

## \* Flavours of Python =>

- CPython = C + Python
- Jython / JPython = Java + Python
- IronPython = C# + Python
- Ruby Python = Ruby + Python
- PyPy = Python + Python
  - To increase the performance of Python
  - ↓
  - PVM (Python Virtual Machine)
  - +
  - JIT (Just In Time Compiler)

→ Anaconda Python => Python  $\Rightarrow$   
Huge programs  
can't Run ~~correctly~~

→ Panda Python => Lower than  
Anaconda

## "PYTHON"

### ★ History ⇒

Started In = 1991

Started By = Guido Van Rossum (Netherlands)

First Version = Labeled version (0.9.0)  
1991

Name 'PYTHON' = Monty Python's Flying Circus

## \* Python Versions →

Python 1.0 = 1994

Python 2.0 = 2000

Python 3.0 = 2008 ✓

Python 3.10.4 = 2022

\* Python 3.13.5 = June 2025 (Latest Version)

## \*Features of Python =>

1. Easy to learn & Use  $\Rightarrow$  सबसे Easy Syntax
2. Freeware & Open Source  $\Rightarrow$  Source Code भी निलगा है
3. High level Programming Language  $\Rightarrow$  Coding English में
4. Platform Independent  $\Rightarrow$  No Dependancy on OS & uses Interpreter
5. Architecture Independent  $\Rightarrow$  No Dependancy on Hardware
6. Dynamically Typed  $\Rightarrow$

→ Python में अन्य languages की तरह variable को declare करने की Need नहीं होती है।  
→ इसमें variable को direct value assign कर दी जाती है तथा Runtime पर उसके DataType  
का निर्धारण होता है।  
→ इसमें एक ही Variable की Program में कलज़-2 प्रकार की Values Assign की जा सकती है।

Ex:-      a=5                  a=6.5                  a="Ram"

## 7. Cross Platform Language

→ Python में किसी अन्य Platform पर बनाए गए Code को ऑप्स में लिया जा सकता है।

## 8. Extensible

→ Python में दुसरी languages में बनाए गए program को भी use में लिया जा सकता है।

### 2 लाभ -

A. Application की performance को increase किया जा सकता।

B. Non-Python Code को Python की नई Application में use में लिया जा सकता।

## 9. Interpreted language

## 10. Both Procedural and Object Oriented language

## 11. Embeddable $\Rightarrow$

$\rightarrow$  Python के code में अन्य language के code को Embed कर application develop किया जा सकता है।

## 12. Extensive libraries $\Rightarrow$

$\rightarrow$  Python में Rich set of Libraries होता है, जो भलग-<sup>2</sup> Applications को develop करने में help करता है।

## \* Python Framework for Application Development ⇒

- 1. Django
- 2. TurboGears
- 3. Numpy (Numerical Python)
- 4. Pandas
- 5. Matplotlib
- 6. Web2Py
- 7. Flask
- 8. Bottle
- 9. CherryPy
- 10. AIOHTTP
- 11. CubicWeb - Fullstack
- 12. Dash
- 13. Falcon
- 14. Giotto - Full stack

## \*Python Programming Modes ⇒

- A . Interactive Mode
- B . Script Base Mode

### A. Interactive Mode ⇒

- Line by Line Execution
- इसमें प्रत्येक Line >>> के साथ होती है।
- Testing & Debugging में use.

