#Python Operators

1.Arithmetic Operater

2.Comparison

3.Assignment

4.Logical

5.Membership

6.Bitwise

In [3]:
```
#Arithmetic Operator
a=12
b=2
#'+'Operator
print('a + b =',a+b)
print('a - b =',a-b)
print('a * b =',a*b)
print('a / b =',a/b)#
print('a ** b =',a**b)
print('a//b=',a//b)
```

```
a + b = 14
a - b = 10
a * b = 24
a / b = 6.0
a ** b = 144
a//b= 6
```

In [5]:
```
#Comparison Operator
a=12
b=2
print('a > b is',a>b)
print('a < b is',a<b)
print('a == b is',a==b)
print('a >= b is',a>=b)
print('a <= b is',a<=b)
print('a != b is',a!=b)
```

```
a > b is True
a < b is False
a == b is False
a >= b is True
a <= b is False
a != b is True
```

In [10]:
```
#Assignment Operator:
## x = x  +  5
# '=' Operator
x = 5
print('x = ',x)
# '+' Operator
x += 5
print("x = ",x)
x -= 5
```

```
print("x = ",x)
x *= 5
print("x = ",x)
x /= 5
print("x = ",x)
```

```
x =  5
x =  10
x =  5
x =  25
x =  5.0
```

In [16]:

```
# Logical Operator
# 'and Operator
# segragating two statements
x = 11
a = x > 10 and x < 20
b = x > 10 and x == 12
print(a)
print(b)
# 'or' Operator
# one of the statement is true than output is true
c = x > 10 or x == 12
print(c)
# not Operator
print(not a)
```

```
True
False
True
False
```

In [1]:

```
# Membership Operator (checking something)
# String
a = 'Sure Trust'
print('L' not in a)
print('S' in a)
print('F' in a)
```

```
True
True
False
```

In [ ]:

Python Flow Control

three ways

1.Conditional Statements

2.Loops

3.Function Calls

Conditional Statement

if condition syntax

if test_expression1: statement1

elif test_expression2: statement2

In [22]:
```
x = 5
if x < 0:
    print('x is Negative')
else:
    print('x is Positive')
# more than two statements we use elif
x = 4
if x < 0:
    print('x is Negative')
elif x % 2:
    print('x is Positive and odd')
else:
    print('x is even and nonnegative')
```
```
x is Positive
x is even and nonnegative
```

In [ ]:
```
#Assignment 08-03-23
# Write a program to accept percentage from the User and display acc
a = int(input("enter the percentage:"))
if a >= 90:
    print('Excellent')
elif a >= 60 and a < 90:
    print('good')
elif a >= 40 and a < 60:
    print('average')
else:
    print('fail')
```

In [4]:
```
# input type
name = input('Enter your Name :') #name = input()
print(name)
```
```
Enter your Name :nandini
nandini
```

In [8]:
```
# Calculate by using input Keyword:
a = int(input('Enter a Number1 : '))
b = int(input('Enter a Number2 : '))
c = int(input('Enter a Number3 : '))
d = a + b + c
print('Result:', d)
```
```
Enter a Number1 : 23
Enter a Number2 : 1
Enter a Number : 4
Result: 28
```

In [11]:
```
# write a program to calculate Simple BMI by using  Heights and Weig
Weight = int(input('Enter a weight : '))
Height = float(input('Enter a Height : '))

BMI = (Weight/(Height)**2)
```

```
print(BMI)
```

```
Enter a weight : 60
Enter a Height : 170.0
0.0020761245674740486
```

In [24]:
```
# Adding to Strings
first_name = 'chelimi '
last_name = 'Nandini'

full_name = first_name +  last_name
print(full_name)

text = 'Age is '
age ='22'

combine = text + age
print(combine)


price = 120
discount = 35.5

net_price = price - discount
print(net_price)

price = 120
quantity = 15

total_cost = price * quantity
print(total_cost)

total_cost = 130000
quantity = 25
price_per_unit = total_cost // quantity
print(price_per_unit)
```

```
chelimi Nandini
Age is 22
84.5
1800
5200
```

Assignment 09-07-23

Calculate Principle, Tenure and Rate of Interest

Output should be calculated of Simple interest by using user input format

In [27]:
```
Principle_Amount = int(input("Enter the Amount : "))
Tenure = int(input("Enter time in Years : "))
Rate_of_interest = int(input("Enter the Rate of Interest : "))

simple_interest = (Principle_Amount * Tenure * Rate_of_interest) / 10
print("simple interest",simple_interest)

#Find the Even number by using input User method
```

```
a = int(input("Enter the number : "))
if a % 2 ==0:
    print("Given number is even")
else:
    print("Given number is odd")
```

```
Enter the Amount : 25000
Enter time in Years : 3
Enter the Rate of Interest : 5
simple interest 3750.0
Enter the number : 6
Given number is even
```

Data Types Conversions:

1.Implicit conversion : Done by python interpreter with programmer's Intervention

2.Explicit conversion : User-defined that forces an Expression to be of Specific Data type

In [4]:
```
# Implict Conversion
income = 2000
print(type(income))
additional_income = 200.0
print(type(additional_income))
total = income + additional_income
print(total)
```

```
<class 'int'>
<class 'float'>
2200.0
```

In [5]:
```
# Explicit Conversion
price = 10000
tax = '20'

total_cost = price + int(tax)
print(total_cost)
```

```
10020
```

In [6]:
```
# importing math library for square root

import math

math.sqrt(49)
```

Out[6]:
```
7.0
```

In [10]:
```
string = 'The Quick Brown Fox'
print(string.count('The Quick Brown Fox'))
len(string)
```

```
1
```

Out[10]:
```
19
```

In [11]:
```
string = 'How are You'
print(string)
print(string.replace('o','i'))
print(string.upper())
print(string.lower())
print(string.title())
```

```
How are You
Hiw are Yiu
HOW ARE YOU
how are you
How Are You
```

In [15]:
```
# Concatenation  of Each Element
 #(webscraping  company give 200 url per day to webscraping )
s1 = '08' , '03' , '2000' # number into date using webscraping
sep = '/'

date = sep.join(s1)
print(date)
```

```
08/03/2000
```

In [2]:
```
text  = 'Cereals Groceries Veggies'
text1 = text.count('Cereals Groceries Veggies')
print(text.split(' '))
print(text.split(' / '))
print(text1)
len(text)
```

```
['Cereals', 'Groceries', 'Veggies']
['Cereals Groceries Veggies']
1
```

Out[2]:
```
25
```

Assignment

10-03-23

print the Message

Calculater :

```
1. Add
2. Subtract
3. Multiply
4. Divide
```

#input

Enter Choice (1-4) : 3 Enter A:10 Enter B:20

output

product = 200 by using input Format

In [1]:
```
choice = int(input("enter the choice in between 1-4 : "))
b = int(input("Enter the number1 :"))
c = int(input("Enter the number2 :"))
```

```
if choice == 1:
    d = b + c
    print(d)
elif choice == 2:
    d = b - c
    print(d)
elif choice == 3:
    d = b * c
    print(d)
elif choice == 4:
    d = b / c
    print(d)
else:
    print("not in choice")
```

```
enter the choice in between 1-4 : 3
Enter the number1 :10
Enter the number2 :20
200
```

In [3]:
```
#remove symbols are letters in paragraph
text = '?*?/Rakesh?*/'
print(text.strip('?*/'))

#reverse the string
string = 'I am in Python Class'
print(string[::-1])
```

```
Rakesh
ssalC nohtyP ni ma I
```

Data Container

1.List

2.Tuple

3.Dictionary

4.Sets

List

1.list is a Container Object 2.Heterogeneous Sequence of Elements 3.It can have Duplicate values 4.It is **mutable** 5.Elements are separated by commas and enclosed by Paranthesis Square[] 6.Supportive **Elements in List* *

-Append -concatenation -len(),min(),max() -Access the Values and Indexs

In [8]:
```
# Creating  Empty List

customer = []
print(customer)
```

```
customer_age = [19,22,32]
print(type(customer_age))

customer_details = [19,'john',86.2]
print(customer_details)
```

```
[]
<class 'list'>
[19, 'john', 86.2]
```

In [ ]:
```
<font color = blue
```

In [27]:
```
# A list inside a List is called as Nested List
amount_spent_by_a_customer = ['john',[18,23,44,56],67.0]
print(amount_spent_by_a_customer)

#concat and Append
states_in_india = ['haryana','karnataka','Tamilnadu','Andhra Pradesh
states_in_USA = ['texas','Alaska','New Jersey']

#concatination
print(states_in_india + states_in_USA)

#Append
states_in_USA.append('Florida')
print(states_in_USA)

# sort
average_price = [100, 700,450,567,900,233]
average_price.sort(reverse = True)
print(average_price)
average_price.sort()
print(average_price)
```

```
['john', [18, 23, 44, 56], 67.0]
['haryana', 'karnataka', 'Tamilnadu', 'Andhra Pradesh', 'texas', 'A
laska', 'New Jersey']
['texas', 'Alaska', 'New Jersey', 'Florida']
[900, 700, 567, 450, 233, 100]
[100, 233, 450, 567, 700, 900]
```

