char==" "])
a
```

```
7
```

Out[54]:
```
7
```

In [55]:
```
list1=["M","na","i","ke"]
list2=["y","me","s","lly"]




list3=[list1[i]+list2[i] for i in range(len(list1))]

print(list3)
```

```
['My', 'name', 'is', 'kelly']
```

In [56]:
```
list1=["Mike", " ", "Ema", "kelley", " ", "brad"]

list2=[char for char in list1 if char!=" "]

print(list2)
```

```
['Mike', 'Ema', 'kelley', 'brad']
```

Tuple

In [57]:
```
tuple_1= ("python" , 12. , 0 , "#" , "introduction to python for architects", "grassho
```

In [58]:
```
tuple_2=  "java" , 18 , "rhino" , 17.36 ,  "grasshopper"
```

In [59]:
```
list1=[ "java" , 18 , "rhino" , 17.36 ,  "grasshopper"]

list1[0]= "c++"
list1
```

Out[59]:
```
['c++', 18, 'rhino', 17.36, 'grasshopper']
```

In [60]:
```
tuple_1[3]
```

Out[60]:
```
'#'
```

In [61]:
```
tuple_1[1:3]
```

Out[61]:
```
(12.0, 0)
```

In [62]:
```
tuple_1[:]
```

Out[62]:
```
('python',
 12.0,
 0,
 '#',
 'introduction to python for architects',
 'grasshopper')
```

In [63]:
```
tuple_1[-1:-5:-1]
```

Out[63]:
```
('grasshopper', 'introduction to python for architects', '#', 0)
```

In [64]:
```
tuple_3= 20,
```

In [65]:
```
# tuple is immutable
```

In [66]:
```
tuple_2[1] = 200
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_12228\1211235198.py in <module>
----> 1 tuple_2[1] = 200

TypeError: 'tuple' object does not support item assignment
```

Nested tuple

In [67]:
```
tuple_4=(("rhino", "grasshopper"), ("3dmax" ,"vray","corona"),("revit", "dynamo"))

tuple_4[0]
```

Out[67]:
```
('rhino', 'grasshopper')
```

In [68]:
```
tuple_4[0][0]
```

Out[68]:
```
'rhino'
```

built-in methods in tuple

In [69]:
```
tuple_5= (1,2,3,12,1,45,87,96,36,12,10,10,52,45,45,2,1,96)
```

In [70]:
```
tuple_5.count(1)
```

Out[70]:
```
3
```

In [71]:
```
tuple_5.index(45)
```

Out[71]:
```
5
```

built-in function in tuple

In [72]:
```
sum(tuple_5)
```

Out[72]:
```
556
```

In [73]:
```
min(tuple_5)
```

Out[73]:
```
1
```

In [74]:
```
max(tuple_5)
```

Out[74]:
```
96
```

In [75]:
```
len(tuple_5)
```

Out[75]:
```
18
```

In [76]:
```
sorted(tuple_5, reverse=True)
```

Out[76]:
```
[96, 96, 87, 52, 45, 45, 45, 36, 12, 12, 10, 10, 3, 2, 2, 1, 1, 1]
```

In [77]:
```
print(tuple_5)
```

```
(1, 2, 3, 12, 1, 45, 87, 96, 36, 12, 10, 10, 52, 45, 45, 2, 1, 96)
```

zip

In [78]:
```
n_tuple=("morteza","maryam","mina")
f_tuple=("akbari","sharifi","karimi")


total=zip(n_tuple, f_tuple )

#print(tuple(total))
#print(list(total))
#print(dict(total))
```

In [79]:
```
list_n=[1,2,3,4,5]
list_m=[4,5,6,9,7]
list_k= [6,9,8,7,5]

a=zip(list_n , list_m , list_k)
#print(tuple(a))
#print(list(a))
#print(dict(list(a)))
```

In [80]:
```
list_k=[1,2,3,4]
list_p=[4,5,6,9,7]
```

```
b=zip(list_k , list_p)
print(tuple(b))
```

```
((1, 4), (2, 5), (3, 6), (4, 9))
```

In [81]:
```
list(zip('abcdefg', range(3), range(4)))
```

Out[81]:
```
[('a', 0, 0), ('b', 1, 1), ('c', 2, 2)]
```

In [82]:
```
list_n=[1,2,3,4,5]
list_m=[4,5,6,9,7]
for i,j in zip(list_n , list_m):
    print(i ,"--", j )
```

```
1 -- 4
2 -- 5
3 -- 6
4 -- 9
5 -- 7
```

In [83]:
```
list_n=[1,2,3,4,5]
list_m=[4,5,6,9,7]

list_factor=[]

for i,j in zip(list_n , list_m):
    list_factor.append(i*j)

list_factor
```

Out[83]:
```
[4, 10, 18, 36, 35]
```

In [84]:
```
#exercise: list comprehension of previous exersice
list_factor=[i*j for i,j in zip(list_n , list_m)]

list_factor
```

Out[84]:
```
[4, 10, 18, 36, 35]
```

Set

In [85]:
```
set1={"python", "grasshopper", 12 , 45.36, "$" }
```

In [86]:
```
set1[1]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_12228\4054153366.py in <module>
----> 1 set1[1]

TypeError: 'set' object is not subscriptable
```

In [87]:
```
set1["python"] = "rhino"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_12228\1704955146.py in <module>
----> 1 set1["python"] = "rhino"

TypeError: 'set' object does not support item assignment
```

In [88]:
```
set2={1,2,3,5,5,6}
set2
```

Out[88]:
```
{1, 2, 3, 5, 6}
```

In [89]:
```
list_1=[1,2,3,4,5,6,6,7,8,8,9,10]              #change list to set

list_2=[]
for i in (list_1):
    if i not in list_2:
        list_2.append(i)
```

In [90]:
```
numbers= str(set(list_1))

numbers
```

Out[90]:
```
'{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}'
```

built-in methods in sets

In [91]:
```
set_new={1,5,6,9,8,7,14,78,96,52,10}

set_new.remove(78)

set_new
```

Out[91]:
```
{1, 5, 6, 7, 8, 9, 10, 14, 52, 96}
```

In [92]:
```
set_new.add(13)
set_new
```

Out[92]:
```
{1, 5, 6, 7, 8, 9, 10, 13, 14, 52, 96}
```

In [93]:
```
set_new.discard(125)              #remove a data if exist
set_new
```

Out[93]:
```
{1, 5, 6, 7, 8, 9, 10, 13, 14, 52, 96}
```

In [94]:
```
set_new.pop()

set_new
```

Out[94]:
```
{1, 5, 6, 7, 8, 9, 10, 13, 14, 52}
```

In [95]:
```
set_new.clear()
```

```
set_new
```

Out[95]:
```
set()
```

In [96]:
```
M1 ={12,13,15,14,17,19}
M2={12,85,15,14,17,10,23}