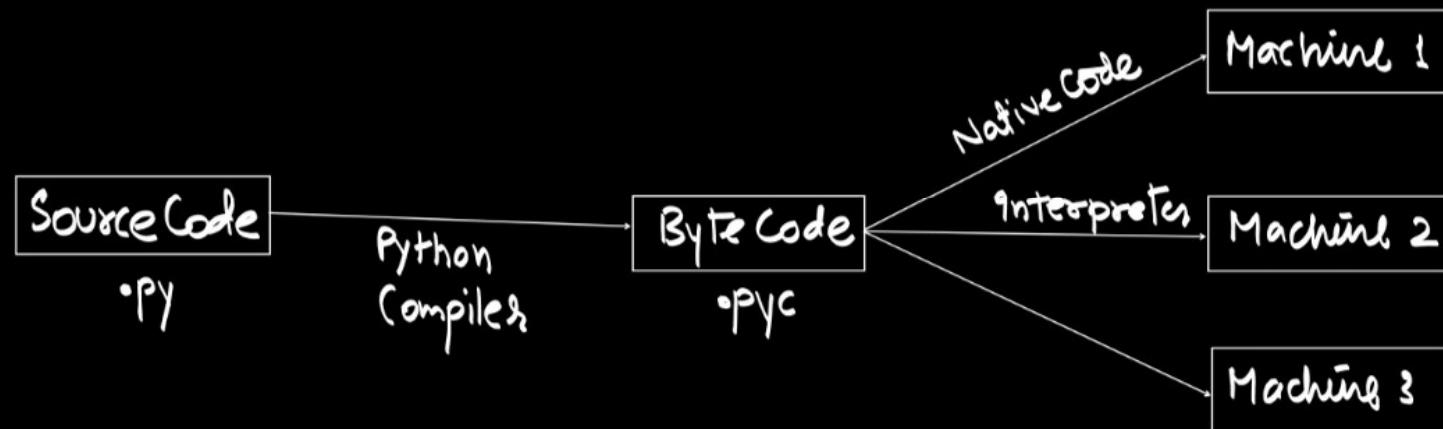
Ex ⇒      `>>> print ("Hello")`

O/P = Hello

## B. Script Based Mode ⇒

→ इस Mode में पूरे Python Program को एक file में लिखकर .py extension से save कर किया जाता है।

## \* Translation Policy of Python Program ⇒



## \* Character set of Python →

{ C, C++ ⇒ ASCII ⇒ 256  
JAVA ⇒ UNICODE-16 ⇒ 65536  
PYTHON ⇒ UNICODE-8 ⇒ 256

## \* Python Tokens $\Rightarrow$

- Building block of program.
- प्रोग्राम की smallest unit जिसे further divide नहीं किया जा सकता है
- Types  $\Rightarrow$

- A. Identifiers
- B. Keywords
- C. Literals
- D. Datatype
- E. Operators

## A. Identifier →

- Python case sensitive होती है।
- मात्र किसी identifier के पहले underscore (-) लगा दिया जाए, तो वह private बन जाता है।  
उद्दीप्ति-    \_a = 5  
                      ↓  
                      PRIVATE
- newline, \n & other special symbols are not allowed.

## B. Keywords →

- Originally = 33
- After python 3.8 = 35

## C. Literals $\Rightarrow$

$\rightarrow$  Python में Constants की Literals कहा जाता है।

$\rightarrow$  प्रकार =

a. Integer Based

b. Float Based  $\rightarrow$  Double is not allowed.

c. Complex Numbers  $\rightarrow$   $i + j$   
 $\downarrow$                      $\searrow$   
Real                    imaginary Number  
Number

Ex =  $3i + 4j$

d. Boolean Literals  $\rightarrow$  true & false

e. String Based Literals  $\rightarrow$  'Shubham'  
"whoshubhamsis"  
'''Swarnkar'''

{char X}

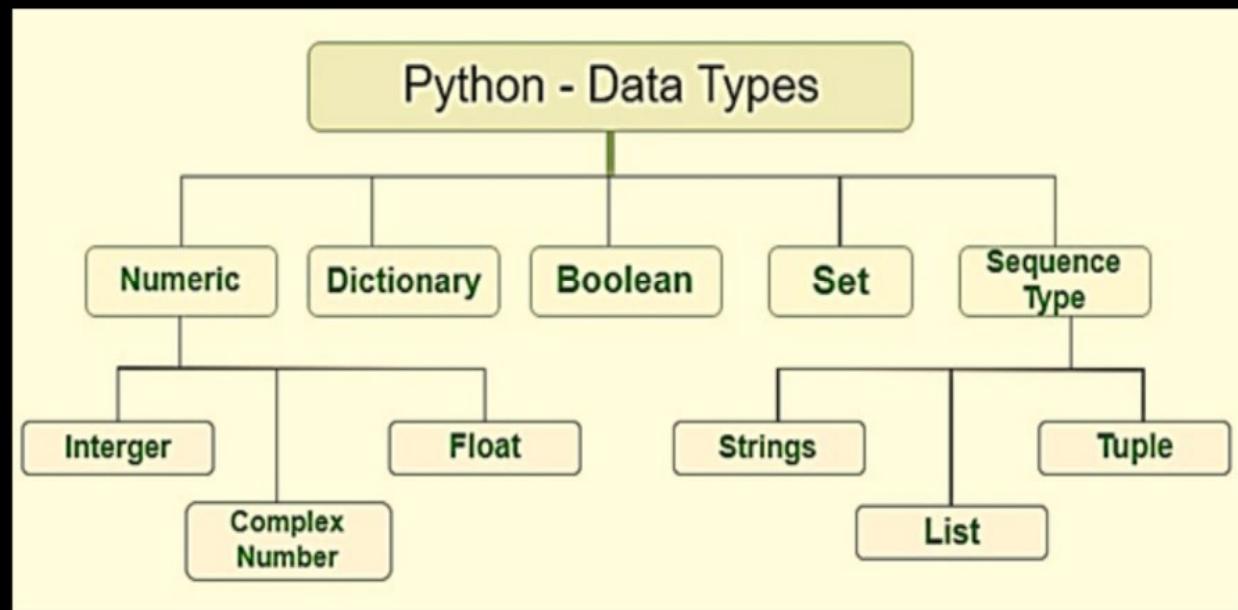
## D. DataTypes

- Python में datatype की variable की present value से बदला जाता है।
- Python में datatypes explicitly नहीं बताये जाते हैं, केवल variable की value के अनुसार program के execution के दौरान निर्धारित किया जाता है, इसी कारण Python को Dynamically typed language कहते हैं।

a = 5

a = 5.9

a = 'Ram'



