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 , generati ai everywhere we will learn pyton from start to end || basic to expert. if you are not done programm then that is totally fine. I will explain from starting from scratch. python software - pycharm || vs code || jupyter || spyder

PYTHON INTERPRETTER

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 wnated 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 -->

- Translates program one statement at a time
- Interpreter run every line item
- Execut 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

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

In [1]:
```
1 + 1 # ADDITION
```

Out[1]:
```
2
```

In [2]:
```
2-1
```

Out[2]:
```
1
```

In [3]:
```
3*4
```

Out[3]:
```
12
```

In [4]:
```
8 / 4 # Division
```

Out[4]:
```
2.0
```

In [5]:
```
8 / 5 #float division
```

Out[5]:
```
1.6
```

In [6]:
```
8/4 ## float division
```

Out[6]:
```
2.0
```

In [7]:
```
8 // 4 #integer division
```

Out[7]:
```
2
```

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

Out[8]:
```
10
```

In [9]:
```
8 + 8 - #syntax error:
```

```
  Input In [9]
    8 + 8 - #syntax error:
            ^
SyntaxError: invalid syntax
```

In [10]:
```
5 + 5 * 5
```

Out[10]:
```
30
```

In [11]:
```
(5 + 5) * 5 # BODMAS (Bracket || Oders || Divide || Multiply || Add || Substact)
```

Out[11]:
```
50
```

In [12]:
```
2 * 2 * 2 * 2 * 2 # exponentaion
```

Out[12]:
```
32
```

In [13]:
```
2 * 5
```

Out[13]:
```
10
```

In [14]:
```
2 ** 5
```

Out[14]:
```
32
```

In [15]:
```
15 / 3
```

Out[15]:
```
5.0
```

In [16]:
```
10 // 3
```

Out[16]:
```
3
```

In [17]:
```
15 % 2 # Modulus
```

Out[17]:
```
1
```

In [20]:
```
10 % 2
```

Out[20]:
```
0
```

In [21]:
```
15 %% 2
```

```
  Input In [21]
    15 %% 2
        ^
SyntaxError: invalid syntax
```

In [22]:
```
3 + 'nit'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [22], in <cell line: 1>()
----> 1 3 + 'nit'

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

In [23]:
```
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 [24]:
```
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 [25]:
```
type(c)
```

Out[25]:
```
str
```

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

In [26]:
```
'Naresh IT'
```

Out[26]:
```
'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 [27]:
```
print('Max it')
```

```
Max it
```

In [28]:
```
"max it technology"
```

Out[28]:
```
'max it technology'
```

In [29]:
```
s1 = 'max it technology'
s1
```

Out[29]:
```
'max it technology'
```

In [30]:
```
a = 2
b = 3
a + b
```

Out[30]:
```
5
```

In [31]:
```
c = a + b
c
```

Out[31]:
```
5
```

In [32]:
```
a = 3
b = 'hi'
type(b)
```

Out[32]:
```
str
```

In [33]:
```
print('max it's"Technology"') # \ has some special meaning to ignore the error
```

```
  Input In [33]
    print('max it's"Technology"') # \ has some special meaning to ignore the error
                  ^
SyntaxError: invalid syntax
```

In [34]:
```
print('max it\'s"Technology"') #\ has some special meaning to ignore the error
```

```
max it's"Technology"
```