M1.union(M2)
```

Out[96]:
```
{10, 12, 13, 14, 15, 17, 19, 23, 85}
```

In [97]:
```
M1.intersection(M2)
```

Out[97]:
```
{12, 14, 15, 17}
```

Data structures comparison

Set	Dictionary	Tuple	List	Sequence	
{ }	{ }	( )	[ ]	" "	Sign
+	+	-	+	+	Mutable
-	key	From 0 - n	From 0 - n	From 0 - n	Index
-	+	+	+	+	Repetitive member
-	+	+	+	+	call

Built-in functions

In [ ]:
```
abs()   Returns the absolute value of a number

all()   Returns True if all items in an iterable object are true

any()   Returns True if any item in an iterable object is true

dict()  Returns a dictionary (Array)

dir()   Returns a list of the specified object's properties and methods

enumerate()     Takes a collection (e.g. a tuple) and returns it as an enumerate objec

exec()  Executes the specified code (or object)

filter()        Use a filter function to exclude items in an iterable object

float() Returns a floating point number

globals()       Returns the current global symbol table as a dictionary

help()  Executes the built-in help system

id()    Returns the id of an object

input() Allowing user input
```

```
int()   Returns an integer number

len()   Returns the length of an object

list()  Returns a list

map()   Returns the specified iterator with the specified function applied to each ite

memoryview()    Returns a memory view object

min()   Returns the smallest item in an iterable

next()  Returns the next item in an iterable

open()  Opens a file and returns a file object

pow()   Returns the value of x to the power of y

print() Prints to the standard output device

range() Returns a sequence of numbers, starting from 0 and increments by 1 (by default

reversed()      Returns a reversed iterator

round() Rounds a numbers

set()   Returns a new set object

slice() Returns a slice object

sorted()        Returns a sorted list

str()   Returns a string object

sum()   Sums the items of an iterator

tuple() Returns a tuple

type()  Returns the type of an object

vars()  Returns the __dict__ property of an object

zip()   Returns an iterator, from two or more iterators
```

Function

def functionName (): statement return...

In [98]:
```
#even numbers

number=10
if number%2==0:
    print("even number")
```

```
even number
```

In [99]:
```
def even_number(number):
    if number%2==0:
        return "even_number"

even_number(12)
```

Out[99]:
```
'even_number'
```

In [100…
```
even_number(14)
```

Out[100]:
```
'even_number'
```

In [101…
```
def prime_number (x):                    #prime number
    for i in range(2 , x):
        if x%i==0:
            return False
        else:
            return True

prime_number(54)
```

Out[101]:
```
False
```

In [102…
```
def f (a, b):
    return a**2 + b*2

f(2,3)
```

Out[102]:
```
10
```

In [103…
```
def gh12():
    print("hello")

gh12()
```

```
hello
```

In [104…
```
def mean(list_2):
    return sum(list_2)/ len(list_2)

mean([1,2,3,4,5,6,12,15])
```

Out[104]:
```
6.0
```

In [105…
```
def function(list_1):
    k=[]
    for i in range(len(list_1)):
        k.append(list_1[i]**2)
    return k

function([1,2,3,4,5,6])
```

Out[105]:
```
[1, 4, 9, 16, 25, 36]
```

In [106…
```
#exercise : write previous funcion as a list comprehension
```

```
def function(list_1):
    return [list_1[i] **2 for i in range(len(list_1))]

function([1,2,3,4,5,6])
```

Out[106]: [1, 4, 9, 16, 25, 36]

In [107…
```
def h(x):
    return x+2
    return x+4

h(3)
```

Out[107]: 5

In [108…
```
def g(y):
    return y+2 , y+3

g(3)
```

Out[108]: (5, 6)

In [109…
```
def maximum (a,b,c,d):
    return max(a,b,c,d)


maximum(1,2,3,4)
```

Out[109]: 4

a) function- argumnet

An argument is the value that is sent to the function when it is called.

In [110…
```
a,b,c = 1,2,3
print(c)

#unpack
```

3

In [111…
```
a,b,c = (1,2,3,4,5)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_12228\3950196508.py in <module>
----> 1 a,b,c = (1,2,3,4,5)

ValueError: too many values to unpack (expected 3)
```

In [112…
```
a,b,*c= 1,2,3,4,5      #Asterisk

print(c)
print(type(c))
```

```
[3, 4, 5]
<class 'list'>
```

In [113…
```
a,b,*c= [1,2,3,4,5]      #Asterisk

print(c)
print(type(c))
```

```
[3, 4, 5]
<class 'list'>
```

In [114…
```
n=[1,2,3,4,5]

print(*n)
```

```
1 2 3 4 5
```

In [115…
```
def f (a,b,c):
    print(a)
    print(b)
    print(c)

f(1,2,3)                      #positional
```

```
1
2
3
```

In [116…
```
def f (a,b,c):
    print(a)
    print(b)
    print(c)

f(a=1,b=2,c=3)
```

```
1
2
3
```

In [117…
```
def f (a,b,c):
    print(a)
    print(b)
    print(c)

f(b=2,c=3,a=1)
```

```
1
2
3
```

In [118…
```
def f (a,b,c):
    print(a)
    print(b)
    print(c)

f(1,b=2,c=3)
```

```
1
2
3
```

In [119…
```
def f (a,b,c):
    print(a)
    print(b)
    print(c)

f(a=1,2,3)
```

```
  File "C:\Users\Morteza\AppData\Local\Temp\ipykernel_12228\4080309904.py", line 6
    f(a=1,2,3)
            ^
SyntaxError: positional argument follows keyword argument
```

In [120…
```
def f (a,b,c):
    print(a)
    print(b)
    print(c)

f(a=1,c=2,3)
```

```
  File "C:\Users\Morteza\AppData\Local\Temp\ipykernel_12228\2769006299.py", line 6
    f(a=1,c=2,3)
            ^
SyntaxError: positional argument follows keyword argument
```

In [121…
```
def f (a,b,c):
    print(a)
    print(b)
    print(c)

n=[1,2,3]

f(n[0] , n[1] , n[2])

f(*n)                              # "*" unpack  and specify more than one
```

```
1
2
3
1
2
3
```

In [122…
```
def f (a,b,c):
    print(a)
    print(b)
    print(c)

dic={"a": 7 , "b": 4 , "c": 5}


f(dic["a"], dic["b"], dic["c"])



f(**dic)
```

```
7
4
5
7
4
5
```

b) function- parameter

A parameter is the variable listed inside the parentheses in the function definition.

In [123…
```
def f (a=1,b=2,c=3):
    print(a)
    print(b)
    print(c)

f()
```

```
1
2
3
```

In [124…
```
def f (a=1,b=2,c=3):
    print(a)
    print(b)
    print(c)

f(a=5,b=2,c=4)
```

```
5
2
4
```

In [125…
```
def f (a=1,b=2,c=3):
    print(a)
    print(b)
    print(c)

f(5,b=6,c=4)
```

```
5
6
4
```

In [126…
```
def f (a,b,c=3):
    print(a)
    print(b)
    print(c)

f(5,8)
```

```
5
8
3
```

In [127…
```
def power(a , b , c , d=3 , e = 2):
    return (a**2 + b**2 + c**2+d**2+e**2)
power(1,5,4)
```

Out[127]:
```
55
```

In [128…
```
def summ(a=1 , b=2 ,c=5):
    return (a+b+c)

print(summ())
print(".........")

print(summ(4))
print(".........")

print(summ(5,4))
print(".........")

print(summ(c=10))
```

```
8
.........
11
.........
14
.........
13
```

In [129…
```
def f (a,b,c):
    print(a)
    print(b)
    print(c)

f(1,2,3)

f(2,3,4,5)
```

```
1
2
3
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_12228\3034060258.py in <module>
      6 f(1,2,3)
      7
----> 8 f(2,3,4,5)
      9
     10