\* program में किसी variable का datatype पता करने के लिए "type()" method का use किया जाता है।

जैसे = a = 10  
print(type(a)) = <class "int">

\* किसी variable की location पता करने के लिए "id()" method का use किया जाता है।

जैसे = a = 10  
print(id(a)) = 43921436  
RandomNumber

## टाइप प्राप्ति प्रूफली

1. `a = 100  
type(a)`

O/p - <class 'int'>

2. `b = true  
type(b)`

O/p - <class 'bool'>

3. `c = -49.678  
type(c)`

O/p - <class 'float'>

4. `d = 4i + 5j  
type(d)`

O/p - <class 'complex'>

5.  $a = 5i + 3j$   
 $b = 3i + 4j$   
 $c = a + b$   
`print(c) → O/p = 8i + 7j`  
`type(c) → O/p = <class 'complex'>`  
`print(c.real) → 8`  
`print(c.imag) → 7`

## \* Python input Statement ⇒

```
a = input() ← 10 ←  
print(type(a))  
O/P = <class 'string'>
```

→ Python `input()` method का use करके जो भी value read करता है, वह default से string के रूप में read किया जाता है।

```
a = int(input()) ← 10 ←  
print(type(a))  
O/P = <class 'int'>
```

\* <sup>float</sup>  
b = float(input()) ← 10 ←  
print(type(b))  
O/P = <class 'float'>

## \* Python Input Statement ⇒

```
a = input() ← 10 ←  
print(type(a))  
O/P = <class 'string'>
```

→ Python `input()` method का use करके जो भी value read करता है, वह default से string के रूप में read किया जाता है।

```
a = int(input()) ← 10 ←  
print(type(a))  
O/P = <class 'int'>
```

\* ← 10 ←  
Float  
b = double(input())  
print(type(b))  
O/P = Error  
↓ <class 'float'>

→ Python में Multiple Values की भी `input()` method के द्वारा लिया जा सकता है

उदाहरण -

```
fname, Sname = input()
```

O/P = Error

Shubham Swarupkar  
Value 1

→ `input()` method के साथ `split()` method का use करके multiple values को Read किया जाता है।

→ `split()` method default रूप से space विलेने वाला string को split करता है।

उदाहरण -

```
fname, Sname = input().split()
```

O/P = Shubham Swarupkar

Shubham Swarupkar  
fname      Sname

\* split() method के bracket में उस symbol को लिखा जा सकता है, जिसे strings के बीच display किया जाता है।

उदाहरण- fname, Sname = input().split('@')

print(fname) = KC

print(Sname) = gmail.com

Split(-)  
KC@gmail.com  
f      S

## \* Python Output Statement ↴

print("Hello")                    O/P = Hello

a=10  
print(a)                        O/P = 10

a=10  
b=20  
print(a+b)                    O/P = 30

## ★ Python Data Types ⇒

### A- Numeric Data Types ⇒

a. int = 54, 23, 5

b. float = 45.23, 54.5, 5.23

c. Complex = यदि  $i+j$  का एक Paired Pattern होता है, जिसमें  $i=real\ number$  &  $j=imaginary\ number$  को denote करता है।

जैसे -  $\frac{2i+5j}{j}$   
                ↓      ↓  
                real    imaginary

## B. Sequence DataType $\Rightarrow$

### a. String $\Rightarrow$

→ Group of characters

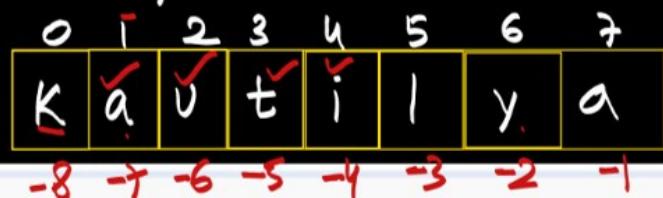
→ Python में string को ` ', " ", ''' ''', के त्रिपल quotes declare किया जाता है

जैसे - 'Shubham'  
"Shubham"  
''' Shubham '''

→ Python में NULL character को support नहीं किया जाता है

## \* String Slice in Python ⇒

- किसी string को trim करके display करना।
- [ ] का use किया जाता है।



print[4:8]  
[... : -]  
whole  
Kautilya  
-2: [ ]  
Star [ ]

```
main.py
1 a = "Kautilya"
2 print(a[0])
3 print(a[-2])
4 print(a[1:5])
5 print(a[:5])
6 print(a[:])
7 print(a[1:])
```

Shell

1	print(a[0])	→ K
2	print(a[-2])	→ y
3	print(a[1:5])	→ auti✓
4	print(a[:5])	→ Kauti✓
5	print(a[:])	→ Kautilya✓
6	print(a[1:])	→ autilya✓
7		>