Assignment 11-03-2023

Create a Vault by using input user method

Input

Enter a String = xyz Enter a String - xyz123

Output

Either it could be 1.Successfully logged in!!! 2.Please Check your userid Or Password Use by if and Else Statement

In [2]:
```
A = input("Enter a String :")
B = input("Enter a String")
```

```
if A == 'xyz' and B == 'xyz123':
        print("Successfully Logged")
else:
    print("Please Check your userid Or Password")
```

```
Enter a String :xyz
Enter a Stringxyz123
Successfully Logged
```

In [6]:
```
a = [10,12,16,17,23,9,8,100]
b = [11,23,56,78,45,8,6,200]
print('The original list : ',a)
a.sort()
print('the Modified list :',a )
print("\n")
print('The original list : ',b)
b.sort()
print('the Modified list :',b)
print("\n")
c = [1.2,34,1,0,5,6]
c.sort()
print(c)
```

```
The original list :  [10, 12, 16, 17, 23, 9, 8, 100]
the Modified list : [8, 9, 10, 12, 16, 17, 23, 100]


The original list :  [11, 23, 56, 78, 45, 8, 6, 200]
the Modified list : [6, 8, 11, 23, 45, 56, 78, 200]


[0, 1, 1.2, 5, 6, 34]
```

In [4]:
```
list = [12,34,'Cereals',100]
print(list.sort())

# we could not able to use Sort Function for mixed data types
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent ca
ll last)
<ipython-input-4-4c8d264a3445> in <module>
      1 list = [12,34,'Cereals',100]
----> 2 list.sort()

TypeError: '<' not supported between instances of 'str' and 'int'
```

In [16]:
```
# The difference is sort functon it modifies the Existing list
# Sorted Function - It creates on Another List(or output might be an
a = [10,12,16,17,9,8,100]
a = sorted(a)
print(a)
```

```
[8, 9, 10, 12, 16, 17, 100]
```

In [17]:
```
print(a)
```

```
[8, 9, 10, 12, 16, 17, 100]
```

In [14]:
```
# More List Operators - with Membership Operator
product_prices = [120,1000,2000,3000,4000,340]
print(2000 in product_prices)
print(670 not in product_prices)
print(len(product_prices))
print(max(product_prices))
print(min(product_prices))
```

```
True
True
6
4000
120
```

In [28]:
```
# Define a List
retail_products = ['Beverages','cereals','Biscuits']
retail_products.append('20') # insert something it will be good for p
# let s consider 5 columns we have to insert null values not use for
print(retail_products)
retail_products.remove('Beverages')
print(retail_products)
retail_products.clear()
print(retail_products)
retail_products = ['Beverages','cereals','Biscuits']
retail_products1 = ['Cosmetics','Veggies']
retail_products.extend(retail_products1)
print(retail_products)
retail_products = ['Beverages','cereals','Biscuits']
retail_products.extend('Milk') # extnd will work for more then two st
print(retail_products) # then we have use append for single string
```

```
['Beverages', 'cereals', 'Biscuits', '20']
['cereals', 'Biscuits', '20']
[]
['Beverages', 'cereals', 'Biscuits', 'Cosmetics', 'Veggies']
['Beverages', 'cereals', 'Biscuits', 'M', 'i', 'l', 'k']
```

In [ ]:
```
# Assignment 13-03-22
WAP Currency Converter should be able to convert a Specific Currency
# Inputs :
USD (U.S.Dollars)(1 USD = 82.16 INR)
YEN (japanese YEN)(1 YEN = 0.62 INR)
EURO (1 EURO = 88.04 INR
U.K. POUND(1 U.K.Pound = 99.86 INR)

Enter the currency which you want to convert- eg USD-EURO Enter the v
the converted value-
```

In [33]:
```
currency = int(input("Enter the indian rupees you want to convert :")
choice = int(input("enter the choice 1-4:"))
if choice == 1:
    USD = currency * 82.16
    print("USD currency : ",USD)
elif choice == 2:
    YEN = currency * 0.62
    print("YEN currency : ",YEN)
elif choice == 3:
    EURO = currency * 99.86
    print("EURO currency : ",EURO)
```

```
elif choice == 4:
    UK = currency * 99.86
    print("UK currency : ",UK)
else:
    print("correct choice")
```

```
Enter the indian rupees you want to convert :5000
enter the choice 1-4:4
UK currency :  499300.0
```

In [11]:

```
# define a list

course1 = ['data science','Machine Learning','Python','html','big dat

#   index       0              1                   2        3     4
#              -5             -4                  -3       -2    -1
course1.append('Statistics')
print(course1)
course1.insert(0,'Statistics')
print(course1)
print(course1[-3])
print(course1[0:3])
```

```
['data science', 'Machine Learning', 'Python', 'html', 'big data',
'Statistics']
['Statistics', 'data science', 'Machine Learning', 'Python', 'html
', 'big data', 'Statistics']
html
['Statistics', 'data science', 'Machine Learning']
```

In [12]:

```
course1 = ['data science' , 'Machine Learning',
           'Python' , 'html' , 'big data']
course2 = ['Statistics', 'hadoop']
course1.extend(course2)
print(course1)
```

```
['data science', 'Machine Learning', 'Python', 'html', 'big data',
'Statistics', 'hadoop']
```

In [15]:

```
income = [25000,34000,4500,7500,45,34000]
income.sort(reverse = True)
print(income)
#Delete the list Elements
course1 = ['data science' , 'Machine Learning',
           'Python' , 'html' , 'big data']
del course1[2:4]
print(course1)
course1.append('html')
print(course1)
course2 = ['data science' , 'Machine Learning',
           'Python' , 'html' , 'big data']
course2.reverse()
print(course2)
# Pop Function removes and returns the last item of a list
course1.pop()
print(course1)
course2*2 # repetation function
```

```
[34000, 34000, 25000, 7500, 4500, 45]
['data science', 'Machine Learning', 'big data']
```

Out[15]:
```
['big data',
 'html',
 'Python',
 'Machine Learning',
 'data science',
 'big data',
 'html',
 'Python',
 'Machine Learning',
 'data science']
```

Introduction for Tuple

1.A tuple is a collection of ellements which are ordered

2.ATuple once if it is defined that cannot be modified(immutable)

3.Paranthesis() and Seperatd by commas('Round brackets')

4.for iteration tuples are actually faster than list

In [27]:
```
# Create a Tuple

my_tuple = ('hello','Python','hello','World')
print(my_tuple)

#mixed data types
my_tuple = (123,2.23,'Hello Python')
print(my_tuple)

#A tuple can list inside
my_tuple = ('Python',[123,45,67,23,5])
print(my_tuple)

# A tuple can hold a tuple inside it: and also mixed datatype
my_tuple = ((3245,334,8,9),('Python',23,3,500))
print(my_tuple)

#how we can able to access the tuple elements
my_tuple = ('mango','yellow','green','blue')
print(my_tuple[0])
print(my_tuple[-3:-1])

#change the tuple values gives error but we can do some modification
# tuple object does not support item assignment
# we couldn't delete particular function and we couldn't add anything

my_tuple = (123,['s','v','a'],'world')
my_tuple[1][0] = 99
print(my_tuple)
# delete enter tuple
del my_tuple
my_tuple
```

```
('hello', 'Python', 'hello', 'World')
(123, 2.23, 'Hello Python')
('Python', [123, 45, 67, 23, 5])
((3245, 334, 8, 9), ('Python', 23, 3, 500))
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-27-f3e981df8a32> in <module>
     30 # delete enter tuple
     31 del my_tuple
---> 32 my_tuple

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

In [ ]:
```
# Assignment 14-03-22
Accept the three numbers form the user and display the Second largest
```

In [46]:
```
number1 = int(input("Enter the Number1 :"))
number2 = int(input("Enter the Number2 :"))
number3 = int(input("Enter the Number3 :"))
if (number1 <= number2) and (number2 <= number3):
    print(number2)
elif (number1 <= number3 and number3 <= number2:
    print(number3)
else:
    print(number1)
```

```
Enter the Number1 :1
Enter the Number2 :3
Enter the Number3 :2
2
```

Tuple Methods:

1. Count
2. Index()

In [5]:
```
my_tuple = ('a','p','p','l','y','r','e','t','s')
print(my_tuple.count('e'))
print(my_tuple.count('p'))
print(my_tuple.index('t'))
```

