Python

Page No.
Date

→ What is python?
- Python is a general purpose and Powerful Programming language.
- Python is Considered as one of the most versatile programming language as it can be used to develop almost any kind of application including.
  → desktop Applications.
  → Web App.
  → IoT App.
  → AI, ML and Data Science app.
  and many more...

→ Who Created python?
- Developed by Guido van Rossum, a Dutch Scientist.
- Created at Centre for Mathematics and Research, Netherland.
- It is inspired by another programming language called ABC.

→ Why was python Created?
- Guido started python development as a hobby in 1989.
- But since then it has grown to become one of most polished languages of the Computing World.

→ How Python got it's name?
- The name Python is inspired from Guido's favourite Comedy TV Show Called "Monty Python's Flying Circus.
- Guido Wanted a name that was short, Unique and Slightly mysterious, so he decided to call the language Python.

Page No. | Date

→ Who manages python today?
- From version 2.1 onwords, Python is managed by python Software foundation situated in Delaware, USA.
- It is a non-profit organization devoted to the growth and enhancement of python language.
- Their website is : http://www.Python.org.

→ Famous websites Developed using python :
→ NASA → Instagram → udemy → spotify
→ Mozilla → Dropbox and above all YouTube.
- There are many excellent python frameworks like Django, Flask for web application development.

→ Why should I learn python?
- 3rd should Popular Programming.
- Fastest growing language.
- Opens lots of doors.
- Big Corporates prefer python.
- Means, PYTHON IS THE FUTURE.

→ Features of python :
i) Simple
ii) Dynamically Typed
iii) Robust
iv) Supports multiple Programming paradigms.
v) Compiled as well as Interpreted.
vi) Cross Platform.
vii) extensible.
viii) Huge library.

Page No.
Date

⇒ Multiparadigm

```
def add (a,b);
    return a+b
    result = add(3,5)
    Print (result)
```

⇒ Script Model/Programming

```
for number in range (1,6);
    Print(number)
```

O/P

```
1
2
3
4
5
```

⇒ Tokens in Python
- i) Keywords
- ii) identifiers (Names)
- iii) Literals
- iv) Operators
- v) punctuators

- These are also Known as reserve words.
- These are the words whose meaning is predefined by interpreter and Cannote be used like general purpose words.

Page No.
Date

- This are basic building blocks ~~of~~ a program used to Construct instructions.
- Keywords Cannot be used as variable name ~~of~~ or function name or as a identifier.
- 36 Keyword in 3.12

- import Keyword
  Print (Keyword.Kwlist)
  ['False', 'None', 'True', 'and', 'as', 'if', 'for', 'try', 'while', 'else' etc.]

⇒ Identifiers
- User defined name
- Sequence of letters and digits.
- The First character must be a letter or _.
- Upper & lower-case letters are different.
- 0 to 9 Can't be the First letter of identifier.
- Must not be a Keyword.
- Cannot Contain any Special character except for _.
  MyFile ✓ My-File ✗

⇒ Literals (-Constant values)
- String
- Numeric
- Boolean
- Special Literal - None (value = None #None literal)

⇒ Key Points about python variables:
- Variable Declaration
- Naming Rules
- Reassing variables
- Dynamic Typing
- Multiple Assignment

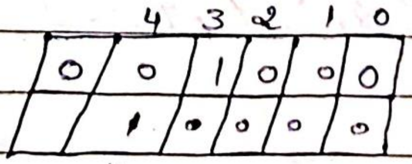
⇒ Operators
- Unary oper.
- Binary oper.
    - Arithmetic oper.
    - Bitwise oper.
    - Shift oper.
    - Identity oper.
    - Relational oper.
    - Logical oper.
    - Assignment oper.
    - Membership oper.

Shift Operators

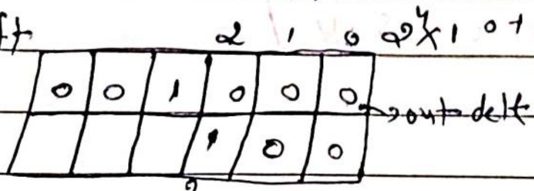
<< Shif left

Shift Right >>

Left Shift Example : (8)



Right Shift



$2^4 \times 1 + 0 + 0 + 0 + 0 = 16$

$2^2 \times 1 + 0 + 0$

$= 4$

index Num.

$2^4 \times 1$

binary Base

Binary digit

Page No.
Date

Logical op.

Both condition TRUE

Logical AND (Both Condition chock

Logical OR (one condition chock

when one condition True then ok and Second False / TRue OK. (Min. One condition TRUE

Assignment op.

= Assignment

Ex: a = 5

$\uparrow$ assignment

*= 5+= 2 a= 7

%= a+ =b / a= a+b

-= Value ~~e~~assign left side varible automatic.

** =

// =

Membership op.

in Whether varible in Sequence

not in Whether not in

2 in [2, 5, 6, 7, 8]

TRue

5 in [2, 5, 7, 8, 9]

TRue

6 not in [7, 8, 6, 5, 4]

False

Page No.
Date

Identity operator

is   is the identity Same?
isnot   is the identity not same?
i) a is b
= False
ii) b is b
True

→ for output
ex: name = "Sahil"
age = 20
Print("my Name is {name} Age is {age}.")
output my name is Sahil Age is 20.

Ex: a = 2;
b = 3;
Sum = a+b;
Print (F"Sum of two number is: {Sum}")

O/P Sum of two number is: 5

→ for input
i) name = input("enter your name:")
Print (name)

output enter your name: Sahil
Sahil

Page No.
Date

ii) num =

iii) Print ('what is your name:')
name = input ()
Print("hello ", name)

<u>output</u>
```
what is your name:
Sahil Kumawat
hello Sahil Kumawat
```

iv) Print('Sahil' , 'Kumawat')
↑
Space

<u>output</u> Sahil Kumawat

v) Print('Sahil', Kumawat', end = '##')
Print ('by', 'Sahil', Sep = '@')
Print ('with sep', 'and end', Sep = '', end = '\n')

<u>output:</u>
```
Sahil Kumawat##by@Sahil
with sepand end
```

Print ('1', '+', '2', '=', '3') // 1+2=3
Print(1+2, 3/2) // 3 1.5
Print ("a", "b", "c", "d", Sep = ",") // a,b, c,d
Print ("a" "b" "c" "d", Sep = ",") // abcd
↳ Sep don't work because ,(Coumo) is not avelible in code.

Page No.
Date

→ Data type and operators:

- 16 / 52
  // 0.3076923076923077

- Print ('%.f '%.(16/52))
  // 0.307692 (Show six digit /Point Value.)

- Print ("%.2f" %(16/52))
  // 0.31

Real Imaginary

- (3+3j) + (2+2j)
  // (5+5j)

- x = '''Sahil ← Three Time Semicolon use for multi line
  Kumawat'''
  Print (x) // Sahil
  Kumowat

- x = 'one line'
  Print (x, x) // one line one line

- x = 'one line'
  Print (x+x) // one Lineone Line

- x = 'one line'
  Print ("my answer is", 3 * "No!") // my answer is No! No! No!

- n = int (input ())
  Print (n * "Python") // 3
  PythonPythonPython

Page No.
Date

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
- name = "Python Programming"
Print(name [0], name [7], len (name)) // P P 18

This is unmutable → ~~ython[x]~~ not valid
- name [0] = "c" (in Python not work Replacement)

(Reverse)
- Print (name [-1], name [-11], name [-len(name)])
// g P P

→ Slice   starting value, ending but skip it.

First = name [1:5]
Second = name [7: len (name)]
Print(First, Second, Sep=", ")

// ython, Programming

- First = name [:6]
Second = name [7:]
Print (First, Second, Sep = " ")
name = Second + " " + First;
Print (name)
// Python Programming
Programming Python

- First = name [:6]
Second = name [7:]
Print (Frist, Second, Sep = " ")
name = Second + " " + First;
Print (name)
// Python Programming
Programming Python

Page No.
Date

Starting end but skip step count

- Print (name [1 : : 2]) // Yhnpormig
  name [4 : -3] // on Programm
  name [-3 : -4] // No Answer print because not count value
  name [3 : -1] // hon Programmin
  name [-3 : -1] // in
  name [-1 : 3] // No Ans.

→ Complex Num

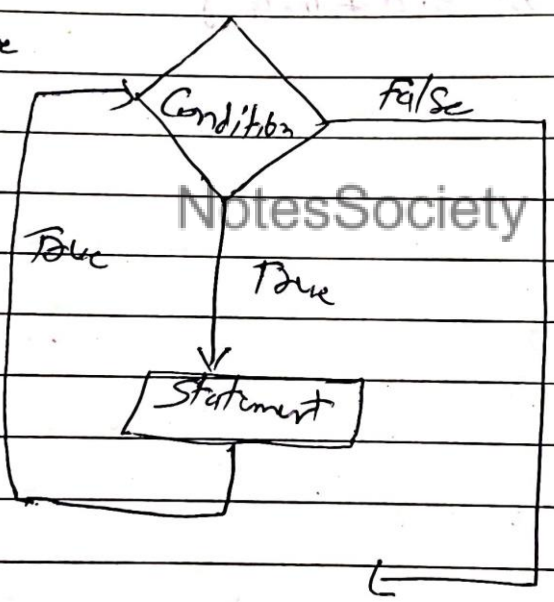
- Z = 3 + 4j
  Print (Z.real, '+', Z.imag, 'j')
  // 3.0 + 4.0 j

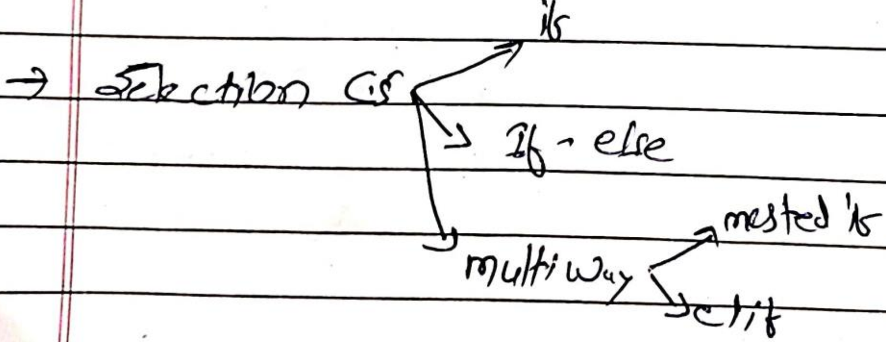
Control Structures

- Sequential C.S
- Selection C.S (If else)
- Iterative C.S

```
Print ("Hellow world")
Print ("This is our World")
name = Input ();
Print ("I m Kaib");
Print ("Thank you")
```

Iterative





Page No.
Date

```
if <condition>:
        <Statement(s)>
        else:
              <Statement()>
```

→ Indition is very important.

```
n = int(input("Enter a num"))
if n< =10:
    Print ("value is:", n)
    Print ("Thankyou")
```

multiway:

elif

```
    if (n<=10):
        Print ("Grade A")  <Condition one is Print becouse
    elif (n<=20):                Starting Top to bottom where
          point ("Grade B")      find condition Stop there.
    elif (n<=30):
          print("Grade C")
    else:
          Print("You have not Cleared the exam.")
```

→ Even odd

```
n = int(input("Enter a num"))
  if ~~n~~ (n % 2 ==0)
        Print ("Number is even");
  else
        Print ("Number is odd").
```

Page No.
Date

```
→ if (n <= 40):
    Print("Fail")
  elif (n <= 50):
    Print("Pass")
  elif (n <= 60):
    Print("First div")
  elif (n <= 80):
    Print("good!")
  else:
    Print(" Enter value 1 to 80")
```

⇒ nested if

```
College = "PCU"          { if in if is Nested if
Course = "MCA"

if College == "PCU":
    if Course == "MCA":
        print("He/She is MCA Student")
```

Looping

```
n = 1
→ While n < 10:        { Print 1 to 9 but you can Print
    Print(n)             10 out of Program but not Print
    n = n+1              bcas condition n < 10.
```

~~ppr~~

output

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

Program) very important

```
While m<10:
    Print(n)
    n=n+1
    if m==5:
      break
else:
    print ("n is ~~greater~~ equal than 10")
```

{You can use in python while loop if else statement.

output

```
1 2 3 4
```

→ for loop

```
    L = [2, 5, 7, 8]
    for num in L:
        Print(num)
      // print("Hello python")
```

for <var> in <Seq.>

→ range ( )

```
    for n in range (1, 11)
        print (n)
```

{you can use (2, 101, 2) Slicing according run/Step 50 times.

Table of 2.

```
→ for n in range (2, 11, 2)
      print (n)
```

→ Print table any num.

```
num = int(input ("Enter a number:"))
for ~~num~~ in (1, 11)
T = num x n
  print (T)
```

Page No.
Date

⇒ User ~~&~~ giving value Print Table.

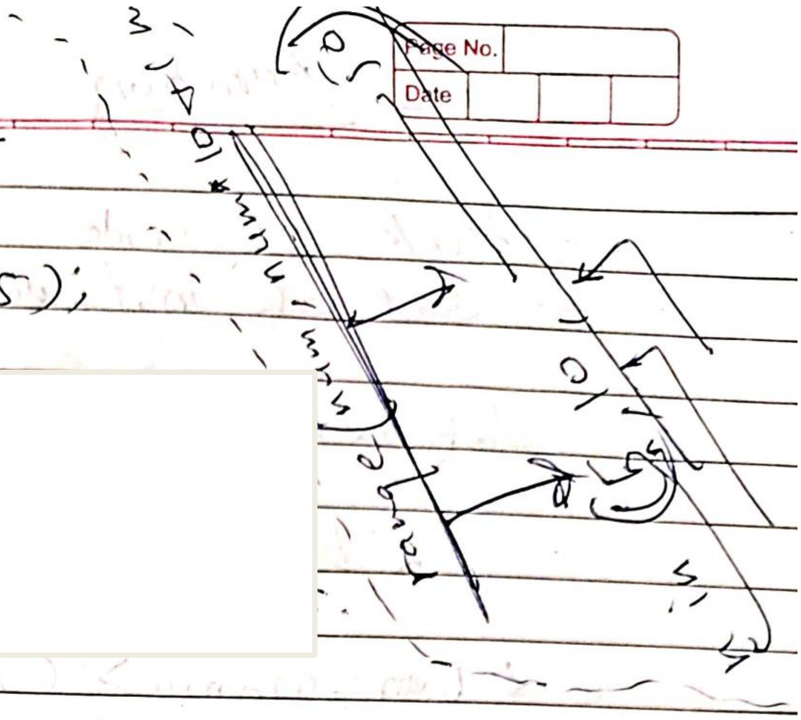
```
→ no = int(input("enter no to get table"))
  ran = int(input("enter a num"))

  for n in range(1, ran+1)          (1, no*n, ran+1)
                                    (1, no* = n, ran+1)
     table = no*n
     print(table)
           print(f" {no} x {n} : {table}")
```

5
+
1, no * 10,

9-10-24

```
for n in range(5):          | output
                            | 1
   # if n==3:               | 2
      print(n)              | 3
                            | 4
```

```
for n in range(5):          | output
                            | 0
      if n==3               | 1
          break:            | 2
      print(n)              | Thank you
   print("Thank you")
```



```
=> for n in range (5);
       if n = = 3
       Continue
       Print(n)
```

output

```
0
1
2
4
```

```
-> for n in range(5):
       if n = = 3
          Pass
          Point (n)
```

output

```
0
1
2
3
4
```

```
-> i=0
   Sum=0
   While i<10
   n=input ( )
      sum = Sum + n
          if n<0
             break
             i =i+1
   else print (" Successfully run")
```

Page No.
Date

Function

-> block of Code
-> Set of instruction

```
def <fun_name> ():

    <body>

<fun_name> ()
```

=> Two type
i) User define
ii) bultin fun / Pridef.

```
-> def addition ():  (a,b)       //def add (a+b)    Positiona
       Sum = a+b                                   Paramete
         Print(Sum)
   addition ()                    add(2, 4)
```

=> Four Parameters :

-) Positional parameter

2) Default "
3) Keyword "
4.) Arbitrary "

Page No.
Date

Default Par.

```
                     Default
def disp(name, age=22)
    Print(f"my Name is {Name} and age is {age}")
    disp("Sahil", 23) {current time Print 23 but age not
                      give So Default value automatic
                      bach age 22.

def add.(a=5, b, c=6)
    sum = a+b+c

    (a, b=5, c=6) ✓    { We can give First value
    (a=5, b=5, c) X      So give Error.
    (a, b, c=6) ✓
    (a=5, b, c) X

• def add (a=5, b=6): / (a=5, b) {Show Error
    sum = a+b
      Print(sum)
    add (7, 8)
      add (7)
      add()
```

Keyword:

```
        def disp(name, age):
    Print(f"my name is {name} and age is {age}").

disp(age=22, name='Sahil')
```

Page No.
Date

Arbitrary

~~Fit~~ (**args)

```
def addi(*args)   { we can use when we don
for i in args:    { Know range of value.

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

* Sum two num:

```
def add(a,b):
    return a+b
result = add (2, 3)
Print(result)
```

16-10

Page No.
Date

Anonymous function :

Lambda :

i) Lambda functions have no Name.
ii) Lambda can take any number arguments.
iii) " Can return Just one value in the form of an Expression
iv) lambda function def. doesn't have an explicit return, but it can always contain expression.
v) " are one-line version of a function and hance can't multiple expression.
vi) They can't access varible other then.
vii) " Can't ~~g~~ access global varible.

Syntax

$\boxed{\text{lambda arguments : expression}}$

```
// lambda x,y : x+y
// lambda : print("Hello Python")

// add (x,y,z):
       s = x+y+z;
   return s

// res = lambda x,y : x*y
   res(5,3) -> This method only for lambda fun.

Print ("using lambda")
L = [lambda x: x**2, lambda x: x**3, lambda x: x**4]
  Print ("Lambda list:")
   For F in L
    Print(F(2))
```

```
| o/p
| 4
| 8
| 16
```

Page No. | Date

fun. (using def)

```
Print("using def:")

  def f1(x):
        return x**2
  def f2 (x):
        return x**3
  def f3 (x)'
        return x**4
     l = [f1, f2, f3]
    for f in l
        Print ( f(2))
```

```
using def:
4
8
16
```

Lambda with map()

The map() fun. is a built-in fun. in Python that applies a given fun. to all items in an input iterable.

- returns a map object, which is an iterator.

```
// num = [1, 2,3,4]
  def Sqr (num):
      return num**2

l num = [1, 2,3,4]
Sqr_res = map(Sqr,num)
Print(list(sqr_res))

//  numbers = [1, 2, 3,4]
    Squares = list (map (lambda num : num ** 2, numbers)
      Print (Squares)
```

output

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

Page No.
Date

```
// list_num = [1, 2, 3, 4]
list_num = map(lambda x: x*10, list num)
  For num in list_num:
      Print(num, end=" ")        /output
                                  10 20 30 40
```

```
// list_num=[1,2,3,4]
list_num = map(lambda X: X*10, list_num)
print(list_num)          output
                   // <map object at 0x105313dc0>
```

```
// list_numbers = [1,2,3,4]
list_numbers = map(lambda X: X*10, list_numbers)
  # L=List(List_numbers)
  # Print (L)
   print (next(list_numbers))
   print(next(list_numbers))
                              /output
                                10
                                20
```

```
   Define two num lists
//  num1=[1,2,3,4]
    num2= [10,20, 30,40]
       # map with lambda fun add.
  result = map (lambda x, y: x+y, num1, num2)
  print (list(result)) # Convert map obj to List
                            output
                          [11, 22, 33,44]
```

Page No.
Date

filter(fun , iterable)

```
Lnum = [1,2,3,4]
even_n =list(filter(lambda x : x%2 ==0 ,Lnum))
  print(even_n)
```

/output
[2,4]

Unit III

24-10

Regular Expression (Reg Ex , RE)

Page No.
Date

-> import re
-> Specifies a Pattern

* matching Characters

```
import re
pattern = 'N....a'
test_string = 'Namita is my name!'
result = re.Match(Pattern, test_string)
Print (result)
if result:
    Print('Search successful')
else:
    Print('Search Unsuccessful')
```

(Namita starting hona chahiye / Check begning)

O/P
Search Succesful

-> method

Match()	-> Determine if the RE matches at the beginning of str
Search()	-> looking of any location where this RE matches.
Findall()	-> RE matches, and returns them as a list.
Finditer()	

findall > Search() > Match()

-> [abc] [cb] -> Mach
-> [a-e] is the Same as [abcde] (- to mean a to e Print)
-> [0-39] is the Same as [1239]

```
import re
Pattern = '[0-9]'
Str = "Hi my Passwords are 123 and 5abc"
```

Page No.
Date

^ - Caret

→ String Ke Starting me Hi search Krega.

→ $ - Dollar

→ String Ke last word Ko Search करने ~~[illegible]~~ Ke liye.