

Python

=====

=>Python is general purpose high level programming.

=>Python was developed Guido Van Rossan in 1989 while working at National Research institute at Netherlands.

=>It was made available to public in 27 Feb 1991.

=>The name python was selected from a comedy Show "The Monty Python Flying circus" which was broadcasted at BBC channel in 1969 to 1974.

Top Applications based on Python

=====

Google Search Engine

gmail

yahoo search engine

nasa

www.youtube.com

www.instagram.com

etc.

Where We can use Python ?

=====

1. For developing Web Applications
2. GUI Application (Desktop Application .exe)
3. Console Application
4. Mobile Application
5. ML Application
6. Database Application

7. AI Applications

etc.

=====

Features of Python.

=====

1. Simple and Easy to learn
2. Interpreted
3. freeware and open source

NGO PSF(Python Software Foundation)

www.python.org

=

4. Platform Independent
 5. Extensible
 6. Embeddable
 7. Dynamically Typed
 8. Object Oriented.
- etc.

=====

1. Statically Typed || strongly Typed

Java C# C C++

int a=10

float b=10.5

string c="Techpile"

```
boolean d=true
```

2. Dynamically Typed || Loosly Typed

```
a=10
```

```
b=30.5
```

```
c="RAM"
```

```
d=true
```

Command to check version of Python

```
=====
```

```
python --version
```

Basic Command

```
=====
```

```
mkdir directoryName
```

```
mkdir Python Batch2
```

```
print()
```

```
=====
```

print function is used to display output on command prompt.

Syntax

```
=====
```

```
print("Message")
```

```
print("Welcome to The world of Python")
```

identifier

=====

Any name in python program is known as identifier. it can be variable name or function name or class name or modulename.

Rule to define identifier in Python

=====

1. Python variable should not start with digit.
2. Python variables are case_sensitive.
3. We can not define keyword as identifier
4. We can define identifier with alphabets(A-Z or a-z) or Alpha-Numeric

123Name

123

for

if

True

Name

Ca\$h

Cash

name_abc

name@abc

_name //private

Keyword

variable

=====

=>variable is a container to store data value.

=>python variables are dynamically typed.

```
x=10
```

```
y=30
```

```
z="RAM"
```

```
a=True
```

Datatype

=====

=>datatype is used to represent type of data present inside variable.

=>In python we are not required to define data type explicitly. based on provided value datatype will be assigned automatically hence python is dynamically typed language.

1. Fundamental Datatype

=====

1. int => to represent integer(whole number 10,30,40,100) type value.

```
x=20
```

```
y=30
```

```
type()
```

=====

type function is used to check type of data present inside variable.

```
x=20
```

```
y=30
```

```
z=30.5
```

```
print(type(x))#int
print(type(y))#int
print(type(z))#float
```

2. float => to represent floating type(decimal value 50.3344,10.44344) .

```
x=10.5
y=10.433
print(type(x))#float
print(type(y))float
```

3. str

=====

=>str datatype is used to represent string type value.

string

=====

string is a collection of character enclosed with single or double.

'RAM'

"RAM"

"10"

'wedwehf23284'

'30'

"True"

=====

String slicing

=====

```
a="TECHPILE"
```

```
print(a)#TECHPILE
```

```
print(a[0])#T
```

```
print(a[5])#I
```

```
print(a[-4])#P
```

```
print(a[-2])#
```

```
str[startIndex:endIndex]
```

```
a="TECHPILE"
```

```
print(a[1:4])#ECH
```

```
print(a[5:7])#IL
```

```
print(a[1:7])#ECHIPIL
```

```
print(a[2:5])#CHP
```

```
print(a[-8:-2])#-2-(-8)=>TECHPI
```

```
print(a[-5:])#-8 to last =>HPILE
```

```
print(a[:4])#from 0 to 4th index TECH
```

```
print(a[:])#from 0 to last TECHPILE
```

4. bool

=====

bool datatype is used to represent boolean type(True or False 1 or 0).

```
x=True
```

```
y=False
```

```
print(type(x))#bool
```

```
print(type(y))#bool
```

Advance Datatype

```
=====
```

1. list

2. set

3. range

4. tuple

5. dict

```
=====
```

```
a=["RAM",10,10.5,True]
```

```
for i in a:
```

```
    print(i)
```

```
=====
```

```
input
```

```
=====
```

input function is used to take input by user on command propmt.

=>Return type of input function is string.

Syntax

```
=====
```

```
x=input("message")
```

```
x=input("Enter Your Name :")#40
```

```
#print(type(x))# str
```

```
print("Name : ",x)
```


Type Casting || Type Conversion

=====

=>The process of converting one data type value to another data type is known as type casting or type conversion.

1. int()

```
print(type(10.5667))#float
```

```
print(int(10.5667))#10
```

```
x=23.56
```

```
print(type(x))#float
```

```
x=int(x)
```

```
print(type(x))#int
```

2. float()

=====

```
print(float(10))#10.0
```

```
print(float("10.5"))#10.5
```

```
print(float(True))#1.0
```

```
print(float(False))#0.0
```

3. bool()

=====

```
print(bool(0))#False
```

```
print(bool(0.0))#False
```

```
print(bool(""))#False
```

```
print(bool("Ram"))#True
```

```
print(bool("0"))#True
```

```
print(bool("100"))#True
```

```
print(bool("10.5"))#True
```

Operator

=====

=>operator is symbol to perform operation over the operands.

Type of Operator

=====

1. Arithmetic Operator

+, -,

*

20+30=50

40-30=10

3*2=6

/ Normal Division

50/2 =25.0

100.0/3=33.333

100/50 =2.0

100//3 =

100/3 =33.333

50//2.0 =25.0

30/3.0 =10.0

600/2 =300.0

600//2 =300

$600.0/2.0 = 300.0$

$1000/2 = 500.0$

$3*3 = 9$

$3**3 = 3*3*3 = 27$

$2**4 = 2*2*2*2 = 16$

$10\%3 = 1$

// Floor Division

** exponent Operator || power Operator

% Modulo Operator

=====

2. Relational Operator

<

>

<=

>=

`print(50<30)#False`

`print(50>30)#True`

`print(50>=30)#True`

`print(50<=2)#False`

`print(2<=2)#True`

`print(True<=True)#True`

`print(True>=False)#True`

`print(True<=False)#False`

`print(False<=False)#True`

```
print(False>=False)#True
```

```
print(False<False)#False
```

```
print(True<False)#False
```

=====

3. Logical Operator

and =>it returns True if both conditions(argument) are True otherwise it return False.

True and True => True

False and True =>False

True and False => False

False and False => False

```
print(20>10 and 30<50)#True
```

```
print(10<=100 and 30>=30)#True
```

```
print(10>=100 and 30>=30)#False
```

```
print(True>=True and 30>=30)#True
```

```
print(True>=False and False>=True)#False
```

```
print(False>True and True<False)#False
```

=====

or =>It returns True of at least one argument is True otherwise it returns False.

True or True # True

True or False # True

False or True # True

False or False # False

print(20>10 or 30<50)#True

print(10<=100 or 30>=30)#True

print(10>=100 or 30>=30)#True

print(True>=True or 30>=30)#True

print(True>=False or False>=True)#True

print(False>True or True<False)#False

not True =>False

not False => True

print(not True>True)#True

print(not True>=True)#False

print(not 50>=60)#True

=====

4. Assignment Operator

=>Assignment operator is used to assign value to the variable.

=

x=10

x+=20 #x=x+20 => x=10+20

print(x)

+=

-=

a=1000

a-=500#a=a-500

```
print(a)#500
```

```
/=
```

```
a1=1000
```

```
a1/=500# a1=a1/500 1000/500=>2.0
```

```
print(a1)#2.0
```

```
a2=10
```

```
a2/=3 #a2/3 10/3 =>3.33 a2=a2/3
```

```
print(a2)#3.33
```

```
a3=10.0
```

```
a3//=3.0 # a3=a3//3.0 10.0//3.0 =>3.0
```

```
print(a3)
```

```
//=
```

```
*=
```

```
a4=50
```

```
a4*=2 # a4=a4*2
```

```
print(a4)#100
```

```
**=
```

```
a5=2
```

```
a5**=3 # a5=a5*a5*a5
```

```
print(a5)#8
```

```
a6=10
```

```
a6%=3#a6=a6%3
```

```
print(a6)#1
```

%=

=====

5. Membership Operator

Membership operator is used to check given object present in main object or not.

in =>It returns True if given object present in main object otherwise it returns False.

not in =>It returns True if given object is not present in main object otherwise it returns False.

```
str="Techpile Technology Pvt Ltd."
```

```
x="Techno"
```

```
print(x in str)#True
```

```
print("PVT" in str)#False
```

```
print("PVT" not in str)#True
```

```
print("Ltd." not in str)#False
```

=====

6. Ternary Operator / Conditional operator

Syntax

=====

first value if condition else second value

if condition is True then first value will be considered otherwise second value will be considered.

```
print("Welcome to Techpile Old Building") if(True>False) else print("Welcome to Techpile new Building")
```

=====

7. equality operator

==

!=

```
print(100==100)#True
```

```
print(1000==100)#False
```

```
print(100!=100)#False
```

```
print(100!=10)#True
```

=====

Flow control

=====

=>flow controls describes order in which statement will be executed.