In [35]:
```
print('max it',  'Technology')
```

```
max it Technology
```

In [36]:
```
print("max it',  'Technology")
```

```
max it',  'Technology
```

In [37]:
```
# print the nit 2 times
'nit' + ' nit'
```

Out[37]:
```
'nit nit'
```

In [38]:
```
'nit'  ' nit'
```

Out[38]:
```
'nit nit'
```

In [39]:
```
#5 time print
5 * 'nit'
```

Out[39]:
```
'nitnitnitnitnit'
```

In [40]:
```
5*' nit' # soace between words
```

Out[40]:
```
' nit nit nit nit nit'
```

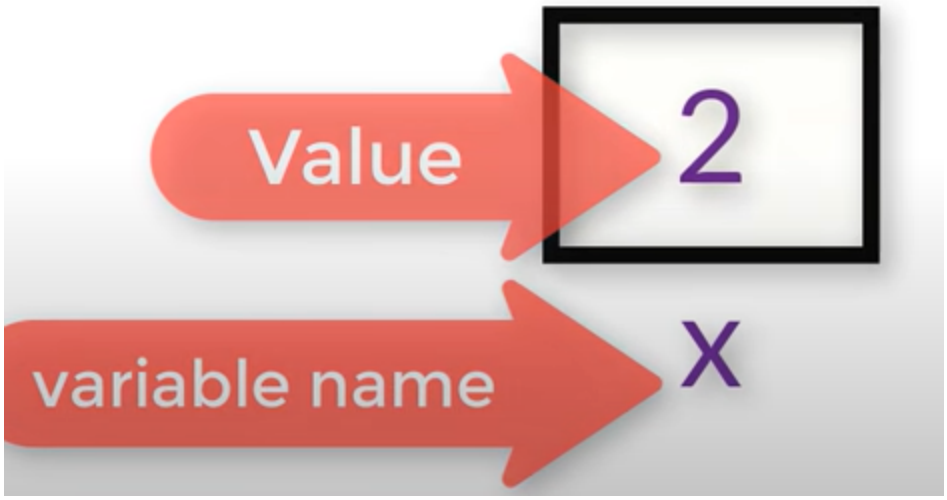
In [41]:
```
print('c:\nit') #\n -- new line
```

```
c:
it
```

In [42]:
```
print(r'c:\nit') #raw string
```

```
c:\nit
```

variable || identifier || object



In [43]:
```
2
```

Out[43]:
```
2
```

In [44]:
```
x = 2 #x is variable/identifier/objec, 2 is the value
x
```

Out[44]:
```
2
```

In [45]:
```
x + 3
```

Out[45]:
```
5
```

In [46]:
```
y = 3
y
```

Out[46]:
```
3
```

In [47]:
```
x + y
```

Out[47]:
```
5
```

In [48]:
```
x = 9
x
```

Out[48]:
```
9
```

In [49]:
```
x + y
```

Out[49]:
```
12
```

In [50]:
```
x + 10
```

Out[50]:
```
19
```

In [51]:
```
_ + y # _ understand the previous result of the
```

Out[51]:
```
22
```

In [52]:
```
_ + y
```

Out[52]:
```
25
```

In [53]:
```
_ + y
```

Out[53]:
```
28
```

In [ ]:
```
_ + y
```

In [ ]:
```
y
```

In [54]:
```
_ + y
```

Out[54]:
```
31
```

In [55]:
```
_ + y
```

Out[55]:
```
34
```

In [56]:
```
_ + y
```

Out[56]:
```
37
```

In [57]:
```
# string variable
name = 'mit'
```

In [58]:
```
name
```

Out[58]:
```
'mit'
```

In [59]:
```
name + 'technology'
```

Out[59]:
```
'mittechnology'
```

In [ ]:
```
name + ' technology'
```

In [60]:
```
name 'technology'
```

```
  Input In [60]
    name 'technology'
         ^
SyntaxError: invalid syntax
```

In [61]:
```
name
```

Out[61]:
```
'mit'
```

In [62]:
```
len(name)
```

Out[62]:
```
3
```

In [63]:
```
name[0] #python index begins with 0
```

Out[63]:
```
'm'
```

In [64]:
```
name[5]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Input In [64], in <cell line: 1>()
----> 1 name[5]

IndexError: string index out of range
```

In [65]:
```
name[7]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Input In [65], in <cell line: 1>()
----> 1 name[7]

IndexError: string index out of range
```

In [66]:
```
name[-1]
```

Out[66]:
```
't'
```

In [67]:
```
name[-2]
```

Out[67]:
```
'i'
```

In [69]:
```
name[-6]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Input In [69], in <cell line: 1>()
----> 1 name[-6]

IndexError: string index out of range
```

slicing

In [70]:
```
name
```

Out[70]:
```
'mit'
```

In [71]:
```
name[0:1] #to print 2 character
```

Out[71]:
```
'm'
```

In [72]:
```
name[0:2]
```

Out[72]:
```
'mi'
```

In [73]:
```
name[1:4]
```

Out[73]:
```
'it'
```

In [74]:
```
name[1:]
```

Out[74]:
```
'it'
```

In [75]:
```
name[:4]
```

Out[75]:
```
'mit'
```

In [ ]:
```
name[3:9]
```

In [76]:
```
name
```

Out[76]:
```
'mit'
```

In [77]:
```
name1 = 'fine' # change the string fine to dine
name1
```

Out[77]:
```
'fine'
```

In [78]:
```
name1[0:1]
```

Out[78]:
```
'f'
```

In [79]:
```
name1[0:1] = 'd' # i want to change 1st character of naresh (n) - t
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [79], in <cell line: 1>()
----> 1 name1[0:1] = 'd'

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

In [80]:
```
name1[0] = 'd' #strings in python are immutable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [80], in <cell line: 1>()
----> 1 name1[0] = 'd'

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

In [81]:
```
name1
```

Out[81]:
```
'fine'
```

In [82]:
```
name1[1:]
```

Out[82]:
```
'ine'
```

In [83]:
```
'd' + name1[1:] #i want to change fine to dine
```

Out[83]:
```
'dine'
```

In [84]:
```
num1.insert(2,'nit') #insert the value as per index values i.e 2nd index we are assign
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [84], in <cell line: 1>()
----> 1 num1.insert(2,'nit')

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

introduce to ID()

In [85]:
```
# variable address
num = 5
id(num)
```

Out[85]:
```
1241317337520
```

In [86]:
```
name = 'nit'
id(name) #Address will be different for both
```

Out[86]:
```
1241369225008
```

In [ ]:
```
a = 10
id(a)
```

In [87]:
```
b = a #thats why python is more memory efficient
```

In [88]:
```
id(b)
```

Out[88]:
```
1241317337456
```

In [89]:
```
id(10)
```

Out[89]:
```
1241317337680
```

In [90]:
```
k = 10
id(k)
```

Out[90]:
```
1241317337680
```

In [91]:
```
a = 20  # as we change the value of a then address will change
id(a)
```

Out[91]:
```
1241317338000
```

In [92]:
```
id(b)
```

Out[92]:
```
1241317337456
```

what ever the variale we assigned the memory and we 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 [93]:
```
PI = 3.14  #in math this is alway constant but python we can chang
PI
```

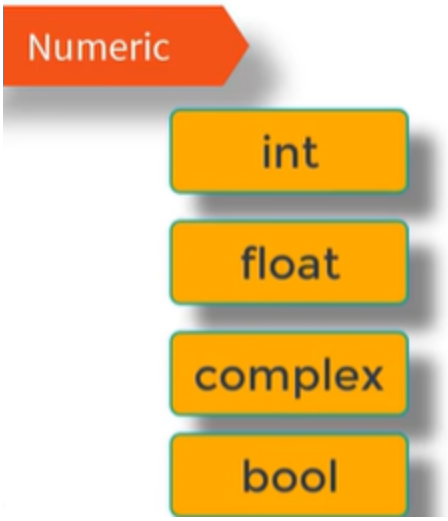
Out[93]:
```
3.14
```

In [94]:
```
PI = 3.15
PI
```

Out[94]:
```
3.15
```

In [95]:
```
type(PI)
```

Out[95]:
```
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 [96]:
```
x1, y1 = 10, 5
```

In [97]:
```
#x1 ^ y1
```

In [98]:
```
x1  + y1
```

Out[98]:
```
15
```

In [99]:
```
x1 - y1
```

Out[99]:
```
5
```

In [100…
```
x1 * y1
```

Out[100]:
```
50
```

In [101…
```
x1 / y1
```

Out[101]:
```
2.0
```

In [102…
```
x1 // y1
```

Out[102]:
```
2
```

In [103…
```
x1 % y1
```

Out[103]:
```
0
```

In [104…
```
x1 ** y1
```

Out[104]:
```
100000
```

In [105…
```
x2 = 3
y2 = 2
x2 ** y2
```

Out[105]:
```
9
```

Assignment operator

In [106…
```
x = 2
```

In [107…
```
x = x + 2 # if you want to increment by 2
```

In [108…
```
x
```

Out[108]:
```
4
```

In [109…
```
x += 2
x
```

Out[109]:
```
6
```

In [110…
```
x += 2
x
```

Out[110]:
```
8
```

In [111…
```
x *= 2
```

In [112…
```
x
```

Out[112]:
```
16
```

In [113…
```
x -= 2
```

In [114…
```
x
```

Out[114]:
```
14
```

In [115…
```
x /= 2
x
```

Out[115]:
```
7.0
```

In [116…
```
x //= 2
x
```

Out[116]:
```
3.0
```

In [117…
```
a, b = 5,6 # you can assigned variable in one line as well
```

In [118…
```
a
```

Out[118]:
```
5
```

In [119…
```
b
```

Out[119]:
```
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 [120…
```
n = 7 #negattion
n
```

Out[120]:
```
7
```

In [121…
```
m = -(n)
m
```

Out[121]:
```
-7
```

In [122…
```
n
```

Out[122]:
```
7
```

In [123…
```
-n
```

Out[123]:
```
-7
```

Relational operator

we are using this operator for comparing

In [124…
```
a = 5
b = 6
```

In [125…
```
a<b
```

Out[125]:
```
True
```

In [126…
```
a>b
```

Out[126]:
```
False
```

In [127…
```
# a = b # we cannot use = operatro that means it is assigning
```

In [128…
```
a == b
```

Out[128]:
```
False
```

In [129…
```
a != b
```

Out[129]:
```
True
```

In [130…
```
# hear if i change b = 6
b = 5
```

In [131…
```
a == b
```

Out[131]:
```
True
```

In [132…
```
a
```

Out[132]:
```
5
```

In [133…
```
b
```

Out[133]:
```
5
```

In [134…
```
a >= b
```

Out[134]:
```
True
```

In [135…
```
a <= b
```

Out[135]: True

In [136…
```
a < b
```

Out[136]: False

In [137…
```
a>b
```

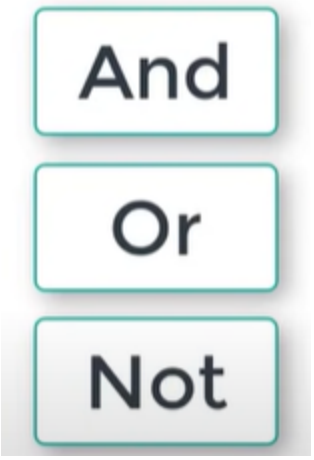
Out[137]: False

In [138…
```
b = 7
```

In [139…
```
a != b
```

Out[139]: True

LOGICAL OPERATOR



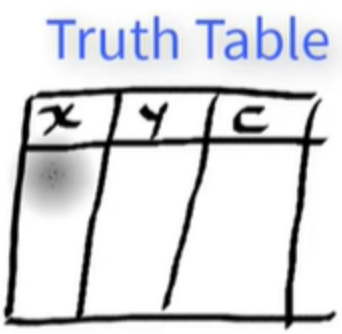
- logical operator you need to understand about true & false table
- 3 importand part of logical operator is --> AND, OR, NOT

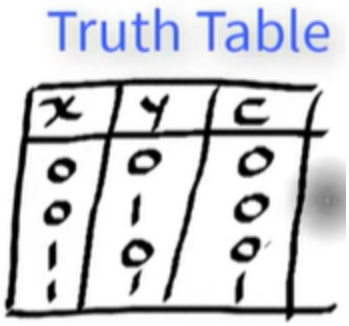
- lets understand the truth table:- in truth table you can represent (true-1 & false means- 0)

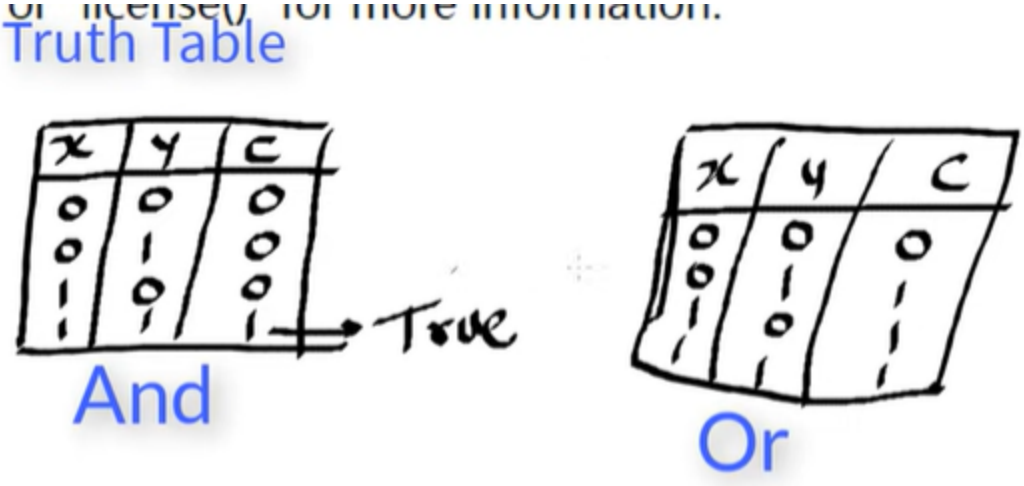
4

8 and b < 5

8 and b<2







In [140…

```
a = 5
b = 4
```

In [141…
```
a < 8 and b < 5 #refer to the truth table
```
Out[141]:
```
True
```

In [142…
```
a < 8 and b < 2
```
Out[142]:
```
False
```

In [143…
```
a < 8 or b < 2
```
Out[143]:
```
True
```

In [144…
```
a>8 or b<2
```
Out[144]:
```
False
```

In [145…
```
x = False
x
```
Out[145]:
```
False
```

In [146…
```
not x  # you can reverse the operation
```
Out[146]:
```
True
```

In [ ]: