```
#Learning python from scratch
SAGAR KALAUNI
southern Illinois University Edwardsville
statistics || Data Science
Teaching Assistant @SIUE

Chapter-1:5 PYTHON PROGRAMMING BASICS
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# How to print the command in the python
print("Hello world, I am sagar")

# how to assign the value in python
x=1
print(x)

x=x+1
print(x)
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Adding two numbers and two stings using python
# adding two numbers
a=2+3
print(a)

# adding two strings
Name= "sagar" + " " + "Kalauni"
print(Name)
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
# how to look our variable data type in python
x=32
type(x)

y="Sagar"
type(y)
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Data type conversion using python
x=35
float(x)

sval="123"
type(sval)

ival=int(sval)
type(ival)

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
# There are 4 basic type of coding practise, out of which we will discuss about first 3 now, they are
# Sequential coding, conditional coding and Repeted coding. and fourth one is store and reuse


# sequential code
# will be excuted in a sequence
x=2
print(x)
x=x+2
print(x)
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
# conditional steps code
x=5
if x<10:
```

```python
 print("smaller")
if x>20:
 print("greater")
print("finish")

-------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Repeted step coding
n=5
while n>0:
 print(n)
 n=n-1
print("done")

# A big and nice programming code containg all these above in one
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Modulo operator in python
# let's not talk about simple addition and subtraction operator in python, instead directly talk about the modulo operator in python
y=53
z= y % 2  # this will return the reminder when divided by 2, can be used to pick the small number from big, and other code development ideas
------------------------------------------------------------------------------------------------------------------------------------------------------------------

#operator precedence in python: paranthesis--power--multiplication--addition-Left to right.
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
# INPUT FUNCTION in python
# We can instruct python to pause and read data from the user using the input()
# The input function always return strings

name=input("who are you?")
print("Welcome", name)
-------------------------------------------------------------------------------------------------------------------------------------------------------------------

# Time to write our very first code on the basis of the knowledge above
# Europe floor to US floor conversion in Elevitor

europe_floor=input("Europe floor")
us_floor= int(europe_floor) + 1
print("The equivalent Us floor is:", us_floor)

-------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Homework-1
# Write a program to prompt the user for hour and rate per hour to compute a gross pay?

Hour=input("Enter Hour: ")
Rate=input("Enter Rate per Hour: ")
Gross_pay= float(Hour) * float(Rate)
print(Gross_pay)
-------------------------------------------------------------------------------------------------------------------------------------------------------------------

# Try and except in python

string="sagar"
try:
    print("this lines runs")
    print("this also")
    print("this too")
    integer=int(string)                # At this line boom happens, so after that it will directly go to except part of the code
    print("let's see this line runs or not")
except:
    integer= 32

print(string, integer)

--------------------------------------------------------------------------------------------------------------------------------------------
# Homework-2

# Rewrite your pay program using try and except so that your program can handles non-numeric input gracefully by printing a message and exiting the program?

Hour= input("Enter hour: " )
Rate= input("Enter Rate: " )
try:
   float_Hour=float(Hour)       # the code blows at this point then will not even look at bellow part and directly go to the except part of the code
   float_rate=float(Rate)
except:
      print("Enter the Numeric Values!")
      quit()
gross_pay= float_Hour * float_rate
print(gross_pay)
--------------------------------------------------------------------------------------------------------------------------------------------

# Now here we will discuss about our fourth coding way i.e store and Reuse.
# For doing this we will create our own function having certain code, so that we did not have to type the same chunk of code again and again for the next time

def thing():
    print("do your stuff over here")
    print("print your stuff over here")          # till this part we have just created our own customized function

print("---------------------below I will call my thing() function--------------------")
thing()       # it's time to call our function
print("---------------------did you just see, I did not wrote code to print above stuff just call my thing function--------------------")
--------------------------------------------------------------------------------------------------------------------------------------------

# building a function and calling a function are two completely different things.
# once we built our own function, it will not give any output untill we call the function

# building a function[store part]
def song_lyrics():
            print("give me some sun shine \n give me some rain \n give me another chanse \n I wanna gorw up once again")

# Calling a function[reuse part]
song_lyrics()
--------------------------------------------------------------------------------------------------------------------------------------------
# Program to find the relationship
# defining a function
def relation(friend):
        if friend=="kashi":
            print("Friend")
        elif friend=="biru":
            print("Brother")
        elif friend=="yogesh":
            print("Nephew")
        elif friend=="owen":
            print("classmate")
        else:
            print("Unknown")
# calling our function back
relation('yogesh')
----------------------------------------------------------------------------------------------------------------------------------------------------------

# talking about return in our own customized function

def greet():
    return "The number you have dailed is currently busy\nplease dail later"
def system(call):
    if call=="busy":
        print(greet())

# calling the function back
system("busy")

#print(greet(), "sagar")
----------------------------------------------------------------------------------------------------------------------------------------------------------
# LOOPS AND ITERATIONS IN python
# while and for are the keywords used for doing this [definite and indefinite loop]

# infinite loop
# this may happen because of error in the code or sometimes you may need it, but definitely stay away from this if your computer is not good enough
# Its never a good idea to put your computer in a infinite loop
n=10
while n>0:
    print("Hie hello world")
    n=n+1   # for infinite loop

# Zero loop code
n=0
while n>0:
    print("hie hello world")
    Print("loop is working")
print("I am outside of the loop")
----------------------------------------------------------------------------------------------------------------------------------------------------------
# How to go out of the loop and execute the code after loop
# we can use the break function to exit loop
# let's write an infinite loop and and get out of the loop using break statement

# infinite loop with break to come out of the loop
while True:
    password=input("Enter your birth city: ")
    if password=="baitadi":
        break
print("Wellcome to your mobilephone")
----------------------------------------------------------------------------------------------------------------------------------------------------------
# continue statement in Python
# the continue statement ends the current iteration and jumps to the top of the loop and start the next iteration.
# break completely exit the loop while continue just stop at that point go back to the top to do next iteration. (does not move forward in ceratin iteration)

while True:
    password=input("Enter a password:")
    if password[0]="#":
        continue
    elif password="baitadi":
        break
print(your password)
----------------------------------------------------------------------------------------------------------------------------------------------------------
```

```
# definite loops in python/ looping using for

for i in [1,2,3,4]:
        print(i)
print("done")

for name in ["sagar", "dipendra", "suraj", "bijay"]:
    print(name)
-----------------------------------------------------------------------------------------
--------------------------------------------------------------------------
# Using the loop ideas to solve our problems
# 3,    41,    12,     9,     74,      15  find the largest among these numbers

# writing the code for find the lagrest number
largest_num_till_now=-1
print("befor", "max:-",largest_num_till_now)
for num in [1,5,96,2,17,99]:
    if largest_num_till_now< num:
        largest_num_till_now=num
        print(largest_num_till_now, num)
print("after", "max:-",largest_num_till_now)
-----------------------------------------------------------------------------------------
--------------------------------------------------------------------------
# Loop Idioms
# [1]counting the number of items in the list using the for loop

count=0
print("befor", count)
for item in [0,9,65,85,66,38,36,649,2165,5,32]:
    count= count + 1
print("after", count)
-----------------------------------------------------------------------------------------
--------------------------------------------------------------------------
# [2]adding the number of items in the list using the for loop

sum=0
print("befor", sum)
for items in [0,9,65,85,66,38,36,649,2165,5,32]:
    sum= sum + items
    Running_sum=sum
    print("Running sums",Running_sum)
print("after", sum)
-----------------------------------------------------------------------------------------
--------------------------------------------------------------------------
# Homework-3
# [3]finding the average of items in the list using the for loop
# Just remember the average is total_sum/ total_count

total_count=0
total_sum=0
for item in [0,9,65,85,66,38,36,649,2165,5,32]:
    total_count= total_count + 1
    total_sum= total_sum + item
print("the average is" total_sum/total_count)

-----------------------------------------------------------------------------------------
--------------------------------------------------------------------------
# [4]filtering the item from the list using the for loop
# Let's find the items which are greater then 100 from our list

count=0
for item in [0,9,65,85,66,38,36,649,2165,5,32]:
    if item > 100:
        count=count + 1
        print(count," greater then 100 number:-", item)

-----------------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------
# [5]searching for the particular item in the list using the boolean variable and for loop.
# boolean is the another variable type beside int, float and string and it has its value true or false

found=False
#print("before", found)
for item in [0,9,65,85,66,38,36,649,2165,5,32]:
    if item==649:
        found=True
print("List contain the number:-", found)
------------------------------------------------------------------------------------------------------------------------------------------------------------
# quiz- like we use largest_num_till_now=-1 in the code for finding the maximum number in the list. how can we modify that code to find the smallest number in the list
# It is way harder to say this is the maximum number so we introduce the concept of new number type call None.

smalles_num_till_now=None
for value in [55,9,65,85,66,38,36,649,2165,5,32]:
    if smalles_num_till_now is None:
        smalles_num_till_now= value
    elif value < smalles_num_till_now:
        smalles_num_till_now= value
    print(smalles_num_till_now, value)
print("smallest number in our list is:-", smalles_num_till_now)
------------------------------------------------------------------------------------------------------------------------------------------------------------
Homework-4
#Write a program that repeatedly prompts a user for input numbers until the user enters 'done'.
