Python  World

From the book you can learn the core Python programming.

MD. Delwar Hossen Asik

BSC In CSE

Diploma In Computer Technology

Course Introduction

For Python setup watch the video http://youtube.com

Chapter-1

- Getting Started
- Base Command
- Comments
- Output Format
- Input System
- Precedence
- Operator
  (Comparison, Logical, Assignment, Identity, In)
- Data Types
- String
- Escape Sequences

Chapter-2

- Function And Method
  (Length, Count, Lower, Upper,islower,isupper)
- (Strip | Replace | Find | Title | Center) Method
- If else condition
- Elif Condition
- Short Hand if | if.. else
- Logical Condition
- Condition game
- Float After (.) Value limit
- Random Number
- (in | not in) Keyword
- While Loop
- For Loop
- Break and Continue
- One line print
- Loop Game And Pattern
- For Loop String
- Import Random Method
  (Randrange|Choice|Shuffle)

Chapter-3

- Multi Integer Input
- Using Created Function
- Fibonacci Using Function
- Odd Even Function
- Palindrome Number Using Function
- No Parameters Function
- Function Inside Function
- Default Parameters Function
- Scope Variable (Global | Local)
- Important Function Argument Default Genarate

Chepter-4

- List
- List Control

Add List Data

- Append Method
- Insert Method
- Combine Two List
- Extend Method: Combine Two List
- List Inside List Append Method

Delete List Data

- Pop Method
- Delete Method
- Remove Method

Important List Method And Function

- Length Function:
- Count Menthod:
- Sorted Function:
- Sort Method:
- Reverse Method:
- Copy Method:
- Clear Method:

More 2nd page Right collum

Chapter 5

Chapter 6

Chapter 8

Chapter 7

Chapter 10

Chapter 9

- Dictionary Comprehension
- Dictionary Comprehension IF Else
- Dictionary Comprehension (Nested If / elif)
- Set Comprehension
- Function *args operation
- *args using with normal parameter
- *args As Argument Unpack (List | Set | Tuple)
- **kwargs (Keyword args)
- Parameter Order Inside Function

Chapter-1

Getting Started

```
print("Hello, World!")

 Output:
 Hello, World!
```

Base Command

----------→(print working Directory)

----------→(See All List On This Folder)

r ----------→ For Screen Clear)

:// ---------- ---→(Change Directory)

ir "" ----------→ (make Directory Create Folder)

oldername ----------→ (open Folder)

h Filename_With_Extension ----------→(Create a file)

---------- -----→(For Back)

yt (TAB) ----------→ (write First three word then tab “Full Word Show”)

file name" ----------→(Remove File)

rf "Folder name" ----------→ (remove folder) (Remove Current Folder)

"file name.py" ----------→ (open file)

on filename.py ----------→(run python)

---------- -→(by default open sell)

oldefilename.type" "newname.type" ----------→(Rename File Or folder)

"file name.type ./foldername ----------→(move file same Directory Folder)

file name.type" ./foldername ----------→ (copy File ../foldername back folder)

ntry code reuse >> Up Arrow

Comments

Single line comments:

```
#Single Comments
```

Full comments:

```
"""All
Line
Comments"""
```

Python (3 |3.6) Output Format

Python 3 output:

```
print("Your Name Is {}.Roll {} And Age {}".format(Name,Roll,Age))
```

Advanced:

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))
```

Advanced:

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."
print(myorder.format(quantity, itemno, price))
```

Python 3.6 Output:

```
print(f"Your Name Is {Name}.Roll {Roll} And Age {Age}")
```

Input And Multi Input System

Single Input:

```
Name=input("Enter Your Name: ")
```

Multi Input:

```
Name,Roll,Age=input("Name (,) Roll (,) Your Age:  ").split(",")
#Its Give Us List
List=input("Enter: ").split(",")
```

Input Always String Type. For Multi Integer Float Click

To make it int, float other data type:

```
Name=int(input("Enter Your Name: "))
```

Precedence

Precedence rule

OPERATORS	PRECEDENCE AND ASSOCIATIVITY RULE
PARENTHESE	HIGHEST
EXPONENT	RIGHT TO LEFT
*,/ ,//, %	LEFT TO RIGHT
+, -	LEFT TO RIGHT

This table lists the operator with highest precedence at the top and the lowest precedence at the top.

What is precedence rule?

To evaluate complex expression python follows the rule of precedence, it governs the order in which the operation take place.

EXAMPLE – print(3-1*2)

In this expression there are two operators – **and** * but python will do multiplication before than subtraction because of precedence of multiplication is higher than subtraction.

print(3-1*2) → print(3-2) → print(1) → final output is 1 as shown below.

Operator

+	Adder
–	Sub
*	Multi
/	Float Division Not equal
//	Integer Division
%	Mod
**	Power

Comparison operators

!=	Not equal
==	equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equa

Logical operators Example Program

and
Or
Not

Assignment operators

+=
-=
*=
/=
//=
%=
**=

Identity operators Example Program

Is (Is Work with storage same location)
Is not
Output always show True or False

In , Not In operators Example Program

In (In Use with value present or not)
Not in

Data Types

Example	Data Type
x = "Hello World"	str
x = 20	int
x = 20.5	float
x = 1j	complex
x = ["apple", "banana", "cherry"]	list
x = ("apple", "banana", "cherry")	tuple
x = range(6)	range
x = {"name" : "John", "age" : 36}	dict
x = {"apple", "banana", "cherry"}	set
x = frozenset({"apple", "banana", "cherry"})	frozenset
x = True	bool
x = b"Hello"	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview

Index String

Format:

```
Name=123456

print(Name[Start : End : step])
```

Program:

```
Num="123456"
print("Number " + str(Num))
#Reverse
print(Num[-1::-1])   #If I Don't Know Length (last -1+,2 Incrase)
print("Delwar Hossen"[::-1])   # Reverse
print(Num[5::-1])    # Count 0-5=6 #(-1) Means 6
print(Num[5:2:-1])

_____________________________________

output:
Number 123456
654321
nessoH rawleD
654321
654
```

Escape Sequences

```
print("Name: \n A \t B \\\\ ")
print("Name: \\n A \\t B \\\\ ")  #print Normal Text ES. Seq.
```

Output:

```
Name:
A       B \\
Name: \n A \t B \\
```

Escape Sequences	
\\	Backslash (\)
\’ And \”	Single(‘) And Double Quote(“)
\a	ASCII bell (BEL)
\b	ASCII backspace (BS)
\t	Horizontal tab (TAB)
\n	New Line

```
t(r"Name: \n A \t B \\\\ ")
ut:
: \n A \t B \\\\
```

Avoid Escape

Example Program

GO Back

```
# IS | is Not Operator(Is Work With Same Location Not Same Value)

A="Delwar"
B="Hossen"
C=A

print(A is C)

print(A is B)

print(A is not B)
```

Out Put: True

False

True

Example Program

GO Back

```
# in | not in Operator (Value Present Or Not)

A="I Am A STudent"

print("Am" in A)

print("Delwar" not in A)
```

out put:

True

True

Also See It Up Next Topic

Chapter-2

Function And Method Basic Introduction

Length Function:

```
Name="Md. Delwar Hossen"
print(len(Name))
```

Count Method:

```
Name="Md. Delwar Hossen"
print(Name.count("s"))
```

Lower Case Method: for check islower()

```
Name="Md. Delwar Hossen"
print(Name.lower())
```

Upper Case Method: for check isupper()

```
Name="Md. Delwar Hossen"
print(Name.upper())
```

String + Integer Not Allow

```
Number=int(input("Input Your Number: "))
print("Number Is: " + str(Number))
```

Multi Duble “ ” or ‘ ’ Not Allow

```
print("His Name Is 'Delwar Hossen'.")
```

(Strip | Replace |Find) Method

Left Strip And Rigth Strip:

```
Name="     Md. Delwar Hossen    "
Dots="."
print(Name.lstrip()+Dots) #Dot for See Right Space
print(Name.rstrip())
print(Name.strip())
```

Replace Method:

```
print(Name.replace(" ","")) #All Space Replace By None
```

Replace Method 2:

```
text="My name is asik.It is a nice day."
print(text.replace("is","Was"))
```

Perfect Length:

```
Name="        Md. Delwar Hossen      "
print(len(Name.replace(" ","")))
```

Find Method:

```
P="My Name Is Asik . My Country Is Bangladesh"
P1=(P.lower().find("is"))
print(f'{P1} {P.lower().find("is", P1+1)}')
```

Title Method:

```
Name="it was a nice day."
print(Name.title()) #First Word Capital
```

Center Method:

```
Name="Delwar Hossen"
print(Name.center(len(Name)+8,"*")) # Text Print Center And Two Side * Also use Space
```

If.. Else Condition

```
if 10>6:
    print("Big Number")
else:
    print("Small Number")
```

Elif Condition

```
if 10>5:
    print("Big Number: ")
elif 10<5:
    print("Small Number: ")
elif 10==5:
    print("Equal Number: ")
```

Short Hand if | if.. else

If:

```
a=10
b=5
if a > b: print("a is greater than b")
```

if else:

```
a = 2
b = 330
print("A") if a > b else print("B")
```

First output for if And second output for else…….