TypeError: f() takes 3 positional arguments but 4 were given
```

In [130…
```
def f (*c):
    #print(a)
    #print(b)
    print(c)

f(1,2,3,4,5,6)
```

```
(1, 2, 3, 4, 5, 6)
```

In [131…
```
def maxi(*args):
    return max(args)

maxi(1,2,5,12,48,75,9,86,3)
```

Out[131]:
```
86
```

In [132…
```
def mean(*args):
    return sum(args)/len(args)

mean (1,2,3,4)
```

Out[132]:
```
2.5
```

In [133…
```
def number(*args):
    return max(args)

n=[12,10,5,9,8,7,4,14,7,0]
number(*n)
```

Out[133]:
```
14
```

In [134…
```
def f (a,b,c):
    print(a)
    print(b)
    print(c)



f(a=7 , b=4, c=5, d=7)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_12228\701758096.py in <module>
      6
      7
----> 8 f(a=7 , b=4, c=5, d=7)

TypeError: f() got an unexpected keyword argument 'd'
```

In [135…
```
def f (**kwargs):
    print(kwargs)



#dic={"a": 7 , "b": 4 , "c": 5 , "d": 6 , "e": 9}

f(a=2 , b=4 , c=5 ,d=6 , e=9)
```

```
{'a': 2, 'b': 4, 'c': 5, 'd': 6, 'e': 9}
```

In [136…
```
def h(*args , **kwargs):
    pass

h(1,2,3,4,5,a=8,b=9,d=12)
```

In [137…
```
def p (a,b,*args , **kwargs):
    pass

p(14,-5,9,45,72, m=12.5,n=-32)
```

In [138…
```
def function(a, b, * , c, d):                    #after
    return a+b+c+d

function(5,7,c=10,d=6)
```

Out[138]:
```
28
```

In [139…
```
def function(a, b, / , c, d):                    #before
    return a+b+c+d

function(5,7,c=10,d=6)
```

Out[139]:
```
28
```

In [140…
```
def function(a, b, / ,c,*, d,e):
    return a+b+c+d+e

function(5,7,c=10,d=6,e=5)
```

Out[140]:
```
33
```

c) function annotation

In [141…
```
def t (x:int, y:int , z:int)-> int:                #definitaion can be string    pytho
    return (x+y+z)

t(2,5,9)
```

Out[141]:
```
16
```

In [142…
```
def t (x:int, y:int , z:int=8):
    return (x+y+z)

t(2,5)
```

Out[142]:
```
15
```

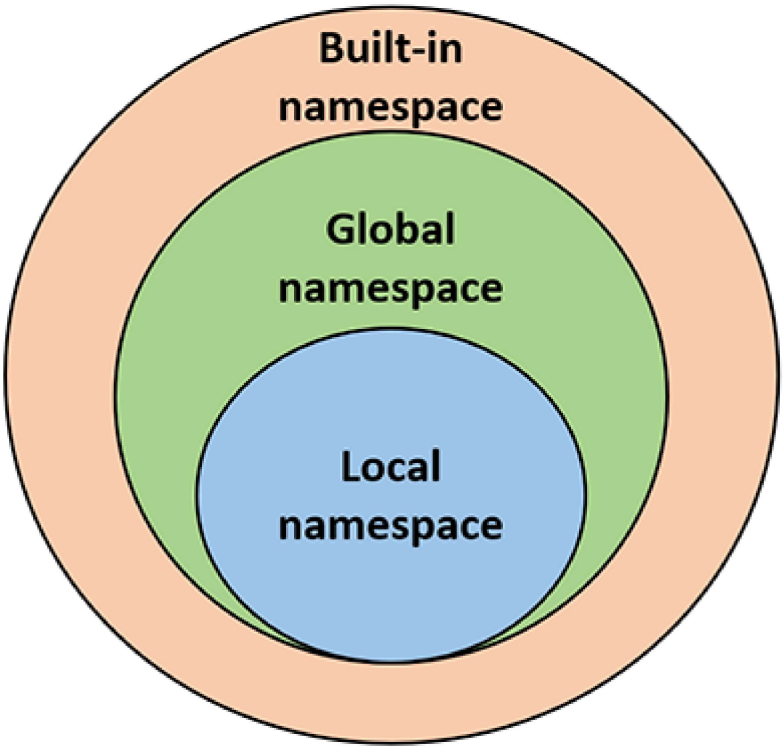
In [143…
```
def maxi(*args:int)-> float:
    return max(args)

maxi(1,2,5,8,9,12)
```

Out[143]:
```
12
```

In [144…
```
print(maxi.__annotations__)
```

```
{'args': <class 'int'>, 'return': <class 'float'>}
```

Name space



Type of Namespaces

In [145…
```
#built_in

print(type(len))

print("-------------")
print(type(print))

print("-------------")
print(type(sum))
```

```
<class 'builtin_function_or_method'>
-------------
<class 'builtin_function_or_method'>
-------------
<class 'builtin_function_or_method'>
```

In [146…
```
#global
def f(x):
    y=x**2
    z=x+5
    return y+z

print(f(10))
#..................
a=3
```

```
115
```

In [147…
```
#local
def f(x):
    global y, z
    y=x**2
    z=x+5
    return y+z
```

```
print(f(10))
print(y)                                    #enclose
```

```
115
100
```

Lambda

In [148…
```
def f (x):
    return x+10

print(f(2))

a=lambda x : x+10

print(a(2))
```

```
12
12
```

In [149…
```
f=lambda x,y: x*y
f(3,5)
```

Out[149]:
```
15
```

In [150…
```
power = lambda x: x ** 2

power(3)
```

Out[150]:
```
9
```

In [151…
```
add = lambda x, y: x + y
add(3,5)
```

Out[151]:
```
8
```

In [152…
```
# Use if-else in Lambda Functions

# check if number is even or odd
result = lambda x : x if x %2==0 else f"{x} is odd"

# print for numbers
print(result(12))
print(result(11))
```

```
12
11 is odd
```

In [153…
```
result = lambda x :  x %2==0

# print for numbers
print(result(12))
print(result(11))
```

```
True
False
```

Map

Python's map() is a built-in function that allows you to process and transform all the items in an iterable without using an explicit for loop

map (function , list)

In [154…
```
def f (*args):
    a=[]
    for i in args:
         a.append(i**2)
    return a

list_1 = [1,2,3,4,5]

f(*list_1)
```

Out[154]:
```
[1, 4, 9, 16, 25]
```

In [155…
```
def f (x):
    return x**2

list_1 = [1,2,3,4,5]

map_number= map(f , list_1)

print(list(map_number))
```

```
[1, 4, 9, 16, 25]
```

In [156…
```
#function
def addition(n):
    return n + n

#list
numbers = (1, 2, 3, 4)

# We double all numbers using map()
result = list(map(addition, numbers))
print(result)
```

```
[2, 4, 6, 8]
```

In [157…
```
def func(n):
    return abs(n)

abso_numbers=list(map(func , range(-10, 0)))

abso_numbers
```

Out[157]:
```
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

In [158…
```
def up(word):
    return word.upper()

word="introduction to python"
```

```
# map() can listify the list of strings individually

a=tuple(map(up , word))
print(a)
```

```
('I', 'N', 'T', 'R', 'O', 'D', 'U', 'C', 'T', 'I', 'O', 'N', ' ', 'T', 'O', ' ', 'P',
'Y', 'T', 'H', 'O', 'N')
```