→ Python में निम्न 5 Basic Data Types मात्रे जाते हैं, क्योंकि casting function द्वारा इनकी values की interchange किया जा सकता है।

int, float, bool, complex, str

Complex to int  
X

main.py      *Float*

1 print(int(123.45))	→ 123
2 print(int(True))	→ 1
3 print(int(False))	→ 0
4 print(int(10))	→ 10
5 print(int(10.3))	→ 10
6 print(int(0x46ef))	→ 18159
7 print(int("1020"))	→ 1020
8 print(int(3+4j))	→ Traceback (most recent call last): File "<string>", line 8, in <module> TypeError: can't convert complex to int <i>real</i> <i>Imaginary</i>

main.py



Run

Shell

```
1 print(float(123.45))           → 123.45
2 print(float(True))             → 1.0
3 print(float(False))            → 0.0
4 print(float(10))               → 10.0
5 print(float(10.3))              → 10.3
6 print(float(0x46ef))            → 18159.0
7 print(float("1020"))             → 1020.0
8 print(float(3+4j))
```

T  
real    imaginary

Traceback (most recent call last):  
File "<string>", line 8, in <module>  
TypeError: can't convert complex to float  
>

main.py

```
1 print(complex (123.45)) → (123.45+0j)
2 print(complex (True)) → (1+0j)
3 print(complex (False)) → 0j
4 print(complex (10)) → (10+0j)
5 print(complex (10.3)) → (10.3+0j)
6 print(complex (0x46ef)) → (18159+0j)
7 print(complex ("1020")) → (1020+0j)
8 print(complex ("3+4j")) → (3+4j)
9 print(complex ("4j")) → 4j
10 print(complex ("j")) → 1j
```



Shell

## B. List $\Rightarrow$

- Python का Array
- List items are ordered, changeable & can contain duplicate values.
- क्षेत्र-2 प्रकार के data types का storage.
- List के प्रत्येक element को Comma से क्षेत्र-2 छिपा जाता है और पुरी List को [ ] में बन्द कर दिया जाता है।

जैसे-

```
mylist = ["Shubham", "swankar", 1, True]
```

```
type(mylist)
```

```
O/p = <class "list">
```

```
BPSC = ["STET", 270, False]  
print(type(BPSC))
```

```
O/p = <class "list">
```

### C. Tuple $\Rightarrow$

- Collection of different data types
- Tuple items are ordered, unchangeable & can contain duplicate values.
- Tuple के प्रत्येक elements को Comma से जालग-2 किया जाता है और पुरे Tuple को () में बना किया जाता है।

Ex  $\Rightarrow$  KC = (456, "Ram", True)

print(type(KC))

O/p = <class "Tuple">

KC = [456, "Ram", True]

print(type(KC))

O/p = <class "List">

#### D. Set $\Rightarrow$

- Collection of different data types
- Set items are → Unordered  
Unchangeable  
No duplicate value

Ex  $\Rightarrow$   $kc = \{1, 2, 3, 5, 7, 8\}$

$kc1 = \{2, 4, 5, 7, 9, 10\}$

→ Set के प्रत्येक element को comma से अलग<sup>2</sup> कर {} के बीच रखा जाता है।

## E. Dictionary $\Rightarrow$

- इसमें items को Key:Value के pair में रखा जाता है।
- Dictionary items are = Ordered,  
Changeable,  
No duplicate value
- Dictionary के elements को comma से अलग  $\&$   $\{ \}$  के बीच बन्द कर दिया जाता है।