#Once 'done' is entered, print out the total sum, total number and average of the numbers
#If the user enters anything other than a valid number catch it with a try/except
#and put out an appropriate message and ignore the number.
#Enter 6, 5, bob, and 1 and match the output below.
# total_sum: 12, total_num: 3, average: 4

# infinite loop with break to come out of the loop
total_sum=0
total_count=0
while True:
    sval=input("Enter the number:-")
    if sval=="done":
        break
    try:
        fval= float(sval)
    except:
        print("Invalid Input")
        continue
    total_sum=total_sum + fval
    total_count= total_count + 1
    #print(total_sum,total_count)
print("total sum: ", total_sum, "total num: ",total_count, "average: ", total_sum/total_count)



Homework-5
#Write a program that repeatedly prompts a user for integer numbers until the user enters 'done'.
#Once 'done' is entered, print out the largest and smallest of the numbers.
#If the user enters anything other than a valid number catch it with a try/except
#and put out an appropriate message and ignore the number.
#Enter 7, 2, bob, 10, and 4 and match the output below.
#Invalid input
#Maximum is 10
```

```
#Minimum is 2

largest=None
smallest=None

while True:
    sval= input("inter a number:")
    if sval=="done":
        break
    try:
        int_val=int(sval)
        if smallest is None:
            smallest= int_val
        if int_val < smallest:
            smallest= int_val
        if largest is None:
            largest= int_val
        if int_val > largest:
             largest= int_val
    except:
        print("Invalid input")
        continue
print(smallest, largest)
-------------------------------------------------chapter-1 The End
------------------------------------------------------------------------------------------
--------
##########################################################################################
###########################################################################
#Learning python from scratch
SAGAR KALAUNI
southern Illinois University Edwardsville
statistics || Data Science
Teaching Assistant @SIUE

Chapter-6:10 PYTHON DATA STRUCTURE
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Looking inside the string
# How to access to character inside the string
# python start with zero position, keep that in mind before doing any position accessing
coding

fruit= "Apple"
fruit[3]         # can be numeric

x=5
fruit[x-2]       # can be equation
------------------------------------------------------------------------------------------
---------------------------------------------------------------------------
# length function in python

fruit= "pineapple"
len(fruit)         # it returns the number of character in the variable

# quiz- Write your own function to find the length of the character string
def length_count(variable):
    count=0
    for i in variable:
        count = count + 1
    return count

length_count('apple')
------------------------------------------------------------------------------------------
---------------------------------------------------------------------------
# Looping through the index

fruit= "Pineapple"
```

```python
index= 0
while index < len( fruit):
    letter= fruit[index]
    print(index, letter)
    index= index + 1
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------
# accessing the letters of the string using the for loop
fruit= "Pineapple"
for letter in fruit:
    print(letter)
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Homework-6
# Counting the numbers of letters (certain) in the words

fruit= "Pineapple"
count=0
for letter in fruit:
    if letter=="p":
        count= count + 1
print(count)
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Slicing of the string in the python
# [a:b] --> this means start with a and go upto b but not does not include b. {b is not included}

string= "Hello my name is sagar"

string[0:4]  # this means from H and go upto o but do not include it. {in simple 4th character is not included}

string[5:7]

string[5:1000000] # this is also okay for python but it will take more memory

string[ :4]    # from starting and up ot 4.

string[5: ]     # from 5th place till the end

string[ : ]     # whole string
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------

# string concatenation in python

name = "sagar"
full_name= name + "kalauni"
print(full_name)

full_name= "Sagar" + " " + "Kalauni"
print(full_name)
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------

# Using in as a logical operator in python
fruit = "Pineapple"

'p' in fruit

"s" in "sagar"

"t" in "sagar"

" " in "sagar"
-----------------------------------------------------------------------------------------------
```

```
---------------------------------------------------------------------------
# quiz- Write a program that will look for certain letter in the string and print out found is the letter is in the string

string= " Mahabharat"
if 'M' in string:
    print("found it")
else:
    print("not found")
---------------------------------------------------------------------------------------------------------------------------------------------------------------------

# string comprasion
# comparing the string means comparing the each letter of the string with other string
# a=1 and z=26 is the value for them, we will go in detail later  so 'z' > 'a'; bigger letter bigger number

'sagar'> 'bharat'

'suraj' < 'sagar'
---------------------------------------------------------------------------------------------------------------------------------------------------------------------
# quiz- short the name Ram, shyam and hari using string comprasion

name= input("Enter name")
if name == "Ram":
    print("my refrence name is:-", name)

if name > "Ram":
    print("this name", name, "comes after Ram")

if name < "Ram":
    print("this name", name, "comes before Ram")
---------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Lower and upper function for string

greet= " Hello world"
greet.lower()          # orginal is not changed, orginal is still same

greet                  # cross checked

converted= greet.lower()
converted

# for upper

greet= " Hello world"
greet.upper()          # orginal is not changed, orginal is still same

greet                  # cross checked

converted= greet.upper()
converted
---------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Similarly there are a lot of other libraries which we can use in the string, to know more check out theses

stuff= "hellow world"
type(stuff)

dir(stuff)
# you can play with all of these string libraries, i will some few of them, below

"AAAA".capitalize()            # make first letter capitalize

```

```
-----------------------------------------------------------------------------------------------------------------------------------------------------------
# find function in string libraries

fruit= "Pineapple"

fruit.find('l')
position=fruit.find('l')

print(position)

fruit.find('z') # if you look for something that is not there you will get -1

# find function can take two arguments: first one is what to find and second one is from where should I start
fruit.find('a', 'i')
-----------------------------------------------------------------------------------------------------------------------------------------------------------
# search and replace function in the string library
# seach and replace function works in such a way that, first argument we have to give is the iteam which need to be seached from the string and second argument is what is need ot be replaced with

string= "Hello Sagar"
string.replace('Sagar', 'Bharat')

string.replace('S', 'B')                # search for S in our string and replace it with B
-----------------------------------------------------------------------------------------------------------------------------------------------------------
# Strapping whitespace from the string using the strip() function in string library

word="     hello     sagar     "
word.lstrip()
word.rstrip()
word.strip()
-----------------------------------------------------------------------------------------------------------------------------------------------------------
# startswith() function in the string library
# sometime we need to find weather the line that start with particular word. we can use it over there

line= "Breadth there the man whose soul so die"
line.startswith('Breadth')

line.startswith('P')
-----------------------------------------------------------------------------------------------------------------------------------------------------------
# Homework-7
# bellow is the line given from the line only slice and extract the university short format

line= "From Augustin.Marcus@siue.edu Jan 4 2024 06:32PM"

# since we want the slice siue only from the above line, first we will find (look) for the starting position

spos = line.find('@')

# now secondly we need to find the ending position of the line

lpos = lind.find('.', spos)     # this code will look for the '.' starting from spos value in the line

# findally we get the exect slice we just have to slice it with proper start and end point
line[spos + 1 : lpos]
------------------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------
# Homework-8
# Take the following python code that store a string: string = "X-DSPAM-Confidence: 0.8475";
#Write code to extract the number at the end of the line below and convert the extracted value to a floating point number and print it out.

string = "X-DSPAM-Confidence:    0.8475"

spos= string.find(':')
lpos=string.find('5', spos)

sliced= string[spos + 1: lpos + 1]

num = sliced.strip()

num=float(num)
print(num)
----------------------------------------------------taking data from our computer----------------------------------------------------------------------------------
# new line character in python,
print('x\ny')

len('x\ny')     # new line character is a single character \n
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
# How to read file using python
# impnote, python treat filehandle as a sequence of lines
# file handle act as sequence of string where eaach line in the file is a string in the sequence
# open() does not actually read the file, it kind of give us the protal where we can look for the file
fhandle= open("C:/Users/Dell/Desktop/Learn with Friedns/Learning python from Zero/Data folder/mbox.txt")

print(fhandle) # it is a python object, we will talk about it later, for now just remember open does nto actually read the file but only give us protal to look at it
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
# WORKING DIRECTORY IN Python
# How to look for what is my working directory right now in python
pwd()

import os
os.getcwd()    # full form of cwd is current working directory

# how to change the working directory in python

import os
os.chdir('INside this write the path you want to set working directory to')
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------

# counting the number of lines in the paragraph

fhandle= open("C:/Users/Dell/Desktop/Learn with Friedns/Learning python from Zero/Data folder/mbox.txt")

count=0
for line in fhandle:
    count= count + 1
print(count)
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------

# how to read file using python

```

```python
fhandle= open("C:/Users/Dell/Desktop/Learn with Friedns/Learning python from Zero/Data folder/mbox.txt")
file=fhandle.read()
-----------------------------------------------------------------------------------------------------------------------------------------------
# searching through a file

fhandle= open("C:/Users/Dell/Desktop/Learn with Friedns/Learning python from Zero/Data folder/mbox.txt")

for line in fhandle:
    if line.startswith('From'):
        print(line)
-----------------------------------------------------------------------------------------------------------------------------------------------
# if you look at the output of the above code you can see that there is space between each line. this is because print statements adds new line (every time, always true)
# and also each line from the file has newline at the end
# quiz-How to deal with them?

fhandle= open("C:/Users/Dell/Desktop/Learn with Friedns/Learning python from Zero/Data folder/mbox.txt")

for line in fhandle:
    line= line.rstrip()
    if line.startswith("From"):
        print(line)

# we will do this kind of coding a lot in the future class so you need to understand first three line of code completely i.e loading a data file, looking through the data file and removing \n
-----------------------------------------------------------------------------------------------------------------------------------------------
# quiz- How you will write a python code that will print the exect the same output as above but with different logic?
# Idea is we can skip all the line that does not startswith From clause.
# skipping with continue

fhandle= open("C:/Users/Dell/Desktop/Learn with Friedns/Learning python from Zero/Data folder/mbox.txt")

for line in fhandle:
    line = line.rstrip()
    if not line.startswith("From"):
        continue
    print(line)
-----------------------------------------------------------------------------------------------------------------------------------------------
# using in to select lines

fhandle= open("C:/Users/Dell/Desktop/Learn with Friedns/Learning python from Zero/Data folder/mbox.txt")

for line in fhandle:
    line= line.rstrip()
    if "umich.edu" in line:
        print(line)
-----------------------------------------------------------------------------------------------------------------------------------------------
# quiz- how to get the same code with different logic
# exectly by skipping line with continue

fhandle= open("C:/Users/Dell/Desktop/Learn with Friedns/Learning python from Zero/Data folder/mbox.txt")

for line in fhandle:
    line= line.rstrip()
    if not "umich.edu" in line:
        continue
    print(line)
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
# proper coding format

fname= input("Enter a file name: ")
fhandle= open(fname)                # becasue input() always give string as the output

count= 0
for line in fhandle:
    count = count + 1
print(count)
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Homework-9
# Write a code to find how many lines starts with 'Subject' in mbox.txt and mbox-short.txt files?

fname= input("Enter a file name:")
fhandle= open(fname)

count=0
for line in fhandle:
    line= line.rstrip()
    if line.startswith("Subject"):
        count = count + 1
print(count)
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
# quiz- The above code will blow up if the user input the wrong file name, we don't want code to blow up, what can be done.
# Correct we can us try and except

fname= input("Enter a file name:")
try:
    fhandle= open(fname)
except:
    print("Invalid file name:-", fname)
    quit()

count=0
for line in fhandle:
    line= line.rstrip()
    if line.startswith("Subject"):
        count= count + 1
print(count)
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Homework-10
# Write a program to read through a file[mbox.txt] and print the content of the file ( line by line) all in upper case.

fname= input("Enter a file name:")
fhandle= open(fname)
for line in fhandle:
    line=line.rstrip()
    print(line.upper())
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
# LIST in the python
# till now we are talking about single variable, but python data structure can contain more then one variable in the form of list and other stuffs
# List in the python means the collection of the variables and the variables can be any string or numeric
# list is denoted by []
```

```python

num_list= [1,2,3,4]
print(num_list)
type(num_list)

var_list=['Ram','shyam','kanha']
print(var_list)
type(var_list)
------------------------------------------------------------------------------------------
---------------------------------------------------------------------------
# It is not necessary that list should contain same data type

mix_list= ['ram', 32, 36, 44, 'x']
print(mix_list)
------------------------------------------------------------------------------------------
---------------------------------------------------------------------------
# List inside list
# A list can have list inside it

new_list= [ 32, "ram", [1,3, 4], 55, 6545]
print(new_list)

# Empy list in python
emp_list=[]
print(emp_list)
------------------------------------------------------------------------------------------
---------------------------------------------------------------------------
# List and definite loop

friends= ["kashi", "Dipendra", "Suresh", "Hemu"]
for friend in friends:
    print(friend)
------------------------------------------------------------------------------------------
---------------------------------------------------------------------------
# Looking inside the list

friends= ["kashi", "Dipendra", "Suresh", "Hemu"]

print(friends[1])  # how to read it: it is friends of 1
------------------------------------------------------------------------------------------
---------------------------------------------------------------------------
# List ar Mutable
# if you remember variables are not mutable but list are mutable.

string= "SAGAR"
string.lower()    # this code does not actully change the Sagar to lower so not mutable
string

string= string.lower()  # now this code does change, because here change is being saved
to the orginal. this automatically happens in list so list is called mutable
string

# for list

item_list= ["pen", "car", "mobilephone", "laptop"]

item_list[1]= "toyota car"     # this actually change the orginal item_list so list is
called mutable.
# At this point list is truly change
------------------------------------------------------------------------------------------
---------------------------------------------------------------------------

# Range() function in python
# Range() function in python a sequence of number from 0 to value minus 1.

print(range(4))

```

```python
for i in range(3):
    print(i)


flist=["Ram", "laxman", "bharat"]
print(range(len(flist)))
----------------------------------------------------------------------------------------------------------------------------------------------------------------
# Normal old code what we have used before

friends = [ "Kashi", "Dipendra", "Surja", "Bijay"]
for friend in friends:
    print("Happy new year", friend)

# Same code and same output using range() function

friends = [ "Kashi", "Dipendra", "Surja", "Bijay"]
for i in list(range(len(friends))):                    # first find out len(friends) --4 and then use range(3) --0,1,2,3  and then make it a list [0, 1, 2,3]
    print("Happy new year", friends[i])
----------------------------------------------------------------------------------------------------------------------------------------------------------------
# Operations on lists
# Adding two lists
# Most of the operations in the list are same as the string
x = [ "hi", 1, 32, 5]
y = [ "Hello", 15, 20 ]

new_list= x + y      # the order of the entity in the list will be same as the order in which list are added.
print(new_list)
----------------------------------------------------------------------------------------------------------------------------------------------------------------
# Slicing of the list
# Remember as in string also, the last number in the slacing repersent upto but not included
num_list = [ 1,5,7,9,6,36,84,66,25,94,22,16,546]

num_list[0:6]   # the number inside the slacing are talking about the position

num_list[:3]     # first three

num_list[5:]      # fifth positin entity till last

num_list[:]       # whole list
----------------------------------------------------------------------------------------------------------------------------------------------------------------
# Just like there are a lot of library methods in string we have many library methods for list too (eg append(), count(), len() etc)

demo_list = [ 1,5,7, "bharat",9,6,36, "sagar", "suraj", "hemant"]
dir(demo_list)          # these are the libraries you can use in the list

# we will try some of these libraries by building our own list from scratch
----------------------------------------------------------------------------------------------------------------------------------------------------------------
# Building a list from scratch
# list maintain the order i.e the order you append the item, same order it will appear in the list

check_list = list()  # list is like empty pocket
check_list.append("tomatos")           # oder of append will decide the order of entity in the list.
check_list.append("potatos")          # how many times you append sthg that will be in the list that very time
check_list.append("chicken")         # since list are mutable, so every time we do append our orginal list is being changed
```

```python

print(check_list)

# quiz- can we append this this later to middle of the list? ASK THEM yes by insert()
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------
# IN to find items in the list

demo_list = [ 1,5,7, "bharat",9,6,36, "sagar", "suraj", "hemant"]

5 in demo_list

"sagar" in demo_list

15 in demo_list
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Sorting in the list

friends = [ "Kashi", "Dipendra", "Surja", "Bijay"]
friends.sort()
print(friends)
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Some built in function we can use in the list
# if you have the numeric list built-in function are far better to use then using loop

num_list = [ 1,5,7,9,6,36,84,66,25,94,22,16,546]

print(max(num_list))

print(min(num_list))

print(sum(num_list))

print(len(num_list))
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Homework-10
# Write a program that pop up for the number to enter untill you enter done. and prints out the average of the entered numbers without using list and with using list.

total_sum = 0
total_count = 0
while True:
    num = input("Enter a number:")
    if num == "done":
        break

    num = float(num)
    total_sum = total_sum + num
    total_count = total_count + 1

print("The average is:", total_sum/total_count)

# Now trying the same code with list
num_list = list()

while True:
    num = input("Enter a number:")
    if num == "done":
        break

    num = float(num)
    num_list.append(num)

Average= sum(num_list)/len(num_list)
```

```python
print("The average is:", Average)
---------------------------------------------------------------------------------------------------------------------------------------------------------------------
# split() function in strings and list
# eg:- break a line(or string), split it and look for the each words in the line
# split() function can split the string on certain basis and return back us the list
# by default split() function split by the blank space

song = "give me some sunshine give me some rain"
song.split()              # this output is a list, where breaking cretria on the string was blank space
---------------------------------------------------------------------------------------------------------------------------------------------------------------------
# more then one blank space will also be counted as single blank space while splitting

line = "A                  lot of blank space we have"
line.split()
---------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Other splitting cretria

demo_line = "A;what's;your;Name"
demo_line.split()                # no blank space means no splitting will happens, single entity inside list

demo_line = "A;what's;your;Name"
demo_line.split(';')             # the splitting cretria is now ';', so splitting will happens where where there is ';'
---------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Homework-11
# Write a python program which ask for file to read, then then read the mbox-short.txt file, get all the lines that starts from From cluse and check all those email
# and print out the date are are reviced on

file = input("Enter a file name:")
fhandle = open(file)

for line in fhandle:
    line = line.strip()
    if not line.startswith('From '):
        continue

    word_list= line.split()
    word_list[2]

---------------------------------------------------------------------------------------------------------------------------------------------------------------------

# Homework-11 [Alternative good approach]
# Write a python program which ask for file to read, then then read the mbox-short.txt file, get all the lines that starts from From cluse and check all those email
# and print out the date are are reviced on

file = input("Enter a file name:")
fhandle = open(file)

for line in fhandle:
    line = line.rstrip()
    if not line.startswith('From'):
        continue

    #print(line)
    words=line.split()
    if len(words) <= 2:
            continue
    #print(words)
    print(words[2])
---------------------------------------------------------------------------------------------------------------------------------------------------------
# Homework-11 [alternative approach]
# Write a python program which ask for file to read, then then read the mbox-short.txt file, get all the lines that starts from From cluse and check all those email
# and print out the date are are reviced on

file = input("Enter a file name:")
fhandle = open(file)

for line in fhandle:
    line = line.rstrip()
    words=line.split()
    #print(words)
    if len(words) <= 2 or words[0] !="From":
        continue
    #print(words)
    print(words[2])
---------------------------------------------------------------------------------------------------------------------------------------------------------
# Homework-12
# Redo the old code (Homework-7) for getting the university name from the line using split() function

line= "From Augustin.Marcus@siue.edu Jan 4 2024 06:32PM"

words = line.split()
sub_words = words[1].split('.')
sub_words[1].split('@')[1]

--------------------------------------------------------------------------------------------------------------------------------------------------------
# Dictionories in Python
# Dictionaries in python are denoted by {} inside which there is a key-value pair
# Secodn type of data structure
# difference between list and dictionary is: List is a linear collection of values that stay in order while dictionary is a beg of value each with its own label
# In list index is position but in dictionary index is key (of key-value pair)
# so Dictionary is a key value pair.
# let's look the template code of dictionary

purse = dict()

purse['Key'] = 'value-1'

print(purse)
--------------------------------------------------------------------------------------------------------------------------------------------------------
# quiz- Make a dictionary that has subject as the key and their subject code as a value

Statistics = dict()

Statistics['Machine learning'] = 'Stat 561'
Statistics['Data Science'] = 'Stat 560'
Statistics['SQL an Oracal'] = 'CMIS 563'

# ordering of the items in the dictionary does not have any pattern but key value pair always comes together.
print(Statistics)

# accessing the dictionary

Statistics['Machine learning']
--------------------------------------------------------------------------------------------------------------------------------------------------------

```

```
# Another easy way to create a dictionary

demDict = {'Stat': 561, 'OR': 585, 'QR': 101, 'CMIS': 563}

print(demDict)

print(type(demDict))

------------------------------------------------------------------------------------------------------------------------------------------------------------
# directory is not mutable same as string, i.e orginal dictionary will not change if you apply any operation on it
# to make the change in the orginal dictionary you need to save the change to the orginal dictionary

Statistics = dict()

Statistics['Machine learning'] = 'Stat 561'
Statistics['Data Science'] = 'Stat 560'
Statistics['SQL an Oracal'] = 'CMIS 563'

Statistics['Machine learning'] + ' and Stat 562'    # this will not change our orgianl dictionary

print('Without Saving:-',Statistics)

# but this bellow code will change the dictionary as we are saving the change made in the dictionary too

Statistics['Machine learning'] = Statistics['Machine learning'] + ' and Stat 562'

print('With Saving:-',Statistics)
------------------------------------------------------------------------------------------------------------------------------------------------------------
# quiz- IF you are given a list of names, how do you know which one is the highest repeted? ---by drawing histogram or by counting each time we saw same name.
# we can do that using the python dictionary, by updating the value of the name (key) each time we encounter.

friends = ["Kashi", "Dipendra", "Surja", "Bijay", "Kashi", "Dipendra","Dipendra", "Surja","Surja", "Bijay", "Kashi","Dipendra"]

fdictionary = dict()

fdictionary["Kashi"] = 0
fdictionary["Dipendra"] = 0
fdictionary["Surja"] = 0
fdictionary["Bijay"] = 0

for key in friends:
    #print(key)
    if key == "Kashi":
        fdictionary["Kashi"] =  fdictionary["Kashi"] + 1
    elif key == "Dipendra":
        fdictionary["Dipendra"] = fdictionary["Dipendra"] + 1
    elif key == "Surja":
        fdictionary["Surja"] = fdictionary["Surja"] + 1
    else:
        fdictionary["Bijay"] = fdictionary["Bijay"] + 1

fdictionary
------------------------------------------------------------------------------------------------------------------------------------------------------------
# looking if something is inside the dictionary or not
# the search is done on the basis of key not value

demo_dict= dict()
```

```
demo_dict["STAT"] = 561
demo_dict["CMIS"] = 563
demo_dict["OR"] = 585

demo_dict

print(type(demo_dict))

'CMIS' in demo_dict

'Stat' in demo_dict

'QR' in demo_dict

# this is very important because, with help of this we can do sthg in the dictionary, if key is there or do something else
-----------------------------------------------------------------------------------------------------------------------------------------------------
# using the above key in the dictionary or not concept, we will count the names of the friends that how many times they apears

friends = ["Kashi", "Dipendra", "Surja", "Bijay", "Kashi", "Dipendra","Dipendra", "Surja","Surja", "Bijay", "Kashi","Dipendra"]

fdictionary = dict()

for name in friends:
    if name not in fdictionary:
        fdictionary[name] = 1
    else :
        fdictionary[name] = fdictionary[name] + 1

print(fdictionary)
-----------------------------------------------------------------------------------------------------------------------------------------------------
# if there is no key in the dictionary, put it in. if there is key in the dictionary do sthg to the existing value that are already there.
#  This is something we will do a lot in the dictionary, we have to do this so many times such that they have developed function for this
# the get() Methods for dictionary
# the pattern of checking to see if a key is already in the dictionary and assuming a default value if the key is not there is so common that there is a method
# called get() that does this for us.

# eg.   x = counts. get(name, 0) :-- here counts is the dictionary on which we have used the get function which look for the key as a name and 0 is the default value
# that mean if key does not exist, it will have value 0 for it
-----------------------------------------------------------------------------------------------------------------------------------------------------
# Using this get() function concept to sovle the above problem of counting the name of the friends how many times they appers

friends = ["Kashi", "Dipendra", "Surja", "Bijay", "Kashi", "Dipendra","Dipendra", "Surja","Surja", "Bijay", "Kashi","Dipendra"]

counts = dict()
for name in friends:
    counts[name] =counts.get(name, 0) + 1
print(counts)

-----------------------------------------------------------------------------------------------------------------------------------------------------
# Homework-13
# write a python code to use all above concepts to find all the words from the mbox.txt file and also give their count that how many times they appear?

file = input("Enter a file")
```

```python
fhandle = open(file)
counts = dict()

for line in fhandle:
    line = line.strip()
    words = line.split()
    #print(words)
    for word in words:
        counts[word] = counts.get(word, 0) + 1

print(counts)
----------------------------------------------------------------------------------------------------------------------------------------------------------------
# for loops in dictionary

counts = { 'Sagar' : 100, 'Kashi' : 55, 'Dipendra' : 32, 'Ram'  : 105 }

for key in counts:                                                    # for making my life easier I use the word key, it is not necessary
    print('the key value pairs are: ', key , '--', counts[key])

----------------------------------------------------------------------------------------------------------------------------------------------------------------
# getting list of key's from the dictionary

counts = { 'Sagar' : 100, 'Kashi' : 55, 'Dipendra' : 32, 'Ram'  : 105 }

print(list(counts))

print(counts.keys())

print(counts.values())

print(counts.items())    # tuple: we will talk about this a lot in next chapter,
                         # the output of this will be the key-value pair as (key,value) inside list

----------------------------------------------------------------------------------------------------------------------------------------------------------------
# items() bonus trick
# Bonus: Two iteration variables in python, no other programming language can handle this

counts = { 'Sagar' : 100, 'Kashi' : 55, 'Dipendra' : 32, 'Ram'  : 105 }

for key, value in counts.items():       # note here key is the first iteration variable and value is the second iteration variable
    print(key, value)
----------------------------------------------------------------------------------------------------------------------------------------------------------------
# Homework-14
# write a python code to use all above concepts to find all the words from the mbox.txt file and also give their count that how many times they appear?
# modify this homework-13 to give only the word which is highest frequent and also give its number.

file = input("Enter a file name:")
fhandle = open(file)

counts = dict()
for line in fhandle:
    line = line.strip()
    words = line.split()
    #print(words)

    for word in words:
        counts[word] = counts.get(word, 0) + 1

```

```
#print(counts)
bigword = None
bigcount = None

for key, value in counts.items():
    #print(key, value)
    if bigcount is None or value > bigcount:
        bigword = key
        bigcount = value
print(bigword, bigcount)
--------------------------------------------------------------------------------------------------------------------------------------------------------------

# Cross-checking the above code with string (single) line
fhandle = "hello hello what is is name is your name what is is"

counts = dict()
for line in fhandle:
    line = line.strip()
    words = line.split()
    #print(words)

    for word in words:
        counts[word] = counts.get(word, 0) + 1

#print(counts)
bigword = None
bigcount = None

for key, value in counts.items():
    #print(key, value)
    if bigcount is None or value > bigcount:
        bigword = key
        bigcount = value
print(bigword, bigcount)
--------------------------------------------------------------------------------------------------------------------------------------------------------------
# showing how program is actually working with a easy example

file = input('Enter a file Name: ')
fhandle = open(file)

count = dict()
for line in fhandle:
    line = line.strip()
    print(line)
    words = line.split()
    print(words)
    for word in words:
        if word in count:
            print(word)
            print('**Existing word**')
            print(count[word])
            count[word] = count[word] + 1
        else:
            print(word)
            print('**new word**')
            count[word] = 1
            print(count[word])
print(count)
--------------------------------------------------------------------------------------------------------------------------------------------------------------
# tuple in python
# tuples are like list, but here we use () parenthesis instead of [] squrebracket, note formation wise only
# index in tuple is postition and position starts from zero, all same as list
# the only difference is they are not mutable
```

```
demo_tuple = ('Ram', 'Krishna', 'bishnu', 'Hanuman', 'Mahadev', 'kali')

demo_tuple[1]
------------------------------------------------------------------------------------------------------------------------------------------------------------------
# quiz- what does tuple are not mutable means

x_list = [ 5, 6, 32, 9, 0]

x_list[1] = 69
print(x_list)  # look list is changed

y_tuple = ( 5, 6, 32, 9, 0)

y_tuple[1] = 69  # traceback error: item assignment does not support
------------------------------------------------------------------------------------------------------------------------------------------------------------------
# there are lot of things that you can not do in tuple like
# 1) you can not short the tuple
# 2) you can not append into the tuple
# 3) you can not reverse the tuple

# for list you can do
x_list = [ 5, 6, 32, 9, 0]
x_list[1] = 69
x_list.append(5000)
x_list.sort(reverse=True)
x_list.reverse()  # if you want to do descending
print(x_list)  # look list is changed


# for tuple you can not do
y_tuple = ( 5, 6, 32, 9, 0)
y_tuple[1] = 69  # traceback error: item assignment does not support
y_tuple.append('sagar')
y_tuple.sort()
y_tuple.reverse()
------------------------------------------------------------------------------------------------------------------------------------------------------------------
# so the question is what we can do with tuple

t = tuple()
dir(t)
------------------------------------------------------------------------------------------------------------------------------------------------------------------
# comparing waht we can do with list vs what we can do with tuple

l = list()
print(dir(l))

t = tuple()
print(dir(t))
------------------------------------------------------------------------------------------------------------------------------------------------------------------
# tuple for assignment or assignment in tuple

(x, y) = ( 1, 100)
print(x)
print(y)

(first_name, last_name) = ( 'sagar' , 'kalauni')
print(first_name)
print(last_name)
------------------------------------------------------------------------------------------------------------------------------------------------------------------
# tuples from dictionary
# use items() function to find list of the tuples from the dictionary

demoDic = { 'stat' : 560, 'CMIS': 563, 'OR': 585 }

print(type(demoDic))
demoDic.items()
------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Comprasion in tuple

(1, 5, 7) < (2, -5 -10)    # it will only look for the most significant digit in the tuple, first one is most significant, similar as comparing numbers

(1, 5, 7) < (1, -5 -10)     # if the first one is same it will look for the second and so on

('ram' , 'sita', 'hanuman') < ( 'hanuman', 'sita', 'ram')

('ram' , 'sita', 'hanuman') < ( 'ram', 'hanuman', 'sita')
------------------------------------------------------------------------------------------------------------------------------------------------------------------
# shorting the list of tuples using the sorted() function
# we can obtain the list of tuple by using the items() function in the dictionary
# sorting is done on the basis of key

demo_dict =  { 'Stat' : 560, 'CMIS': 563, 'OR': 585 }

tuple_list = demo_dict.items()

# sorting the list o tuples.

sorted(tuple_list, reverse = True)
------------------------------------------------------------------------------------------------------------------------------------------------------------------
# using sorted() and loop to print the key value pair in certain order
# sorting is done on the basis of key

demo_dict =  { 'Stat' : 560, 'CMIS': 563, 'OR': 585 }

tuple_list = demo_dict.items()

sorted(tuple_list)

for key, value in sorted(tuple_list):
    print (key, value)

# same code in descending order

demo_dict =  { 'Stat' : 560, 'CMIS': 563, 'OR': 585 }

tuple_list = demo_dict.items()

sorted(tuple_list, reverse = True)

for key, value in sorted(tuple_list, reverse = True):
    print (key, value)

# alway remember shorting by default is done on the basis of key
------------------------------------------------------------------------------------------------------------------------------------------------------------------
# quiz- how to do the sorting on the basis of value of the tuple.
# idea is same that by default sorting is done on the basis of key of the tuple, so why not change the order of our tuple to make key as value and value as key
# and then do sorting. those changed order tuples should be put in the new empty list

demo_dict =  { 'a' : 560, 'c': 563, 'd': 585, 'z' : 99 }

```

```
temp_list = list()
tuple_list = demo_dict.items()

for key, value in tuple_list:
    temp_list.append((value, key))
#print(temp_list)
    for key, value in sorted(temp_list):
        print(key, value)
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
# quiz- how to do the sorting on the basis of value of the tuple.
# idea is same that by default sorting is done on the basis of key of the tuple, so why not change the order of our tuple to make key as value and value as key
# and and put them on one empty list then do sorting.
demo_dict =  { 'a' : 560, 'c': 563, 'd': 585, 'z' : 99 }

temp_list = list()
tuple_list = demo_dict.items()

for key, value in tuple_list:
    temp_list.append((value, key))
#print(temp_list)
for key, value in sorted(temp_list):
    print(key, value)
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Homework-15
# Using all above concepts, find out the top 10 most used words in mbox-shor.txt file with standard coding format.

file = input("Enter a file: ")
fhandle = open(file)

count = dict()
for line in fhandle:
    line = line.strip()
    words = line.split()

    for word in words:
        count[word] = count.get(word, 0) + 1
#print(count)
temp_list = list()
tuple_list = count.items()

for key, value in tuple_list:
    temp_list.append((value, key))
#print(temp_list)

for key, value in sorted(temp_list, reverse = True)[:10]: # just to limit 10 items from the list
    print(key, value)
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
# quiz- Can you do the same above coding of finding the sorted list of highly frequented data from this dictionary: demo_dict = {'a': 10, 'b':2, 'c':99}
# doing it with even shorter version, not necessary to understand right now, just showing you when you are comfortable with programming how it can be too short also
# List comprasion

demo_dict = {'a': 10, 'b':2, 'c':99, 'd':11, 'z':81}

print( sorted([(v, k) for k, v in demo_dict.items()], reverse= True)[0:3])
--------------------------------------------------------------------------------------------------------------------------------------------------------------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Chater-3 Accessing web data through python~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
# Regular Expression in python
# Regular expression is someting you can skip also
# Regular expression quick guide
^ match the begining of the line
$ mathch the end of the line
. match any character
\s match whitespace
\S match any non-whitespace character
* repetes a character zero or more times
*? repetes a character zero or more times (non greedy)
+ repetes a character one or more times
+? repetes a character one or more times(non greedy)
...
etc
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Regular expression in not built in python, so we kind of import it in the working environment before using
# re.search()  ---> same as the find function in the string [Used for finding the match]
# re.findall()  ---> same as the combination of find and slicing in the stirng [Used for extracting]

# using find()
file = input("Enter a file name: ")
fhandle = open(file)

for line in fhandle:
    line = line.strip()
    if line.find('From:') >=0:
        print(line)

# using regular expression search
import re

file = input("Enter a file name: ")
fhandle = open(file)

for line in fhandle:
    line = line.strip()
    if re.search('From:', line):
         print(line)
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
# if we want our first character to be certain to startwith, we can using the regular expression
# as we used to use startswith() function for string, here we will use ^ before the thing we should search for indicating that this should be the first character.

# using startswith()
file = input("Enter a file name: ")
fhandle = open(file)

for line in fhandle:
    line = line.strip()
    if line.startswith('From:'):
        print(line)

# using regular expression search
import re

file = input("Enter a file name: ")
fhandle = open(file)

for line in fhandle:
    line = line.strip()
    if re.search('^From:', line):
         print(line)
```

```
-------------------------------------------------------------------------------------------------------------------------------------------------------
# wild card characters
# ^X.*: ---> looking for lines, (^) X at the begning,(.) match any character, (*) any number of character. [with no condition at the end]

# using regular expression search
import re

file = input("Enter a file name: ")
fhandle = open(file)

for line in fhandle:
    line = line.strip()
    if re.search('^X.*:', line):
         print(line)
-------------------------------------------------------------------------------------------------------------------------------------------------------
# Being more preicse
# ^X-\S+: ---> looking for lines, (^) X at the begning,(.) match any character, (*) any number of character,

# using regular expression search
import re

file = input("Enter a file name: ")
fhandle = open(file)

for line in fhandle:
    line = line.strip()
    if re.search('^X-\S+:', line):
         print(line)
-------------------------------------------------------------------------------------------------------------------------------------------------------
# Now extracting the data with the help of Regular expression
# anything inside squre bracket is one character to look for but it can inside have anything like range, list etc
# so [0-9]+, this means look for any one digit from 0-9, by saying + it means it can have more character too. so look for one or more digit

import re
x = 'Hell my name is Sagar. My age is 26. I am right now in 4 th semester of my masters. I have courses stat 579, stat 561 this semester'

y = re.findall('[0-9]+', x)
print(y)


-------------------------------------------------------------------------------------------------------------------------------------------------------
# looking for how many vowels are in the given string (line)
import re
x = 'Hell my name is Sagar. My age is 26. I am right now in 4 th semester of my masters. I have courses stat 579, stat 561 this semester '

y = re.findall('[aeiou]', x)
print(y)

count = dict()
for letter in y:
    count[letter] = count.get(letter, 0) +1
print(count)
-------------------------------------------------------------------------------------------------------------------------------------------------------
# Homework-16
# Use this code to find the number in the each line of mbox-short.txt
import re

```

```
file = input("Enter a file name: ")
fhandle = open(file)

for line in fhandle:
    line = line.strip()
    y= re.findall('[0-9]+', line)
    print(y)
----------------------------------------------------------------------------------------------------------------------------------
# greedy matching in the expression
import re
x = 'From: using the:'

y=re.search('^F.+:', x)
print(y)

# how to remove the gredyness of the expression
# greedy matching in the expression is removed by adding ?
import re
x = 'From: using the:'

y=re.search('^F.+?:', x) # (.)any character, (+) one or more time, (?) but don't be greedy
print(y)
----------------------------------------------------------------------------------------------------------------------------------
# Homework-17
# find all the email address provided in the mbox.txt file using expression. hint: email has non-blank character followed by @ followed by non-blank character.

file = input('Enter a file: ')
fhandle = open(file)

for line in fhandle:
    line = line.strip()
    y = re.findall('\S+@\S+', line)
    if y == []:
        continue
    print(y)

----------------------------------------------------------------------------------------------------------------------------------
# for more exect
# find all the email address provided in the mbox.txt file using expression. hint: email has non-blank character followed by @ followed by non-blank character.

import re
file = input('Enter a file: ')
fhandle = open(file)

for line in fhandle:
    line = line.strip()
    y = re.findall('^From (\S+@\S+)', line) # what i am saying inside extration, go look word starting with From but I dont want that, after that there should be
    if y == []:                              # space and the thing to be extracted will be inside the parenthesis which is non-blank character followed by some
        continue                             # some character followed by @ sign, again followed by non-blank character and other characters
    print(y)
----------------------------------------------------------------------------------------------------------------------------------
# Recall Coding problem
# Recalling the university or company name from the email address
# Method-1 (Already discussed)

line= "From Augustin.Marcus@siue.edu Jan 4 2024 06:32PM"

```

```python
Spoint = line.find('@')

Epoint = line.find('.', Spoint)

line[Spoint + 1 : Epoint]

# Method-2 (already discussed)

line= "From Augustin.Marcus@siue.edu Jan 4 2024 06:32PM"

new_line = line.split()[1]

new_line_1 = new_line.split('.')[1]

print(new_line_1.split('@')[1])

# Method-3

import re
line= "From Augustin.Marcus@siue.edu Jan 4 2024 06:32PM"

y=re.search('@([^.]+)', line).group(1)
print(y)
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Homework-18
# Extracting data from the mbox-short.txt file
# write a python code to extract the float number in each line after the word X-DSPAM-Confidence. finally give the list of all the float numbers

file =  input('Enter a file: ')
fhandle = open(file)

for line in fhandle:
    line = line.strip()
    word = re.findall('^X-DSPAM-Confidence [0-9].*', line)
    print(word)
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Homework-18
# Extracting data from the mbox-short.txt file
# write a python code to extract the float number in each line after the word X-DSPAM-Confidence. finally give the list of all the float numbers

import re
file =  input('Enter a file: ')
fhandle = open(file)

num_list = list()
for line in fhandle:
    line = line.strip()
    num = re.findall('^X-DSPAM-Confidence: ([0-9.]+)', line)  # look for the line having X-DSPAM-Confidence, start extracting after the space after X-DSPAM-Confidence
    if num == []:                                             # look for any one character from 0-9 and ., (+) with one or more character
        continue
    else:
        num_list.append(num[0])
print(num_list)

# if you want the maximum and minimum of this
print(max(num_list))
print(min(num_list))
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
# same question little different approach
import re
```

```
file =  input('Enter a file: ')
fhandle = open(file)

num_list = list()
for line in fhandle:
    line = line.strip()
    num = re.findall('^X-DSPAM-Confidence: ([0-9.]+)', line)
    if len(num) != 1:
        continue
    else:
        num_list.append(num[0])
print(num_list)

# if you want the maximum and minimum of this
print(max(num_list))
print(min(num_list))
-------------------------------------------------------------------------------------------------------------------------------
# What if you have to extract some symbol like($) which have a special meaning in the expression. use \ (back-slash) infront of it

line = 'Hey my name is sagar and I work as a teaching assistent in siue and they pay me $1060 as stipend monthly'

import re

my_salary = re.findall('^\$+', line)
print(my_salary)
-------------------------------------------------------------------------------------------------------------------------------
# Quick review of the expression and their use in python

^    Matches the beginning of a line
$    Matches the end of the line
.    Matches any character
\s   Matches whitespace
\S   Matches any non-whitespace character
*    Repeats a character zero or more times
*?   Repeats a character zero or more times (non-greedy)
+    Repeats a character one or more times
+?   Repeats a character one or more times (non-greedy)

[aeiou]   Matches a single character in the listed set
[^XYZ]    Matches a single character not in the listed set
[a-z0-9]  The set of characters can include a range
(         Indicates where string extraction is to start
)         Indicates where string extraction is to end
-------------------------------------------------------------------------------------------------------------------------------
# Now onward let's talk to the internet with the help of python

# TCP connections/ Sockets
# In a computer networking, an internet socket or network socket is an endpoint of bidirectional inter-process communication flow across an internet protocol-based