In [159…
```
sequences = [10,2,8,7,5,4,11]

squared_result = map (lambda x: x**2, sequences)

print(list(squared_result))
```

```
[100, 4, 64, 49, 25, 16, 121]
```

In [160…
```
list_1 = [1,2,3,4,5]

map_number= list(map(lambda x : x+5 , list_1))

map_number
```

Out[160]:
```
[6, 7, 8, 9, 10]
```

In [161…
```
abso_numbers=list(map(lambda n : abs(n) , range(-10, 0)))

abso_numbers
```

Out[161]:
```
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

In [162…
```
numbers1 = [1, 2, 3]
numbers2 = [4, 5, 6]

result = map(lambda x, y: x + y, numbers1, numbers2)
print(list(result))
```

```
[5, 7, 9]
```

In [163…
```
# List of strings
list_3 = ['sat', 'bat', 'cat', 'mat']

test = list(map(len, list_3))
print(test)
```

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

Filter

Filter() is a built-in function in Python. The filter function can be applied to an iterable such as a list or a dictionary and create a new iterator. This new iterator can filter out certain specific elements based on the condition that you provide very efficiently.

filter (function , list)

In [164…
```
def function (number):
    if number>=4:
        return True
```

```
    else:
        return False

list_2 = [0,2,6,9,4,1,-2,5,4,6,10]

filtered_number= list(filter(function , list_2))
filtered_number
```

Out[164]:
```
[6, 9, 4, 5, 4, 6, 10]
```

In [165…
```
#exrecise
def function (number):
        return number>=4

list_2 = [0,2,6,9,4,1,-2,5,4,6,10]
filtered_number= list(filter(function , list_2))
filtered_number
```

Out[165]:
```
[6, 9, 4, 5, 4, 6, 10]
```

In [166…
```
# function that filters vowels

def fun(variable):
    letters = ['a', 'e', 'i', 'o', 'u']
    if (variable in letters):
        return True
    else:
        return False

# sequence
sequence = ['g', 'e', 'e', 'j', 'k', 's', 'p', 'r',"i","o"]

list_4= "morteza khorsand nikoo"

# using filter function
filtered_1 = filter(fun, sequence)
filtered_2 = filter(fun, list_4)


print(list(filtered_1))
print(list(filtered_2))
```

```
['e', 'e', 'i', 'o']
['o', 'e', 'a', 'o', 'a', 'i', 'o', 'o']
```

In [167…
```
list_2 = [0,2,6,9,4,1,-2,5,4,6,10]

filtered_n= list(filter(lambda number:number>=4 , list_2))

filtered_n
```

Out[167]:
```
[6, 9, 4, 5, 4, 6, 10]
```

In [168…
```
# a list contains both even and odd numbers.
seq = [0, 1, 2, 3, 5, 8, 13]

# result contains odd numbers of the list
result = filter(lambda x: x % 2 != 0, seq)
print(list(result))
```

```
# result contains even numbers of the list
result = filter(lambda x: x % 2 == 0, seq)
print(list(result))
```

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

In [169…
```
result = filter(None,(1,0,6,-1,None, [], " ", True))       # returns all non-zero valu
print(list(result))
```

```
[1, 6, -1, ' ', True]
```

In [170…
```
# Python filter() function example
def mulof3(val):
    if val%3==0:
        return val
# Calling function
result = filter(mulof3,(1,3,5,6,8,9,12,14))
# Displaying result
result = list(result)
print(result) # multiples of 3
```

```
[3, 6, 9, 12]
```

In [171…
```
seq = [0, 1, 2, 3, 5, 8, 13]

result = filter(lambda x: x % 2 , seq)
print(list(result))
```

```
[1, 3, 5, 13]
```

Reduce

Python's reduce() is a function that implements a mathematical technique called folding or reduction. reduce() is useful when you need to apply a function to an iterable and reduce it to a single cumulative value.

reduce (function(x,y) , list)

In [172…
```
# importing functools for reduce()
from functools import reduce

numbers=[10,20,30,40]
result= reduce(lambda x, y : x+y , numbers)

result
```

Out[172]:
```
100
```

In [173…
```
# initializing list
lis = [1, 3, 4, 10, 4]


# printing summation using reduce()
print("The summation of list using reduce is :", end="--")
print(reduce(lambda x, y: x+y, lis))
```

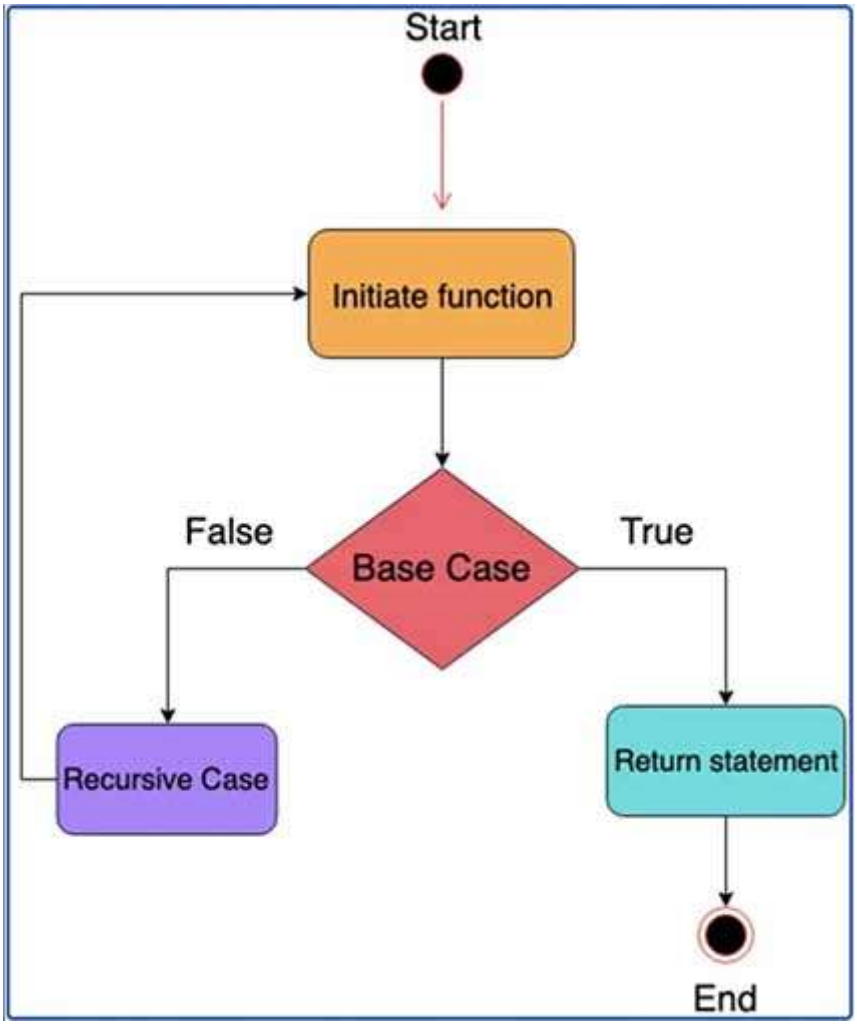
```
The summation of list using reduce is :--22
```

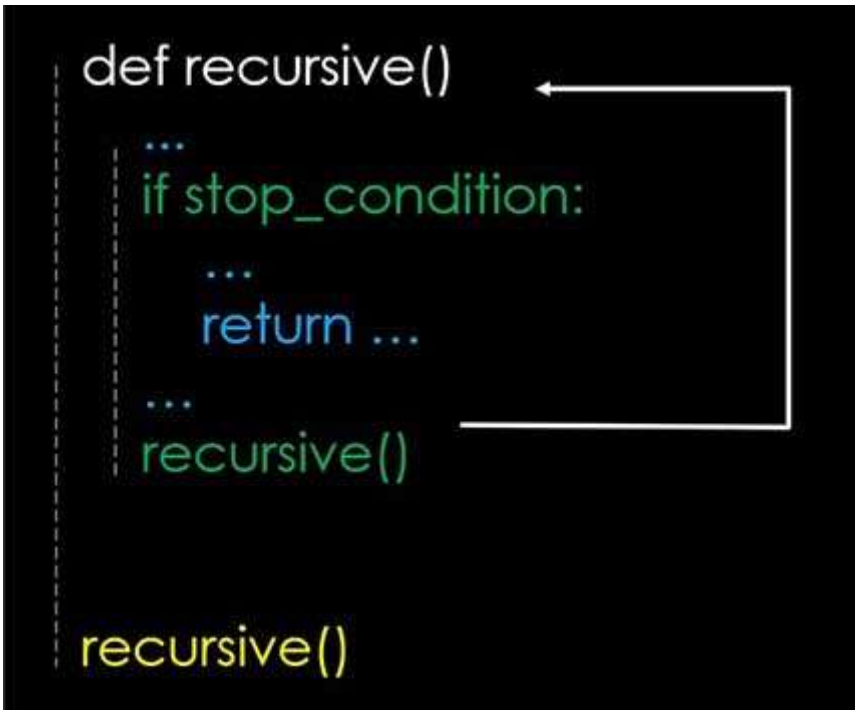
In [174…