Logical Condition

And:

```
a="A"
b="A"
if a and b:
    print("a And b Same")
else:
    pass
```

Here “pass” means no print line avoid the print statement…

Or:

```
a="A"
b="B"
if a or b:
    print("Condition true")
else:
    pass
```

Condition game

Input a number If equal 45 Then print You win If blow 45 print Too Low!!! If upper 45 print Too High

```
win=45
n=int(input("Enter a number 1 to 100: "))
if n==win:
    print("You win!!!")
elif n>win:
    print("Too High!!!")
elif n<win:
    print("Too Low!!!")
```

Float After (.) Value limit

Round Founction:

```
print(round(5**.25,4))
```

```
print(round(2**0.5,4)
```

Randeom Number

```
import random

print(random.randrange(1,10)) #We will see more about import library Up next
```

(in | not in) Keyword

In Keyword:

```
Name="my name is asik"
if "is" in Name:
    print("The Keyword In The String")
else:
    print("The Keyword Not In The String")
```

Not In Keyword:

```
Name="my name is asik"
if "is" not in Name:
    print("The Keyword In The String")
else:
    print("The Keyword Not In The String")
```

While Loop

```
i=1
while i<=10:
    print("hello world")
    i=i+1
```

1 to 10 sum:

```
i=1
sum=0
while i<=10:
    sum+=i
    i=i+1
print(sum)
```

Input A String Number (ex. 12345) Sum the number (1+2+3+4+5)

```
num=input("Enter Your Number: ")
n=len(num)
i=0
sum=0
while i<n:
    sum+=int(num[i])
    i+=1
print(sum)
```

Input a name count every character without duplicate

```
Name=input("Enter A Name: ").lower()
n=len(Name)
i=0
temp=""
while i<n:
    if Name[i] not in temp:
        c=Name.count(Name[i])
        print(f"{Name[i]} = {c}")
    temp+=Name[i]
    i+=1
```

Infinite Loop

```
i=0
while i<=10:
    print("Hello World!!")
```

OR

```
while True:
    print("Hello World!!")
```

If You run this program your IDE will be hang……

For Loop

Syntax:

```
for i in range(i value,end length,increse or decrese):
    print()
```

10 time print "Hello world"

```
for i in range(10):
    print("Hello world!!")
```

By default i=0

Input a number ex. 12345 Sum it 1+2+3+4+5

```
name=input("Enter A Number: ")
n=len(name)
sum=0
for i in range(n):
    sum+=int(name[i])
print(sum)
```

Break and Continue

Break:

```
for i in range(10):
    if i==5:
        break
    print(i)
```

continue:

```
for i in range(10):
    if i==5:
        continue
    print(i)
```

Break =(Loop Stop) And continue=(avoid Only This Number)

Number Guessing Game With Loop

```
import random
win=random.randint(1,100)
n=int(input("Enter A Number Between 1 to 100: "))
for i in range(11):
    if n==win:
        print(f"You Win!!! => You try {i} times")
        break
    elif n<win:
        print("Too Low")
    elif n>win:
        print("Too High")
    elif 1>n<100:
        print(f"{n} Not Allowed")
    n=int(input(f"Try Again you have {11-i} Times: "))
```

One line print

```
print("* ",end=(""))
```

Loop Pattern Program 1

```
n=int(input("Enter max pattan number: "))
for i in range(n+1):
    for j in range(i):
        print('* ', end="")
    print("")
for i in range(n+1,0,-1):
    for j in range(i):
        print('* ', end="")
    print("")
```

out put:

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

Loop Pattern Program 2

```
n=int(input("Enter your pattan max number: "))
for i in range(n+1,0,-1):
    for j in range(i):
        print('* ',end="")
    print("")
for i in range(1,n+1):
    for j in range(i):
        print('* ',end="")
    print("")
```

```
out put:   * * * * * *
            * * * * *
            * * * *
            * * *
            * *
            *
            *
            * *
            * * *
            * * * *
            * * * * *
```

Loop Pattern Program 3

```
n=int(input("Enter maximum pattarn number: "))
e=""
for i in range(n,0,-1):
    e+=" "
    for j in range(i):
        print("* ",end=(""))
    print()
    print(end=e)
```

```
out put:



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

Loop Pattern Program 4

```
n=int(input("Enter maximum pattarn number: "))
e=""
f=""
for i in range(n,0,-1):
    e+=" "
    print(end=e)
    for j in range(i):
        print("*",end=(" "))
    print()
l=len(e)
for i in range(1,n+1):
    f=e[0:l]
    print(end=f)
    for j in range(i):
        print("*",end=(" "))
    print()
    l=len(e)-i
```

Out Put:

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

For Loop String

```
name="Delwar"
for i in name:
    print(i)
```

Out Put:

```
D
e
l
w
a
r
```

Import Random Method (Randrange|Choice|Shuffle)

```
#Already we ware seen how import random number
#Now we will see advances import method
import random
List=[8,9,6,2,5,7,4,1,8]      #We can also use tuple
print(random.choice(List))    #choice for List [String Or Number]
```

Output:

#Every Time Different number of Listed

```
#Older Random Number Generate Modify
import random
Win=random.randrange(0,100,2)
#Here randrange(Start,End,Step)
print(Win)
```

Output:

#Every Time Different Even Number Print

Shuffle List Data

```
import random

Color=["Yello","Red","Blue","Black","Green"]

random.shuffle(Color)   #Shuffle For List [String Or Number]

print(Color)
```

Output:

#Every Time Different Color Sequence Print

Shuffle For String

```
import random
Name='Delwar'
Name=list(Name)
#Now String Separated by ['D', 'e', 'l', 'w', 'a', 'r']
random.shuffle(Name)
Shuffle_Str="".join(Name)
print(Shuffle_Str)
```

Output:

#Every Time Different Sequence Print

Checking Multiple Condition

```
C=19
D=23
if C<20 and D>10:
    print("If Ok")
```

Chapter-3

Multi Integer Input

```
n1,n2,n3=map(int,input("Enter Your n1,n2,n3: ").split(","))
#map(function name without(),List or tuple) #Here int function And input split like list tuple
```

For more see next Map topic.....

Using Created Function

```
# Average funtion

def ave_three(a,b,c):
    return ((a+b+c)/3)

n1,n2,n3=map(int,input("Enter Your n1,n2,n3: ").split(" "))
print(ave_three(n1,n2,n3))
```

Output:

Enter Your n1,n2,n3: 9 8 5

7.333333333333333

Example 2: Sum Two Number

```
def print_fun(a,b):
    print(a+b)
print_fun(4,5)
```

Example 3: Find Big Number

```
def big_small(a,b):
    return f"Big Number {a}" if a>b else f"Big Number {b}"

n1,n2=map(int,input("Enter n1 space n2: ").split(" "))
print(big_small(n1,n2))
```

Output:

Enter n1 space n2: 44 55

Big Number 55

Fibonacci Using Function

```
def fibo(n):
    f1=0
    f2=1
    for i in range(n):
        print(f1, end=" ") # No need return Becouse only one valu return it
        sum=f1+f2
        f1=f2
        f2=sum
n=int(input("Enter your Number: "))
fibo(n)
```

Output:

Enter your Number: 10

0 1 1 2 3 5 8 13 21 34

Odd Even Function

```
def sort_oddeven(n):
    return n%2==0
#Here n%2==0 is ok then print True Or Not Ok Print False

print(sort_oddeven(9))
```

Output: False

OR:

```
def odd_even(a):
    if a%2 == 0:
        return "even"
    else:
        return "odd"
print(odd_even(10))
```

OR:

```
def odd_even(a):
    if a%2 ==0:
        return "Even"
    return "Odd"
print(odd_even(9))
```

OR:

```
def odd_even(a):
    return "even" if a%2==0 else "odd"
print(odd_even(9))
```

Palindrome Number Using Function

If input number or name reverse name or number same then its Palindrome Number

```
def pal(word):
    return  word==word[-1::-1]
name=input("Enter keyword: ")
print(pal(name))
```

Output: Enter keyword: ASSA

True

No Parameters Function

```
def Hello():
    return "Hello World!!!!!"
print(Hello())
```

Out Put:

Hello World!!!!!

EXTRA :

Input A string print last value

```
def last_valu(a):
    return a[-1]
name=input("Enter Your Name: ")
print(last_valu(name))
```

Function Inside Function

```
def big(a,b):
    return a if a>b else b
def gater(a,b,c):
    return big(big(a,b),c)

a,b,c=map(int,input("Enter a,b,c: ").split(","))
print(gater(a,b,c))
```

Output:

Enter a,b,c: 8,9,10

10

Important Function Argument Defult Genarate

```
def my_function(*kids): #Its Generate Tuple
  print("The youngest child is " + kids[2])

my_function("Emil", "Tobias", "Linus")
```

Output:

```
The youngest child is Linus
```

Default Parameters Function

```
def Def(first_name,Last_name='unkonwn',age=None):

    #Without Last Perameter default other perameter never default
    #like Not Possible def Def(first_name,Last_name='unkonwn',age)

    print(f"First Name {first_name}")
    print(f"Last Name {Last_name}")
    print(f"Age is {age}")
Def("Asik")
```

Out Put:

```
First Name Asik

Last Name unkonwn

Age is None
```

Scope Variable (Global | Local)

```
# But we are not using this method
# not change golobal into function
#only time weast
X=7 #Global Variable
Z=8
def var():
    global Z #using global variable into fuction
    Z=55
    y=5 #local variable
    return y
print(Z) #Before Call var() function global variable not chaneg Z=8
print(X) #X global Print 7
print(var()) #function call print 5
print(Z) #After call Var() function call (global Z) value change Z=55
print(y)  #not work direct local variable call
```

OutPut:

```
8

7

5
```

```
55
Traceback (most recent call last):
  File "e:/code/C4/Var_scope.py", line 16, in <module>
    print(y)  #not work direct local variable call
NameError: name 'y' is not defined
```

Chapter-4

List

```
Num=[1,2,3,4] #Integer
print(Num)

Str=["Asik",'asik'] #String
print(Str)

Mixed=[1,"asik",3.5,True,None]
#Integer String Float Boolean Nul
print(Mixed)
print(Mixed[2]) #Single Value Print
```

Out Put: [1, 2, 3, 4]

['Asik', 'asik']

[1, 'asik', 3.5, True, None]

3.5

List Control

```
Mixed=[1,"asik",3.5,True,None]
print(Mixed[-1])
Mixed[1]="Asik" #Replace
print(Mixed)
Mixed[1:]=["asik",5.6,False] #Re Add List Value
print(Mixed)
```

Output:

None

[1, 'Asik', 3.5, True, None]

[1, 'asik', 5.6, False]

Add List Data

Append Method:

```
Program=["java","C"]
print(Program)

Program.append("Python") #Add Data in list Last position
print(Program)
```

Output:

['java', 'C']

['java', 'C', 'Python']

TO Replace A Item (((Program[0]=”Python”))))

Real Life Program:

```
#Real Life Program
Program=[]
Program.append("java")
Program.append("Python")
print(Program)
```

Output:

['java', 'Python']

Insert Method:

```
Program1=["Java","C#"]
Program1.insert(0,"Python")
#Inset Method (Position Num, Data)
print(Program1)
```

OutPut: ['Python', 'Java', 'C#']

Combine Two List:

```
#Combine Two List
Program1=["Java","C#"]
Program2=["Go","C","Ruby"]
Program=Program1+Program2     #Build New Location Like Copy
print(Program)
```

Output: ['Java', 'C#', 'Go', 'C', 'Ruby']

Extend Method: Combine Two List

```
Program1=["Java","C#"]
Program2=["Go","C","Ruby"]
#Extend Method Add A List To other list Permanently
Program1.extend(Program2)
print(Program1)
```

Output:

['Java', 'C#', 'Go', 'C', 'Ruby']

List Inside List Append Method:

```
Program1=["Java","C#"]
Program2=["Go","C","Ruby"]
#If We Use Append Method to add A list To other list
Program1.append(Program2)
print(Program1) # Add also With Brakets Full List
```

Output:

[,'Java', 'C#', ['Go', 'C', 'Ruby']]

Delete List Data

Pop Method:

```
Program=["java","Python","C#"] #0,1,2
Program.pop() #Pass No Position Default Last Value pop
print(Program)

Program.pop(1) #Pass a position poped
print(Program)
```

Output:

['java', 'Python']

['java']

Delete Operator: or statement

```
Program=["Ruby","Go","Ruby","Python"]
del Program[1] #Delete Operator

print(Program)
```

Output:

['Ruby', 'Ruby', 'Python']

Remove Method:

```
Program=["Ruby","Go","Ruby","Python"]
Program.remove("Ruby") #If we don't know the location but know data then use remove()
print(Program)
```

Output:

['Go',Ruby', 'Python']

Important List Method And Function

Length Function:

```
#Length Function
List=["Apple",3.25,5,True,None] #You Can Enter Any Type Data

print(len(List))
```

Output:

5

Count Menthod:

```
#Count Method
List=["Apple",3.25,5,"Apple",None]
print(List.count("Apple"))
```

Output:

2

Sorted Function:

```
#Sorted Function Temporary
List=["Mango","Banana","Orange","Apple"]
#Not Permanent Change

print(sorted(List))
print(List)
```

Output:

['Apple', 'Banana', 'Mango', 'Orange']

['Mango', 'Banana', 'Orange', 'Apple']

Sort Method:

```
#sort method
List=["Mango","Banana","Orange","Apple"]
#Permanent Change
print(List.sort()) #Becouse data return old data not lost
print(List) #Then Permanent Change Data
```

OutPut: None

['Apple', 'Banana', 'Mango', 'Orange']

Reverse Method:

```
#Reverse method
List=["Mango","Banana","Orange","Apple"]
#Permanent Change
List.reverse() #Data Revese data return no lost orginal data
print(List) #Then Permanent Changeed Data
```

OutPut:

['Apple', 'Orange', 'Banana', 'Mango']

Copy Method:

```
#Copy method
List=["Mango","Banana","Orange","Apple"]
List2=List.copy()  #Copy Make A New Location But if (=) Use save same location
print(List2)
```

Output:

['Mango', 'Banana', 'Orange', 'Apple']

Clear Method:

```
#Clear method
List=["Mango","Banana","Orange","Apple"]
List.clear()
print(List)
```

Output:

[]

Loop In List

For Loop:

Same Like String For Loop

Example

```
Name="Asik"
for i in Name:
    print(i)
print()
```

output:

A
S
I
K

Foor Loop In List

```
#For Loop
List=["Apple","Mango","Orange","Banana","Licu"]
for item in List:
    print(item)
```

Output:

Apple
Mango
Orange
Banana
Licu

OR: For Loop

```
#For Loop
List=["Apple","Mango","Orange","Banana","Licu"]
for item in range(len(List)):
    print(List[item])
```

Output:

Apple

Mango

Orange

Banana

Licu

While Loop:

```
#While Loop
List=["Apple","Mango","Orange","Banana","Licu"]
item=0
while item<len(List):
    print(List[item])
    item+=1
```

Output:

Apple

Mango

Orange

Banana

Licu

Loop In (2D and 3D) List

2D List Print:

```
#loop in 2D List
List=[[1,2,3,4],[4,5,6],[7,8,9,0]]
for item in List:
    print(item)
```

Output:

[1, 2, 3, 4]

[4, 5, 6]

[7, 8, 9, 0]

2D List Every Element Print:

```
#loop in 2D List
List=[[1,2,3,4],[4,5,6],[7,8,9,0]]
for item in List:
    for j in item:
        print(j)
```

OutPut:

1

2

3

4

4

5

6

7

8

9

0

3D list Loop:

```
#loop in 3D List
List=[[[1,2],[3,4]],[[4,5],[6]],[[7,8,9],[0]]]
for item in List:
    for item2d in item:
        for item3d in item2d:
            print(item3d)
```

Output:

1

2

3

4

4

5

6

7

8

9

0

Access 2D And 3D List Data

```
#Access 2d 3d list data
List2D=[[1,2,3,4],[4,5,6],[7,8,9,0]]
List3D=[[[1,2],[3,4]],[[4,5],[6]],[[7,8,9],[0]]]
print(List2D[1][2]) #List2D[List Item number][Inside List Item Num]
print(List3D[2][0][2])
```

Output :

6

9

Split And Join Method In List

Split Method

String To List

```
List="My Name Is Asik".split(" ")
```

Output:

['My', 'Name', 'Is', 'Asik']

```
#Split Method
List1,List2,List3,List4="My Name Is Asik".split(" ") #also use Input
#method Change permanently
print(List1)
print(List2)
print(List3)
print(List4)
```

Output:

My

Name

Is

Asik

Join Method:

```
#join Method
List0=["Apple","Mango","Orenge"] #Only For String
List0=','.join(List0) #Output Apple,Mango,Orenge
print(List0)
print(List0.replace(","," "))  #Replace Space To , coma
```

Output:

Apple,Mango,Orenge

Apple Mango Orenge

(in | not in) List

```
List=["Apple",3.2225,None,True,8]
if "Apple" in List:
    print("Ok")
else:
    print("Not Ok")
```

output:

Ok

Not in:

```
List=["Apple",3.2225,None,True,8]
if "Mango" not in List:
    print("Ok")
else:
    print("Not Ok")
```

OutPut: Ok

(IS | Equal) List

```
List1=["apple","Orange","Mango"]
List2=["apple","Orange","Mango"]
List3=List1
print(List1==List2) #Equal Check Same valu or not
print(List1 is List2) #Is Check Same Location store Or Not
print(List1 is List3)
```

OutPut: True

False

True

Check Type Data

```
List=[[1,2,3,4],[5,6,7],[8,9,0]]
Name="Delwar"
Num=20
print(type(List))
print(type(Name))
print(type(Num))
```

OutPut:

<class 'list'>

<class 'str'>

<class 'int'>

String Vs List immutable And mutable

String Immutable

```
Name="delwar"
print(Name.title()) #No Change Immutable
print(Name) #No Change
Name2=Name.title() #Store Other Variable
print(Name2) #Changed
```

Output:

Delwar

delwar

Delwar

Mutable List

```
List=["Asik","Rofiq","Sofiq"]
List.pop() #Changed Mutable
print(List)

List.append("Rohim") #Changed
print(List)
```

Output:

['Asik', 'Rofiq']

['Asik', 'Rofiq', 'Rohim']

List Index Method

```
List1=[1,2,3,6,5,4,8,9,7,1]
#List indexing if Same Data in The List
print(List1.index(8,2,len(List1)))
print(List1.index(1,2,len(List1))) #find position
#List1.index(Finded value,Start Position,End Position)
```

Output:

6

9

Pass List A Function

```
#Pass List In A Function
List0=[1,2,3,6,5,4,8,9,7,1]
def neg(l):
    neg=[]
    for i in l:
        neg.append(-i) #Make The List Negetive
    return neg
print(neg(List0))
```

Output:

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

List Duplicate Remove

```
def duplicate(l):
    Temp=[]                 #Using  Temp Value
    for item in l:
        if item  not in Temp:
            Temp.append(item)
    return Temp

List=['abc', 'xyz', 'abc', '121','123','121']
print(duplicate(List))
```

Output:

['abc', 'xyz', '121', '123']

List Max Min Function

```
#Find Max Min
List=[55,44,99]
def min_max(l):
    print(f"Minimum Value Is= {min(l)}")
    print(f"Maximum Valie Is= {max(l)}")
min_max(List)
```

```
#Find Max Min Sub
List2=[54,44,99]
def min_max(l):
    return max(l) - min(l)

print(min_max(List2))
```

Output:

Minimum Value Is= 44

Maximum Valie Is= 99

55

List Max Min Without Function

```
def large(l):
    max=l[0]
    for item in l:
        if item > max:
            max=item
    return max

List=[99,55,77,22,33]
print(large(List))
```

output:

99

Exercise

1.If A list item length upto 2 or more and item 1st and last value same then print the item length

```
def match_words(l):
    Count=0
    for item in l:
        if (len(item)>=2) and (item[0]==item[-1]):
            Count+=1
    return Count

List=['abc', 'xyz', 'aba', '1221']
print(match_words(List))
```

Output: 2

2.Check How Many List Inside List

```
def Check_List(l):
    c=[]
    for item in l:
        if type(item)==list:
            c.append(item)
    return len(c)
List=[1,2,3,[1,2,3],[4,5,6]]
```

```
print(Check_List(List))
```

Output:

2

3.Two List Print The Common Item

```
def common(l1,l2):
    n=[]
    for item in l1:
        if item in l2:
            n.append(item)
    return n
List1=[1,2,5,6,9]
List2=[2,3,6,5,8,4,6]
print(common(List1,List2))
```

Output:

[2, 5, 6]

4.Filter Odd Even Number

```
def odd_even(l):
    odd=[]
    even=[]
    for item in l:
        if (item%2==0):
            even.append(item)
        else:
            odd.append(item)
    n=[even,odd]
    return n
List=[1,2,3,4,5,6,7,8,9]
print(odd_even(List))
```

Output:

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

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Chapter 5

Tuple Immutable | Indexing | Slicing | Count | Len

```
Tuple=tuple(range(1,11))
Tuple2=(15,6,8,47,5,10,10)

print(Tuple[5]) #Indexing
print(Tuple2[-1::-1]) #Slicing
print(Tuple2.count(10)) #Count Method
print(len(Tuple)) #len Function
```

Output: 6

(10, 10, 5, 47, 8, 6, 15)

2

10

Important Note: Tuple Is immutable so item add delete not possible

Also many Method Function Not possible which we used in list

Tuple Loop

```
Tuple=(1,3,6,5,4,7,8,9,5)
for item in Tuple:
    print(item,end=" ")
```

Output: 1 3 6 5 4 7 8 9 5

Single Element in tuple And Check Type

```
Tuple=("Apple") #Type List
print(type(Tuple))
Tuple2=tuple("Apple")
print(type(Tuple2))
Tuple3=("Apple",)
print(type(Tuple3))
```

Output: <class 'str'>

<class 'tuple'>

<class 'tuple'>

Function Return Two values

```
def sum_multi(int1,int2):
    Sum=int1+int2
    Multi=int1*int2
    return Sum, Multi
print(sum_multi(2,4))
Sum,Multi=sum_multi(2,4)
print(Sum)
print(Multi)
```

Out Put:

(6, 8) (Note: Return two output showing tuple)

6

8

Tuple To List (append) To Tuple To String | Delete Tuple

```
Tuple=tuple(range(1,8))
print(Tuple)

#Tuple to list to tuple
List=list(Tuple)
print(List)

List.append(6) #Append Item
Tuple=tuple(List)
print(Tuple)

Tuple2=tuple(range(1,8))
Str=str(Tuple2)
print(Str)
print(type(Str))

#Join Tuple
T1=(1,2,3,4)
T2=(5,6,7,8)
T3=T1+T2
print(T3)

#Delete tuple
del T1
print(T1) #Error Becouse Tuplw Doesn't Exist
```

Output:

(1, 2, 3, 4, 5, 6, 7)

[1, 2, 3, 4, 5, 6, 7]

(1, 2, 3, 4, 5, 6, 7, 6)

(1, 2, 3, 4, 5, 6, 7)

<class 'str'>

(1, 2, 3, 4, 5, 6, 7, 8)

print(T1) #Error Becouse Tuplw Doesn't Exist

NameError: name 'T1' is not defined

Without parentheses called Tuple

```
Tuple="apple","mango","banana"
print(Tuple)
print(type(Tuple))
```

OutPut:

('apple', 'mango', 'banana')

<class 'tuple'>

Tuple Unpacking

```
Tuple2=("Dhasik","Malik","karim")
E1,E2,E3=Tuple2
print(E1,E2,E3)
```

Output:

Dhasik Malik karim

List Inside Tuple Access

```
Tuple3=(1,2,3,["Apple","Mango","Orange"])
Tuple3[3].pop() #Access Tuple Inside List
print(Tuple3)
Tuple3[3].append("Apple")
print(Tuple3)
```

Output:

(1, 2, 3, ['Apple', 'Mango'])

(1, 2, 3, ['Apple', 'Mango', 'Apple'])

Max Min Sum Function Tuple

```
T=tuple(range(1,11))
print(min(T))
print(max(T))
print(sum(T))
```

OutPut:

1

10

55

Tuple deepcopy() function Import

```
import copy
Tuple=(1,2,3,5,6,8,9)
Tuple2=copy.deepcopy(Tuple)
print(Tuple2)
```

Output:

1, 2, 3, 5, 6, 8, 9)

Tuple Data Add No Method

```
T=(1,2,3,6,5,4)
T+=(8,)
print(T)
```

Output:

(1, 2, 3, 6, 5, 4, 8)

Chapter 6

Dictionary Intro [Most important here keys works for index values]

Anything You can store like List (String, int, keyword, list, tuple, Boolean etc). Important Note: Like "Name" Calles Keys : "DHAsik" Calles Values. ((Indexing >>List D[1] Same Dictionary D["Name"]))

```
#Create Dictionary
#First System
DIC={"Name":"DHAsik","Age":23,"Sex":None,"Movie":['COCO','Robot','Mowna']}
print(DIC)
```

Output:

{'Name': 'DHAsik', 'Age': 23, 'Sex' : None, 'Movie': ['COCO', 'Robot', 'Mowna']}

```
#2nd System
DIC={
    "Name":"DHAsik",
    "Age":23,
    "Sex":None,
    "Movie":['COCO','Robot','Mowna']
}

print(DIC)
```

Output:

{'Name': 'DHAsik', 'Age': 23, 'Sex': None, 'Movie': ['COCO', 'Robot', 'Mowna']}

```
#3rd System
DIC=dict(Name="DHAsik", Age=23)
print(DIC)
```

Output:

{'Name': 'DHAsik', 'Age': 23}

Access Dictionary Data

```
#Dictionary unhashable Slice
#Unhashable to indexing
DIC={
    'Name': 'DHAsik',
    'Age': 23,
    'Sex' : None,
    'Movie': ['COCO', 'Robot', 'Mowna']
    }
# print(DIC[1]) #Output Error
print(DIC["Name"])
```

Output: DHAsik

Access Dictionary Inside List Data

```
DIC={
    'Name': 'DHAsik',
    'Age': 23,
    'Sex' : None,
    'Movie': ['COCO', 'Robot', 'Mowna']
    }
# print(DIC[1]) #Output Error
print(DIC["Movie"][1])
```

Output:

Robot

Add Data Into Dictionary

```
D={}        #Empty Dictionary
D["Name"]="DHAsik"
D["Age"]=23
D["Movie"]=["Coco","Robot"]
print(D)
```

Output: {'Name': 'DHAsik', 'Age': 23, 'Movie': ['Coco', 'Robot']}

Delete Data Into Dictionary (pop | popitem) Poped Tuple

```
D={'Name': 'DHAsik', 'Age': 23, 'Movie': ['Coco', 'Robot']}
# D.pop() #Show Error Not Like List
#Must be pass 1 Argument
poped=D.pop("Movie")
print(poped) #Itshow Tuple keys and vluess
print(type(poped_item))
print(D)
```

output: ('Movie', ['Coco', 'Robot'])

<class 'tuple'>

{'Name': 'DHAsik', 'Age': 23}

```
D={'Name': 'DHAsik', 'Age': 23, 'Movie': ['Coco', 'Robot']}
poped_item=D.popitem() #No Argument Pass Delete last value
print(poped_item) #poped item saved as tuple 1st value keys 2nd value Data
print(type(poped_item))
print(D)
```

Output: ('Movie', ['Coco', 'Robot'])

<class 'tuple'>

{'Name': 'DHAsik', 'Age': 23}

We Can Also Use >>>> del Statement <<<<<<

Copy With get() method Normal Error Handle

Normal Data error

```
D={'Name': 'DHAsik', 'Age': 23, 'Movie': ['Coco', 'Robot']}
Copy=D["namesss"] #Wrong Key enter output show error
print(Copy)
```

output:

Traceback (most recent call last):

File "test.py", line 2, in <module>

Copy=D["Names"]

KeyError: 'Names'

Get method

```
D={'Name': 'DHAsik', 'Age': 23, 'Movie': ['Coco', 'Robot']}
copy=D.get('Name')
print(copy)
```

Output:

DHAsik

Get() Method Error

```
D={'Name': 'DHAsik', 'Age': 23, 'Movie': ['Coco', 'Robot']}
copy=D.get('namess') #Enter Wrong Keys
print(copy)
```

Output:

None

Get() Method Error Replace User Text

```
D={'Name': 'DHAsik', 'Age': 23, 'Movie': ['Coco', 'Robot']}
copy=D.get('namess',"Data Not Found") #Enter Wrong Keys
print(copy)
```

Output:

Data Not Found

If Condition With Get method

```
D={'Name': 'DHAsik', 'Age': 23, 'Movie': ['Coco', 'Robot']}
if D.get('Name'):
    print("Present")
else:
    print("Not Present")
```

Output:

Present

Duplicate Data Not Store Dictionary

```
D={
    'Name': 'DHAsik',
    'Age': 23,
    'Movie': ['Coco', 'Robot']
    }
D["Name"]="Delwar Hossen"     #Its Replace older  Data
print(D)
```

Output:

{'Name': 'Delwar Hossen', 'Age': 23, 'Movie': ['Coco', 'Robot']}

Replace Older Data of Name

Keys And Values Method Using in

What is the out put keys and values method

```
DIC={'Name':'Delwar','Age':23,'Movie':['Coco','Robot','2.0'],'Player':['Shakib','Tamim']}
item=DIC.values()
item2=DIC.keys()
print(item)
print(type(item))
print(item2)
```

Output: Output Looks Like List But Its Not List

dict_values(['Delwar', 23, ['Coco', 'Robot', '2.0'], ['Shakib', 'Tamim']])

<class 'dict_values'>

dict_keys(['Name', 'Age', 'Movie', 'Player'])

If You want to check with keys method

```
DIC={'Name':'Delwar','Age':23,'Movie':['Coco','Robot','2.0'],'Player':['Shakib','Tamim']}
#For Keys Normal Method
if 'Name' in DIC:
    print("Present")
else:
    print("Not Present")


# Or Keys Method
if 'Name' in DIC.keys():
    print("Present")
else:
    print("Not Present")
```

Output: Present

Present

Values() Method For Check Value In the Dictionary or not

```
DIC={'Name':'Delwar','Age':23,'Movie':['Coco','Robot','2.0'],'Player':['Shakib','Tamim']}
# VAlues Method
if (23 or ['Shakib','Tamim']) in DIC.values():
    print("Present")
else:
    print("Not Present")
```

Output:

Present

Looping into dictionary

```
DIC={'Name':'Delwar','Age':23,'Movie':['Coco','Robot','2.0'],'Player':['Shakib','Tamim']}
for item in DIC: # We Can also use DIC.keys() Method Same Work
    print(item)
```

Output:

Name

Age

Movie

Player

For Print Values

```
DIC={'Name':'Delwar','Age':23,'Movie':['Coco','Robot','2.0'],'Player':['Shakib','Tamim']}
for item in DIC.values():
    print(item)
```

Out Put:

Delwar

23

['Coco', 'Robot', '2.0']

['Shakib', 'Tamim']

Or

```
DIC={'Name':'Delwar','Age':23,'Movie':['Coco','Robot','2.0'],'Player':['Shakib','Tamim']}
for item in DIC:
    print(DIC[item])
```

Out Put:

Delwar

23

['Coco', 'Robot', '2.0']

['Shakib', 'Tamim']

Items Method

Very UseFull Method

Items method Select a Dictionary and Make it Tuple (Keys,Values) But Tuple Type dict inside [] like list

But [] Its not a list its called 'dict_items'. So you can't use it like list (Add Delete Or other)

[(,),(,),(,),(,)] But We Can Looping

```
DIC={'Name':'Delwar','Age':23,'Movie':['Coco','Robot','2.0'],'Player':['Shakib','Tamim']}
Data=DIC.items()
print(Data)
print(type(Data))
```

Output:

dict_items([('Name', 'Delwar'), ('Age', 23), ('Movie', ['Coco', 'Robot', '2.0']), ('Player', ['Shakib', 'Tamim'])])

<class 'dict_items'>

Print Keys And Values Using items() Method

```
DIC={'Name':'Delwar','Age':23,'Movie':['Coco','Robot','2.0'],'Player':['Shakib','Tamim']}
for key,value in DIC.items():
    print(f"{key} value is {value}")
```

Output:

Name value is Delwar

Age value is 23

Movie value is ['Coco', 'Robot', '2.0']

Player value is ['Shakib', 'Tamim']

Print Keys And Values Without Using items Method

```
DIC={'Name':'Delwar','Age':23,'Movie':['Coco','Robot','2.0'],'Player':['Shakib','Tamim']}
for item in DIC:
    print(f"{item} value is {DIC[item]}")
```

Output:

Name value is Delwar

Age value is 23

Movie value is ['Coco', 'Robot', '2.0']

Player value is ['Shakib', 'Tamim']

Fromkeys() Method in Dictionary

```
# Crate A Dictionary with fromkeys() Method
# use List On Keys
DIC=dict.fromkeys(["Name","Age","Movie"],"Unknwon")
print(DIC)
```

Output:

{'Name': 'Unknwon', 'Age': 'Unknwon', 'Movie': 'Unknwon'}

```
# Also use tuple On Keys
DIC2=dict.fromkeys(('Name','Age','Hight'),'Unknown')
print(DIC2)
```

Output:

{'Name': 'Unknown', 'Age': 'Unknown', 'Hight': 'Unknown'}

IF We Don't Use List Or Tuple On Keys

```
#Why Useing List or Tuple
DIC3=dict.fromkeys("Name",'Unknown')
print(DIC3)
```

Output:

{'N': 'Unknown', 'a': 'Unknown', 'm': 'Unknown', 'e': 'Unknown'}

Using Range Function To Generate Keys

```
DIC4=dict.fromkeys(range(1,11),"Unknown")
print(DIC4)
```

Output:

{1: 'Unknown', 2: 'Unknown', 3: 'Unknown', 4: 'Unknown', 5: 'Unknown', 6: 'Unknown', 7: 'Unknown', 8:

'Unknown', 9: 'Unknown', 10: 'Unknown'}

Copy() Method in Dictionary

```
#Copy Method
D={"Name":"Delwar","Age":23,}
D1=D.copy()
print(D1)
print(D1 is D) #print false becouse diffrent location
print(D1 == D) #print true becouse check same value or not
```

Output:

{'Name': 'Delwar', 'Age': 23}

False

True

Clear() Method in Dictionary

```
D={"Name":"Delwar","Age":23,}
D1=D #Its no copy its same access one location
#Clear Method
D.clear()
print(D)
print(D1) #Output Empty
# So do not use =
#Must use copy()
```

Output:

{}

{}

Update() Method in Dictionary

```
#Update Like Append method
DIC={
    'Name':'Delwar',
    'Age':23,
    'Movie':['Coco','Robot','2.0'],
    'Player':['Shakib','Tamim'],
}
more_DIC={
    'city':['Bheramara','Rongpur','Dhaka'],
    'Vill':'candipur',
}
more_DIC2={
    'Name':'Delwar Hossen', #duplicate Name Replace Older data
    'city':['Bheramara','Rongpur','Dhaka'],
    'Vill':'candipur',
}
DIC.update(more_DIC)
print(DIC)

DIC.update(more_DIC2)
print(DIC) #Name Riplaced
```

Output:

{'Name': 'Delwar', 'Age': 23, 'Movie': ['Coco', 'Robot', '2.0'], 'Player': ['Shakib', 'Tamim'], 'city': ['Bheramara', 'Rongpur', 'Dhaka'], 'Vill': 'candipur'}

{'Name': 'Delwar Hossen', 'Age': 23, 'Movie': ['Coco', 'Robot', '2.0'], 'Player': ['Shakib', 'Tamim'], 'city': ['Bheramara', 'Rongpur', 'Dhaka'], 'Vill': 'candipur'}

Cube Finder

```
def cube_find(l):
    D={}
    for item in range(1,l+1):
        D[item]=item**3
    return D

n=int(input("Enter cube Value: "))
print(cube_find(n))
```

Output:

{1: 1, 2: 8, 3: 27, 4: 64, 5: 125, 6: 216, 7: 343, 8: 512}

User Input get data and print it like Dictionary

```
D={}
Name=input("Enter Your Name: ")
Age=int(input("Enter Your Age: "))
Movie=input("Enter Your Fev Movie list use ,:").split(",")
D["Name"]=Name
D["Age"]=Age
D["Movie"]=Movie
print(D)

for n1,n2 in D.items():
    print(f"{n1}:{n2}")
```

Output:

Enter Your Name: Delwar

Enter Your Age: 23

Enter Your Fev Movie list use ,:Coco,Robot

{'Name': 'Delwar', 'Age': 23, 'Movie': ['Coco', 'Robot']}

Name:Delwar

Age:23

Movie:['Coco', 'Robot']

Word Counter

```
def word_count(n):
    #No Need Temp variable to store check item like list
    #Becouse Dictionary Removed Overwrite Item
    D={}
    for item in n:
        D[item]=n.count(item)
    return D
```

```
Name="Delwar Hossen"
print(word_count(Name.lower()))
```

Output:

{'d': 1, 'e': 2, 'l': 1, 'w': 1, 'a': 1, 'r': 1, ' ': 1, 'h': 1, 'o': 1, 's': 2, 'n': 1}

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Sum Method In Dictionary

Write a Python program to sum all the items in a dictionary

```
D={5:9,9:8,6:5,5:2}

print(sum(D.values()))
```

Output:

15

Chapter 7

Set (Create | Syntex)

```
#Set Immuatble
#Set unorderd Collection
#Set Create
s={1,2,3,5}
s1=set(range(1,6))
print(s)
print(s1)
```

OutPut:

{1, 2, 3, 5}

{1, 2, 3, 4, 5}

Set Duplicate Not Allow

```
#Duplicate Not Allow
s={1,2,3,5,6,6,4,4,8}
print(s)
```

Output:

{1, 2, 3, 4, 5, 6, 8}

List Duplicate Removed By Set

```
List=[1,2,3,6,12,2,5,112,2,2,3,5]
S=set(List)
List=list(S)
print(List)
```

Output:

[1, 2, 3, 5, 6, 12, 112]

Create Empty Set

```
S={} # its Not Set Its Dictionary
print(type(S))
```

Output:

<class 'dict'>

So How We Create Empty Set

```
S=set()
print(S)
print(type(S))
```

Output: set()

<class 'set'>

What type of data store in set

```
S={"Apple",1,3.1,None,(1,3,5)}
#String Int Float None Tuple Boolean
print(S)
```

Output:

{1, None, 3.1, (1, 3, 5), 'Apple'}

Not Stored Data

```
S={[2,6,5,9],{"A":6,3:6}}
#List Dictionary
```

Set Add Data (Add Method)

```
S=set()
for i in range(1,6):
    S.add(i)
print(S)
```

Output:

{1, 2, 3, 4, 5}

Delete Set Data

Remove Method

```
S={2,6,5,4,8,9,7}
S.remove(6)     #6 Removed
print(S)
```

Output:

{2, 4, 5, 7, 8, 9}

If Remove Method Data Not Present Show Error

```
S={2,6,5,4,8,9,7}
S.remove(10)
print(S)
```

Output:

Error

To Avoid Error We Use Discard

```
S={2,6,5,4,8,9,7}
S.discard(10)
print(S)
```

Ouput:

{2, 4, 5, 6, 7, 8, 9}

No Change

PoP Method

```
S={2,6,5,4,8,9,7}
S.pop()          #Last Data Poped
print(S)
```

Output:

{4, 5, 6, 7, 8, 9}

Delete Statement

```
S={2,6,5,4,8,9,7}
del S
print(S)
```

Output:

Error Becouse Set No More

Set Data | Copy | Clear | Method

Copy

```
S={2,6,5,4,8,9,7}
S2=S.copy()
print(S2)
```

Ouput:

{2, 4, 5, 6, 7, 8, 9}

Clear

```
S={2,6,5,4,8,9,7}
S.clear()
print(S)
```

Output:

set()

Set Data Update Method

```
S1={2,3,6,5,4}
S2={1,6,9,2,3,7}
S1.update(S2)
print(S1)
```

Output:

{1, 2, 3, 4, 5, 6, 7, 9}

Set | len | Sum |Max|Min| Function

```
S={2,3,6,5,4}
print(len(S))
print(sum(S))
print(min(S))
```

```
print(max(S))
```

Output:

5

20

2

6

Set | Union | Intersection | Difference

```
S1={2,3,6,5,4}
S2={1,2,5,6,9}
#Union
U=S1 | S2
U1=S1.union(S2) #Use Any One
print(U,U1)

#InterSection
I= S1 & S2
I1=S1.intersection(S2)  #Use Any One
print(I,I1)

#Difference
D=S1-S2
D1=S1.difference(S2)    #Use Any One
print(D,D1)
```

Output:

{1, 2, 3, 4, 5, 6, 9} {1, 2, 3, 4, 5, 6, 9}

{2, 5, 6} {2, 5, 6}

{3, 4} {3, 4}

Set | If Else | Looping

Condition

```
s={"a",1,2,3.0,(5,6)}

if "a" in s:
    print("Present")
else:
    pass
```

Output:

Present

Looping

```
s={"a",1,2,3.0,(5,6)}
for item in s:
    print(item)
```

Output:

1

2

3.0

(5, 6)

a

Chapter 8

List Comprehension

Simple Method Data Append Into List

```
L=[]
for i in range(1,6):
    L.append(i)
print(L)
```

Output:

[1, 2, 3, 4, 5]

Now Comprehension system

```
L=[i for i in range(1,6)]
print(L)
```

Output:

[1, 2, 3, 4, 5]

List Comprehension (Square Every append data) And (Negative Append)

```
List=[(i**2) for i in range(1,10)]
print(List)
```

Output:

[1, 4, 9, 16, 25, 36, 49, 64, 81]

```
Lisi1=[-i for i in range(1,10)]
print(Lisi1)
```

Output:

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

List Comprehension |If+For| Find Number Print It String

```
L=["A",None,"Delwar",5,6,9.1]
def Num(l):
    return [str(i) for i in l if (type(i)==int or type(i)==float)]
print(Num(L))
```

Output:

['5', '6', '9.1']

List Comprehension |If else+For| If string reverse it else negative it

```
List=["Delwar","Asik",1,5,6,2.0]
List2=[i[::-1] if type(i)==str else (-i) for i in List]
print(List2)
```

Output:

['rawleD', 'kisA', -1, -5, -6, -2.0]

List Comprehension | Multi If + Loop |

```
l = [1, 2, 3, "Asik",None]
List2=[f'{i} integer' if type(i) == int else f'{i} String' if type(i) == str else f'{i} Other Type' for i in l]
print(List2)
```

Output:

['1 integer', '2 integer', '3 integer', 'Asik String', 'None Other Type']

List Comprehension Example

```
#Reverse String by comprehension
List=["abc","xyz","hft"]
Rev_Stri=[i[-1::-1] for i in List]
print(Rev_Stri)

#using function
def Rev_Str(l):
    return [i[::-1] for i in l]
print(Rev_Str(List))
```

Output:

['cba', 'zyx', 'tfh']

['cba', 'zyx', 'tfh']

Nested List Comprehension

```
List=[[i for i in range(1,4)] for j in range(6)]
print(List)
```

Outptu:

[[1, 2, 3], [1, 2, 3], [1, 2, 3], [1, 2, 3], [1, 2, 3], [1, 2, 3]]

Chapter 9

Dictionary Comprehension

Create Dictionary By Comprehension

```
#Key i And Value i square
DIC={i:i**2 for i in range(1,9)}
print(DIC)
```

Output:

{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}

EX.2

```
#count Charecter Without Duplicate
Name="delwar hossen asik"
DIC={i:Name.count(i) for i in Name}
print(DIC)
```

Output:

{'d': 1, 'e': 2, 'l': 1, 'w': 1, 'a': 2, 'r': 1, ' ': 2, 'h': 1, 'o': 1, 's': 3, 'n': 1, 'i': 1, 'k': 1}

Dictionary Comprehension IF Else Condition

```
#Crate Odd Even Dictionary
DIC={i:("Even" if i%2==0 else "Odd") for i in range(1,10)}
print(DIC)
```

Output:

{1: 'Odd', 2: 'Even', 3: 'Odd', 4: 'Even', 5: 'Odd', 6: 'Even', 7: 'Odd', 8: 'Even', 9: 'Odd'}

Dictionary Comprehension (Nested If / elif) Condition

```
List=["Asik","Delwar",'Coco',4,6,2.0,True,(1,2,3,6)]
DIC2={i:("String" if type(i)==str else "Integer" if type(i)==int else "Boolean" if type(i)==bool else
"Float" if type(i)==float else "Tuple" if type(i)==tuple else "Other") for i in List}
print(DIC2)
```

Output:

{'Asik': 'String', 'Delwar': 'String', 'Coco': 'String', 4: 'Integer', 6: 'Integer', 2.0: 'Float', True: 'Boolean', (1, 2, 3, 6): 'Tuple'}

Set Comprehension

Create Set:

```
S={i for i in range(1,9)}
print(S)
```

Output:

{1, 2, 3, 4, 5, 6, 7, 8}

EX2::::::::

```
S2={i**2 for i in range(1,9)}
print(S2)                          #Set Unordered
```

Output:

{64, 1, 4, 36, 9, 16, 49, 25}

EX3:::::::

```
S3={"Apple","Mango","Banana"}
#We need First Char Set
S4={i[0] for i in S3}
print(S4)                           #Set Unordered
```

Output:

{'A', 'M', 'B'}

(Set, Tuple) Other Operation And Looping Same like List And Dictionary

Function *args operation (args all value store into tuple)

```
#To Flexible Function
def Total(*args):           #All Data Make Tuple Like(1,2,3)
    Sum=0
    for i in args:
        Sum+=i
    return Sum
print(Total(2,3,6,5,89,8))
```

Output:

113

*args using with normal parameter

```
def multiply(*args):        #Tuple (2,3,6,5,4)
    #Its Called Perameter
    multi=1
    for i in args:
        multi*=i
    return multi
print(multiply(2,3,6,5,4)) #multiply (2,3,6,5,4)
#Its Called Argument
```

Output:

720

Now with Normal Perameter

```
def multiply(Num1,*args):        #Num1=2 args=Tuple (3,6,5,4)
    #Num1 One perameter *args One Perameter
    multi=1
    for i in args:
        multi*=i
    return multi
print(multiply(2,3,6,5,4))      #multiply Only (3,6,5,4) without 2
```

Output:

360

If No Argument Pass

```
def multiply(Num1,*args):
    multi=1
    for i in args:
        multi*=i
    return multi
print(multiply()) #Error Becouse Num1 must be fillup but *args create empty tuple()
```

Output:

Error

If Normal Perameter after *args

```
def multiply(*args,Num1):
    multi=1
    for i in args:
        multi*=i
    return multi
print(multiply(2,3,6,5,4))
#Its Also Error Becouse all inputed argument going to args tuple so num1 have no argument
```

Ouput:

Error

*args As Argument Unpack (List | Set | Tuple)

```
List=[1,2,3,6,5,4]
def Multiply(*args): #Now Tuple Has One Value List Like ([1,2,3,6,5,4])
    multi=1
    for i in args:
        multi*=i
    return multi
print(Multiply(List)) #Nothing Happening
# So How We Unpack
```

Output:

[1,2,3,6,5,4] Not Working

So How To Unpack The List As Argument And Pass Every value in to *args

```
# To Unpack The List As A Argument
List=[1,2,3,6,5,4]
def Multiply(*args): #Now List Unpacked (1,2,3,6,5,4)
    multi=1
    for i in args:
        multi*=i
    return multi
print(Multiply(*List)) #Use Unpaking (*) *List
```

Output: 720

We Can Also Use Tuple And Set

Example 1

***Crate A Function(Ex. Power) With one Normal Parameter And *args**
***Input normal parameter as a power of *args every value**
***And Store It In a list**
***If Empty List Or Tuple Pass In to *args Print (You Didn't pass Args)**

```
def Power_List(power,*args):
    return [i**power for i in args] if args else "You Didn't pass any Args"
Num=(1,2,3,6,5,4,8,7)
Num2=()
print(Power_List(3,*Num)) #Pass *Num Unpacked tuple
print(Power_List(3,*Num2)) #Pass *Num2 Empty tuple
```

Output:

[1, 8, 27, 216, 125, 64, 512, 343]

You Didn't pass Args

OR::::::::::

```
def Power_List(power,*args):
    if args:
        return[i**power for i in args]
    else:
        return "You didn't passed any value"
Num=(1,2,3,6,5,4,8,7)
Num2=()
print(Power_List(3,*Num)) #Pass *Num Unpacked tuple
print(Power_List(3,*Num2)) #Pass *Num2 Empty tuple
```

Output:

[1, 8, 27, 216, 125, 64, 512, 343]

You Didn't pass Args

**kwargs (Keyword args) (Store As Dictionary)

Only Of Dictionary

```
def DIC(**kwargs):                  #Read A Dictionary And Sotre As Dictionary
    print(type(kwargs))
    for k,v in kwargs.items():
        print(f"{k}:{v}")
DIC(Name="Delwar",Age=25)
```

Output:

<class 'dict'>

Name:Delwar

Age:25

OR:::::::::

```
DIC1={
    "Name":"Delwar",
    "Age":23,
    "Movie":"COCO"
}
def DICT(**kwargs):
    for k,v in kwargs.items():
        print(f"{k}:{v}")
DICT(**DIC1)
```

Output:

Name:Delwar

Age:23

Movie:COCO

We Can Also Use A Normal Parameter Like

```
def DICT(Name,**kwargs):
```

Parameter Order Inside Function

1. Perameter
2. *args
3. Default Perameter
4. **kwargs

At The Same Time Use

```
def order(Normal,*args,default="Unknown",**kwargs):
    print(Normal)
    print(args)
    print(default)
    print(kwargs)
List=[1,2,3,5,4]
DIC={"Name":"Asik","Age":23}
order(3,*List,**DIC)
```

Output:

3

(1, 2, 3, 5, 4)

Unknown

{'Name': 'Asik', 'Age': 23}

Exercise

```
Names=["Delwar","Hossen"]
def Rev_T(name,**kwargs):
    if kwargs["Rev"]==True:
        return [i[::-1].title() for i in name]
    else:
        return [i.title() for i in name]

print(Rev_T(Names,Rev=True))
print(Rev_T(Names,Rev=False))
```

Output:

['Rawled', 'Nessoh']

['Delwar', 'Hossen']

Chapter 10

Lambda Expression

Lambda Expression is a very usefull method. We can use it like function in one line.

Function Like:

```
def add(a,b):
    return a+b
print(add(2,3))
```

But Lambda Expression

```
add=lambda a,b: a+b
print(add(2,3))
```

Output:

5

```
lambda parameter_list: expression
```

<<< lambda Parameter: Return expression >>> And Store It A Variable

To Call This Function read the variable name like <<< add(arguments) >>>

Its anonymous Function lambda have no name….. Only Have <lambda> location

How Small A Function

Bigger:

```
def Stri(a):
    if len(a)>5:
        return True
    else:
        return False
print(Stri("Delwar"))
```

Output: True

Smaller:

```
def Stri(a):
    return len(a)>5
print(Stri("Delwar"))
```

Output: True

Use it in lambda

```
A=lambda a:  len(a)>5
print(A("Delwar"))
```

Output: True

If Else In Lambda

```
A=lambda a:  "Bigger" if len(a)>5 else "Samll"
print(A("Delwar"))
```

Output: Bigger

Nested If else In Lambda

```
A=lambda a:  "Bigger Then 5" if len(a)>5  else "Equal TO 5" if len(a)==5 else "Smaller Then 5"
print(A("Delwar"))
```

Output: Bigger Then 5

Chapter 11

Enumerate Function

Suppose You Have A List Name Is List

Now Print The list with positin like

0--> "Apple"

0--> "Mango"

Simple Way:::::::::

```
L=["apple","Mango","Orenge"]
pos=0
for item in L:
    print(f"{pos}--> {item}")
    pos+=1
```

Output:

0--> apple

1--> Mango

2--> Orenge

USING Enumerate Function::::::::::::

```
L=["apple","Mango","Orenge"]
for pos,item in enumerate(L):
    print(f"{pos}--> {item}")
```

Output:

0--> apple

1--> Mango

2--> Orenge

Find The Position with enumerate() Function

```
L=["Apple","Mango","Orange"]
A="Mango"
def find_pos(l,n):
    for pos,item in enumerate(l):
        if n==item:
            return pos
    return -1
print(find_pos(L,A))
```

Output:

1

Map Function

Crate A List with Other List Items Squares

Map Function

Syntax:::

```
map(Function name without():List Name)
```

Map Passing step by step list item into function

Using Function::::::::

```
List=[1,2,3,4,5,6]
def square(n):
    return n**2
Squares= list(map(square,List))
print(Squares)
```

Output:

[1, 4, 9, 16, 25, 36]

Using Lambda Expression::::

```
List=[1,2,3,4,5,6]
S2=list(map(lambda n: n**2,List))
print(S2)
```

Output:

[1, 4, 9, 16, 25, 36]

Using List Comprehension::::::::::

```
List=[1,2,3,4,5,6]
S3=[item**2 for item in List]
print(S3)
```

Output:

[1, 4, 9, 16, 25, 36]

Using Map Function Into Input

```
LIST=list(map(int,input("Enter List Item , seprate: ").split(",")))
TUPLE=tuple(map(int,input("Enter Tuple Item , seprate: ").split(",")))
print(LIST,TUPLE)
```

Output:

Enter List Item , seprate: 1,2,3,6,5,4

Enter Tuple Item , seprate: 1,2,3,6,4

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

Looping Into Map Function

Count List string item length….

