PYTHON PROGRAMMING LANGUAGE

Python Became the Best Programming Language & Fastest programming language. Python is used in Machine Learning, Data Science, Big Data, Web Development, Scripting, LLM , Generative AI everywhere. Python software - pycharm || vs code || jupyter || spyder

PYTHON INTERPRETER

IDE (INTEGRATED DEVELOPMENT ENVIRONMENT)

IDE (INTEGRATED DEVELOPMENT ENVIRONMENT) =>

- Using IDE - One can write code, run the code, debug the code
- IDE takes care of interpreting the Python code, running python scripts, building executables, and debugging the applications.
- An IDE enables programmers to combine the different aspects of writing a computer program.
- If you want to be Python Developer only then you need to install (IDE -- PYCHARM)

PYTHON INTERPRETER --> What is Python interpreter? A python interpreter is a computer program that converts each high-level program statement into machine code. An interpreter translates the command that you write out into code that the computer can understand

PYTHON INTERPRETER EXAMPLE --> You write your Python code in a text file with a name like hello.py . How does that code Run? There is program installed on your computer named "python3" or "python", and its job is looking at and running your Python code. This type of program is called an "interpreter".

PYTHON INTERPRETER & COMPILER

Both compilers and interpreters are used to convert a program written in a high-level language into machine code understood by computers. Interpreter ->

Interpreter -->

- Translates program one statement at a time
- Interpreter run every line item
- Execute the single, partial line of code
- Easy for programming

Compiler -->

- Scans the entire program and translates it as a whole into machine code.
- No execution if an error occurs
- You can not fix the bug (debug) line by line

Is Python an interpreter or compiler? Python is an interpreted language, which means the source code of a Python program is converted into bytecode that is then executed by the Python virtual machine. Python is different from major compiled languages, such as C and C++, as Python code is not required to be built and linked like code for these languages.

How to create python environment variable 1- cmd - python (if it not works) 2- find the location where the python is installed -- > C:\Users\A3MAX SOFTWARE TECH\AppData\Local\Programs\Python\Python39\Scripts 3- system -- env - environment variable screen will pop up 4- select on system variable - click on path - create New 5- C:\Users\kdata\AppData\Local\Programs\Python\Python311 6- env - sys variable - path - new - C:\Users\kdata\AppData\Local\Programs\Python\Python311\Scripts 7- cmd - type python -version 8- successfully python install in cmd

ANACONDA

```
In [27]: 1 + 1 # ADDITION
```
```
Out[27]: 2
```
```
In [29]: 2-1
```
```
Out[29]: 1
```
```
In [31]: 3*4
```
```
Out[31]: 12
```
```
In [33]: 8 / 4 # Division
```
```
Out[33]: 2.0
```
```
In [35]: 8 / 5 #float division
```
```
Out[35]: 1.6
```
```
In [37]: 8/4 ## float division
```
```
Out[37]: 2.0
```
```
In [39]: 8 // 4 #integer division
```
```
Out[39]: 2
```

```
In [41]: 8 + 9 - 7

Out[41]: 10

In [43]: 8 + 8 - #syntax error:

  Cell In[43], line 1
    8 + 8 - #syntax error:
            ^
SyntaxError: invalid syntax

In [45]: 5 + 5 * 5

Out[45]: 30

In [47]: (5 + 5) * 5 # BODMAS (Bracket || Orders || Divide || Multiply || Add || Substact

Out[47]: 50

In [51]: 2 * 2 * 2 * 2 * 2 # exponentiation

Out[51]: 32

In [53]: 2 * 5

Out[53]: 10

In [57]: 2 ** 5 # This is 2 power 5

Out[57]: 32

In [59]: 15 / 3

Out[59]: 5.0

In [61]: 10 // 3

Out[61]: 3

In [63]: 15 % 2 # Modulus -> Modulus is the remainder value of the division  --> 15/2 = 7

Out[63]: 1

In [65]: 10 % 2 # 10/2 = 5 thus remainder is 0

Out[65]: 0

In [67]: 15 %% 2

  Cell In[67], line 1
    15 %% 2
        ^
SyntaxError: invalid syntax

In [69]: 3 + 'nit'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[69], line 1
----> 1 3 + 'nit'

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [71]:
```
a,b,c,d,e = 15, 7.8, 'nit', 8+9j, True

print(a)
print(b)
print(c)
print(d)
print(e)
```

```
15
7.8
nit
(8+9j)
True
```

In [73]:
```
print(type(a))
print(type(b))
print(type(c))
print(type(d))
print(type(e))
```

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'complex'>
<class 'bool'>
```

In [75]:
```
type(c)
```

Out[75]:
```
str
```

- So far we code with numbers(integer)
- Lets work with string

In [78]:
```
'Naresh IT'
```

Out[78]:
```
'Naresh IT'
```

Python inbuild function - print & you need to pass the parameter in print()

A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result.

In [81]:
```
print('Max it')
```

```
Max it
```

In [83]:
```
"max it technology"
```

Out[83]:
```
'max it technology'
```

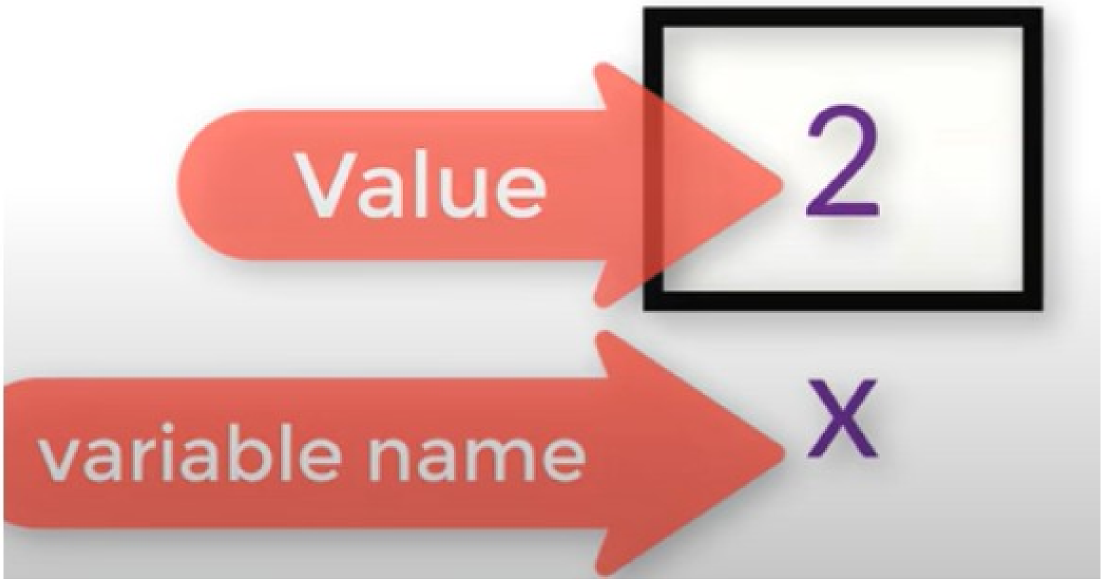
```
In [85]: s1 = 'max it technology'
         s1

Out[85]: 'max it technology'

In [87]: a = 2
         b = 3
         a + b  # Prints the value

Out[87]: 5

In [89]: c = a + b
         c

Out[89]: 5

In [91]: a = 3
         b = 'hi'
         type(b)

Out[91]: str

In [93]: print('max it's"Technology"') # \ has some special meaning to ignore the error

  Cell In[93], line 1
    print('max it's"Technology"') # \ has some special meaning to ignore the erro
r
                                 ^
SyntaxError: unterminated string literal (detected at line 1)

In [105… print('max its\' "Technology"') #\ has some special meaning to ignore the error

         max its' "Technology"

In [107… print("max it',  'Technology") # Prints ' ' as it is within " "

         max it',  'Technology

In [109… # print the nit 2 times
         'nit' + ' nit'

Out[109… 'nit nit'

In [111… 'nit'  ' nit'

Out[111… 'nit nit'

In [113… #5 time print
         5 * 'nit'

Out[113… 'nitnitnitnitnit'

In [115… 5*' nit' # soace between words

Out[115… ' nit nit nit nit nit'

In [117… print('c:\nit') #\n -- new line
```

```
c:
it
```

In [119…
```
print(r'c:\nit') #raw string bec r is written before
```

```
c:\nit
```

variable || identifier || object



In [123…
```
2
```

Out[123…
```
2
```

In [125…
```
x = 2 #x is variable/identifier/object, 2 is the value
x
```

Out[125…
```
2
```

In [127…
```
x + 3
```

Out[127…
```
5
```

In [129…
```
y = 3
y
```

Out[129…
```
3
```

In [131…
```
x + y
```

Out[131…
```
5
```

In [133…
```
x = 9
x
```

Out[133…
```
9
```

```
In [135…  x + y
Out[135…  12

In [137…  x + 10
Out[137…  19

In [139…  _ + y # _ understand the previous result  --> Works in Google colab but not Jupi
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[139], line 1
----> 1 _ + y

TypeError: can only concatenate str (not "int") to str
```

```
In [141…  _ + y
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[141], line 1
----> 1 _ + y

TypeError: can only concatenate str (not "int") to str
```

Works in Google colab but not Jupiter

- _ + y
- _ + y
- y
- _ + y
- _ + y
- _ + y

string variable

```
In [148…  name = 'mit'

In [150…  name
Out[150…  'mit'

In [152…  name + 'technology'
Out[152…  'mittechnology'

In [154…  name + ' technology'
Out[154…  'mit technology'
```

```
In [156…  name 'technology'

  Cell In[156], line 1
    name 'technology'
         ^
SyntaxError: invalid syntax
```

```
In [158…  name

Out[158…  'mit'
```

```
In [160…  len(name)

Out[160…  3
```

```
In [162…  name[0] #python index begins with 0

Out[162…  'm'
```

```
In [164…  name[5]

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[164], line 1
----> 1 name[5]

IndexError: string index out of range
```

```
In [166…  name[7]

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[166], line 1
----> 1 name[7]

IndexError: string index out of range
```

```
In [168…  name[-1]

Out[168…  't'
```

```
In [170…  name[-2]

Out[170…  'i'
```

```
In [172…  name[-6]

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[172], line 1
----> 1 name[-6]

IndexError: string index out of range
```

Slicing

```
In [175…  name
Out[175…  'mit'

In [177…  name[0:1] #to print 2 character
Out[177…  'm'

In [179…  name[0:2]
Out[179…  'mi'

In [181…  name[1:4]
Out[181…  'it'

In [183…  name[1:]
Out[183…  'it'

In [185…  name[:4]
Out[185…  'mit'

In [187…  name[3:9]
Out[187…  ''

In [189…  name
Out[189…  'mit'

In [191…  name1 = 'fine' # change the string fine to dine
          name1
Out[191…  'fine'

In [193…  name1[0:1]
Out[193…  'f'

In [195…  name1[0:1] = 'd' # I want to change 1st character of naresh (n) - t
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[195], line 1
----> 1 name1[0:1] = 'd'

TypeError: 'str' object does not support item assignment
```

```
In [197…  name1[0] = 'd' #strings in python are immutable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[197], line 1
----> 1 name1[0] = 'd'

TypeError: 'str' object does not support item assignment
```

In [199…
```
name1
```
Out[199…
```
'fine'
```

In [201…
```
name1[1:]
```
Out[201…
```
'ine'
```

In [203…
```
'd' + name1[1:] #i want to change fine to dine
```
Out[203…
```
'dine'
```

In [205…
```
num1.insert(2,'nit') #insert the value as per index values i.e 2nd index we are
```
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[205], line 1
----> 1 num1.insert(2,'nit')

NameError: name 'num1' is not defined
```

Introduction to ID()

In [208…
```
# variable address
num = 5
id(num)
```
Out[208…
```
140717689875000
```

In [211…
```
name = 'nit'
id(name) #Address will be different for both
```
Out[211…
```
2397806140544
```

In [213…
```
a = 10
id(a)
```
Out[213…
```
140717689875160
```

In [217…
```
b = a #thats why python is more memory efficient
```

In [219…
```
id(b)
```
Out[219…
```
140717689875160
```

In [221…
```
id(10)
```
Out[221…
```
140717689875160
```

In [223…
```
k = 10
id(k)
```
Out[223…
```
140717689875160
```

In [225…
```
a = 20  # as we change the value of a then address will change
id(a)
```
Out[225…
```
140717689875480
```

In [227…
```
id(b)
```
Out[227…
```
140717689875160
```

What ever the variable we assign, if the memory is not assigned anywhere then we can use as garbage collection.|| VARIABLE - we can change the values || CONSTANT - we cannot change the value

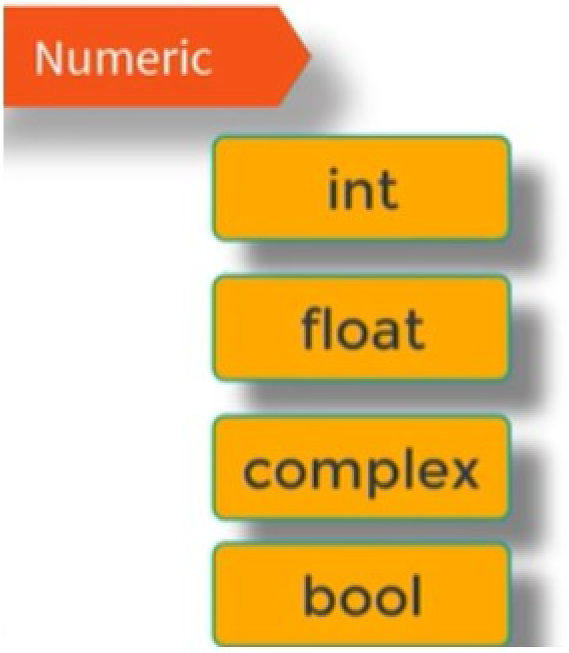
- can we make VARIABLE as a CONSTANT (note - in python you cannot make variable as constant)

In [233…
```
PI = 3.14  #In math this is alway constant but python we can change
PI
```
Out[233…
```
3.14
```

In [235…
```
PI = 3.15
PI
```
Out[235…
```
3.15
```

In [237…
```
type(PI)
```
Out[237…
```
float
```

1- NUMERIC :- INT || FLOAT || COMPLEX || BOOL



Operator

- 1- ARITHMETIC OPERATOR (+ , -, , /, %, %%, *, ^
- 2- ASSIGNMEN OPERATOR (=)
- 3- RELATIONAL OPERATOR
- 4- LOGICAL OPERATOR
- 5- UNARY OPERATOR

Arithmetic operator

In [249…
```
x1, y1 = 10, 5
```

In [251…
```
#x1 ^ y1
```

In [253…
```
x1  + y1
```

Out[253…
```
15
```

In [255…
```
x1 - y1
```

Out[255…
```
5
```

In [257…
```
x1 * y1
```

Out[257…
```
50
```

In [259…
```
x1 / y1
```

Out[259…
```
2.0
```

In [261…
```
x1 // y1
```

Out[261…
```
2
```

In [263…
```
x1 % y1
```

Out[263…
```
0
```

In [265…
```
x1 ** y1
```

Out[265…
```
100000
```

In [267…
```
x2 = 3
y2 = 2
x2 ** y2
```

Out[267…
```
9
```

Assignment operator

In [270…
```
x = 2
```

In [272…
```
x = x + 2 # if you want to increment by 2
```

In [274…
```
x
```

Out[274…
```
4
```

In [276…
```
x += 2
x
```
Out[276…
```
6
```
In [278…
```
x += 2
x
```
Out[278…
```
8
```
In [280…
```
x *= 2
```
In [282…
```
x
```
Out[282…
```
16
```
In [284…
```
x -= 2
```
In [286…
```
x
```
Out[286…
```
14
```
In [288…
```
x /= 2
x
```
Out[288…
```
7.0
```
In [290…
```
x //= 2
x
```
Out[290…
```
3.0
```
In [292…
```
a, b = 5,6 # you can assigned variable in one line as well
```
In [294…
```
a
```
Out[294…
```
5
```
In [296…
```
b
```
Out[296…
```
6
```

Unary Operator

- Unary means 1 || Binary means 2
- Here we are applying unary minus operator(-) on the operand n; the value of m becomes -7, which indicates it as a negative value.

In [299…
```
n = 7 #negattion
n
```
Out[299…
```
7
```

```
In [301…  m = -(n)
          m

Out[301…  -7

In [303…  n

Out[303…  7

In [305…  -n

Out[305…  -7
```

Relational operator

- It is used for comparing

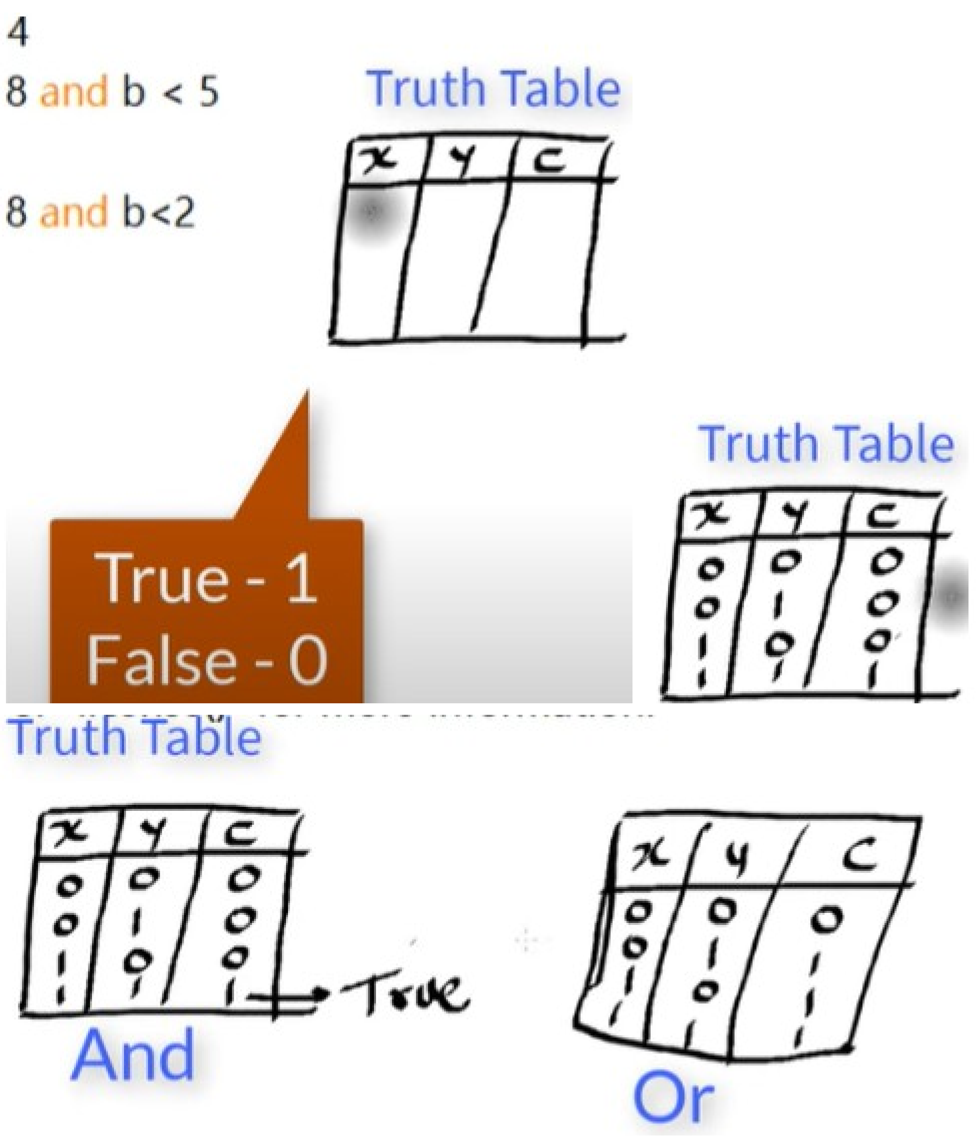
```
In [3]:   a = 5
          b = 6

In [4]:   a<b

Out[4]:   True

In [5]:   a>b

Out[5]:   False

In [9]:   # a = b # we cannot use = operator as that means it is assigning

In [11]:  a == b

Out[11]:  False

In [13]:  a != b

Out[13]:  True

In [15]:  # Here if I change b = 6
          b = 5

In [17]:  a == b

Out[17]:  True

In [19]:  a

Out[19]:  5

In [21]:  b

Out[21]:  5
```

```
In [23]: a >= b

Out[23]: True

In [25]: a <= b

Out[25]: True

In [27]: a < b

Out[27]: False

In [29]: a>b

Out[29]: False

In [31]: b = 7

In [33]: a != b

Out[33]: True
```

LOGICAL OPERATOR

- For Logical operator you need to understand about true & false table
- 3 important parts of logical operator is --> AND, OR, NOT
- AND - If both conditions are True only, then True
- OR - If one of the condition is True, then True

- Let's understand the truth table:- in truth table you can represent (True = 1 & False = 0)



In [42]:
```
a = 5
b = 4
```

In [44]:
```
a < 8 and b < 5 #refer to the truth table
```

Out[44]:
```
True
```

In [47]:
```
a < 8 and b < 2
```

Out[47]:
```
False
```

In [49]:
```
a < 8 or b < 2
```

Out[49]:
```
True
```

In [51]:
```
a>8 or b<2
```

Out[51]:
```
False
```

In [53]:
```
x = False
x
```

Out[53]:
```
False
```

In [55]:
```
not x  # you can reverse the operation
```

Out[55]:
```
True
```

In [ ]:
```
```