```python
# initializing list
listt = [1, 3, 5, 6, 2]


# using reduce to compute maximum element from list


print(reduce(lambda a, b: a if a > b else b, listt))
```

```
6
```

Recursive function





In [175…
```python
#factorial

def fact(number):
    if number==1:
        return 1
    return number * fact(number-1)

fact(5)
```

Out[175]:
```
120
```

In [176…
```python
from time import sleep

def reverse_number(n):
    #conditon
    if n==10:
        return None
    print(n)
    sleep(0.5)
    n+=1
    reverse_number(n)

reverse_number(0)
```

```
0
1
2
3
4
5
6
7
8
9
```

In [177…

```
#fibonacci
# 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144...

def fib (n):
    if n==0 or n==1:
        return n
    return fib(n-1)+ fib(n-2)

fib(10)
```

Out[177]:

```
55
```

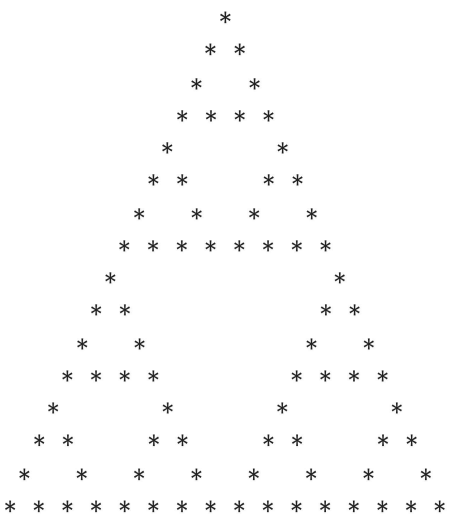
In [178…

```
list_fib= [fib(i) for i in range(6)]

print(list_fib)
```

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

In [179…

```
# sierpinski triangle.

def printSierpinski( n) :
    y = n - 1
    while(y >= 0) :

        # printing space till the value of y
        i = 0
        while(i < y ):
            print(" ",end="")
            i = i + 1

        # printing '*'
        x = 0
        while(x + y < n ):

            # printing '*' at the appropriate
            # position is done by the and
            # value of x and y wherever value
            # is 0 we have printed '*'
            if ((x & y) != 0) :
                print(" ", end = " ")
            else :
                print("* ", end = "")
            x =x + 1

        print()
        y = y - 1

# Driver code
n = 16

# Function calling
printSierpinski(n)
```



Enumerate

In [180…
```
numbers=[12,18,75,23,10]
for index,num  in enumerate (numbers):
    print(index ,"--" ,num)
```

```
0 -- 12
1 -- 18
2 -- 75
3 -- 23
4 -- 10
```

In [181…
```
# Python program to illustrate
# enumerate function in loops
list_5 = ["Mahsa", "Sara", "Ashkan"]

# changing index and printing separately
for i, j in enumerate(list_5, 100):
    print (i,j)
```

```
100 Mahsa
101 Sara
102 Ashkan
```

Turtle

developed by Wally Feurzeig, Seymour Papert and Cynthia Solomon in 1967

In [ ]:
```
pip install turtle
```

In [ ]:
```
import turtle

#from turtle import *
```

Turtle motion turtle.forward(distance) turtle.fd(distance) turtle.position() turtle.back(distance) turtle.bk(distance) turtle.backward(distance) turtle.right(angle) turtle.rt(angle) turtle.left(angle) turtle.lt(angle) turtle.goto(x, y) turtle.home()

turtle.circle(radius, extent=which part of the circle is drawn(number or none) , steps=number or None)

In [ ]:
```
turtle.circle(25, extent=360 , steps=15)
```

turtle.dot(size=integer, *color = a colorstring or a numeric color tuple)

In [ ]:
```
turtle.dot(20, "blue")
```

turtle.undo() Undo (repeatedly) the last turtle action(s).

In [ ]:
```
turtle.reset()
```

In [ ]:
```
for i in range(4):
    turtle.fd(50); turtle.lt(80)

for i in range(8):
    turtle.undo()
```

turtle.speed(speed=None) “fastest”: 0 #speed = 0 means that no animation takes place. forward/back makes turtle jump and likewise “fast”: 10 “normal”: 6 “slow”: 3 “slowest”: 1

In [ ]:
```
turtle.speed(speed=1)
for i in range(5):
    turtle.fd(50); turtle.lt(80)
```

Pen control turtle.pendown() turtle.pd() turtle.down() turtle.penup() turtle.pu() turtle.up()
turtle.pensize(width=None)¶ turtle.width(width=None)

In [ ]:
```
turtle.pensize(3)
for i in range(4):
    turtle.fd(50); turtle.lt(80)
```

In [ ]:
```
turtle.reset()
```

In [ ]:
```
turtle.pen(fillcolor="black", pencolor="red", pensize=10)
for i in range(4):
    turtle.fd(50); turtle.lt(80)
```

turtle.pencolor(*args) 1. pencolor(colorstring) 2. pencolor(r, g, b)
turtle.fillcolor(*args) 1. fillcolor(colorstring) 2. pencolor(r, g, b)
turtle.reset() Delete the turtle’s drawings from the screen, re-center the turtle and set variables to the default values.
turtle.clear() Delete the turtle’s drawings from the screen. Do not move turtle. State and position of the turtle as well as drawings of other turtles are not affected.
turtle.write(arg, move=False, align='left', font=('Arial', 8, 'normal'))

In [ ]:
```
turtle.showturtle()
turtle.write("women, life, freedome ", True, align="left")
```

turtle.hideturtle() turtle.ht() turtle.showturtle() turtle.st()
turtle.shape(name=None) “turtle”, “circle”, “square”, “triangle”, “classic”

In [ ]:
```
turtle.shape("turtle")
turtle.write("women, life, freedome ", True, align="left")
```

In [ ]:
```
turtle.reset()
```

In [ ]:
```
Methods of TurtleScreen/Screen