```
Box=["Apple","Mangoos","Orengesss"]
Counted= map(len,Box) #Iterator
print(Counted)           #Its Iterator Not Iterable
```

Output:

Error

If We Covert The Map Into List

```
Box=["Apple","Mangoos","Orengesss"]
Counted= map(len,Box) #Iterator
for i in Counted:
    print(i)
```

Output:

5

7

9

Iterator Only One Time Use It Loop

```
Box=["Apple","Mangoos","Orengesss"]
Counted= map(len,Box) #Iterator
for i in Counted:
    print(i)
for j in Counted:
    print(j)
```

Output:

5

7

9

Here Only print 1st i value 2nd j value Not Printed

Only one time looping

But After Make it list you can looping many times

Filter() Function

```
Filter Function Same To Same Map Function
Suppose We Have A List Now Make A List all even number
def even(A):
    if A%2==0:
        return A
LIST=[1,2,3,6,5,4,9,8,7]
Even= list(filter(even,LIST))
print(Even)
```

Output:

[2, 6, 4, 8]

Deference Between Map() And Filter() Function

```
LIST=[1,2,3,6,5,4,9,8,7]
def even(A):
    if A%2==0:
        return A
Even2= list(map(even,LIST))
print(Even2)
```

Output:

[None, 2, None, 6, None, 4, None, 8, None]

```
#Map Work With Every Item Of List
#Out Put [None, 2, None, 6, None, 4, None, 8, None]
#If Function Return Come append the item into list If not return Append
the item NONE into list
```

Lambda Expression Filter() Function

```
LIST=[1,2,3,6,5,4,9,8,7]
Even3= list(filter(lambda a: a%2==0,LIST))
print(Even3)
```

Output:

[2, 6, 4, 8]

List Comprehension Same Output Without Filter

```
LIST=[1,2,3,6,5,4,9,8,7]
Even4= [i for i in LIST if i%2==0]
print(Even4)
```

Output:

[2, 6, 4, 8]

Looping Into Filter() Function

Filter Same to same Map Both are Iterator

```
#Looping Same Like Map  One Time Without Make It List and tuple
LIST2=[1,2,3,6,5,4,9,8,7]
Even= filter(even,LIST2)
for i in Even:
    print(i)
#But only one time
```

Output:

2

6

4

8

Iterator Vs Iterable

```
LIST=[1,2,3,6,5,4,7,8,9]------------------------->Iterables
Square= map(lambda a: a**2,LIST)----------------->Iterator
```

```
LIST=[1,2,3,6,5,4,7,8,9]
Square= map(lambda a: a**2,LIST)
```

Output:

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

<map object at 0x02E77220>

So How For Loop Work Into Iterator And Iterables
For List As Iterables:::::::

```
LIST=[1,2,3,6,5,4,7,8,9]
for i in LIST:
    print(i,end=" ")
```

Output:

1 2 3 6 5 4 7 8 9

For Map As Iterator::::::::

```
Square= map(lambda a: a**2,LIST)
for i in Square:
    print(i,end=" ")
```

Output:

1 4 9 36 25 16 49 64 81

Who Is Iterator And Iterable

ITERABLE

```
#So How Its Work
# For LIST Iterable
# First Call iter Function
    LIST=[1,2,3,6,5,4,7,8,9] #For Loop List Convert In to iterator by Calling iter() Function
    ITER=iter(LIST) #Now Its Iterator
    #Then Call Next Funcction
    print(next(ITER))
    print(next(ITER))
    print(next(ITER))
    print(next(ITER))
    print(next(ITER))
    print(next(ITER))
    print(next(ITER))
    #We directly not call next() Function Because It is (list,tuple,set)
    #After Call iter() Funtion Then next() Funtion
```

Output:
1
2
3
6
5
4
7

ITERATOR

```
#FOR Square map() filter() function raw data iter() Function data
#Its already Called
#So We Directly Call into next() Function
    Square= map(lambda a: a**2,LIST)
    print(next(Square))
    print(next(Square))
    print(next(Square))
    print(next(Square))
    print(next(Square))
    print(next(Square))
    print(next(Square))
    #If we Call It iter(Square) Its Show Us Error
    #So We Directly Call next() Function
```

Output:

1

4

9

36

25

16

49

Next Topic We Know More About Iterator…….

Exercise

1. Write a Python function to find the Max And Min numbers.

```
def Max_Min(l):
    Max_Num=l[0]
    Min_Num=l[0]
    for item in l:
        if item>Max_Num:
            Max_Num=item
        elif item<Min_Num:
            Min_Num=item
    return Max_Num,Min_Num

List=[]
n=int(input("Enter How Many Number You Want Enter: "))

for item in range(n):
    Num=int(input("Enter Your Number: "))
    List.append(Num)

print(f"Enterd All Number {List}")
Max,Min=Max_Min(List)
print(f"Max Num {Max} And Min Num {Min}")
```

Output:

Enter How Many Number You Want Enter: 2

Enter Your Number: 6

Enter Your Number: 8

Enterd All Number [6, 8]

Max Num 8 And Min Num 6

2. Write a Python function to sum all the numbers in a list.

```
def Sum_List(l):

    sum=0
    for item in l:
        sum+=item
    return sum

List=[]

for item in range(100):
    n=int(input("Enter Your Number If you Want To Stop Pree 00: "))
    if n==00:
        break
    else:
        List.append(n)

print(f"Enterd All Number Is {List}")
print(f"The Sum Is {Sum_List(List)}")
```

Output:

Enter Your Number If you Want To Stop Pree 00: 55

Enter Your Number If you Want To Stop Pree 00: 99

Enter Your Number If you Want To Stop Pree 00: 88

Enter Your Number If you Want To Stop Pree 00: 11

Enter Your Number If you Want To Stop Pree 00: 00

Enterd All Number Is [55, 99, 88, 11]

The Sum Is 253

3. Write a Python function that accepts a string and calculate the number of upper case letters and lower case letters.

```
Text=input("Enter Your Text: ")

def Up_Low(l):
    count1=0
    count2=0
    for item in l:
        if item.isupper():
            count1+=1
        elif item.islower():
            count2+=1
    return count1,count2
Upper,Lower= Up_Low(Text)
print(f"Upper {Upper}\nLower {Lower}")Output:
```

Output:

Enter Your Text: My Name Is aaaaa

Upper 3

Lower 10

4. Write a Python program to print the numbers of a specified list after removing even numbers from it.

```
num = [7,8, 120, 25, 44, 20, 27]

num2 = [x for x in num if x%2!=0]
print(num2)
```

Output:

[7, 25, 27]

5. Shuffle a string and count how many loop its take match oreginal Input

```
import random

Name='Delwar'
Name=list(Name)
#Now String Separated by ['D', 'e', 'l', 'w', 'a', 'r']
for i in range(1000):
    random.shuffle(Name)
    Shuffle_Str="".join(Name)
    if Shuffle_Str=="Delwar":
        print(f"Gotted By {i} Times")
        break
    else:
        pass
```

OutPut:

#Output shows Different Different

6. Write a Python script to add a key to a dictionary

```
DIC={0:1,1:2,10:8}

DIC[5]=5
DIC[9]=8
print(DIC)
```

Output:

{0: 1, 1: 2, 10: 8, 5: 5, 9: 8}

7. Write a Python script to concatenate following dictionaries to create a new one.

```
 dic1={1:10, 2:20}

dic2={3:30, 4:40}
dic3={5:50,6:60}
for item in (dic2,dic3):
    dic1.update(item)
print(dic1)
```

Output: {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

8. Write a Python program to iterate over dictionaries using for loops.

```
D={1:2,2:5,9:5,8:3,6:1}

for keys,value in D.items():
    print(f"{keys}-->{value}")
```

Output:

1-->2

2-->5

9-->5

8-->3

6-->1

9. Write a Python script to print a dictionary where the keys are numbers between 1 and 15 (both included) and the values are square of keys.

```
D={}

for item in range(1,16):
    D[item]=item**2
print(D)
```

Output:

{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100, 11: 121, 12: 144, 13: 169, 14: 196, 15: 225}

10. Write a Python program to sum all the items in a dictionary

```
D={5:9,9:8,6:5,5:2}

print(sum(D.values()))
```

Output:

15

11. Write a Python program to get the maximum and minimum value in a dictionary.

```
D={"X":55,"Y":66,"Z":22}

MAX=max(D.values())
print(MAX)
MIN=min(D.values())
print(MIN)
```

Output:

66

22

12.Write a Python program to check a dictionary is empty or not.

```
my_dict = {}

if bool(my_dict)==False:
    print("Dictionary is empty")
```

Output:

Dictionary is empty

13.Write a Python program to combine two dictionary adding values for common keys.

```
d1 = {'a': 100, 'b': 200, 'c':300}

d2 = {'a': 300, 'b': 200, 'd':400}
for item in d2:
    if item in d1:
        d1[item]=d1[item]+d2[item]
    else:
        d1[item]=d2[item]
print(d1)
# print(d1['a']+d2['d']) For test
```

OutPut:

{'a': 400, 'b': 400, 'c': 300, 'd': 400}