```
1
2
7
```

Dictionary:

1.Python Dictionary is an Unordered Collections 2.It maps keys to values and these keys - value pairs provide a useful way to store data in pytho

In [14]:
```
balance  = {'mla' : 77654,
           'jhone' : 85763,
           'mike' :566664 }
```

```
print(balance)
#Access the Dictionary
x = balance['mike']
print(x)
x = balance.get('mla')
print(x)
```

```
{'mla': 77654, 'jhone': 85763, 'mike': 566664}
566664
77654
```

Creating a Dict:

Syntax

dictionary_name = {key_1:value1,key_2:value2,key_3:value3}

Assignment

Write a Python program to check if a triangle is a equilateral,isosceles or Scalene

An equilateral triangle is a triangle in which all three sides are equal An scalene triangle is a triangle in which all three sides are unequal An isosceles triangle is a triangle in which any two sides are equal

In [16]:

```
a = int(input("enter the first side of triangle : "))
b = int(input("enter the second side of triangle : "))
c = int(input("enter the third side of triangle : "))
if a == b == c:
    print("equilateral triangle")
elif a != b != c:
    print("scalene triangle")
else:
    print("isosceles triangle")
```

```
enter the first side of triangle : 2
enter the second side of triangle : 2
enter the third side of triangle : 4
isosceles triangle
```

In [8]:

```
year_sale = {2014:5000,2015:200,2017:4000}
year_sale[2015] = 4500
print(year_sale)
print(len(year_sale))
year_sale[2018] = 50000
print(year_sale)
```

```
{2014: 5000, 2015: 4500, 2017: 4000}
3
{2014: 5000, 2015: 4500, 2017: 4000, 2018: 50000}
```

In [ ]:

```
# Dictionary Methods

.keys()
.values()
.items()
.pop()
```

In [32]:
```
horsepower = {'BMW' : 900,
             'Mercades' : 945,
             'Ferrari' : 967,
             'Renault' : 889}
print(horsepower)
print(horsepower.keys())
print(horsepower.values())
# It is possible to check key is belongs to dictionary
print('Honda' in horsepower)
print('Mercades' in horsepower)
print(967 in horsepower)
print(horsepower.items())
# Access the items:

a = horsepower.items()
a = list(a)
print(a[0])

# pop()  Remove the elements with the Specified key name
print(horsepower.pop('BMW'))
print(horsepower)

# pop_item: it removes the last inserted item
print(horsepower.popitem())
print(horsepower)

# del keyword It is same as pop but with the Specified key name
del horsepower['Ferrari']
print(horsepower)

# copy method() use copy() method to copy dictionary
horsepower = {'BMW' : 900,
             'Mercades' : 945,
             'Ferrari' : 967,
             'Renault' : 889}
copy_dict = horsepower.copy()
# Update method
# Add and modify dictionaries by using the func dict.update()
horsepower.update({'BWMR' : 700})
print(horsepower)
print(horsepower.clear())
print(horsepower)
```

```
{'BMW': 900, 'Mercades': 945, 'Ferrari': 967, 'Renault': 889}
dict_keys(['BMW', 'Mercades', 'Ferrari', 'Renault'])
dict_values([900, 945, 967, 889])
False
True
False
dict_items([('BMW', 900), ('Mercades', 945), ('Ferrari', 967), ('Re
nault', 889)])
('BMW', 900)
900
{'Mercades': 945, 'Ferrari': 967, 'Renault': 889}
('Renault', 889)
{'Mercades': 945, 'Ferrari': 967}
{'Mercades': 945}
{'BMW': 900, 'Mercades': 945, 'Ferrari': 967, 'Renault': 889, 'BWMR
': 700}
None
```

```
{}
```

```
# SETS
Creating a set:
1.In python sets are written with curly Brackets
2.It is used to eliminate the duplicate items
3.It is collection of unique items
```

In [34]:
```
age_set = {11,12,23,34,50}
print(age_set)
# Another way
set([1,2,3,4])
```

```
{34, 11, 12, 50, 23}
```

Out[34]:
```
{1, 2, 3, 4}
```

Assignment 16-03-23

Enter the Selling Price and caluculate discount based on the range of Selling Price The discount rate is: a)5% if Selling Price is atmost 5000. b)12% if Selling Price is atmost 15000. c)20% if Selling Price is atmost 25000. d)30% if Selling Price is atmost 25000.

In [ ]:
```
a = int(input("Enter the Selling_Price"))
if a > 0 and a <= 5000:
    discount  = 5
    discountAmount = (a*discount)/100
    print(discountAmount)
elif a > 0 and a <= 15000:
    discount  = 12
    discountAmount = (a*discount)/100
    print(discountAmount)
elif a > 0 and a <= 25000:
    discount  = 20
    discountAmount = (a*discount)/100
    print(discountAmount)
else:
    discount = 30
    discountAmount = (a*discount)/100
    print(discountAmount)
```

```
Conditional Statement
 if statement
 if else statement
 if elif  else statement
 nested if and if else condition
 for loop
 while loop
 Break Statement
 Continue Statement
```

if condition :It decides whether given statement needs to be executed or not It checks for a Given Condition, if the condition is True then the code present inside,if block will be executed

In [1]:
```
# Create a String
my_string = 'Python'

# Create a list:
my_list = ['Data Science','Machine Learning','Artificial Intelligence
if my_string in my_list:
    print(my_list + ',' + 'Tutorial')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent ca
ll last)
<ipython-input-1-bbc29443d91b> in <module>
      5 my_list = ['Data Science','Machine Learning','Artificial In
telligence','Python']
      6 if my_string in my_list:
----> 7     print(my_list + ',' + 'Tutorial')

TypeError: can only concatenate list (not "str") to list
```

In [25]:
```
# if- else Statement:

# Create an integer:
x = 200
print(x)

# Use if - else Statement:
if not x == 500:
    print('the value x is not equal to 500')
else:
    print('the value x is equal to 500')
```

```
200
the value x is not equal to 500
```

In [23]:
```
# Nested if and if else Statement:

num = float(input('Enter a Number-'))
if num >=0:
    if num == 0:
        print('Zero')
    else:
        print('Positive Number')
else:
    print('Negative Number')
```

```
Enter a Number-3
Positive Number
```

```
# For Loop :

For loop in python is used to execute a block of statements or code
several times until the given condition is false

It is used to iterate over a Sequence (list,tuple,string)or other
iterable objects
```

In [16]:
```
# for loo
string = 'Introduction to python'
for i in string:
    print(i,end = ' /') # for vertical print(i)
```

```
I /n /t /r /o /d /u /c /t /i /o /n /  /t /o /  /p /y /t /h /o /n /
```

In [22]:

```
for i in range(1,11): # print until 10 that means it will print until
    print(i,end =' ')
```

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

In [13]:

```
# Find sum of all the numbers in a given list:
# Given list
num_list = [45,455,677,34,35]
# Variable to store the sum:
total = 0
# Iteration over the list:
for i in num_list:
    total = total + i

print('The sum is - ',total)
```

```
The sum is -  1246
```

Assignment 17-03-23

Write a Python programe to convert a month name to a number of days.Expected Output: List of months:January,February,March,April,May,June,August,September,October,November,Decem
Input the name of Month:February no of days:28/29 days

In [15]:

```
month = input('enter the month')
a = ['January','March','May','July','September','November']
b = ['April','june','August','October','December']
if month in a:
    print("No of days : 31")
elif month in b:
    print("No of days :30")
else:
    print("No of days :28/29")
```

```
enter the monthFebrauary
No of days :28/29
```

In [12]:

```
## Nested for loop
list_off_list = [['Pune','Mumbai','Delhi'],[23,45,12],[9,29,5,4,3]]
for i in list_off_list:
    for item in i:
        print(item)

# print a number pattern using a for loop and range function:

for numbers in range(15):
    for i in range(numbers):
        print(numbers,end = ' ')
    print('\n')


for i in range(0,5):
    for j in range(0,i+2):
        print('*',end = ' ')
    print()
```

```
for i in range(0,5):
    for j in range(0,10):
        print('*',end = ' ')
    print()
```

```
Pune
Mumbai
Delhi
23
45
12
9
29
5
4
3


1

2 2

3 3 3

4 4 4 4

5 5 5 5 5

6 6 6 6 6 6

7 7 7 7 7 7 7

8 8 8 8 8 8 8 8

9 9 9 9 9 9 9 9 9

10 10 10 10 10 10 10 10 10 10

11 11 11 11 11 11 11 11 11 11 11

12 12 12 12 12 12 12 12 12 12 12 12

13 13 13 13 13 13 13 13 13 13 13 13 13

14 14 14 14 14 14 14 14 14 14 14 14 14 14

* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
```

In [5]:
```
word = ['Apple','banana','Car','lion']
for i in word:
    print('The Following Lines to be Appeared')
    for letter in i:
        print(letter)
    print(i)
```