1. Conditional Statement

(a). if

=====

Java ,C++ c,C# JS ,React

```
if(condition)
```

```
{
```

```
statement
```

```
statement
```

```
statement
```

```
}
```


Syntax

=====

if condition:statement

or

if condition:

statement

statement

or

if(condition):

statement

statement

statement

if(10>30):

print("Welcome To techpile")

print("Out of if")

(b). if-else:

=====

Syntax

=====

if(condition):

Action1

else:

Action2

if condition is True then Action1 will be executed otherwise Action2 will be executed.

```
name=input("Enter your Name :")#ROHAN
```

```
if(name=="ROHAN"):
```

```
    print("***30)
```

```
    print("Hello Rohan Good After noon")
```

```
else:
```

```
    print("***30)
```

```
    print("Hello Guest GN...")
```

```
print("out of if")
```

(c). if-elif-else

```
if(condition1):
```

```
    Action1
```

```
elif(condition2):
```

```
    Action2
```

```
elif(condition3):
```

```
    Action3
```

```
...
```

```
elif(conditionN):
```

```
    ActionN
```

```
else:
```

```
    elseAction
```

=>based on condition corresponding Action will be executed.

```

name=input("Enter your Name :")#Rohit
if(name=="Rohan"):
    print("#"*30)
    print("Hello Rohan")
    print("#"*30)
elif(name=="Mohan"):
    print("#"*30)
    print("Hello Mohan")
    print("#"*30)
elif(name=="Rohit"):
    print("#"*30)
    print("Hello Rohit")
    print("#"*30)
else:
    print("Please Enter Valid Name...")

```

```

print("Out of if")

```

2.Iterative /Looping Statement

```

=====

```

if we want to execute a group of statement multiple times then we have to use iterative/looping statement.

(a). for

```

=====

```

if we want to execute a group of statement for every element present in given collection(string,list,tuple,set,dict,range) then we have to use for loop in python.

Syntax

=====

for variableName in collection:

 statement

 statement

 statement

str="RAM"

for i in str:

 print("ABC")

=====

range

=====

range datatype is used generate sequence of number.

Form1

=====

range(n)#it generate sequence of number from 0 to n-1.

range(10)#0,1,2,3,4,5,6,7,8,9

Form 2

=====

range(x,y)#it generate sequence of number form x to y-1.

Exa:

range(5,9)#5,6,7,8

Form 3

=====

`range(x,y,z)`

it generate sequence of number from x to y-1 incremented by z

Exa:

`range(10,18,3)#10,13,16,`

Form 4

=====

`range(x,y,-z)`

it generate sequence of number from x to y+1 decremented by z

`range(10,5,-2)#10,8,6`

`range(20,5,-3)# 20 to 6 =>20 17 14 11 8`

while

=====

if we want to execute a group of statement until condition False then we have to use wwhile loop.

Syntax

=====

while condition:

 statement

Infinite Loop

```
while True:
```

```
    print("RAM")
```

```
i=1
```

```
while i<=3:
```

```
    print("RAM",end="\t")
```

```
=====
```

Transfer Control Statement

(a). break

```
=====
```

if we want to break execute of loop based on some condition then we have use break statement.

Exa

```
=====
```

```
i=1
```

```
while i<=10:
```

```
    if i==5:
```

```
        print("Hello I am breaking the loop")
```

```
        break
```

```
    print(i)
```

```
    i+=1
```

```
l=[10,20,40,60,3,0,50,60,2]
```

```
num=int(input("Enter Your Number : "))
```

```
for i in l:
```

```
    if i==0:
```

```
        print("I am breaking the loop")
```

```
break

res=num/i

print(res)
```

(b). continue

=====

if we want to skip current iteration and continue for next iteration based on some condition in loop then we have to use continue statement.

```
for i in range(2,11,2):

    if i==8:

        continue

    print(i)#
```

(c). pass

=====

if we want to create empty function or class then we have to use pass statement.

```
def functionName()
```

```
functionName()# error
```

```
def demo()

    pass
```

```
demo()
```

```
class Test:
```

```
    pass
```

=====

Module

=====

Module is collection of Variable function and classes.

=>Every .py extension files are treated like module.

module1.py

variable, function, classes=> member of module

How to import a module

=====

import keyword is used to import module.

import moduleName

import module1,module2,module3,...ModuleN

How to access member from a module

1. modulename

2. moduleName.memberName

import p42

print(p42.x)

print(p42.y)


```
print(p42.z)
```

Module name aliasing || renaming Module name

=====

3. import modulename as aliasname

```
import module as m1
```

4. import module1 as m1, module2 as m2,....moduleN as Mn.

How to import member from a module

=====

5. from modulename import membername

Exa:

===

6. from modulename import member1,member2,member3.... memberN.

7. from modulename import * (for Importing all member of a module)

```
from p44 import demo
```

```
demo()
```

```
from p42 import x,y,z
```

Member Aliasing || Member renaming

=====

8. from modulename import member as m

9. from modulename import member1 as m1,member2 as m2.....,memberN as mn.

=====

1. import modulename

2. import module1,module2,...moduleN

3. import module as m

4. import module1 as m1,module2 as m2,...moduleN as mn.

5. from modulename import member

6. from modulename import member1,member2,...memberN

7. from modulename import member as m1

8. from modulename import member1 as m1,member2 as m2... ,memberN as mn.

9. from modulename import *

list

====

if we want store multiple value as single entity where duplicates are allowed and order is required to preserve then we have to use list data.

1. list is denoted By []

2. Heterogenous elements(Different data type value) are allowed.

3. duplicates are allowed here.

4. indexing and slicing concept is applicable here.

5. List is mutable(Changeable)

l=[item1,item2,item3...itemN]

l=[10,20,30,10.5,"RAM",10]

print(l[0])#10

```
print(l[1])#10
print(l[2])#30
print(l[1:5])#20,30,10.5,"RAM"#slicing Concept
```

```
l=[10,20,30,10.5,"RAM",10]
```

```
print(type(l))#list
```

```
len()
```

```
====
```

len function is used to find length of any collection(list,tuple,set,dict,str).

```
x="techpile"
```

```
print(len(x))#8
```

```
l=["RAM","ROHAN","MOHAN",True]
```

```
print(len(l))#4
```

```
l.append("Techpile")
```

```
print(len(l))
```

```
=====
```

tuple

```
=====
```

=>tuple is a same as list but tuple is immutable.

=>tuple is ready only version of list.

1. tuple is denoted by ()
2. duplicates are allowed
3. collection of heterogeneous elements.

4. Slicing and indexing concept is applicable.

5. tuple is immutable.

```
x=(10,20,30,20,20.5,"RAM")
```

```
print(type(x))#tuple
```

```
for i in x:
```

```
    print(i,end="\t")
```

```
t1=(50)
```

```
print(type(t1))#int
```

```
t2=(50,)
```

```
print(type(t2))#tuple
```

```
x=(10,20,30,20,20.5,"RAM")
```

```
x.append(True)#immutable
```

```
x.remove(10)#immutable
```

set

====

if we want to represent a group of value as single entity where duplicates are not allowed and order is not applicable then we have to use set datatype.

1. set is denoted by {}

2. duplicates are not allowed

3. indexing and slicing concept is not applicable

4. heterogeneous elements are allowed here.

5. set is mutable.

=====

```
s={10,20,30,"RAM",True,10}
```

```
s.add("Techpile")
```

```
print(s)#s #mutable
```

```
=====
```

```
dict
```

```
=====
```

=>if we want to represent a value in the form of Key value pair then we have to use dict datatype.

```
{key1:value1,key2:value2,.....KeyN:valueN}
```

=>here value can be duplicates but key can't be duplicates,if we are trying to duplicates key then old value of key will replaced with new value of key.

Syntax

```
=====
```

```
d={key1:value1,key2:value2,.....KeyN:valueN}
```

Exa:

```
=====
```

```
d={101:"RAM",102:"IET LUCKNOW",103:"Lucknow"}
```

```
d[104]="ROHAN" Mutable
```

```
print(d[104])#Rohan
```

How to access value from Dict

```
=====
```

```
dict[key]
```

Exa:

```
d[101]#RAM
```

```
d[103]#Lucknow
```

d[102]#IET LUCKNOW

=====

random module

=====

=>random is a pre-defined module providing some function related to random object.

1. random()

=====

=>random function is used to return random number between 0(inclusive) to 1(exclusive).

0-.99999

0

0.5

0.56785

0.200

1

2. randint()

=====

=>randint function is used to return random integer number between given range(inclusive).

Syntax

=====

randint(x,y)

randint(10,20)10,20 15 ,14, 9

uniform

=====

uniform function is used to return random float value between given range(exclusive).

```
from random import uniform
```

```
x=uniform(10,20)#
```

```
print(x)
```

choice()

=====

```
l=["HTML","CSS","JS","JQUERY"]
```

```
t=("HTML","CSS","JS","JQUERY")
```

=>choice function is used to return random object from given collection(list,tuple).

```
from random import choice
```

```
l=["HTML","CSS","JS","JQUERY"]
```

```
print(choice(l))
```

=====

math module

=====

=>math is a pre-defined module providing some functions related to mathematical operation.

floor

=====

floor function is used to return lowest interger value of given value.

```
print(floor(10.00054))#10
```

```
print(floor(101.9054))#101
```

```
print(floor(1.9054))#1
```

ceil()

=====

ceil function is used to return largest interger value of given value.

```
import math
```

```
print(math.ceil(10.234))#11
```

```
print(math.ceil(0.3456))#1
```

```
print(math.ceil(1001.2345))#1002
```

sqrt()

factorial()

pow(x,y)

=>x to the y

```
pow(2,3)#2*2*2
```

etc.

=====

function

=====

=> function is a block of re-usable code which used to perform particular task.

type of function

=====

1. Built-in Function

The functions which are coming along with python software are known as Built-in Function.

Exa:

input()

print()

int()

type()

float()

str()

bool()

len()

upper()

lower()

split()

etc.

2. User Defined Function(UDF)

=>In python two keyword are used to create function.

=>The function which are defined by programmer based on bussiness requirement are known as UDF.

1. def(compulsory)

2. return (optional)

def functionName(): # function Declaration

```
print("Welcome to Techpile")
```

Syntax

```
=====
```

```
def functionName():
```

```
    -----
```

```
    -----
```

```
    -----
```

```
    -----
```

```
def demo():
```

```
    x=int(input("Enter Your Base value : "))#2
```

```
    y=int(input("Enter your power value : "))#3
```

```
    res=x**y
```

```
    print(x," to the power ",y," = ",res)
```

```
# 2 to the power 3 = 8
```

```
=====
```

```
function with parameter
```

```
parameter
```

```
=====
```

=>parameter provides inputs to the function. if any function contains any parameter then at the time calling function compulsory we have to provide value to that parameter otherwise we will get error.

Syntax

```
=====
```

```
def functionName(param1,param2,param3....paramN):
```

```
    statement
```

```
    statement
```

```
    statement
```

```
functionName(value1,value2,value3,....valueN)
```

Exa:

```
def demo(x,y):
```

```
    print(x)
```

```
    print(y)
```

```
demo(20,30)
```

x and y is formal argument

20 and 30 is Actual argument

```
def multiply(x,y,z):
```

```
    print("Result is : ",(x*y*z))
```

```
multiply(10,20,30)
```

```
def sum(x,y):
```

```
    print(x)
```

```
    print(y)
```

```
sum()
```

Type of argument in Python

=====

1. Positional argument

=>Here numbers of argument and position of argument must be matched.

=>if we are trying change position of argument then result may be changed and trying to change number of argument then we will get error.

Exa:

```
def demo(x,y):
```

```
    print(x/y)#0.5
```

```
def demo1(a,b):
```

```
    print(a+b)#
```

```
demo(10)#error
```

```
demo(10,15,30)#error
```

```
demo(10,20)#0.5
```

```
demo(20,10)#2.0
```

```
demo1(10)#error
```

```
demo1(20,30)#50
```

```
demo1(30,20)#50
```

=====

2. Keyword argument

=>In keyword argument we can define keyword (Same name as formal argument) as a actual argument.

```
def demo(x,y):
```

```
    sum=x+y
```

```
sub=x-y  
multi=x*y  
div=x/y
```

```
demo(x=20,y=10)
```

```
demo(y=10,x=20)
```

```
x=20
```

```
y=10 keyword argument
```

```
=====
```

3. default argument

some time we provide default value to our form argument such type of argument are knows as default parameter.

=> if we are not provide value to our formal argument then function will be executed with default.

```
def demo(x=10,y=20,z=30):  
    print(x+y+z)
```

```
demo()#60
```

```
demo(40)#90
```

```
demo(40,100)#170
```

```
demo(40,100,100)#240
```

```
demo(40,100,100,50)#error
```

```
=====
```

4. variable length argument

=====

if we want to call any function with multiple value(actual argument) then we have use variable-length argument.

* symbol is used to create variable-length argument.

```
def demo(*x):  
    print(x)
```

demo()

demo(10)

demo(10,20)

demo(10,20,30,40)

demo(10,20,30,40,50)

__name__ => "__main__" direct execution

__name__ => modulename indirect execution

=====