turtle.bgcolor(*args)
```

```
screen.bgcolor("orange")

turtle.screensize(canvwidth=None, canvheight=None, bg=None)
screen.screensize(2000,1500 , "orange)
```

turtle.bye() Shut the turtlegraphics window.

In [ ]:
```
skk = turtle.Turtle()

for i in range(4):
    skk.forward(50)
    skk.right(90)

turtle.done()
```

In [ ]:
```
star = turtle.Turtle()


for i in range(5):
    star.right(144)
    star.forward(100)

turtle.done()
```

In [ ]:
```
polygon = turtle.Turtle()

num_sides = 6
side_length = 70
angle = 360.0 / num_sides

for i in range(num_sides):
    polygon.forward(side_length)
    polygon.right(angle)

turtle.done()
```

In [ ]:
```
import turtle   #Outside_In
wn = turtle.Screen()
wn.bgcolor("light green")
wn.title("Turtle")
skk = turtle.Turtle()
skk.color("blue")

def sqrfunc(size):
    for i in range(4):
        skk.fd(size)
        skk.left(90)
        size = size-5


sqrfunc(146)
sqrfunc(126)
sqrfunc(106)
sqrfunc(86)
sqrfunc(66)
sqrfunc(46)
sqrfunc(26)
```

In [ ]:
```
import turtle
wn = turtle.Screen()
wn.bgcolor("light green")
skk = turtle.Turtle()
skk.color("blue")

def sqrfunc(size):
    for i in range(4):
        skk.fd(size)
        skk.left(90)
        size = size + 5


sqrfunc(6)
sqrfunc(26)
sqrfunc(46)
sqrfunc(66)
sqrfunc(86)
sqrfunc(106)
sqrfunc(126)
sqrfunc(146)
```

In [ ]:
```
import turtle
loadWindow = turtle.Screen()
turtle.speed(2)

for i in range(100):
    turtle.circle(5*i)
    turtle.circle(-5*i)
    turtle.left(i)
```

In [ ]:
```
colors = ['red', 'purple', 'blue', 'green', 'orange', 'yellow']
t = turtle.Pen()
turtle.bgcolor('black')
for x in range(360):
    t.pencolor(colors[x%6])
    t.width(x//100 + 1)
    t.forward(x)
    t.left(59)
```

In [ ]:
```
import colorsys
import turtle

t = turtle.Turtle()
s = turtle.Screen()

s.bgcolor('black')
t.speed(0)

n= 36
h = 0

for i in range (460):
```

```
    #HSV (Hue Saturation Value).

    c = colorsys.hsv_to_rgb(h, 1, 0.8)
    h+=1/n
    t.color(c)
    t.left(145)

    for j in range (5):

        t.forward(300)
        t.left(150)
```

In [ ]:
```
import colorsys
import turtle
t = turtle.Turtle()
s = turtle.Screen().bgcolor('black')
t.speed(0)
n = 70
h = 0
for i in range (360):
    c = colorsys.hsv_to_rgb(h, 1, 0.8)
    h+= 1/n
    t.color(c)
    t.left(1)
    t.fd(1)
    for j in range (2):
        t.left(2)
        t.circle(100)
```

In [ ]:
```
hr=turtle.Turtle()
hr.left(90)
hr.speed(8)

def tree(i: int):
    if i <10:
        return
    else:
        hr.forward(i)
        hr.left(30)
        tree(3* i/4)
        hr.right(60)
        tree(3*i/4)
        hr.left(30)
        hr.backward(i)
tree(100)
turtle.done()
```

In [ ]:
```
###### KOCH CURVE PROGRAM EXAMPLE ######
###### DRAW KOCH SNOWFLAKE ######
import turtle
screen = turtle.Screen() # Create the screen.
screen.setup(480, 480)   # Set Window size.

###### TURTLE SHAPE, SPEED, PEN SIZE, COLOR ######
TTL = turtle.Turtle()
TTL.speed(10) #Set the turtle's speed. 1 is slow, 10 is fast; 0 is fastest.
TTL.color("red") #Set the turtle's color.
TTL.pensize(2) #Set width of turtle drawing pen.
```

```
TTL.shape("turtle")

###### SET TURTLE STARTING POSITION ######
TTL.penup() # Do not let the turtle draw while moving to position (-160, 110).
TTL.setposition(-170,110)
TTL.pendown() # Enable the turtle to draw.

###### VARIABLES ######
# Variable 'length' is the length of side of starting equilateral triangle.
# Variable 'length' is divided by variable 'lengthDivisor' at each recursion iteration
# Variable 'recursionLevel' is the depth of the Koch Curve.


###### KochFractal FUNCTION DEFINITION ######
# Draw a Koch fractal of 'recursionLevel' and 'length'.  # 1
def KochFractal(TTL, recursionLevel, length, lengthDivisor):
    if recursionLevel == 0:          # recursionLevel = 0 is fractal base case.
        TTL.forward(length) # This will draw an equilateral triangle.
    else:
        length = length / lengthDivisor
        # Call 'KochFractal' function inside the function itself - Recursion.
        # Draw Koch pattern of one-third 'length' and decrement 'recursionLevel'.
        KochFractal(TTL, recursionLevel-1, length, lengthDivisor)
        TTL.left(60) #Turn turtle left by 60 degrees.
        KochFractal(TTL, recursionLevel-1, length, lengthDivisor)
        TTL.right(120) #Turn turtle right by 120 degrees.
        KochFractal(TTL, recursionLevel-1, length, lengthDivisor)
        TTL.left(60) #Turn turtle left by 60 degrees.
        KochFractal(TTL, recursionLevel-1, length, lengthDivisor)

###### INITIALIZE KOCH FRACTAL recursionLevel AND length ######
# Initialize variable 'length', the length of side of equvilateral triangle.
length = 360 # Equvilateral triangle length = 360 pixels.   # 2

###### CALL KochFractal FUNCTION ######
# KochFractal function draws Koch fractal for one side of the triangle.
# Call KochFractal function for the first side of the triangle.
KochFractal(TTL, 3, length, 3)                      # 3
TTL.right(120) #Turn turtle right by 120 degrees.
# Call KochFractal function for the second side of the triangle.
KochFractal(TTL, 3, length, 3)                      # 4
TTL.right(120) #Turn turtle right by 120 degrees.
# Call KochFractal function for the third side of the triangle.
KochFractal(TTL, 3, length, 3)                      # 5

screen.exitonclick() # Exit
```

The Walrus Operator :=

In [4]:
```
import sys                                  #python 3.8 and higher
print(sys.version)
```

```
3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)]
```

In [184…
```
w=5
print(w)
```

```
5
```

In [189…
```
print(w:=5)
```

```
5
```

In [190…
```
a=[1,2,3,4,5]
n=len(a)
if n >3:
    print(True)
```

```
True
```

In [191…
```
if n:=len(a) >3:                            #better to use paracentesis
        print(True)
```

```
True
```

In [192…
```
def function (x):
    return x+2

function(2)

function(k:=2)
```

Out[192]:
```
4
```

In [193…
```
n=5

print(f" the number you wrote is {n}" )
```

```
 the number you wrote is 5
```

In [194…
```
print(f" the number you wrote is {(n:=5)}")
```

```
 the number you wrote is 5
```

In [195…
```
number=[1,2,3,4,5]
```

```
dictionary={
            "le" : len(number),
            "su" : sum(number),
          "mean" : len(number) / sum(number)

}
```

In [196…
```
number=[1,2,3,4,5]

dictionary={
          "le": (l:=len(number)),
          "su": (su:= sum(number)),
        "mean": l/su
}
```

In [197…
```
dictionary
```

Out[197]:
```
{'le': 5, 'su': 15, 'mean': 0.3333333333333333}
```

PEP - Python Enhancement Proposal

In [198…
```
#https://peps.python.org/pep-0000/
```

In [199…
```
import this
```

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

In [200…
```
#https://pep8.ir/
```

Iterat

iteration -> for and while ..... iterable -> list - tupple - sequence ..... itrator -> Technically, in Python, an iterator is an object which implements the iterator protocol, which consist of the methods __iter__() and __next__().

In [201…
```
list_1 = [1,2,3,4,5]

print(dir(list_1))            # print("__iter__" in dir(list_1))
```

```
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__delit
em__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '_
_getitem__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_subcla
ss__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__
reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '_
_setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'clear', 'copy',
'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

In [202…
```
list_1 = [1,2,3,4,5]

list_1 = iter(list_1)                          #you can use next method on it

print(next(list_1))
```

```
1
```

In [203…
```
list_number=[1,2,3,4,5,6]
list_number= iter(list_number)

while True:
    try:
        print(next(list_number))
    except StopIteration:
        break
```

```
1
2
3
4
5
6
```

In [204…
```
from itertools import count

counter= count()
for i in counter:
    print(i)
    if i==50:
        break
```

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
39
40
41
42
43
44
45
46
47
48
49
50

Object Oriented Programming (OOP)

class Grey : morteza = Grey() #object morteza.natinality = "iranian" #attribute or properties morteza.language = "persian" morteza.job = " teacher" morteza.hight= 187
class Architecture : project1 = Architecture() project1.name = "Azadi Tower" project1.architect = "Amanat" project1.year= "1353"

In [ ]:
```
#step 1
class Grey:
    self.natinality = "iranian"                  #attribute or properties
    self.language = "persian"
    self.job = " teacher"
    self.hight= 187

morteza = Grey()                                 #object


#step2
class Grey:
    self.natinality = x
    self.language = y
    self.job =  z
    self.hight= k

morteza = Grey()

#step3
class Grey:
    self.natinality = natinality
    self.language = language
    self.job =  job
    self.hight= hight

morteza = Grey()


#step4
class Grey:
    def data (self , natinality , language , job , hight):              #method     #er
        self.natinality = natinality
        self.language = language
        self.job = teacher
        self.hight= hight


morteza = Grey()
morteza.data("iranian" , "persian" , "teacher", "178")
```

In [205…
```
class Architecture(object):
    def f (self, name , architect , year):
        self.name = name
        self.architect = architect
        self.year = year

project1 = Architecture()

project1.f("azadi" , " amanat", "1353")
```

In [206…
```
class Test :
    def var(self , number1 , number2):
        self.number1 = number1
        self.number2 = number2
    def add(self):
        return self.number1 + self.number2
    def mul(self):
```

```
        return self.number1 * self.number2

obj1= Test()

obj1.var(2,3)

print(obj1.add())

print("----------")

print(obj1.mul())
```

```
5
----------
6
```

In [207…

```
class Language (object):
    def Langname(self , name):
        self.name = name
    def display_name (self):
        return self.name
    def message_name (self):
        print("welcome to ", self.name)

l1=Language()

l1.Langname("python")

print(l1.display_name())

print("------------")


l1.message_name()
```

```
python
------------
welcome to  python
```

In [208…

```
list_1=[1,2,3,4,5]

print(type(list_1))

list_1.append(8)
```

```
<class 'list'>
```

In [ ]:

```
class list(object):
    """
    list() -> new empty list
    list(iterable) -> new list initialized from iterable's items
    """
    def append(self,number): # real signature unknown; restored from __doc__
        """ L.append(object) -> None -- append object to end """
        pass

    def clear(self): # real signature unknown; restored from __doc__
        """ L.clear() -> None -- remove all items from L """
```

```
        pass

    def copy(self): # real signature unknown; restored from __doc__
        """ L.copy() -> list -- a shallow copy of L """
        return []

    def count(self, value): # real signature unknown; restored from __doc__
        """ L.count(value) -> integer -- return number of occurrences of value """
        return 0

    def extend(self, iterable): # real signature unknown; restored from __doc__
        """ L.extend(iterable) -> None -- extend list by appending elements from the i
        pass

    def index(self, value, start=None, stop=None): # real signature unknown; restored
        """
        L.index(value, [start, [stop]]) -> integer -- return first index of value.
        Raises ValueError if the value is not present.
        """
        return 0

    def insert(self, index, p_object): # real signature unknown; restored from __doc__
        """ L.insert(index, object) -- insert object before index """
        pass

    def pop(self, index=None): # real signature unknown; restored from __doc__
        """
        L.pop([index]) -> item -- remove and return item at index (default last).
        Raises IndexError if list is empty or index is out of range.
        """
        pass

    def remove(self, value): # real signature unknown; restored from __doc__
        """
        L.remove(value) -> None -- remove first occurrence of value.
        Raises ValueError if the value is not present.
        """
        pass

    def reverse(self): # real signature unknown; restored from __doc__
        """ L.reverse() -- reverse *IN PLACE* """
        pass

    def sort(self, key=None, reverse=False): # real signature unknown; restored from _
        """ L.sort(key=None, reverse=False) -> None -- stable sort *IN PLACE* """
        pass

    def __add__(self, *args, **kwargs): # real signature unknown
        """ Return self+value. """
        pass

    def __contains__(self, *args, **kwargs): # real signature unknown
        """ Return key in self. """
        pass

    def __delitem__(self, *args, **kwargs): # real signature unknown
        """ Delete self[key]. """
        pass

    def __eq__(self, *args, **kwargs): # real signature unknown
```

```
        """ Return self==value. """
        pass

    def __getattribute__(self, *args, **kwargs): # real signature unknown
        """ Return getattr(self, name). """
        pass

    def __getitem__(self, y): # real signature unknown; restored from __doc__
        """ x.__getitem__(y) <==> x[y] """
        pass

    def __ge__(self, *args, **kwargs): # real signature unknown
        """ Return self>=value. """
        pass

    def __gt__(self, *args, **kwargs): # real signature unknown
        """ Return self>value. """
        pass

    def __iadd__(self, *args, **kwargs): # real signature unknown
        """ Implement self+=value. """
        pass

    def __imul__(self, *args, **kwargs): # real signature unknown
        """ Implement self*=value. """
        pass

    def __init__(self, seq=()): # known special case of list.__init__
        """
        list() -> new empty list
        list(iterable) -> new list initialized from iterable's items
        # (copied from class doc)
        """
        pass

    def __iter__(self, *args, **kwargs): # real signature unknown
        """ Implement iter(self). """
        pass

    def __len__(self, *args, **kwargs): # real signature unknown
        """ Return len(self). """
        pass

    def __le__(self, *args, **kwargs): # real signature unknown
        """ Return self<=value. """
        pass

    def __lt__(self, *args, **kwargs): # real signature unknown
        """ Return self<value. """
        pass

    def __mul__(self, *args, **kwargs): # real signature unknown
        """ Return self*value.n """
        pass

    @staticmethod # known case of __new__
    def __new__(*args, **kwargs): # real signature unknown
        """ Create and return a new object.  See help(type) for accurate signature. "
        pass
```

```
    def __ne__(self, *args, **kwargs): # real signature unknown
        """ Return self!=value. """
        pass

    def __repr__(self, *args, **kwargs): # real signature unknown
        """ Return repr(self). """
        pass

    def __reversed__(self): # real signature unknown; restored from __doc__
        """ L.__reversed__() -- return a reverse iterator over the list """
        pass

    def __rmul__(self, *args, **kwargs): # real signature unknown
        """ Return self*value. """
        pass

    def __setitem__(self, *args, **kwargs): # real signature unknown
        """ Set self[key] to value. """
        pass

    def __sizeof__(self): # real signature unknown; restored from __doc__
        """ L.__sizeof__() -- size of L in memory, in bytes """
        pass

    __hash__ = None
```