# computer network, such as the internet.
# process <---- internet ----> process
# web server ko lagi port ho 80
# the protocol we are going to talk about is Http
# url = protocol + host + docoment. eg http://www.dr.chunk.com/page1-htm.
-------------------------------------------------------------------------------------------------------------------------------

# General Rule of how it works
# Create a socket between your application (python code) and web server
# once socket is created, connect it to the web server by giving the host and port.
# after that we need to have our first communication, and first communication is always done from our side [as a rule of http or port 80]
```

```python
# so write a command and send the command through the socket.
# after that web-server will look at the command and give back you the data on the basis of what command you have sent
# revive the data from the web server through socket.

import socket

mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # think of socket as file handle that does not have data
mysock.connect(('data.pr4e.org',80))                          # connect is the function which take single tuple as the input
cmd = 'GET http://data.pr4e.org/romeo.txt HTTP/1.0\r\n\r\n'.encode()   # we are sending as utf-8 data
mysock.send(cmd)

while True:
    data = mysock.recv(512)
    if (len(data) < 1):
        break
    print(data.decode())                      # we will recive as utf-8 data, so need to decode
mysock.close()
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
# the thing that comes at the end of the url is called the get parameter
# http://data.pr4e.org/romeo.txt/guess=1 , here guess=1 is called the get parameter
# status code = 200 means you are good we found the web page for that
# status code = 404 means error not found
# status code = 302 means wrong web browser but will be directed to the correct one
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Since computer does not understand the letters, so we need to have the standard conversion for computer and one of them was ASCII (American standard code for
# information interchange)
# way to look at the number crossponding to letters

print(ord('H'))
print(ord('A'))

print(ord('\n'))  #remember new line is a single character
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
# in the old times, american computer can talk to american computer only and japnease computer can talk to japnease computer only as they have their own stardard
# but it was the problem to communicate between the computer in japan to computer in america. So they have introduced the concept of unicode. hence all the input
# you will give to the computer will be converted to some encode which is same all over the world and computer will give you output in the same encode version
# which need to be decoded to understand.
# basic concept is we are sending our command as bite by encoding it and computer is send back data in the form of bite and we are decoding it back to the string.
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Retriving the data from the web.
# since every time we have to do the same stuffs of creating socket, connecting to webserver, send request command as a bite, reciving a data as a bite. Why not
# make our life easy by defining a function or libraries that do this for us. yes are are some libraries for doing all this and making coding all easy in python
# the libraries will parse the url, figure out what server to talk with, what docoment to retrive, waht http version are are inside the library. it will simply
# open the url and give the fhandle as we used to get in the old cases from our own computer, means it is almost same as open()

import urllib.request, urllib.parse, urllib.error

fhandle = urllib.request.urlopen('http://data.pr4e.org/romeo.txt')
for line in fhandle:
    print(line.decode().strip())       # this line you get is bite arrary, so may be need to decode to string
------------------------------------------------------------------------------------------------------------------------------------------------------------
# comprasion to get same work done with and with out using the library

# without library
import socket

mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # think of socket as file handle that does not have data
mysock.connect(('data.pr4e.org',80))                        # connect is the function which take single tuple as the input
cmd = 'GET http://data.pr4e.org/romeo.txt HTTP/1.0\r\n\r\n'.encode()   # we are sending as utf-8 data
mysock.send(cmd)

while True:
    data = mysock.recv(512)
    if (len(data) < 1):
        break
    print(data.decode())                        # we will recive as utf-8 data, so need to decode
mysock.close()

# With library [quite easy and stright forward]

import urllib.request, urllib.parse, urllib.error

fhandle = urllib.request.urlopen('http://data.pr4e.org/romeo.txt')
for line in fhandle:
    #line = line.strip()
    print(line.decode().strip())            # this line you get is bite arrary, so may be need to decode to string

# with this 4 line of code we are actually reading the web page, that's the power of python
# Now with these ideas, it is no longer a web page for us, it is simply like a file in our computer whcih we can open and do what ever calculation we want to
------------------------------------------------------------------------------------------------------------------------------------------------------------
# Homework-19
# access the web page http://data.pr4e.org/romeo.txt and look for the words on it and count how many word are repeted how many times? [Same as old questions]

import urllib.request, urllib.parse, urllib.error

fhandle = urllib.request.urlopen('http://data.pr4e.org/romeo.txt')

count = dict()
for line in fhandle:
    line = line.decode().strip()
    #print(line)
    word_list = line.split()
    for word in word_list:
        count[word] = count.get(word, 0) + 1
print(count)
------------------------------------------------------------------------------------------------------------------------------------------------------------
# Homework-20
# access teh web page http://data.pr4e.org/romeo.txt and find out the most frequent word and also give us the frequency of the word in the web page.

import urllib.request, urllib.parse, urllib.error

fhandle = urllib.request.urlopen('http://data.pr4e.org/romeo.txt')

```

```
count = dict()
for line in fhandle:
   line = line.decode().strip()
   word_list = line.split()
   for word in word_list:
      count[word] = count.get(word, 0) + 1
#print(count)

emp_list = list()
for k,v in count.items():
    emp_list.append((v,k))

print(emp_list)
print(max(emp_list)) # so the is one of the most repeted words in the above web page if comared tuples (3, 'is'), (3, 'the')i.e
                     # frist seen and secodn tuple with same max value, so compared by looking the second cordinate
------------------------------------------------------------------------------------------------------------------------------------------------------------------
# quiz- what do you think this web scarping can be used for. does this make sense to you that right now we can look for one certain web page, get the link of
# other page inside that page and follow the link and go again and again, This could be one of your homework-13

import urllib.request, urllib.parse, urllib.error

fhandle = urllib.request.urlopen('http://www.dr-chunk.com/page1-htm')

for line in fhandle:
    line = line.decode().strip()
    print(line)
------------------------------------------------------------------------------------------------------------------------------------------------------------------
# What is web scarping?
# when a program or script pretends to be a browser and retrives web pages, looks at those web pages, extract information and then look at more web pages.
# search engine scrape web pages, we call this 'spidering the web' or 'web crawling'
# So in simple web scarping means look at the web pages for some links and look at those links for some other links to get final answers

------------------------------------------------------------------------------------------------------------------------------------------------------------------
# if it is a text file to be retrive from the web page, we did it and was nice but if it was the html or xml file they are poorly managed so need special library
# instaled to parse it and make it more informative called beautiful soup

# Beautiful Soup is a Python library that is used for web scraping purposes to pull the data out of HTML and XML files. It provides Pythonic idioms for iterating,
# searching, and modifying the parse tree, making it easy to scrape information from web pages.

# you can think of this way also, we you retrive html file usig python code and it has links insde it with href:'url of link', one way to get this link is using
# expression in pythons and that is kind of hard, beautiful soup can do this with all one line of code and make our life easier.

# General syntax for data retrival using Beautiful soup

import urllib.request, urllib.parse, urllib.error
from bs4 import beautifulSoup

url = input ('Enter your url: ')
html = urllib.request.urlopen(url).read()     # because data we retrive by html file is messy, so we name it as html
Soup = BeautifulSoup(html, 'html.parser')     # by taking that nasty html data it will parse it into nice tree like object,(we don't need to know what's inside)

```

```
# Retriving all of the anchor tags
tags = Soup('a')
for tag in tags:
   print(tag.get('href', None)
----------------------------------------------------------------------------------------------------------------------------------------------------------
# installing and importing beautifulsoup in the jupiter notebook
!pip install beautifulsoup4
from bs4 import BeautifulSoup

----------------------------------------------------------------------------------------------------------------------------------------------------------
import urllib.request, urllib.parse, urllib.error
from bs4 import BeautifulSoup
import ssl

# ignore ssl certificate error, [you don't need to know it exectly]
ctx = ssl.create_default_context()
ctx.check_hostname = False
ctx.verify_mode = ssl.CERT_NONE

url = input('Enter your url:')
html = urllib.request.urlopen( url, context = ctx).read()  #it returns entrie docomnet in the web page in a single big string with new line at the end of each line
soup = BeautifulSoup(html, 'html.parser')

# retriving all the anchor tags, anchor tag means any things that starts with <a ... to >
tags = soup('a')           # it will give list of anchor tags
for tag in tags:
   print(tag.get("href", None))
----------------------------------------------------------------------------------------------------------------------------------------------------------
# till now we have studied how to communicate between two computer which are completely distenct using socket and then to make our life easier we do have python
# library to do all those line of code in single line.
# now when we are sending our command or data from our computer using python or java any, we are not actually sending python code we are sending the data which
# is first serialize from our computer to particular format and then it is being to to another computer which may be using java where the data is de-serialize in
# that particular format.

# With the http request/response well understood and well supported, there was a natural move towards exchanging data between programs using those protocols.
# we needed to come up with an agreed way to repersent data going between applications and across networks. There are two commonly used format: XML and JSON
----------------------------------------------------------------------------------------------------------------------------------------------------------
# XML (extensible mark up language)
# in rough word, two program agree on one syntex to share there data across the network
# primary purpose is to help information systems share structured data
# XML BASICS

<person>                              ---start tag
 <name>Sagar</name>
 <phone type = "intl">
 +1 (618) 917-9128                   ----text content
 <Phone>
 <email hide="yes" />                ----self closing tag
</person>                            ----end tag
----------------------------------------------------------------------------------------------------------------------------------------------------------
# XML Schema
# So we have two cooperating applications, and they've got to send data to one another, and they have a disagreement as to whether or not the data is right.
# One side might blow up or the other side might blow up and it's like whose fault is it? To stop this happen XML has its own rule called Schema of XML.
--------------------------------------------------------------------------------------------
```

```
-------------------------------------------------------------------------
# How to parse XML in python
# let's create a general syntax for parsing XLM and extracing infromation from it

# again to make things easier for us to do we do have a library to work with it
import xml.etree.ElementTree as ET
#normally we would be reading all of these data with urllib and read and whatever and then we would parse it. But just to make these simple on one screen,
# I've kept it simple. And so I have a string. also triple-quoted string in Python is a potentially multi-line string.
data = '''<person>
 <name>Sagar</name>
 <phone type="intl">
 +1 (618) 917-9128
 </phone>
 <email hide="yes" />
</person>'''

# And what fromstring says is take this string and give us back basically a nice tree.
# if you have syntax error in your data(or sting ) this could blow up
tree = ET.fromstring(data)
print('name:', tree.find('name').text)        # with in that formed tree go find me the tag named- name and give the text part of it
print('attribute', tree.find('email').get('hide'))   # with in that formed tree go find me hte tag named eamil, look for the attribute hide for it
------------------------------------------------------------------------------------------------------------------------------------------------------------------
# in the above example we did extract information from only one tree, but in many cases we may have to extract the infromation from the list of trees.

import xml.etree.ElementTree as ET     # loading library, built in xml parser library
inputs = '''<stuff>                    # creating dummy xml data, latter on we will parse from the website
               <users>
                  <user x="2">
                     <id>001</id>
                     <name>sagar</name>
                   </user>
                   <user x="7">
                     <id>009</id>
                     <name>Kashi</name>
                    </user>
               </users>
            </stuff>'''

stuff = ET.fromstring(inputs)      # making a nice tree named stuff form the dummy xml data
lst = stuff.findall('users/user')   # because we are accessing a multiple tags so making a list to store it
print('user count', len(lst))        # looking how many items are in the list

for item in lst:
   print('Name:', item.find('name').text)  # accessing each item of the list to retrive particular information.
   print('ID:', item.find('id').text)
   print('Attribute', item.get('x'))
------------------------------------------------------------------------------------------------------------------------------------------------------------------
# JSON [javaScrip object Notation]
# So now we're going to talk about a new serialization format. We've talked about XML, which is kind of complex. And there's a simple one called JSON. XML is way
# powerful and we did not use it a lot. JSON is quite popular nowadays.
# Sample code for JSON

import json      # we have a inbuilt function called json to parse data.

data = '''{ "name" : "Sagar",                      # sample demo data we are creating,
later on we will look from the web
            "phone" : {
                    "type": "intl",
                    "number" : "+1 (618) 917-9128"},
            "email" : { "hide": "yes" }
            }'''

info = json.loads(data)                   # json.loads() is a function that arrange that messy data into a nice tree of information
print('Name:', info['name'])
print('Email:', info['email']['hide'])
-------------------------------------------------------------------------------------------------------------------------------------------------
# Making the list of dictionary tree and extracting the information from there

import json

data = '''[
        {
         "name": "Sagar",
         "Id" : "800752***",
         "email" : "skalaun@siue.edu"
        },
        {
          "name" : "kashi",
          "Id" : "800700***",
          "email" : "kashi@gmail.com"
        }
         ]'''

info = json.loads(data)
print('User count', len(info))

for item in info:
    print("Name:", item['name'])
    print("email:", item['email'])
    print("id no.", item['Id'])
-------------------------------------------------------------------------------------------------------------------------------------------------
# this is the general format to give parameter at the end of the url, where (+) means space and (%2C) means comma.
https://maps.googleapis.com/maps/api/geocode/json?address=Ann+arbor%2C+MI

-------------------------------------------------------------------------------------------------------------------------------------------------
# Now using all the concepts we have learned till now and accessing the data from the web, converting them to useful format and retriving data from there

import urllib.request, urllib.parse, urllib.error
import json

# Note that Google is increasingly requiring keys
# for this API
serviceurl = 'http://maps.googleapis.com/maps/api/geocode/json?'    # The web address of the API

while True:
    address = input('Enter location: ')    # Ask the user to input an address
    if len(address) < 1: break    # If the entered line is blank, break

    url = serviceurl + urllib.parse.urlencode(    # Make a url by concatenating the API url and the url form of the address
        {'address': address})

    print('Retrieving', url)
    uh = urllib.request.urlopen(url)    # Get a handle for the url
    data = uh.read().decode()         #call the read method to pull the entire document
```

```
      & decode (from probably UTF-8)
      print('Retrieved', len(data), 'characters')

      try:
          js = json.loads(data)    # Parse the data as string data
      except:
          js = None

      if not js or 'status' not in js or js['status'] != 'OK':     # Check for failures - if js is false, if status key is missing, or status is not equal to "OK"
          print('==== Failure To Retrieve ====')
          print(data)
          continue

      print(json.dumps(js, indent=4))

      lat = js["results"][0]["geometry"]["location"]["lat"]    # Walking down the tree of keys to look for
      lng = js["results"][0]["geometry"]["location"]["lng"]
      print('lat', lat, 'lng', lng)
      location = js['results'][0]['formatted_address']
      print(location)

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Using database with python ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Python string to bytes
# When we talk to external resource like a network socket we sends bytes, so we need to encode python 3 string into a given character encoding.
# When we read data from the external resource, we must decode it based on the character set so it is properly repersented in python 3 as a string.

While True:
   data = mysock.recv(512)
   if (len(data)<1):
      break
    mystring = data.decode()    # usually we have to specify from what to decode from like utf-8, Asci or other,
    print(mystring)
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Object Oriented programming
# Then the concept of dog is like a class. But when you see a dog and you grab the dog, that's an object.
# Some python objects

x = 'abc'
type(x)
```