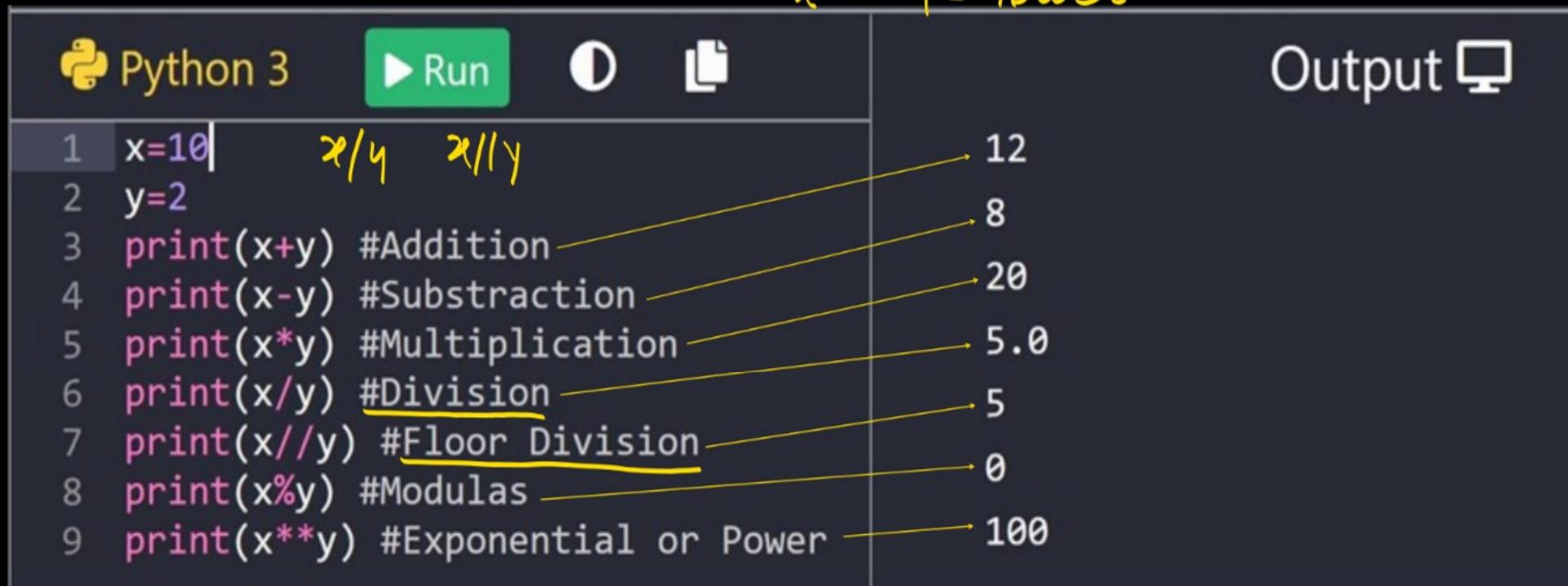
Ex  $\Rightarrow$  KC = {1:"Ram", 2:"Hari"}

List	Tuple	Set	Dictionary
Ordered	Ordered	Unordered	Ordered
[ ]	( )	{ }	{ }
Changeable	Unchangeable	Unchangeable	Changeable
Duplicate Allowed	Duplicate Allowed	No Duplicate	No Duplicate

## \* Python Operators $\Rightarrow$

### 1. Arithmetic Operators $\Rightarrow$

$x * y = \text{Product}$   
 $x ** y = \text{Power}$



The screenshot shows a Jupyter Notebook interface with a code cell and an output cell. The code cell contains the following Python code:

```
Python 3
Run
1 x=10    x/y  x//y
2 y=2
3 print(x+y) #Addition
4 print(x-y) #Substraction
5 print(x*y) #Multiplication
6 print(x/y) #Division
7 print(x//y) #Floor Division
8 print(x%y) #Modulas
9 print(x**y) #Exponential or Power
```

The output cell displays the results of the operations:

12	→	x+y
8	→	x-y
20	→	x*y
5.0	→	x/y
5	→	x//y
0	→	x%y
100	→	x**y

→ Python में divide operators को 2 प्रकार से use करें लिया जाता है-

A. Simple Division ⇒ इस division का उत्तर float value आता है और output में दराखलक के बाद 16 digits तक display किया जाता है।

B. Floor Division ⇒ इस division में यदि दोनों values int हो, तो result उत्तर int में ही आयेगा और यदि दोनों में से सक भी value float हो, तो result float आयेगा।

</> Online Python Code Editor >\_

Python 3	Run	Output
1 print(10/2)		5.0
2 print(10//2)		5
3 print(10.0/2)		5.0
4 print(10.0//2)		5.0
5 print(11.0/2)		5.0
6 print(11.0//2)		5.5 5.0

★ Addition operator का use 2 strings को आपस में जोड़ने (concatenate) में भी किया जाता है।

Ex ⇒ `print("Shubham" + "Swarnkar")` O/p - ShubhamSwarnkar

`Print ("Shubham" + " " + "Swarnkar")` O/p - Shubham Swarnkar  
                        ↑  
                SPACE

`Print ("Shubham" + 10)` O/p: Error

`Print ("Shubham" + "10")` O/p = Shubham10

★ Python में \* operator का use string के रूप कर Multiple Times Repeat कर display किया जाता है।

Ex =

print("KC" \* 5)      O/p= KC KC KC KC KC

## 2. Relational Operators $\Rightarrow$

main.py				Run	Shell
1 a = 10					True
2 b = 20					False
3 print(a < b) #Less than					True
4 print(a > b) #Greater than					False
5 print(a <= b) #Less than Equals to					False
6 print(a >= b) #Greater than Equals to					True
7 print(a == b) #Equals to					>
8 print(a != b) #Not Equals to					

```
1 a = 10
2 b = 20
3 print(a < b) #Less than
4 print(a > b) #Greater than
5 print(a <= b) #Less than Equals to
6 print(a >= b) #Greater than Equals to
7 print(a == b) #Equals to
8 print(a != b) #Not Equals to
```

### 3. Logical Operators

→ इन Operators का प्रयोग Boolean Values के साथ करते हैं।

→ 3 प्रकार →

A. Logical AND - दोनों Input TRUE = Output True

B. Logical OR - एक भी Input TRUE = Output True

C. Logical NOT - Reverse / इन्वर्ट

The screenshot shows a Python code editor interface with a file named "main.py". The code contains five print statements demonstrating logical operators:

```
1 print (True or True) → True
2 print (True and False) → False
3 print (False or True) → True
4 print (not True) → False
5 print (True and not False) → True
```

The "Run" button is highlighted in blue at the top of the editor.

main.py

```
1 print (27 and 47)
2
3 print (27 and 13)
4
5 print (0 and 42)
6
7 print (0 or 42)
8
9 print (29 or 42)
```



Shell

47

13

0

42

29

> |



- \* मग्नि Logical Operators के साथ True & False के अलावा अन्य values हो, तो जिस value की रजाए देंे equation evaluate होती है, वह output आता है।
- \* जब Inverted Comma के बीच न हो कोई string हो और न ही space हो, तो इसे Empty String कहा जाता है व Logical Not Operator के False Read करता है।

main.py		Run	Shell
1 print ( <u>not</u> False) # logic	True		
2 print ( <u>not</u> 'Kautilya')	False		
3 print ( <u>not</u> 43)	False		
4 print ( <u>not</u> True)	False		
5 print ( <u>not</u> "")	True		

#### 4. Bitwise Operator

→ int & boolean के साथ ही work करते हैं।

→ 6 यहाँ

- A. Bitwise AND (&)
- B. Bitwise OR (|)
- C. Bitwise XOR (^)
- D. Bitwise NOT (~)
- E. Bitwise left Shift («)
- F. Bitwise Right Shift (»)

★ Bitwise NOT (~) Positive Value

के बराबर negative value output  
में देता है-

print (~5) = -5

print (~False) = -1

[ print (~True) = -2 ]

main.py

10



Run

Shell

```
1 a = 10  
2 b = 3  
3 print (a & b) 1010  
4 print (a | b) 0011  
5 print (a ^ b) 0010✓  
6 print (True & True) 1011  
7 print (True | False) 1011  
8 print (True & False) 1011  
9 print (True & False) 1011  
10 print (True ^ True) 1011✓ 2  
11  
9  
True  
True  
False  
True  
False  
> |
```

$$\text{print}(10 \ll 1) = 10 \times 2 = 20$$

$$\text{print}(10 \gg 1) = \frac{10}{2} = 5$$

$$10 = \begin{array}{r} 1010 \\ \downarrow \\ 10100 \end{array}$$

$$16 \cancel{+} 4 \cancel{+} 2 \cancel{+} 0 = 20$$

$$10 = \begin{array}{r} 101 \\ \downarrow \\ 101 \end{array}$$

$$101 = 5$$

- \* जितनी बार left shift करता है, उसनी बार 2 से multiply करता पड़ता है।
- \* जितनी बार Right shift करता है, उसनी बार 2 का आज दिया जाता है।

## 5. Assignment Operators $\Rightarrow$

### A. Simple Assignment $\Rightarrow$

$$a = 10$$

$$a = b = c = 10$$

$$a, b, c = 10, 20, 30$$

### B. Compound Assignment $\Rightarrow$

$+ = , - = , *= , /= , // = , \% =$

$$\begin{aligned} a +&= b \\ a &= a + b \end{aligned}$$

$$\begin{array}{c} \text{a} \\ + \\ \text{5} \\ \hline \text{a} = \text{a} + 5 \end{array}$$

$$\begin{array}{l} a = 5 \\ b = 6 \\ \text{print}(a) \\ \text{print}(a + b) \end{array}$$

$$0/p = 11$$

## 6. Increment / Decrement Operator $\Rightarrow$

$a = 5$   
~~print(t+a)~~ X

- Python increment & decrement operators को support नहीं करता है।
- Python में unary + & unary - को support किया जाता है।

## 7. Conditional Operators $\Rightarrow$

Syntax  $\Rightarrow$

$\langle \text{first-value} \rangle \text{ if } \langle \text{Condition} \rangle \text{ else } \langle \text{Second-value} \rangle$

Ex  $\Rightarrow$

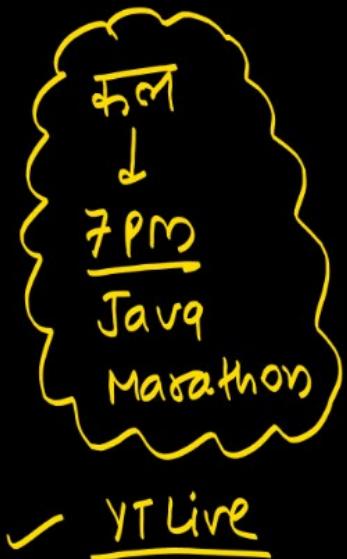
$a = 10$

$b = 20$

$c = 30 \text{ if } a > b \text{ else } 40$

print(c)

Output: 40



## 8. Identity Operator ⇒

→ Variables का Id / Address Compare करते हैं।

- A. is
- B. is not

*id = Address of a variable*

main.py		Run	Shell
1 a = 10			9789280
2 b = 10			9789280
3 print (id(a))			True
4 print (id(b))			False
5			>
6 print (a is b)			
7 print (a is not b)			

## ★ Variable Pooling →

- यदि multiple variables को same value assign की जाती है, तो उन्हें एक ही memory location से pull किया जाता है, इसे variable pooling कहते हैं।
- Memory Management.

## \* Mutable v/s Immutable Data Types ⇒

⇒ ये से Data Types जिनकी Internal State को बदला जा सकता है, उन्हें Mutable कहते हैं।

जैसे - list, set, dictionary.

⇒ ये से Data Types जिनकी Internal State को बदला नहीं जा सकता है, उन्हें Immutable कहते हैं।

जैसे - int, float, boolean, string, Tuple

(C)  $a = 10, c = 11$       (D)  $a = 11, c = 11$

100. Which of the following declarations is incorrect  
in python language?

(A)  $xyzp = 5,000,000$

xyzp

(B)  $xyzp = 5000\ 6000\ 7000\ 8000$

18|6|22

(C)  $x, y, z, p = 5000, 6000, 7000, 8000$

(D)  $x_y_z_p = 5,000,000$

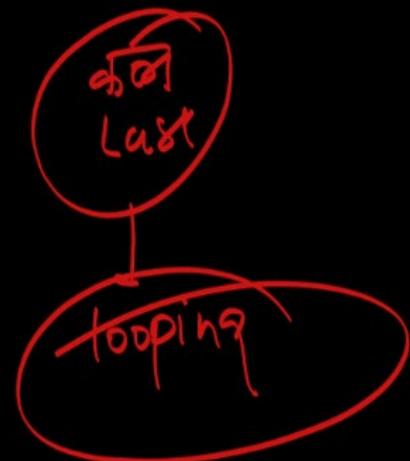
निम्नलिखित में से कौनसी घोषणा python  
language में सही नहीं है?

(A)  $xyzp = 5,000,000$

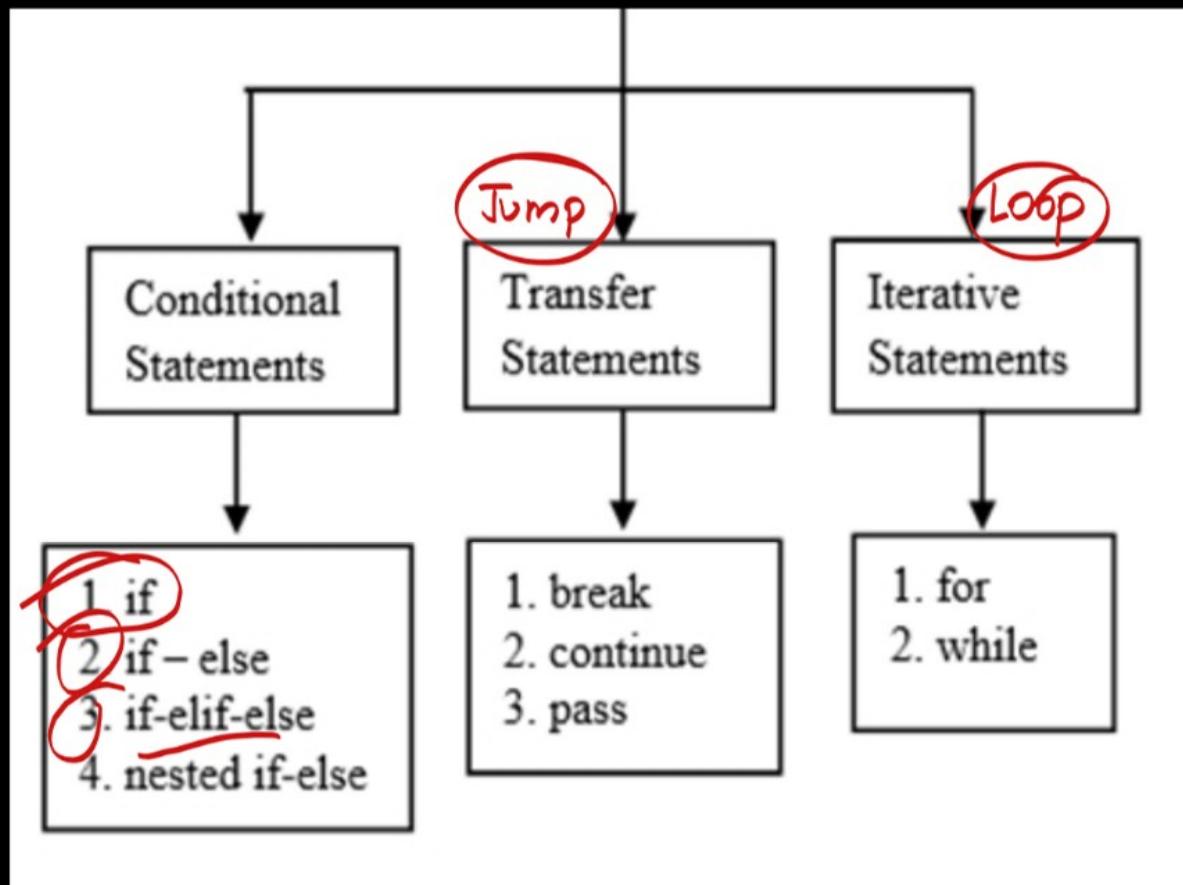
(B)  $xyzp = 5000\ 6000\ 7000\ 8000$

(C)  $x, y, z, p = 5000, 6000, 7000, 8000$

(D)  $x_y_z_p = 5,000,000$



## \* Control Statements in Python =>



no switch  
no goto  
no do-while

main.py

```
1 a = 10
2 b = 20
3 if a<b :
4 print("Hello")
```

Part of main  
if a < b :  
Tab  
not the part of if

Run

Shell

Clear

File "<string>", line 4

print("Hello")

^

IndentationError: expected an indented block

main.py

```
1 a = 10
2 b = 20
3 if a<b :
4 print("Hello")
```

Part of if

Run

Shell

Hello  
> |

- \* Python में किसी भी condition का Block बनाने के लिए Curly Brackets का USE नहीं किया जाता है,  
इसकी जगह Indentation का USE किया जाता है।
- \* Python में किसी भी Condition की statement को Condition से थोड़ा आगे करके लिखा जाता है,  
जिससे इस Condition का Part बना जाते।

main.py

```
1 a = 10
2 b = 20
3 if a<b :
4 print("Hello")
5 else :
6 print("World") X
7 print ("Jai Raam Ji Ki")
```



Run

Shell

Clear

File "<string>", line 6

```
print("World")
^
```

IndentationError: expected an indented block

>

main.py



Run

Shell

```
1 a = 10
2 b = 20
3 if a<b :
4     print("Hello")
5 else :
6     print("World")
7 print ("Jai Raam Ji Ki")
```

Part of main

Part of if

Part of if

Part of else

Hello

Jai Raam Ji Ki

> |

main.py



Run

Shell

```
1 a = 10
2 b = 20
3 if a<b :      print("Hello")
4 else :         print("World")
5 print ("Jai Raam Ji Ki")
```

```
Hello
Jai Raam Ji Ki
> |
```

3. If-elif-else ⇒

$$\boxed{\text{elif} = \text{else} + \text{if}}$$

\* if-elif-else की combinedly if-elif-else कहा जाता है

main.py



Run

Shell

```
1  a = 10
2  b = 20
3  if a<b :
4      print("a is small")
5  elif a>b :
6      print("a is big")
7  else ;
8      print ("a & b are equal")
```

*a = 15  
b = 1*

a is small

>

main.py



Run

Shell

```
1 a = 10
2 b = 20
3 if a<b :
4     print("a is small")
5 else :      elis
6     if a>b :
7         print("a is big")
8     else : i
9     print ("a & b are equal")
```

Nested if

a is small  
> |

main.py



Run

Shell

```
1 a = 1
2 while a<=10 :
3     print(a)
4     a+=1
5
6
7
8
9
10
```

```
1
2
3
4
5
6
7
8
9
10
```

main.py



Run

Shell

```
1 a = 1
2 while a<=10 :
3     print(a)
4 if a == 5 :
5     break
6 a+=1
```

1  
2  
3  
4  
5  
>

5

l

main.py



Run

Shell

```
1 a = 1
2 while a<=10 :
3     print(a)
4     if a==2:
5         break
6     a+=1
7 else:
8     print("Me Else Hu")
```

1  
2

>

- \* Python में while & for के साथ else का भी use किया जा सकता है।
- \* यदि loop condition के false होने तक पुरा execute होता है, तो Else भी execute होगा।
- \* यदि loop को forcefully break कर दिया है, तो Else part execute नहीं होगा।
- \* While की condition false होने पर भी else Run होगा।
- \* for loop की List, Tuple, Set आ Dictionary के साथ use में लिया जाता है तथा एक single variable का use इनके elements को display करने के लिए किया जाता है।

main.py



Run

Shell

```
1 values = ['P', 'y', 't', 'h', 'o', 'n']
2 for val in values:
3     print(val)
4 pass
5
6
7
8
9
```

Pass - Python की NULL Statement  
→ Interpreter इसे Read करता है, परन्तु output नहीं देता है।  
→ इसका use Program में एक line का space रखने के लिए किया जाता है, ताकि future में code add किया जा सके।

P  
y  
t  
h  
o  
n

>