```
The Following Lines to be Appeared
A
p
p
l
e
Apple
The Following Lines to be Appeared
b
a
n
a
n
a
banana
The Following Lines to be Appeared
C
a
r
Car
The Following Lines to be Appeared
l
i
o
n
lion
```

In [18]:

```
# Assignment 18-03-23
# if is used for comparison and for is used for
#you are analyzing house prices.The qiven code declare a list with ho
# heighborhood.you need to calculate and output the number of houses
# is above the average.
house_price = [500000,300000,40000,10000]
total = 0
number = 0
for i in house_price:
    total = total + i
    number = number +1
print(total)
print(number)
average = total/number
print(average)
```

```
850000
4
212500.0
```

While Loop

1. while loop executes the same procedure by checking the given Condition
2. It runs untill the condition is false
3. The condition is given before the loop and the checked before each execution of the loop

In [4]:

```
# 3 condition 1. before the while loop 2.after while loop 3.increment
# Print the Digit of 1 to 5:
```

```
x = 0
while(x < 5):
    # increment the x value
    x = x + 1
    print(x)

a = 0
b = 0
while(a < 10):
    a = a + 1
    b = b + a
    print("print a :",a)
    print("Print b:",b)
else:
    ('the sum of first 9 integers:' b)
```

```
1
2
3
4
5
print a : 1
Print b: 1
print a : 2
Print b: 3
print a : 3
Print b: 6
print a : 4
Print b: 10
print a : 5
Print b: 15
print a : 6
Print b: 21
print a : 7
Print b: 28
print a : 8
Print b: 36
print a : 9
Print b: 45
print a : 10
Print b: 55
```

In [8]:

```
# WAP to print first 10 intergers and theirs squares
# using while loop
x = 1
while(x < 10):
    #print("print the number:",x)
    #print("print the square:",x ** 2)
    print(x, '\t \t \t',x**2)
    x = x + 1
```

```
1                        1
2                        4
3                        9
4                        16
5                        25
6                        36
7                        49
8                        64
9                        81
```

In [18]:
```
#WAP for loop statement
# 10,20,        300
x = 10
while(x <= 300):
    print(x,end = '     ')
    x =x + 10
```

```
10     20     30     40     50     60     70     80     90     100     110
120     130     140     150     160     170     180     190     200     210
220     230     240     250     260     270     280     290     300
```

In [30]:
```
# Assignment 21-03-23
data = {"100-90":25,"42-01":48,"55-09":12,"128-64":71,"002-22":18,"32
x = data.values()
print(x)
for i in x:
    if( i >= 18):
        ticket = 20
        print("t20",i)
    else:
        ticket = 5
        print("5",i)
```

```
dict_values([25, 48, 12, 71, 18, 19])
20 25
20 48
5 12
20 71
20 18
20 19
```

In [37]:
```
data = {"David":["123-321-88","david@test.com"],"James":["241-879-093
name = input("Enter the name")
x = data.keys()
print(x)
if(name == x):
    print("data found")
else:
    print("not found")
```

```
Enter the nameDavid
dict_keys(['David', 'James', 'Bob', 'Amy'])
not found
```

In [ ]:
```
data = {"David":["123-321-88","david@test.com"],"James":["241-879-093
for i in data.keys():
    name = input("enter the name:")
    if(i == name):
        print("data found")
    else:
        print("data not found")
```

In [3]:
```
contacts = {"David":["123-321-88","david@test.com"],"James":["241-879
name = input("Enter the contact name")
if(name in contacts):
    print(contacts[name[1]])
else:
    print('Not fount')
```

```
Enter the contact nameDavid

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-3-79cc173197c4> in <module>
      2 name = input("Enter the contact name")
      3 if(name in contacts):
----> 4     print(contacts[name[1]])
      5 else:
      6     print('Not fount')

KeyError: 'a'
```

In [ ]:

In [ ]:

In [5]:
```
# write a program to enter the numbers till the user wants to end it
# it should display the sum of all number entered
character  = 'yes'
sum = 0
while character.lower() == 'yes':
    number = int(input("enter the number : "))
    sum = sum + number
    character = input("do you want to continue (Yes/No):")
    print(sum)
```

```
enter the number : 1
do you want to continue (Yes/No):yes
1
enter the number : 4
do you want to continue (Yes/No):yes
5
enter the number : 6
do you want to continue (Yes/No):no
11
```

In [8]:
```
character  = 'yes'
sum = 0
while character.lower() == 'yes':
    number = int(input("enter the number : "))
    sum = sum + number
    character = input("do you want to continue (Yes/No):")
    print(sum)
    if(number == 0):
        print(sum)
```

```
enter the number : 1
do you want to continue (Yes/No):yes
1
enter the number : 5
do you want to continue (Yes/No):yes
6
enter the number : 0
```

In [3]:
```python
i = 0
sum = 0
while i <= 4 :
    number = int(input("enter the number : "))
    sum = sum + number
    i = i + 1
print(sum)
```

```
enter the number : 2
enter the number : 3
enter the number : 5
enter the number : 6
enter the number : 7
23
```

In [12]:
```python
i = 0
sum = 0
maximum = 0
minimum = 0
while i <= 9 :
    number = int(input("enter the number : "))
    sum = sum + number
    i = i + 1
    if number >= maximum:
        maximum = number
    if number <= minimum:
        minimum = number
print(sum)
print(maximum)
print(minimum)
```

```
enter the number : 1
enter the number : 2
enter the number : 3
enter the number : 4
enter the number : 5
enter the number : 6
enter the number : 7
enter the number : 8
enter the number : 9
enter the number : 0
45
9
0
```

Break and Continue

- You might be facing a situation in which you need to exit a loop completely
- When an external condition is triggered or there may also be a situatuion when you want to skip a part of the loop and start a new execution

-Python provides break and continue statements.

In [7]:
```
# Break

for i in range(5):
    if i == 3:
        break
    print(i)
for i in range(5):
    if i == 3:
        continue
    print(i)
# Break Statment by using while loop:
# Program to find  first 5 multiples of 6:
number = int(input ("Enter the number of which the user wants to prin
i = 1
# we are using while loop for iterating the multiplication 10 times
print ("The Multiplication Table of: ", number)
while i <= 10:
    if i==5:
        break
    print (number, 'x', i, '=', number * i)
    i += 1
```

```
0
1
2
0
1
2
4
Enter the number of which the user wants to print the multiplicatio
n table: 6
The Multiplication Table of:  6
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
```

In [9]:
```
for letter in 'Python':
    if letter == 'h':
        break
    print('current letter : ',letter)

variable = 10
while(variable > 0):
    print('current variable value - ',variable)
    variable = variable - 1
    if variable == 5:
        break
    print('the loop has been ended')
for letter in 'Python':
    if letter == 'h':
        continue
    print('current letter : ',letter)

variable = 10
while(variable > 0):
    variable = variable - 1
    if variable == 5:
        continue
    print('the loop has been ended' variable)
```

```
current letter :  P
current letter :  y
current letter :  t
current variable value -  10
the loop has been ended
current variable value -  9
the loop has been ended
current variable value -  8
the loop has been ended
current variable value -  7
the loop has been ended
current variable value -  6
current letter :  P
current letter :  y
current letter :  t
current letter :  o
current letter :  n
the loop has been ended 9
the loop has been ended 8
the loop has been ended 7
the loop has been ended 6
the loop has been ended 4
the loop has been ended 3
the loop has been ended 2
the loop has been ended 1
the loop has been ended 0
```

In [ ]:
```
# Assignment
# write the combination of if and else statement
# The statement search for prime numbers
# from 10 through
# Break condition
```

In [7]:
```
for i in range(10,20):
    for j in range(2, i // 2 + 1):
        if (i % j == 0):
            break
        print(i)
```

```
11
11
11
11
13
13
```

In [5]:
```
11 // 2
```

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

.format() method:

In [ ]:
```
- Format Method can be very practical to make Data,Numbers,Strings an
- And Sometimes it may be very useful to replace the values in string
- it can be used to place for multiple string also:
```

In [7]:
```
my_message  = 'Jessi and Jack will marry on {}'
wedding_date = 'may 20'
print(my_message.format(wedding_date))
user_message = 'Baseball Weights {},Football Weights {} and Basketbal
print(user_message.format(0.5,4.0,3.0))
lst = [3.8900,0.9087,5.8904,9.0909098989878687] # reduce decimal poin

