*/ First program to print Hello World.

⇒ print("Hello World")

[save the file → filename.py]

*/ Modules :- A module is a file containing code written by somebody else (usually) which can be imported and used in our programs [import module]

*/ Pip → It is a package manager for python to install a module on our system.

Code	pip install module

[Remember - we write this line of code in terminal].

*/ REPL → Read evaluate print loop.

→ we can REPL to do calculation in our system
→ open powershell → type python → (REPL will get declared) → 5+6 → 11. → exit().
eg [>>> exit()]

*/ Comments → single line #
multi-line '''
line of code
- - -
'''

*/ We can print a multilingual line of code in python using (double code) " ".
for e.g. print(" - - - - -
- - - - ")

- error, for such case we need to use print(''' - - - - -
- - - ''').

* Importing a module to play a music

→ from playsound import playsound
playsound('E:\\ path')

* Write a program to print the contents of a directory using OS module. Search online

→ [function ~~os.rdir~~ os.listdir]

import os
print(os.listdir())

Variables and Data types (chapter 2)

A variable is a name given to memory location in a program.

→ Keywords → Reserved words in python.

→ Identifiers – class / functions / variable name.

code

```
a = "suraj"
b = 34
c = 72.2
```

```
#we can write string in
three ways
"suraj"; '''suraj''';
    '''suraj''':
```

code

```
a = "suraj"
b = 42
c = 3.14
# printing the variable
print(a)
print(b)
print(c)
# type of the variable
print(type(a))
print(type(b))
```

output

```
suraj
42
3.14
<class 'str'>
<class '42'>
```

Operators

1/ Arithmetic operators.
2/ Assignment operators.
3/ Comparison operators.
4/ Logical operators.

Arithmetic operators (+, −, *, /, = etc.)

```
a = 3
b = 4
print("the sum of 3 + 4 : 3+4")
```

→ output - 7

Assignment operators (+=, *=, −=, /=.)

```
a = 34
a += 12
print(a)
```

→ output: 36

(1.11)

Comparison operators ==, >, >=, <, != etc.

Logical operators - and, or, not

AND	
T T	T
T F	F
F T	F
F F	F

OR	
T F	T
F T	T
T T	T
F F	F

Type function and Type Casting

* type function is used to find the data type of a given variable in python.

```
a = 31
print(type(a))  → <class 'int'>
```

* type casting :- It is a function to convert one data type to another :-

str(31) = "31" int to string.

int("31") = 31 string to int.

float(32) = 32.0 Integer to float :

* input function

```
a = input("Enter a number : ")
print(a)
```

remember the number you store will be a string.

Q1/ Write a python program to add two numbers.

```
a = input("Enter a number : ")
a = int(a)        # converting to int
b = input("Enter a number : ")
b = int(b)
print("sum of the two numbers is :", a+b)
```

Q2/ Write a python program to find remainder when a number is divided by 2

```
a = input("Enter a number : ")
a = int(a)
b = input("Enter second number : ")
b = int(b)
print("Remainder dividing", a, "by", b, "is", a%b)
```

4/ Check whether a given variable is greater than 'b' or not using comparison operator (P-5)

ib a = 34, b = 80

⇒

```
a = 34
b = 80
a > b
```

→ output : False

5/ Write a python program to find average of two numbers entered by the user.

⇒

```
a = input("Enter 1st number : ")
a = int(a)
b = input("Enter 2nd number : ")
b = int(b)
c = (a+b)/2
c = int(c)
print("Average is ", c)
```

6/ Write a python program to calculate square of a number.

⇒

```
a = input("Enter the number : ")
a = int(a)
c = a * a
c = int(c)
print("square of number ", c)
```

Chapter - 3 (Strings)

String is a data type in python.

String is a sequence of characters enclosed in quotes.

1.) Single quoted strings → a = 'Harry'.

2) Double quoted strings → b = "Harry".

3) Triple quoted strings → c = '''Harry'''.

String Slicing

→ A string in python can be sliced for getting a part of the string.

Consider the following string

name = "HARRY" ⇒ length = 5.

H	A	R	R	Y
0	1	2	3	4

Code (1)

```
name = "Suraj"
print(name[0])
```

→ output S.

S	U	R	U	J
0	1	2	3	4

Code 2

```
name = "Suraj"
print(name[1:4])
```

→ output URU.

[Here we never consider 4, as the last entity]

a) The index in a string starts from 0 to (length-1)

in python. In order to slice a string, we use the following syntax:

sl = name[ind_start : ind_end]

↙ first index included

↘ last index is not included.

S	U	R	U	J
0	1	2	3	4
(-5)	(-4)	(-3)	(-2)	(-1)

name = "'SURUJ"'

print(name[1:])

print(name[0:])

→ output URUJ

SURUJ

String with skip value

We can provide a skip value as a part of our slice like this:

word = "amazing"

word = [1 : 6 : 2]

0	1	2	3	4	5	6
a	m	a	z	i	n	g

→ mzn

1 2 → 3

3 4 → 5

[Here we are skipping 2.

0 1 → m

2 3 → z

4 5 → n

6 7 → ?

What if we →

name = "amazing"

print(name[0:-3])

0	1	2	3	4	5	6
a	m	a	z	i	n	g

0 → 6 skipping 3

① a → print.

② skip → 3

③ m a 3 → z. print

④ i n g → g print.

→ output . azg.

0 1 2 → a

3 4 5 → n

5 6 7 → ?

What if

print(name[1::3]

mi — output

String functions

Some of the mostly used functions to perform strings manipulations are →

1/ len() function

```
name = "SURUJ"
print(len(name))
```

→ output 5

2/ endswith function

```
name = "AMAZING"
print(name.endswith("G"))
```

→ output True.

3/ count function

```
name = "ALMIGHTY SURUJ"
print(name.count("U"))
```

→ output 2 # it counts the number in the string.

4/ capitalize function

It capitalize the first index of the string.

```
name = "suraj"
print(name.capitalize())
```

output → S.

5/ find function

```
name = "suraj is the one and only GOD"
print(name.find("is"))
```

→ output 6. # is in the six index

6.. String.replace

```
name = "Suraj is the only GOD"
print(name.replace("GOD", "ALMIGHTY"))
```

output ⇒ Suraj is the only ALMIGHTY

*) Escape sequence - characters - comprises of more than one characters - but represents - one character.

Example. \n , \t , \' , \\ etc

- \n → newline
- \t → tab
- \' → single quote
- \\ → backslash.

Practice set on string

1) Write a python program to display a user - entered name followed by Good Afternoon using input() function.

⇒
```
name = "Good Afternoon"
value = input("Enter your name")
c = name + " " + value
print(c)
```

2) Write a program to fill in a letter - template given below - with name and date.

```
letter = '''Dear <NAME>
You are selected!
<DATE>'''
```

⇒
```
letter = '''Dear <NAME>
You are selected
<DATE>
'''
```

```
name = input("Enter your name ")
date = input("Enter your date : ")
letter = letter.replace("<NAME>", name)
letter = letter.replace("<DATE>", date)
print(letter)
```

3) Write a program to detect double spaces in a string & replace double space with single space

```
st = "This is a line with Double spaces"
ds = st.find("  ")
print(ds)
```

→ output : 26

```
st = "This is a line with Double spaces"
st = st.replace("  ", " ")
st = st.replace("Double", single)
print(st).
```

4) Write a program to format a letter using sequence characters.

```
letter = "Dear Suraj, Since you are almighty, bless us. Thanks!"
```

is format

Dear Suraj
since you are Almighty
bless us.
Thanks!

```
letter = "Dear Suraj \n Since you are Almighty! Bless us
\n Thanks \n with regards.
```

print (LL[0]).

Chapter-4 List and Tuples

Python Lists are containers to store a set of values of any data type.

friends = ["Apple", 7, 8.9, False]

List Indexing :- A list can be indexed just like a string

L1 = [7, 9, "Harry"]

L1[0] → 7 , L1[3] = Harry.

L1[1] = 9

(-4)	(-3)	(-2)	(-1)
0	1	2	3

#List slicing

friends = ["Suraj", "Nino", "RAM", "DON"]

print (friends[0:2])

⇒ output → Suraj Nino

List Methods

Consider the following list:

LL = [1, 8, 7, 2, 21, 15]

1 → LL.sort() : update the list to [1, 2, 7, 8, 15, 21]

An important example.

```
LL = [1, 8, 7, 2, 4, 15]
LL_sorted = LL.sort()
print(LL.sorted)
```

output → None

```
L1 = [1, 8, 7, 2, 21, 15]
print(LL.sort())
```

output

[1, 2, 7, 8, 15, 21]

2) → LL.reverse() → It reverse the list

i.e [15, 21, 2, 7, 8, 1]

3. L1.append( )

```
L1 = [1, 5, 7, 6]
L1.append(4)   # add 4 to the end of list.
print(L1)
```

output → [1, 5, 7, 6, 4]

~~4. L1.insert~~

4. L1.insert(3, 8) → this will add 8 at 3 index.

```
L1 = [1, 5, 7, 6]
L1.insert(3, 4)
print(L1)
```

0 1 2 3 4

⇒ output → [1, 5, 7, 4, 6]

add 4 at 3 index

~~5. L1.pop(2)~~

5. L1.pop() → it will delete element at an index and the return the value.

6. L1.remove() → it removes the ~~value~~ data from the table

```
L1 = [1, 2, 3, 4]
L1.remove(2)
print(L1)
```

output [1, 3, 4]

(P-1) Python if ... else → Python support logical conditions from mathematics.

(i) Equal: a == b.

(ii) Not equal a != b.

(iii) Less than: a < b.

(iv) Less than or equal to: a <= b

(v) Greater than: a > b

If statement

(1)

```
a = 33
b = 200
if b > a:
    print("b is greater than a")
```

→ Space, these is called Indentation.

Python relies on Indentation, to define scope in the code. Other programming language often use curly brackets for this purpose.

Elif // It is way of saying, if previous condition is not true, try this condition. //

```
a = 33
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
```

p-2

Else – The else keyword catches anything which isn't caught by the preceding conditions.

Ex:

```
a = 200
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a is equal to b")
else:
    print("a is greater than b")
```

Short Hand

(i) One line if statement

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

(ii) One hand if-else statement

~~if a > b~~

```
a = 300
b = 23
print("a") if a > b else print("B").
```

This technique is known as Ternary operators or conditional statements

(iii) One line if else statement, with 3 conditions

```
a = 330
b = 33
print("a") if a > b else print("=")
if a == b else print("b").
```

And Or

→ The and keyword is a logical and is used combine conditional statements:

→ The OR keyword is also same used to combine conditional statements.

T or F → T
T or T → T
F or T → F
F or F → F

* a = 200
  b = 33
  c = 500
  if a > b or a > c
      print("At least one of the condition are true").

* a = 200
  b = 33
  c = 500
  if a > b and c > a
      print("both condition are true").

Nested If

```
x = 41
if x > 10:
    print("Above ten")
    if x > 20:
        print("above 20")
    else:
        print("but not above 20")
```

The Pass statement

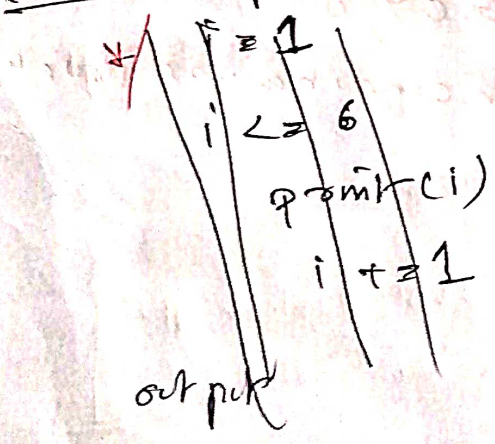
```
a = 33
b = 200

if b > a
    pass.
```

// If statements cannot be empty, but if you for some Reason have an if statement with no content then pass statement will put no error. //.

Python While loop:

- For loop
- while loop.

While loop



Syntax

```
i = initialization
while condition
    print(??)
    i += 1
```

↳ process

Ex-1 : Print i as long as i is less than 6.

```
i = 1
while i < 6
    print(i)
    i += 1.
```

break statement

1/ Exit the loop when i is 3.

⇒

```
i = 1
while i < 6 :
    print(i)
    if i == 3 :
        break
    i += 1.
```

output

1
2
3 ✓

2/ Continue to the ~~iteration~~ next iteration if i is 3.

⇒

```
i = 1
while i <= 6 :
    i += 1
    if i == 3 :
        continue
    print(i).
```

logic

i = 1
for
i = 2
the current value is 3 ✓ which is equal to 3.
hence pair 2.
similarly up to 6.

The else statement

3/ Print a message once the condition is false..

⇒

2.7.6

```
i = 1
while i < 6:
    print(i)
    i += 1
else:
    print("i is no longer smaller than 6")
```

output

1
2
3
4
5
i is no longer smaller than 6.

Python For Loops

A for loop is used for executing/ iterating over a sequence. (that is either a list, tuple, a dictionary, a set, or a string).

Example

```
fruits = ["apple", "orange", "cherry"]
for x in fruits:
    print(x)
```

output

apple
orange
cherry

The break statement

~~for x in fruits~~

```
fruits = ["apple", "banana", "pineapple"]
for x in fruits :
    print(x)
    if x == "banana":
    break
```

output
apple
banana

Ex - Exit the loop when "x" is banana, but this time the break comes before the print.

⇒ ~~fruits~~ fruits = ["apple", "banana", "pineapple"]

```
for x in fruits :
    ~~print(x)~~
    if x == "banana":
        break
    print(x)
```

output → apple

[logic x: apple banana pineapple]

(print) x == banana → break the loop.

The Continue Statement

→ With the continue statement we can stop the current iteration of the loop, and continue with the next :

~~fruits = [~~

Ex:- Do not Print banana in the list.

⇒ ~~for x in~~

fruits = [" apple ", " banana ", " pineapple "]

for x in fruits :

~~print (x)~~

if x == banana :

continue

print (X).

Remember [Just like C++ we use {} for indentation in C++ or Java, we use ~~python~~ space for Iteration]

The Range function

To loop through a set of code a specified number ~~of times~~ of times, we can use the range() function.

The range() function returns a sequence of ~~num~~ numbers, starting from 0 by default and increment by [default 1] 1 and ends at

Ex →

```
for x in range(6)
    print(x)
```

//output
0
1
2
3
4
5 //

// the range() function defaults to 0 starting value, however it is possible to specify the parameters. range(0, 6) // not including 6.

Ex -
```
for x in range(0, 6)
    print(x)
```

output :-
0
1
2
3
4
5

Ex:
```
for x in range(0, 6, 2)
    print(x)
```
→ step: for increment.

output :-
2
4

Else in for loop

The else keyword in a for loop specifies a block of code to be executed when the loop is finished:

Example :- Print all the numbers from 0 to 5 and ends with the message loop have ended.

```
for x in range(6):
    print(x)
else:
    print("loop have ended")
```

```
i = 1
while (i < 6):
    print(i)
    i += 1
else:
    print("loops has ended")
```

1
2
3
4

Note :- Break the loop when x is 3, and see what happens with the else block.

Note :- the else block will NOT be executed if the loop is stopped by a break statement.

Break the loop when x is 3, and see what happens with the else block.

```
for x in range(6):
    if x == 3: break
    print(x)
else:
    print("Finally Finished")
```

Nested loops

A nested loop is a loop inside a loop. The "inner loop" will be executed one time for each iteration of the "outer loop":

Example :-

~~adj = ["red", "banana", "apple"]~~

adj = ["red", "yellow", "tasty"]

fruits = ["apple", "banana", "cherry"]

```
for x in adj:
    for y in fruits:
        print(x, y).
```

output

red apple
red banana
red cherry
yellow apple
yellow banana
yellow cherry
tasty apple
tasty banana
tasty cherry.

Tuples

→ Once defined a tuple elements can't be altered.

→ they are immutable.

a = () → empty tuple

a = (1,) → tuple with only one element needs a comma.

a = (1, 7, 2) ⇒ Tuple with more than one element.

Tuple methods → for more visit python.org

→① a.count(1) : a.count(1) will return number of times 1 occurs in a.

→② a.index(1) : a.index(1) will return the index of first occurence of 1 in a.

Practice set

1) Write a program to store seven fruits in a list entered by the user.

⇒

```
f1 = input("Enter the fruit name 1 : ")
f2 = input("Enter the fruit name 2 : ")
f3 = input("Enter the fruit name 3 : ")
f4 = input("Enter the fruit name 4: ")
f5 = input("Enter the fruit name 5: ")
f6 = input("Enter the fruit name 6 : ")
f7 = input("Enter the fruit name 7 : ")
```

2/ Write a program to accept marks of 6 students and display them in a ~~sorted~~ sorted manner. (P-13)

```
m1 = int(input("Enter the mark of student 1: "))
m2 = int(input("Enter the mark of student 2: "))
m3 = int(input("Enter the mark of student 3: "))
m4 = int(input("Enter the mark of student 4: "))
m5 = int(input("Enter the mark of student 5: "))
m6 = int(input("Enter the mark of student 6: "))
marks = [m1, m2, m3, m4, m5, m6]
marks.sort()
print(marks).
```

3/ Check that a tuple cannot be changed in python.

⇒
```
a = [1, 2, 3, 4]
a[0] = 7
```

→ error

4/ Write a program to sum a list with 4 numbers.

⇒
```
a = [1, 2, 3, 4]
print(a[0] + a[1] + a[2] + a[3])
```

→ output 10

b) Write a program to count the number of zeros in the following tuple.

A = [7, 0, 8, 0, 0, 9]

~~A.count(0)~~

print(A.count(0))

→ output - 2

Dictionary and sets

Dictionary is a collection of key - value pairs

Syntax: a = {"key": "value",
"harry": "code",
"marks": "100"
"list": [1, 2, 9]}

print (a["key"]) → Value

print (a["list"]) = [1, 2, 9].

Code

```
myDict = {
    "Fast": "In a Quick Manner",
    "Harry": "A Coder",
    "Marks": [1, 2, 5],
    "Anotherdict": {'harry': 'Player'}
    }

myDict['Marks'] = [45, 78]

print(myDict['Marks'])  → [45, 78]
```

*/ If we want to know the keys in the Dictionary.

```
print(myDict.keys())
```

↳ It will print the keys.

```
print(myDict.values())
```

↳ It will print the values

code for updating a dictionary

```
myDict = {
    "Fast" : "and furious",
    "high" : "Five".
}

newDict = {
    "Lunio" : "friend"
    "sunny" : "friend"
}

myDict.update(newDict)
print(myDict)
```

```
# get() key.
print(myDict.get("Fast"))
```

↳ output. and Furious

||Imp|| We can use print(myDict["harry"])
then why use get.()?
→ If we are calling a key that doesn't exist in print(myDict["key"]) then it show error whereas if we use get key then it will return none, cos there is no key name in the dictionary.

Sets in Python

Important: This syntax will create an empty dictionary and now an empty set.

```
a = { }
print(type(a))
```
⟶ output <class 'dict'>

Important: An empty set can be created using the below syntax:

```
b = set()
print(type(b))
```

* Set: It is a collection of non-repetable elements.

```
b = set()
b.add(4)
b.add(5)
b.add(5)
b.add(6)
print(b)
```
⟶ output (4, 5)

V.IP → You cannot put list in the set.

i.e

```
b = set()
b = add(4)
b.add([4,5,6])
print(b)
```

⟶ error # as list is mutable, but we can add tuple value in set as they are immutable.

We cannot add dictionary; list in set as they are mutable.

* Ways

```
b = {"key" : "value"}
print(b)
```
⟶ dictionary

```
b = []
print(b)
```
⟶ ~~dict~~ list.

```
b = {( )}
print(b)
```
⟶ tuples.

```
print(len(b))   # print the length of the set.
```

Q1/ Write a program to create a dictionary of Hindi-Words with values as their english translation provide user with an option to look it up.

```
myDict = {
    "Kapre" : "clothes",
    "Vastu" : "Items",
    "Jaal" : "Work."
}
print("choose from the following : ", myDict.keys())
a = input("Enter the name of the Hindi Words ")
print("Meaning is ", myDict[a])
```

2/ Write a program to input eight numbers from the user and display all the unique (set) numbers (once). (18)

⇒
```
num1 = int(input("Enter the number 1 \n"))
num2 = int(input("Enter the number 2 \n"))
.
.
.
num8 = int(input("Enter the number 8 \n"))
s = {num1, num2, ..... num8}
print(s).
```

3/ Can we have a set with 18(int) and 18 (str) as a value in it?

⇒
```
s = {18, "18"}
print(s)
```

Yes it will show, here are int & other is storing.

4/ What will be the length of the following set s
```
s = set()
s.add(20)
s.add(20.0)
s.add("20")
```

⇒ output → (20, '20')

length will be 2.

Coz, the int value 20 & the float value 20.0 are considered to be 1.

Q6/ Create an empty dictionary. Allow 4 friends to enter their favourite language as values and use keys as their names. Assume that the names are unique.

P-19

⇒ farlang = {}

a = input("Enter your fav language Suraj")

b = input("Enter your fav language Arif")

c = input("Enter your fav language Piyush")

d = input("Enter your fav language Harsh")

farlang["Suraj"] = a

farlang["Arif"] = b

farlang["Piyush"] = c

farlang["Harsh"] = d

print(farlang).

output {'Suraj': 'Python', 'Arif': 'C++', 'Piyush': 'C', 'Harsh': 'C'}

Conditionals

Syntax

```
if (condition1):
    print("yes")
elif (condition2):
    print("No")
else:
    print("maybe").
```

Multiple if-else statements

```
if (condition 1):
    print("yes")
if (condition 2):
    print("yes")
if (condition 3):
    print("yes").
if (condition 4):
    print("yes")
else:
    print("No")
```

Write a program to print yes when the age entered by the user is greater than or equal to 18:

```
a = int(input("Enter your age : \n"))
if (a>18):
    print("you are above 18")
else:
    print("you are below 18")
```

Relational operators → used to evaluate conditions

e.g. ==, >=, <= . etc.

Logical operators

Example

and : → true if both operands are true else false

or : → true if atleast one operand is true else false.

not : → inverts true to false and false to true.

3/ is and in

e.g → a = None
if (a is None):
    print("yes")
else:
    print("No")

→ output - yes

e.g → a = [45, 36, 51]
print (45 in a)

→ output - True.

e.g → a = [1, 2, 3]
print (5 in a)

→ output False

4/ Write a program to enter greatest of four numbers entered by user.

```
a = int(input("Enter 1st Number"))
b = int(input("Enter 2nd Number"))
c = int(input("Enter 3rd Number"))
d = int(input("Enter 4th Number"))

if (a > b):
    f1 = a
else:
    f1 = b

if (c > d):
    f2 = c
else
    f2 = d.

if (f1 > f2):
    print("Greatest number is", f1)
else
    print("Greatest number is", f2)
```

92) Write a program to whether a student is pass or fail, if it requires total 40% and atleast 33% to pass any subject taking marks as user input

```
a = input("Enter the student name")
b = int(input("Enter mark obtained in OS"))
c = int(input("Enter the mark obtained in Maths"))
d = int(input("Enter the mark obtained in COA"))
result = (b+c+d)/3
result = int(result)
if(b<33 or c<33 or d<33):
    print("you have failed")
elif(result<40):
    print("you have fail with.", result)
else:
    print(a + " " + "PASS")
```

93) A spam comment is defined as text like "make a lot of money" "buy now" "subscribe this".

```
text = input("Enter your text")
if("make a money" in text):
    spam = True
elif("buy now" in text):
    spam = True
elif("subscribe" in text):
    spam = True
```

```
else:
    spam = False
if (spam):
    print("It is a scam")
else:
    print("Not a scam")
```

Q/ Write a program to find whether a given username contains 10 characters

```
text = input("Enter Your Name")
n = len(text)
if (n>10):
    print("Name contains more than 10 Chas")
else:
    print("Don't contain 10 Chas")
```

Q/ Write a program to know whether a name is in a list or not.

```
a = ["Suraj", "Tanish", "Gokul", "Arib"]
if text = input("Enter a name")
if (text in a):
    print("Yes the name is in the list")
else:
    print("No the name is not in the list")
```

Loops

Loops makes it easy for a programmer to tell the computer, which set of instructions to repeat and how!

Types of loops in Python

1/ while loop. 2/ for loop

e.g → while loop

Print Yes - 10 times

Code

```
i = 0
while i < 10 :
    print ("Yes" + str(i))
    i = i - 1

print("Done")
```

*/ Write a program to print the content of a list using while loops.

⇒

```
Fruits = ["Apple", "Banana", "Mango", "Guava"]
i = 0
while i < len (fruits) :
    print(fruits[i])
    i = i+1
print ("Done")
```

OR. (using For loop)

```
Fruits = ['Banana', 'Watermelon', 'Mango'. ]
for items in Fruits :
    print (item)
```

Range function in Python

The Range function in python is used to generate a sequence of numbers.

We can also specify the start, stop, step size as follows:—

range([start, stop, step-size])

Code (1)

```
for i in range (1, 8):
    print(i)
```

(start = 1, stop = 8)

output
1
2
.
.
8

Code (2)

```
for i in range (1, 8, 2)
    print(i)
```

output
1
3
5
7

For loop with else.

An optional else can be used with a for loop if the code is to be executed when the loop exhausts.

Example

```
l = [1, 7, 8]
for item in l:
    print(item)
else:
    print("Done")
```

→ This is printed when the loop exits.

output
1
7
8
Done.

Break statement — break is used to come out of the loop when encountered. It instructs the program to Exit the loop now.

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

output → 1 2 3 4 5

Continue statement :- It usually skip the iteration if have been assigned.

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

output → 1 2 3 4 6 7 8 9 10

[If there is no 5]

Pass statement

pass is a null statement in python
It instruct to "Do Nothing"

Practice set

Q1/ Write a program to print multiplication table of a given number using for loop.

```
n = int(input("Enter the number \n"))
for i in range(0):
    while i <= 10:
        print(str(n) + " x " + str(i) + " = " + str(i * num))
```

Using For loop

```
num = int(input("Enter your number"))
for i in range(1, 11):
    print(str(num) + "x" + str(i) + "=" +
    str(i * num)).
```

~~print~~

We can write these lines of string.

```
print(f"{num} x {i} = {num * i}")
```

Write a program to greet all the ~~process~~ ~~write a~~ person names stored in a list L1 and which starts with 'S'.

```
L1 = ["Harry", "Sohan", "Sachin",
      "Rahul"]
```

```
L1 = ["Harry", "Sachin", "Sohan", "Rahul"]

for string in L1:
    if string.startswith("S"):
        print("Hello" + " " + string)
```