In [209…

```
a=5.3
print(type(a))
```

```
<class 'float'>
```

kinds of method

1. Instance method (use self)

   1.1 usual method

   1.2 magic method(dunder method) : recall method with a sign

---

1. Class method

   @classmethod - cls

---

1. Static method

   @staticmethod

In [210…

```
class Car:
    def __init__(self, name):                #constructor
        self.name = name



a = Car('benz')
```

```
print(a.name)
```

```
benz
```

In [211…

```
class car (object):
    def __init__ ( self , name , number_of_doors , color):
        self.name= name
        self.number_of_doors = number_of_doors
        self.color= color

benz = car ("sls" , 2 , "red" )
print(benz.color)
```

```
red
```

In [212…

```
class circle:
    def __init__(self , r):
        self.r = r
    def area (self):
        return self.r *self.r +3.14
x=circle(8)
x.area()
```

Out[212]:

```
67.14
```

In [213…

```
class Rectangle:
    def __init__ (self , width , hight):
        self.width= width
        self.hight= hight
    def area(self):
        return self.width * self.hight

rec=Rectangle(5 , 4)
rec.area()
```

Out[213]:

```
20
```

In [214…

```
class Book :
    def __init__ (self , author , title):
        self.author = author
        self.title = title
    def info(self):
        return self.title + " : " + self.author


X= Book("khorsand" , "interactive architecture")
X.info()
```

Out[214]:

```
'interactive architecture : khorsand'
```

In [215…

```
class Dog:

    def __init__(self, dogBreed,dogEyeColor):

        self.breed = dogBreed
        self.eyeColor = dogEyeColor
```

```
tomita = Dog("Fox Terrier","brown")

print("This dog is a",tomita.breed,"and his eyes are",tomita.eyeColor)
```

```
This dog is a Fox Terrier and his eyes are brown
```

In [216…

```
# __str__ return a string representation of its object.  run with print

class Employee:

    def __init__(self, name , salary):
        self.name= name
        self.salary= salary

    def __str__(self):
        return 'name='+self.name+' salary=$'+str(self.salary)


person_1 = Employee("Neda" , 12000)


print(person_1)

person_1
```

```
name=Neda salary=$12000
```

Out[216]:

```
<__main__.Employee at 0x198f747c220>
```

In [217…

```
# __ge__() method   :This method gets invoked when the >= operator is used and returns

class Person:
    def __init__(self, age):
        self.age = age
    def __ge__(self, other):
        return self.age >= other.age



alice = Person(18)
bob = Person(17)
carl = Person(18)


print(alice >= bob)

print(alice >= carl)

print(bob >= alice)
```

```
True
True
False
```

In [218…

```
class c :

    lst=[]

    def f (self , x):
        self.lst.append(x)
```

```
obj1= c()
obj1.f(1)
obj1.f(2)
obj1.f(3)

print(obj1.lst)
```

```
[1, 2, 3]
```

In [219…

```
class Address:
    def __init__ (self , city, street , zipcode):
        self.city= city
        self.street = street
        self.zipcode= zipcode

    def __str__(self):
        lst=[]
        lst.append(f" {self.city} -- {self.street} -- {self.zipcode}")
        return " ".join(lst)

a= Address("Tehran" , "174east" , "1655916993")

print(a)
```

```
 Tehran -- 174east -- 1655916993
```

In [220…

```
text = ['Python', 'is', 'a', 'fun', 'programming', 'language']

# join elements of text with space
print(' '.join(text))
```

```
Python is a fun programming language
```

In [221…

```
class Robot(object):
    def __init__ (self , n , y):
        self.name= n
        self.buildyear= y

    def __str__ (self):
        return "name: " + self.name +  ", buildyear: "+str(self.buildyear)


object2= Robot("rr" ,1980)

print(object2)
```

```
name: rr, buildyear: 1980
```

In [222…

```
class Person:
    def __init__(self, name, height):
        self.name = name
        self.height = height

    def show(self):
        print(f'{self.name} is {self.height}')

p = Person('amir', 180)
p.show()
```

```
amir is 180
```

In [223…

```python
class Test :
    def __init__ (self , a):
        self.a = a
    def __add__ (self , b):
        return self.a + b.a

obj1=Test(1)
obj2=Test(2)

print(obj1+obj2)
```

```
3
```

In [224…

```python
class AB :
    def __init__ (self , age):
        self.age = age

    def __gt__ (self, other):              # __lt__    "lowerthan"       <
        if self.age > other.age :              #__eq__     "equalto"          ==
            return True
        else:  False

Ali= AB(22)
Sara=AB(38)

if (Ali>Sara):
    print("yes")
else:  print("no")
```

```
no
```

In [225…

```python
class K:
    def __init__(self):
        self.lst=[45,89,12]

    def __str__ (self):
        return str(self.lst)

    def __len__ (self):
        return len(self.lst)

    def __getitem__(self , i):
        return self.lst[i]


object1=K()
#len(object1)
#object1[2]

#print(object1)
```

In [226…

```python
class Machine:

    model = "peugeot"                          #class variable

    def __init__ (self , t):
        self.t = t                            #instance variable
```

```
m1= Machine("206")
m2=Machine("207")

print(m1.model)
print(m2.model)

print("-------------")

print(m1.t)
print(m2.t)
```

```
peugeot
peugeot
-------------
206
207
```

In [227…

```
class Test:
    def __init__ (self , num1:int, num2:int):
        self.num1=num1                                  #public
        self.__num2= num2                                #private     #data hiding


ob=Test(2,5)

print(ob.num1)

#print(ob.num2)

print(ob._Test__num2)                          # Calling the private member
                                                # through name mangling
```

```
2
5
```

In [228…

```
class S :
    def __init__ (self , x):
        self.__x = x

    def f (self):
        self.__x+=1

    def g (self , m):
        return (m+ self.__x)

#ob= S(2)
#ob.f()
#ob.g(3)
```

In [229…

```
class H :
    __X=3                          #Private class variable
    def show(self):
        print(self.__X + 2)

ob=H()
ob.show()
ob._H__X
```

```
5
```

Out[229]: 3

In [230…

```
class ABC:
    def __f(self):               #private method
        print(1)

    def g(self):
        return (self.__f())


ob=ABC()
ob.g()
ob._ABC__f()
```

```
1
1
```