# Format with map:
new_lst = map("{:,.2f}".format, lst)
print(list(new_lst))
```

```
Jessi and Jack will marry on may 20
Baseball Weights 0.5,Football Weights 4.0 and Basketball Weights 3.
0

---------------------------------------------------------------------
--------
TypeError                                 Traceback (most recent ca
ll last)
<ipython-input-7-67ed05b2f62e> in <module>
      8 # Format with map:
      9 new_lst = map("{:,.2f}".format, lst)
---> 10 print(list(new_lst))
     11 
     12 # Scientific format

TypeError: 'list' object is not callable
```

In [9]:
```
# Scientific format
#{:e}.format()

a = "{:e}".format(9.000000003433)
print(a)
```

```
9.000000e+00
```

In [13]:
```
 # {:_}.format()
a = "{:_}".format(769434276274)
print(a)
```

```
769_434_276_274
```

#list comprehension:

In [ ]:
```
### list comprehension:
- list Comprehension are used to create a new list from an existing 
- It is very easier to implement when comparing to for loop
- The code is Actually a single line Extension
```

In [15]:
```
letter_list = []
for i in 'sure trust':
    letter_list.append(i)
print(letter_list)

new_list = [i for i in 'sure trust']
new_list
```

```
['s', 'u', 'r', 'e', ' ', 't', 'r', 'u', 's', 't']
```

Out[15]:
```
['s', 'u', 'r', 'e', ' ', 't', 'r', 'u', 's', 't']
```

In [30]:
```
list_1 = ['yes','it\'s','a','beautiful','day']
# normal string has upper lower title
# list doesn't have upper lower
upper_list = [i.title() for i in list_1]
print(upper_list)
upper_list = [i.upper() for i in list_1]
print(upper_list)
upper_list = [i.lower() for i in list_1]
print(upper_list)

list_2 = [1,2,3,4]
square = [[i**2,i**3] for i in list_2]
print(square)
```

```
['Yes', "It'S", 'A', 'Beautiful', 'Day']
['YES', "IT'S", 'A', 'BEAUTIFUL', 'DAY']
['yes', "it's", 'a', 'beautiful', 'day']
[[1, 1], [4, 8], [9, 27], [16, 64]]
```

In [40]:
```
# Assignment
# print the table of number for 13 to 17 by list comprehension
table = [[i,'x',j,'=',i*j] for i in range(13,18) for j in range(1,11
print(table)
```

```
[[13, 'x', 1, '=', 13], [13, 'x', 2, '=', 26], [13, 'x', 3, '=', 3
9], [13, 'x', 4, '=', 52], [13, 'x', 5, '=', 65], [13, 'x', 6, '=',
78], [13, 'x', 7, '=', 91], [13, 'x', 8, '=', 104], [13, 'x', 9, '=
', 117], [13, 'x', 10, '=', 130], [14, 'x', 1, '=', 14], [14, 'x',
2, '=', 28], [14, 'x', 3, '=', 42], [14, 'x', 4, '=', 56], [14, 'x
```

In [21]:

```
# different methodology
products = ['Apparels','Crockeries','Cosmetics','Beverages','Detergen
# Read the first letter of each word:

first_letter = [i for i in products[0:1]]
print(first_letter)
first_letter = [i for i in products[0]]
print(first_letter)
first_letter = [i[0] for i in products]
print(first_letter)
word = [i for i in enumerate(products)]
print(word)
word = [i for i in enumerate(products[0:1])]
print(word)
word = [i for i in enumerate(products[0])]
print(word)
#Keyword
[i for products i in enumerate(products[0:1])]
```

```
['Apparels']
['A', 'p', 'p', 'a', 'r', 'e', 'l', 's']
['A', 'C', 'C', 'B', 'D']
[(0, 'Apparels'), (1, 'Crockeries'), (2, 'Cosmetics'), (3, 'Beverag
es'), (4, 'Detergents')]
[(0, 'Apparels')]
[(0, 'A'), (1, 'p'), (2, 'p'), (3, 'a'), (4, 'r'), (5, 'e'), (6, 'l
'), (7, 's')]
```

Out[21]:

```
['Apparels']
```

In [31]:

```
# read the first three letter of each word:
products = ['Apparels','Crockeries','Cosmetics','Beverages','Detergen
wo = [i[0:3] for i in enumerate(products)]
print(wo)
```

```
[(0, 'Apparels'), (1, 'Crockeries'), (2, 'Cosmetics'), (3, 'Beverag
es'), (4, 'Detergents')]
```

In [30]:

```
# Iteration over more than one iterable in a list
# By using list Comprehension
products = ['Apparels','Crockeries','Cosmetics','Beverages','Detergen
Average_prices = [750,1200,600]
pair_price = [(p,a) for p in products for a in Average_prices]
print(pair_price)
```

```
[('Apparels', 750), ('Apparels', 1200), ('Apparels', 600), ('Crocke
ries', 750), ('Crockeries', 1200), ('Crockeries', 600), ('Cosmetics
', 750), ('Cosmetics', 1200), ('Cosmetics', 600), ('Beverages', 75
0), ('Beverages', 1200), ('Beverages', 600), ('Detergents', 750),
('Detergents', 1200), ('Detergents', 600)]
```

In [3]:

```
# list Comprehension using conditional logic:
item_prices = [56,45,455,23,34,90,567]
# find the prices of items which is more then 100
prices_greater = [i if i>100 else 1 for i in item_prices]
```

```
print(prices_greater)

# extract all the digits from the string:
# By using list comprehension


text = 'I love to have 10 ice creamss along with 30 milk shakes'
number = [i for i in text if i.isdigit()]
print(number)

text = 'I love to have 10 ice creamss along with 30 milk shakes'
number = [int(i) for i in text.split() if i.isdigit() == True]
print(number)
text = 'I love to have 10 ice creamss along with 30 milk shakes'
number = [int(i) for i in text.split() if i.isdigit() == False]
print(number)
# isdigit should be true
```

```
[1, 1, 455, 1, 1, 1, 567]
['1', '0', '3', '0']
[10, 30]

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-3-566bcbf96d8b> in <module>
     17 print(number)
     18 text = 'I love to have 10 ice creamss along with 30 milk shakes'
---> 19 number = [int(i) for i in text.split() if i.isdigit() == False]
     20 print(number)

<ipython-input-3-566bcbf96d8b> in <listcomp>(.0)
     17 print(number)
     18 text = 'I love to have 10 ice creamss along with 30 milk shakes'
---> 19 number = [int(i) for i in text.split() if i.isdigit() == False]
     20 print(number)

ValueError: invalid literal for int() with base 10: 'I'
```

In [6]:

```
# extract all the vowels from the string:
# By using list comprehension


sentence = 'where the mind is without fear'
number = [i for i in sentence if i in 'aeiou']
print(number)
sentence = 'where the mind is without fear'
number = [i for i in sentence if i is 'aeiou']
print(number)
```

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

In [8]:

```
# Nested condition with list comp:
products = ['Apparels','Crockeries','Cosmetics','Beverages','Deterge
# Discount by products category?
```

```
Discount = [10 if i == 'Apparels' else 15 if i == 'Crockeries' else 2
print(Discount)
```

```
[10, 15, 20, 20, 20]
```

In [54]:
```
# Find the common numbers in two lists(without using a tuple or set)
list_a = 1,2,3,4
list_b = 2,3,4,5
common_numbers = [i for i in list_a if i in list_b]
print(common_numbers)
```

```
[2, 3, 4]
```

In [19]:
```
#31/03/22
# 1.find all of the numbers from 1-1000 that are divisible by 7
numbers = [i for i in range(1,1000) if i%7 == 0]
print(numbers)
```

```
[7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98, 105, 112, 1
19, 126, 133, 140, 147, 154, 161, 168, 175, 182, 189, 196, 203, 21
0, 217, 224, 231, 238, 245, 252, 259, 266, 273, 280, 287, 294, 301,
308, 315, 322, 329, 336, 343, 350, 357, 364, 371, 378, 385, 392, 39
9, 406, 413, 420, 427, 434, 441, 448, 455, 462, 469, 476, 483, 490,
497, 504, 511, 518, 525, 532, 539, 546, 553, 560, 567, 574, 581, 58
8, 595, 602, 609, 616, 623, 630, 637, 644, 651, 658, 665, 672, 679,
686, 693, 700, 707, 714, 721, 728, 735, 742, 749, 756, 763, 770, 77
7, 784, 791, 798, 805, 812, 819, 826, 833, 840, 847, 854, 861, 868,
875, 882, 889, 896, 903, 910, 917, 924, 931, 938, 945, 952, 959, 96
6, 973, 980, 987, 994]
```

In [23]:
```
# 2.find all the numbers from 1-1000 that have a 3 in them
numbers = [i for i in range(1,1000) if '3' in str(i)]
print(numbers)
```

```
[3, 13, 23, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 43, 53, 63, 73,
83, 93, 103, 113, 123, 130, 131, 132, 133, 134, 135, 136, 137, 138,
139, 143, 153, 163, 173, 183, 193, 203, 213, 223, 230, 231, 232, 23
3, 234, 235, 236, 237, 238, 239, 243, 253, 263, 273, 283, 293, 300,
301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 31
4, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327,
328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 34
1, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354,
355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 36
8, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381,
382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 39
5, 396, 397, 398, 399, 403, 413, 423, 430, 431, 432, 433, 434, 435,
436, 437, 438, 439, 443, 453, 463, 473, 483, 493, 503, 513, 523, 53
0, 531, 532, 533, 534, 535, 536, 537, 538, 539, 543, 553, 563, 573,
583, 593, 603, 613, 623, 630, 631, 632, 633, 634, 635, 636, 637, 63
8, 639, 643, 653, 663, 673, 683, 693, 703, 713, 723, 730, 731, 732,
733, 734, 735, 736, 737, 738, 739, 743, 753, 763, 773, 783, 793, 80
3, 813, 823, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 843,
853, 863, 873, 883, 893, 903, 913, 923, 930, 931, 932, 933, 934, 93
5, 936, 937, 938, 939, 943, 953, 963, 973, 983, 993]
```

In [25]:
```
#3.count the number of spaces in a string
string = 'iidf jfkddk dfnk'
spaces = [i for i in string if i == ' ']
print(len(spaces))
```

```
2
```

In [46]:
```
# 4.Create a list of all consonants in the string "Yellow Yaks like Y
# they yodled  while eating yuky yams"
string = "Yellow Yaks like Yelling and yawning and yesturday they yod
consonants = [i for i in string if i not in 'aeious']
print(consonants)
```

```
['Y', 'l', 'l', 'w', ' ', 'Y', 'k', ' ', 'l', 'k', ' ', 'Y', 'l', '
l', 'n', 'g', ' ', 'n', 'd', ' ', 'y', 'w', 'n', 'n', 'g', ' ', 'n
', 'd', ' ', 'y', 't', 'r', 'd', 'y', ' ', 't', 'h', 'y', ' ', 'y',
'd', 'l', 'd', ' ', ' ', 'w', 'h', 'l', ' ', 't', 'n', 'g', ' ', 'y
', 'k', 'y', ' ', 'y', 'm']
```

In [36]:
```
# 5.Get the index and the values as a tuple for items in the list "hi
#result would look like(index,value)
items = ("hi",4,8.99,'apple',('t','b','n'))
a = [(items.index(i),i) for i in items]
print(a)
```

```
[(0, 'hi'), (1, 4), (2, 8.99), (3, 'apple'), (4, ('t', 'b', 'n'))]
```

In [45]:
```
#6. Get only the numbers in a sentence like 'in 1984 there were 13 in
text = 'in 1984 there were 13 instances of a protest with over 1000 p
number = [int(i) for i in text.split() if i.isdigit() == True]
print(number)
```

```
[1984, 13, 1000]
```

In [50]:
```
# 7.Given numbers = range(20),products a list containing the word 'ev
# Result would look like 'odd','odd','even'
number = ['even' if i%2 == 0 else 'odd' for i in range(20)]
print(number)
```

```
['even', 'odd', 'even', 'odd', 'even', 'odd', 'even', 'odd', 'even
', 'odd', 'even', 'odd', 'even', 'odd', 'even', 'odd', 'even', 'odd
', 'even', 'odd']
```

In [53]:
```
# 8.Produce a list of tuples consisting of only the matching numbers
# list_b = 2,7,1,12.Result would look like(4,4),(12,12)
list_a = [1, 2, 3,4,5,6,7,8,9]
list_b = [2, 7, 1, 12]

matching_numbers = [(a, b) for a in list_a for b in list_b if a == b]
print(matching_numbers)
```

```
[(1, 1), (2, 2), (7, 7)]
```

In [60]:
```
# 9. Find all of the words in a string that are less then 4 letters
strings = 'dsfsd sdf sdfds sfgds yh d'
word = strings.split()
b = [i for i in word if len(i) < 4]
print(b)
```

```
['sdf', 'yh', 'd']
```

In [64]:
```
# Use a nested list comprehension to find all of the numbers from 1-1
number = [num for num in range(1,1001) if [div for div in range(2,10)
```

```
[2, 3, 4, 4, 5, 6, 6, 6, 7, 8, 8, 8, 9, 10, 10, 12, 12, 12, 12, 14,
14, 15, 15, 16, 16, 16, 18, 18, 18, 20, 20, 20, 21, 21, 22, 24, 24,
24, 24, 24, 25, 26, 27, 28, 28, 28, 30, 30, 30, 30, 32, 32, 32, 33,
34, 35, 35, 36, 36, 36, 36, 38, 39, 40, 40, 40, 40, 42, 42, 42, 42,
44, 44, 45, 45, 46, 48, 48, 48, 48, 48, 49, 50, 50, 51, 52, 52, 54,
54, 54, 55, 56, 56, 56, 56, 57, 58, 60, 60, 60, 60, 60, 62, 63, 63,
64, 64, 64, 65, 66, 66, 66, 68, 68, 69, 70, 70, 70, 72, 72, 72, 72,
72, 74, 75, 75, 76, 76, 77, 78, 78, 78, 80, 80, 80, 80, 81, 82, 84,
84, 84, 84, 84, 85, 86, 87, 88, 88, 88, 90, 90, 90, 90, 91, 92, 92,
93, 94, 95, 96, 96, 96, 96, 96, 98, 98, 99, 100, 100, 100, 102, 10
2, 102, 104, 104, 104, 105, 105, 105, 106, 108, 108, 108, 108, 110,
110, 111, 112, 112, 112, 112, 114, 114, 114, 115, 116, 116, 117, 11
8, 119, 120, 120, 120, 120, 120, 120, 122, 123, 124, 124, 125, 126,
126, 126, 126, 128, 128, 128, 129, 130, 130, 132, 132, 132, 132, 13
3, 134, 135, 135, 136, 136, 136, 138, 138, 138, 140, 140, 140, 140,
141, 142, 144, 144, 144, 144, 144, 145, 146, 147, 147, 148, 148, 15
0, 150, 150, 150, 152, 152, 152, 153, 154, 154, 155, 156, 156, 156,
156, 158, 159, 160, 160, 160, 160, 161, 162, 162, 162, 164, 164, 16
5, 165, 166, 168, 168, 168, 168, 168, 168, 170, 170, 171, 172, 172,
174, 174, 174, 175, 175, 176, 176, 176, 177, 178, 180, 180, 180, 18
0, 180, 182, 182, 183, 184, 184, 184, 185, 186, 186, 186, 188, 188,
189, 189, 190, 190, 192, 192, 192, 192, 192, 194, 195, 195, 196, 19
6, 196, 198, 198, 198, 200, 200, 200, 200, 201, 202, 203, 204, 204,
204, 204, 205, 206, 207, 208, 208, 208, 210, 210, 210, 210, 210, 21
2, 212, 213, 214, 215, 216, 216, 216, 216, 216, 217, 218, 219, 220,
220, 220, 222, 222, 222, 224, 224, 224, 224, 225, 225, 226, 228, 22
8, 228, 228, 230, 230, 231, 231, 232, 232, 232, 234, 234, 234, 235,
236, 236, 237, 238, 238, 240, 240, 240, 240, 240, 240, 242, 243, 24
4, 244, 245, 245, 246, 246, 246, 248, 248, 248, 249, 250, 250, 252,
252, 252, 252, 252, 254, 255, 255, 256, 256, 256, 258, 258, 258, 25
9, 260, 260, 260, 261, 262, 264, 264, 264, 264, 264, 265, 266, 266,
267, 268, 268, 270, 270, 270, 270, 272, 272, 272, 273, 273, 274, 27
5, 276, 276, 276, 276, 278, 279, 280, 280, 280, 280, 280, 282, 282,
282, 284, 284, 285, 285, 286, 287, 288, 288, 288, 288, 288, 290, 29
0, 291, 292, 292, 294, 294, 294, 294, 295, 296, 296, 296, 297, 298,
300, 300, 300, 300, 300, 301, 302, 303, 304, 304, 304, 305, 306, 30
6, 306, 308, 308, 308, 309, 310, 310, 312, 312, 312, 312, 312, 314,
315, 315, 315, 316, 316, 318, 318, 318, 320, 320, 320, 320, 321, 32
2, 322, 324, 324, 324, 324, 325, 326, 327, 328, 328, 328, 329, 330,
330, 330, 330, 332, 332, 333, 334, 335, 336, 336, 336, 336, 336, 33
6, 338, 339, 340, 340, 340, 342, 342, 342, 343, 344, 344, 344, 345,
345, 346, 348, 348, 348, 348, 350, 350, 350, 351, 352, 352, 352, 35
4, 354, 354, 355, 356, 356, 357, 357, 358, 360, 360, 360, 360, 360,
360, 362, 363, 364, 364, 364, 365, 366, 366, 366, 368, 368, 368, 36
9, 370, 370, 371, 372, 372, 372, 372, 374, 375, 375, 376, 376, 376,
378, 378, 378, 378, 380, 380, 380, 381, 382, 384, 384, 384, 384, 38
4, 385, 385, 386, 387, 388, 388, 390, 390, 390, 390, 392, 392, 392,
392, 393, 394, 395, 396, 396, 396, 396, 398, 399, 399, 400, 400, 40
0, 400, 402, 402, 402, 404, 404, 405, 405, 406, 406, 408, 408, 408,
408, 408, 410, 410, 411, 412, 412, 413, 414, 414, 414, 415, 416, 41
6, 416, 417, 418, 420, 420, 420, 420, 420, 420, 422, 423, 424, 424,
424, 425, 426, 426, 426, 427, 428, 428, 429, 430, 430, 432, 432, 43
2, 432, 432, 434, 434, 435, 435, 436, 436, 438, 438, 438, 440, 440,
440, 440, 441, 441, 442, 444, 444, 444, 444, 445, 446, 447, 448, 44
8, 448, 448, 450, 450, 450, 450, 452, 452, 453, 454, 455, 455, 456,
456, 456, 456, 456, 458, 459, 460, 460, 460, 462, 462, 462, 462, 46
4, 464, 464, 465, 465, 466, 468, 468, 468, 468, 469, 470, 470, 471,
472, 472, 472, 474, 474, 474, 475, 476, 476, 476, 477, 478, 480, 48
0, 480, 480, 480, 480, 482, 483, 483, 484, 484, 485, 486, 486, 486,
488, 488, 488, 489, 490, 490, 490, 492, 492, 492, 492, 494, 495, 49
```

```
5, 496, 496, 496, 497, 498, 498, 498, 500, 500, 500, 501, 502, 504,
504, 504, 504, 504, 504, 505, 506, 507, 508, 508, 510, 510, 510, 51
0, 511, 512, 512, 512, 513, 514, 515, 516, 516, 516, 516, 518, 518,
519, 520, 520, 520, 520, 522, 522, 522, 524, 524, 525, 525, 525, 52
6, 528, 528, 528, 528, 528, 530, 530, 531, 532, 532, 532, 534, 534,
534, 535, 536, 536, 536, 537, 538, 539, 540, 540, 540, 540, 540, 54
2, 543, 544, 544, 544, 545, 546, 546, 546, 546, 548, 548, 549, 550,
550, 552, 552, 552, 552, 552, 553, 554, 555, 555, 556, 556, 558, 55
8, 558, 560, 560, 560, 560, 560, 561, 562, 564, 564, 564, 564, 565,
566, 567, 567, 568, 568, 568, 570, 570, 570, 570, 572, 572, 573, 57
4, 574, 575, 576, 576, 576, 576, 576, 578, 579, 580, 580, 580, 581,
582, 582, 582, 584, 584, 584, 585, 585, 586, 588, 588, 588, 588, 58
8, 590, 590, 591, 592, 592, 592, 594, 594, 594, 595, 595, 596, 596,
597, 598, 600, 600, 600, 600, 600, 600, 602, 602, 603, 604, 604, 60
5, 606, 606, 606, 608, 608, 608, 609, 609, 610, 610, 612, 612, 612,
612, 614, 615, 615, 616, 616, 616, 616, 618, 618, 618, 620, 620, 62
0, 621, 622, 623, 624, 624, 624, 624, 624, 625, 626, 627, 628, 628,
630, 630, 630, 630, 630, 632, 632, 632, 633, 634, 635, 636, 636, 63
6, 636, 637, 638, 639, 640, 640, 640, 640, 642, 642, 642, 644, 644,
644, 645, 645, 646, 648, 648, 648, 648, 648, 650, 650, 651, 651, 65
2, 652, 654, 654, 654, 655, 656, 656, 656, 657, 658, 658, 660, 660,
660, 660, 660, 662, 663, 664, 664, 664, 665, 665, 666, 666, 666, 66
8, 668, 669, 670, 670, 672, 672, 672, 672, 672, 672, 674, 675, 675,
676, 676, 678, 678, 678, 679, 680, 680, 680, 680, 681, 682, 684, 68
4, 684, 684, 685, 686, 686, 687, 688, 688, 688, 690, 690, 690, 690,
692, 692, 693, 693, 694, 695, 696, 696, 696, 696, 696, 698, 699, 70
0, 700, 700, 700, 702, 702, 702, 704, 704, 704, 705, 705, 706, 707,
708, 708, 708, 708, 710, 710, 711, 712, 712, 712, 714, 714, 714, 71
4, 715, 716, 716, 717, 718, 720, 720, 720, 720, 720, 720, 721, 722,
723, 724, 724, 725, 726, 726, 726, 728, 728, 728, 728, 729, 730, 73
0, 732, 732, 732, 732, 734, 735, 735, 735, 736, 736, 736, 738, 738,
738, 740, 740, 740, 741, 742, 742, 744, 744, 744, 744, 744, 745, 74
6, 747, 748, 748, 749, 750, 750, 750, 750, 752, 752, 752, 753, 754,
755, 756, 756, 756, 756, 756, 758, 759, 760, 760, 760, 760, 762, 76
2, 762, 763, 764, 764, 765, 765, 766, 768, 768, 768, 768, 768, 770,
770, 770, 771, 772, 772, 774, 774, 774, 775, 776, 776, 776, 777, 77
7, 778, 780, 780, 780, 780, 780, 782, 783, 784, 784, 784, 784, 785,
786, 786, 786, 788, 788, 789, 790, 790, 791, 792, 792, 792, 792, 79
2, 794, 795, 795, 796, 796, 798, 798, 798, 798, 800, 800, 800, 800,
801, 802, 804, 804, 804, 804, 805, 805, 806, 807, 808, 808, 808, 81
0, 810, 810, 810, 812, 812, 812, 813, 814, 815, 816, 816, 816, 816,
816, 818, 819, 819, 820, 820, 820, 822, 822, 822, 824, 824, 824, 82
5, 825, 826, 826, 828, 828, 828, 828, 830, 830, 831, 832, 832, 832,
833, 834, 834, 834, 835, 836, 836, 837, 838, 840, 840, 840, 840, 84
0, 840, 840, 842, 843, 844, 844, 845, 846, 846, 846, 847, 848, 848,
848, 849, 850, 850, 852, 852, 852, 852, 854, 854, 855, 855, 856, 85
6, 856, 858, 858, 858, 860, 860, 860, 861, 861, 862, 864, 864, 864,
864, 864, 865, 866, 867, 868, 868, 868, 870, 870, 870, 870, 872, 87
2, 872, 873, 874, 875, 875, 876, 876, 876, 876, 878, 879, 880, 880,
880, 880, 882, 882, 882, 882, 884, 884, 885, 885, 886, 888, 888, 88
8, 888, 888, 889, 890, 890, 891, 892, 892, 894, 894, 894, 895, 896,
896, 896, 896, 897, 898, 900, 900, 900, 900, 900, 902, 903, 903, 90
4, 904, 904, 905, 906, 906, 906, 908, 908, 909, 910, 910, 910, 912,
912, 912, 912, 912, 914, 915, 915, 916, 916, 917, 918, 918, 918, 92
0, 920, 920, 920, 921, 922, 924, 924, 924, 924, 924, 925, 926, 927,
928, 928, 928, 930, 930, 930, 930, 931, 932, 932, 933, 934, 935, 93
6, 936, 936, 936, 936, 938, 938, 939, 940, 940, 940, 942, 942, 942,
944, 944, 944, 945, 945, 945, 946, 948, 948, 948, 948, 950, 950, 95
1, 952, 952, 952, 952, 954, 954, 954, 955, 956, 956, 957, 958, 959,
960, 960, 960, 960, 960, 960, 962, 963, 964, 964, 965, 966, 966, 96
```

In [16]:
```
# Dictionary
my_dict = {1:'apple',2:'Ball'}
print(my_dict)
# Dictionary comprehension
# Dictionary comprehension is an elegent and concise way to create Di
square_dict = dict()
for num in range(1,11):
    square_dict[num] = num*num
print(square_dict) # old method
square_dict = {num : num*num for num in range(1,11)}
#              key    value      variable  condition
print(square_dict)

# item_price in dollars
old_price = {'Milk' : 1.02,'Coffee' : 2.5,'Bread' : 2.3}
# Dollar to pound:
# Dictionary Comprehension
new_price = {key : num * 0.81  for key in old_price.keys() for num in
print(new_price)

dollar_to_pound = 0.81
new_price = {i : value*dollar_to_pound for (i,value) in old_price.ite
print(new_price)
```

```
{1: 'apple', 2: 'Ball'}
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 10
0}
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 10
0}
{'Milk': 1.863, 'Coffee': 1.863, 'Bread': 1.863}
{'Milk': 0.8262, 'Coffee': 2.0250000000000004, 'Bread': 1.863}
```

In [21]:
```
# Assignment
original_dict = {'jack' : 38,'Michal' : 48,'Sara' : 57,'John' : 33}
print(original_dict)
# 1.Have to take only  even numbers from the key_value pairs
#2.Have to take the values which are not even and lesser than 40
new = {i : value for (i,value) in original_dict.items() if value%2 ==
print(new)
new = {i : value for (i,value) in original_dict.items() if value%2 !=
print(new)
```

```
{'jack': 38, 'Michal': 48, 'Sara': 57, 'John': 33}
{'jack': 38, 'Michal': 48}
{'John': 33}
```

User Define Function

1.A Function that a user defines in a program is know as user define function

2.A user can give your to a user defined function ,Howerver a function name shouldnot have a space or special character

By using def keyword to define a function

In [2]:
```
def area_of_triangle(base,height):
                    #arguments
```

```
        return(0.5*base*height)
# def is the keyword
# fuctional value -area_of_triangle
# Area_of_triangle =1/2*b*h  or 0.5*b*h  base height
# Return - Inbuit keyword in Python -

print('Area of triangle',area_of_triangle(4,5))
```

```
Area of triangle 10.0
```

In [4]:
```
# Write a Function without any arguments:
def greetings():
    print('Hello, Have a wonderful Day Ahead')
greetings()
```

```
Hello, Have a wonderful Day Ahead
```

In [20]:
```
#Write a Function with arguments:
def hello(name):
    print('hallo, '+ name + '  Have a wonderful Day Ahead')
hello('steve')

# A Function with return keyword
def product(x,y):
    c = x*y
    return c
print(product(20,23))

# A Function without return keyword
def product(x,y):
    c = x*y
    return()
a = product(20,23)
print(a)
# A Function without return keyword
def product(x,y):
    c = x*y
    print(c)
print(product(20,23))
```

```
hallo, steve  Have a wonderful Day Ahead
460
()
460
None
```

In [ ]:
```
def prod(x,y):
    x = int(input('enter the number '))
    y= int(input('enter the number'))
    c = x*y
    return c
print(prod(x,y))
```

In [3]:
```
# Assignment 03-04-23
# Find the largest number inside the list without using sort functior
# I =[2,3,4,5,6,7]

def largest(list1):
    lar = list1[0]
    for j in list1:
        if j > lar:
            lar = j
    return lar
list1 = [2,3,4,5,6,7]
print(largest(list1))
```

```
7
```

In [2]:
```
# keyword Aruguments:
def employee(name, designation):
    print(name,designation)
employee(name = 'john',designation = 'CEO')
employee(designation = 'CEO',name = 'john')
employee(name = 'CEO',designation = 'john')
```

```
john CEO
john CEO
CEO john
```

In [8]:
```
def employee(name = 'john',salary = 4000):
    return('Employee Name -',name)
    return('Employee Salary - ',salary)
employee('john')
```

Out[8]:
```
('Employee Name -', 'john')
```

In [15]:
```
# Variable -  length arguments
# (*args)  numerical conversions # it is used for only numbers
# (** args)  to be  used  if name Arguments are to be passed in a fu
def daily_temperature(temp):
    for var in  temp:
        print(var,end = ' ')
daily_temperature(str(10))

# asterix arguments
def daily_temperature(*temp):
    for var in  temp:
        print(var,end = ' ')
daily_temperature(10,20,30,40)

def my_function(**krgs):
    print(type(krgs))
    for i,j in krgs.items():
        print(i, '==',j)
my_function(firstname = 'john' ,second_name = 'alen',salary = 20000,P
```

```
1 0 10 20 30 40 <class 'dict'>
firstname == john
second_name == alen
salary == 20000
PF == 450
```

lambda Function

-Lambda functions are anonymous ie, to say they have no names

-The keyword is lambda

-It is simply one line function

-No def or return keyword to be used with lambda

In [19]:
```
def fun(x,y):
    if(x>y):
        return x
    else:
        return y
print(fun(3,4))

#Using lambda function
fun = lambda x,y: x if x > y else y
print(fun(3,4))

x = lambda a,c : a * c
print(x(5,6))
```

```
4
4
30
```

In [ ]:
```
# Assignment
find life excectancy calculater
def new_life(name,age): ["jane",'Zack','Melissa']
smoker age 40 non smoker age 70
life remaining = life_exp - age
output: hii Jane! your life expectancy:-----years
```

In [32]:
```
list = ["jane",'Zack','Melissa']
def new_life(name,age):
    life_exp = int(input("enter the life_exp:"))
    life_remaining = life_exp - age
    if life_remaining >= 70:
        print('non smoker is:',life_remaining)
    elif life_remaining <= 40:
        print('smoker is:',life_remaining)
    return('hii Jane! your life expectancy:',life_remaining,'years')
print(new_life('jane',40))
```

```
enter the life_exp:90
('hii Jane! your life expectancy:', 50, 'years')
```

In [5]:
```
smoker_age = 40
    non_smoker_age = 70
    if i in krgs:
        v = input("enter he is a smoker or not")
```

```
  File "<ipython-input-5-2dfaa91edc84>", line 2
    non_smoker_age = 70
    ^
IndentationError: unexpected indent
```

In [6]:
```
### Lambda with Map():
-It actually executes the functional objects each element in the sequ
```

```
  File "<ipython-input-6-cffe6473f9d0>", line 2
    -It actually executes the functional objects each element in th
e sequence and returns a sequence
                ^
SyntaxError: invalid syntax
```

In [15]:
```
seq = list()
sample_list1 = [1,2,3,4]
sample_list2 = [5,6,7,8]
sample_tuple = (10,11,12,13)
seq = list(map(lambda x : x*2,(sample_list1,sample_list2,sample_tuple
seq
```

Out[15]:
```
[[1, 2, 3, 4, 1, 2, 3, 4],
 [5, 6, 7, 8, 5, 6, 7, 8],
 (10, 11, 12, 13, 10, 11, 12, 13)]
```

The Lambda with Filter:

-the filter() function expects two arguments -it returns only those elements for which the functional_object returns True

In [13]:
```
num_list = list(range(15))
seq  = list(filter(lambda x : x % 3 == 0,num_list))
seq
```

Out[13]:
```
[0, 3, 6, 9, 12]
```

In [ ]:
```
### The Lambda function with Reduce():

-The reduce() Function in Python takes in a function and a Sequence a
-The Function is called with a lambda Function and a Sequence
-A New Reduce results is Performed.
-This Performe a repetitive operation over the pair of the sequential
```

In [16]:
```
from functools import reduce
reduce(lambda a,b : a + b,[3,5,8,10]) #3+5 = 8 and 8+8 =16 and 16+10=
```

Out[16]:
```
26
```

In [17]:
```
num_tuple = (1,0,3,-1,5,6,10,-5)
reduce(lambda x,y : x if (x>y) else y,num_tuple)
# Note:Reduce() can only have iterables of same type of input
```

Out[17]:
```
10
```

In [ ]:
```
**Note:Reduce() can only have iterables of same type of input**
```

In [2]:
```
# Assignment
#Write a python program to sort a list of tuple using lambda
#original list
list1 = [('English', 88), ('Science', 90), ('Maths', 97), ('Social sc
list1.sort(key = lambda x: x[1])
print(list1)
```

```
[('Social sciences', 82), ('English', 88), ('Science', 90), ('Maths
', 97)]
```

2.write a python programe to find whether a

given string starts with a given character #using lambda

In [ ]:
```
numpy - numerical
statics
panda - virtual
machine learinig algorithm
```

#Write a pyhon program to add two given lists using map and lamdba original list [1,2,3] [4,5,6] Result:after adding two list[5,6,7]

In [6]:
```
list1 = [1,2,3]
list2 = [4,5,6]
list(map(lambda x, y: x + y, list1, list2))
```

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

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: