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. we will learn pyton from start to end || basic to expert. if you are not done programm then th0at is totally fine. I will explain from starting from scratch. python software - pycharm || vs code || jupyter || spyder

PYTHON INTERPRETTER

IDE (INTEGRATED DEVELOPMENT ENVIRONMENT)

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".

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 & 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\kdata\AppData\Local\Programs\Python\Python311\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 # subtraction
```

Out[2]: 1

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

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 # simple math calculation
```

Out[8]: 10

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

```
  Cell In[9], line 1
    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 || Subst
```

Out[11]: 50

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

Out[12]: 32

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

Out[13]: 32

In [14]:
```
15 / 3
```

Out[14]: 5.0

In [15]:
```
10 // 3
```

Out[15]: 3

In [16]:
```
14 % 2 # Modulus
```

Out[16]: 0

In [18]:
```
15 %% 2 # syntax error
```

```
  Cell In[18], line 1
    15 %% 2 # syntax error
       ^
SyntaxError: invalid syntax
```

In [19]:
```
a,b,c,d,e = 15, 7.8, 'nit', 8+9j, True # here a,b,c,d are variable all value

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

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

In [20]:
```
print(type(a)) # here we chaeck all the variable type using type() function
print(type(b))
print(type(c))
print(type(d))
print(type(e))
```

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

In [21]:
```
type(c) # it give type of c variable which is string
```

Out[21]: str

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

In [23]:
```
'Naresh IT' # string are written in single qouet and double quote also in tr
```

Out[23]: 'Naresh IT'

In [24]:
```
"Naresh it"
```

Out[24]: 'Naresh it'

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

Out[26]: 'Naresh it'

In [27]:
```
"""Naresh it"""
```

Out[27]: '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 [29]:
```
print('naresh it') # if you want to exclude the single qouet use print funct
```

```
naresh it
```

In [30]:
```
"max it technology"
```

Out[30]:
```
'max it technology'
```

In [31]:
```
s1 = 'naresh it technology'
s1
```

Out[31]:
```
'naresh it technology'
```

In [32]:
```
# some basic operation using python
a = 2
b = 3

a + b
```

Out[32]:
```
5
```

In [33]:
```
c = a + b
c
```

Out[33]:
```
5
```

In [34]:
```
a = 3
b = 'hi'
type(b) # string type
```

Out[34]:
```
str
```

In [36]:
```
a + b # error cannot concante with int with string
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[36], line 1
----> 1 a + b # error cannot concante with int with string

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

In [37]:
```
print('naresh it's 'Technology')
```

```
  Cell In[37], line 1
    print('naresh it's 'Technology')
                ^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

In [38]:
```
print('naresh it\'s"Technology"') #\ has some special meaning to ignore the
```

```
naresh it's"Technology"
```

In [39]:
```
print('naresh it',  'Technology')
```

```
naresh it Technology
```

In [40]:
```
print("naresh it',  'Technology")
```

```
naresh it',  'Technology
```

In [41]:
```
# print the nit 2 times
'nit' + ' nit'
```

Out[41]:
```
'nit nit'
```

In [42]:
```
'nit'  ' nit'
```

Out[42]:
```
'nit nit'
```

In [43]:
```
#5 time print
5 * 'nit'
```

Out[43]:
```
'nitnitnitnitnit'
```

In [44]:
```
5 *' nit ' # soace between words
```

Out[44]:
```
' nit  nit  nit  nit  nit '
```

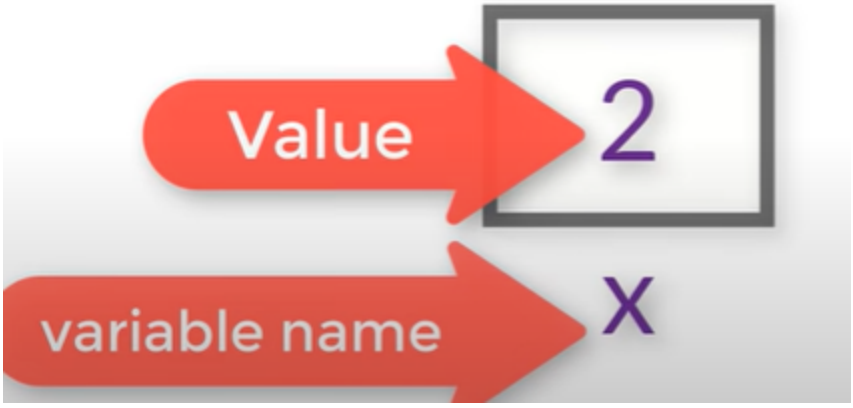
In [45]:
```
print('c:\nit') #\n -- new line
```

```
c:
it
```

In [46]:
```
print(r'c:\nit') #raw string
```

```
c:\nit
```

variable || identifier || object



In [47]:
```
2
```

Out[47]: 2

In [48]:
```
x = 2 #x is variable/identifier/objec, 2 is the value
x
```

Out[48]: 2

In [49]:
```
x + 3
```

Out[49]: 5

In [50]:
```
y = 3
y
```

Out[50]: 3

In [51]:
```
x + y
```

Out[51]: 5

In [52]:
```
x = 9
x
```

Out[52]: 9

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

Out[53]: 12

In [54]:
```
x + 10
```

Out[54]: 19

In [55]:
```
y
```

Out[55]: 3

In [56]:
```
_ + y # _ understand the previous result of the
```

Out[56]: 6

In [57]:
```
y
```

Out[57]: 3

In [58]:
```
_ + y
```

Out[58]: 6

In [59]:
```
_ + y # previous value is 6
```

Out[59]: 9

In [60]:
```
_ + y # previous value 9 y value 3
```

Out[60]: 12

In [61]:
```
# string variable
name = 'mit'
```

In [62]:
```
name
```

Out[62]: 'mit'

In [64]:
```
name + 'technology' # concatination of string
```

Out[64]: 'mittechnology'

In [65]:
```
name + ' technology'
```

Out[65]: 'mit technology'

In [67]:
```
name 'technology' # error rise because of no operation done with the string
```

```
  Cell In[67], line 1
    name 'technology' # error rise because of no operation done with the str
ing
         ^
SyntaxError: invalid syntax
```

In [68]:
```
name
```

Out[68]: 'mit'

In [69]:
```
len(name) # find legth of the variable
```

Out[69]: 3

In [70]:
```
name[0] #python index begins with 0
```

Out[70]: 'm'

In [72]:
```
name[5] # there is no 5 index in the variable name
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[72], line 1
----> 1 name[5] # there is no 5 index in the variable name

IndexError: string index out of range
```

In [73]:
```
name[7]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[73], line 1
----> 1 name[7]

IndexError: string index out of range
```

In [74]:
```
name[-1] # for reverce count index start with -1
```

Out[74]: 't'

In [75]:
```
name[-2]
```

Out[75]: 'i'

In [76]:
```
name[-6]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[76], line 1
----> 1 name[-6]

IndexError: string index out of range
```

slicing

In [79]:
```
name
```

Out[79]: 'mit'

In [80]:
```
name[0:1] #to print 1 character
```

Out[80]: 'm'

In [81]:
```
name[0:2] # start index is 0 end index is 1 , 2nd index excluded
```

Out[81]: 'mi'

In [82]:
```
name[1:4] # start index is 1 end index is 3 , 4th index excluded
```

Out[82]: 'it'

In [83]:
```
name
```

Out[83]: 'mit'

In [84]:
```
name[1:] # starting from 1 index to end index
```

Out[84]: 'it'

In [85]:
```
name[:4] # starting from 0 index to 3 index 4th will excluded
```

Out[85]: 'mit'

In [86]:
```
name[3:9]
```

Out[86]: ''

In [87]:
```
# as string is immutable lets change the character inside the string
name1 = 'fine'
name1
```

Out[87]: 'fine'

In [88]:
```
name1[0:1]
```

Out[88]: 'f'

In [89]:
```
name1[0:1] = 'd' # i want to change 1st character of naresh (n) - t
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[89], line 1
----> 1 name1[0:1] = 'd' # i want to change 1st character of naresh (n) - t

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

In [90]:
```
name1
```

Out[90]: 'fine'

In [91]:
```
name1[0] = 'd' #strings in python are immutable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[91], line 1
----> 1 name1[0] = 'd' #strings in python are immutable

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

In [92]:
```
name1
```

Out[92]:
```
'fine'
```

In [93]:
```
name1[1:]
```

Out[93]:
```
'ine'
```

In [94]:
```
'd' + name1[1:] #i want to change fine to dine
```

Out[94]:
```
'dine'
```

In [95]:
```
len(name1) #python inbuild function
```

Out[95]:
```
4
```

List

- syntax --> list_name=[val1,val2,val3....]

In [96]:
```
# LIST LIST LIST
nums = [10,20,30]
nums
```

Out[96]:
```
[10, 20, 30]
```

In [97]:
```
nums[0] # retrive 0 index from the list
```

Out[97]:
```
10
```

In [98]:
```
nums[-1] # retrive last index of the list
```

Out[98]:
```
30
```

In [99]:
```
nums[1:] # starting index id 1 last index is length of the list
```

Out[99]:
```
[20, 30]
```

In [100…
```
nums[:1] # starting index is 0
```

Out[100…
```
[10]
```

In [101…
```
num1 = ['hi', 'hallo'] # list value with string
```

In [102…
```
num1
```

Out[102…
```
['hi', 'hallo']
```

In [103…
```
num2 = ['hi', 8.9, 34] # we can assign multiple value
num2
```

Out[103… ['hi', 8.9, 34]

In [104…
```
# can we have 2 list together
num3 = [nums, num1] # take 2 list into a single list
```

In [105…
```
num3
```

Out[105… [[10, 20, 30], ['hi', 'hallo']]

In [106…
```
num4 = [nums, num1, num2]
```

In [107…
```
num4
```

Out[107… [[10, 20, 30], ['hi', 'hallo'], ['hi', 8.9, 34]]

In [108…
```
nums
```

Out[108… [10, 20, 30]

In [109…
```
nums.append(45) # append is a inbuilt function in list to add one element at
```

In [110…
```
nums
```

Out[110… [10, 20, 30, 45]

In [111…
```
nums.remove(45) # remove is inbuilt function take value that already inside
```

In [112…
```
nums
```

Out[112… [10, 20, 30]

In [113…
```
nums.pop(1) # pop function use to remove item by help of the index
```

Out[113… 20

In [114…
```
nums
```

Out[114… [10, 30]

In [115…
```
nums.pop() #if you dont assign the index element then it will consider by de
```

Out[115… 30

In [116…
```
nums
```

Out[116… [10]

In [117…
```
num1
```

Out[117… ['hi', 'hallo']

In [118…
```
num1.insert(2,'nit') #insert the value as per index values i.e 2nd index we
```

In [119…
```
num1
```

Out[119…
```
['hi', 'hallo', 'nit']
```

In [120…
```
num1.insert(0, 1) # here at 0 index we adding 1
```

In [121…
```
num1
```

Out[121…
```
[1, 'hi', 'hallo', 'nit']
```

In [122…
```
#if you want to delate multiple value
num2
```

Out[122…
```
['hi', 8.9, 34]
```

In [125…
```
del num2[2:] # it delete from 2 index onwards all the value
```

In [126…
```
num2
```

Out[126…
```
['hi', 8.9]
```

In [127…
```
# if you need to add multiple values
num2.extend([29,15,20])
```

In [128…
```
num2
```

Out[128…
```
['hi', 8.9, 29, 15, 20]
```

In [129…
```
num3
```

Out[129…
```
[[10], [1, 'hi', 'hallo', 'nit']]
```

In [130…
```
num3.extend(['a', 5, 6.7]) # adding in num3 all the values
```

In [131…
```
num3
```

Out[131…
```
[[10], [1, 'hi', 'hallo', 'nit'], 'a', 5, 6.7]
```

In [132…
```
nums
```

Out[132…
```
[10]
```

In [133…
```
min(nums) #inbuild function give minimum value fron the list
```

Out[133…
```
10
```

In [134…
```
max(nums) #inbuild function gives the maximum value from the list
```

Out[134…
```
10
```

In [135…
```
num1
```

Out[135…
```
[1, 'hi', 'hallo', 'nit']
```

In [137…
```
min(num1) # to operate the mathmatical function there should be the integer
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[137], line 1
----> 1 min(num1) # to operate the mathmatical function there should be the
integer or float value

TypeError: '<' not supported between instances of 'str' and 'int'
```

In [138…
```
sum(nums) #inbuild function to add all the element
```

Out[138…
```
10
```

In [141…
```
nums.append(23)
nums.append(3)
nums.append(43)
nums.append(76)
nums.append(99)
nums
```

Out[141…
```
[10, 23, 3, 43, 76, 99]
```

In [142…
```
nums.sort() #sort method use to sort in ascending order
```

In [143…
```
nums
```

Out[143…
```
[3, 10, 23, 43, 76, 99]
```

Tuple

- syntax --> tuple_name=(value1,value2,value3...)

In [144…
```
# TUPLE TUPLE TUPLE
tup = (15,25, 35)
tup
```

Out[144…
```
(15, 25, 35)
```

In [145…
```
tup[0] # it give 0 index from tuple
```

Out[145…
```
15
```

In [147…
```
tup[0] = 10 # update 0th index to 10not allowed because tuple immutable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[147], line 1
----> 1 tup[0] = 10 # update 0th index to 10not allowed because tuple immuta
ble

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

In [148…
```
tup.count(15) # count gives the occurance of the value in the tuple
```

Out[148… 1

In [149…
```
tup.index(25) # it gives the index of the value
```

Out[149… 1

as we are unable to change any value or parameter in tuple so iteration very faster in tuple compare to list

SET

- syntax --> set-name={val1, val2,val3...}

In [150…
```
# SET SET SET
S = {}
```

In [151…
```
type(S) # it return dict
```

Out[151… dict

In [152…
```
# creat empty set
S = set()
```

In [153…
```
type(S)
```

Out[153… set

In [154…
```
s1 = {21,6,34,58,5}
```

In [155…
```
s1
```

Out[155… {5, 6, 21, 34, 58}

In [156…
```
s3= {50,35,53,'nit', 53}
```

In [157…
```
s3
```

Out[157… {35, 50, 53, 'nit'}

In [158…
```
s1[1] #as we dont have proper sequencing thats why indexing not subscriptabl
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[158], line 1
----> 1 s1[1] #as we dont have proper sequencing thats why indexing not subs
criptable

TypeError: 'set' object is not subscriptable
```

DICTIONARY

- syntax--> dict_name={key1:val1, key2:val2,key3:val3...}

In [159…
```
# DICTIONARY DICTIONARY DICTIONARY
data = {1:'apple', 2:'banana',4:'orange'}
data
```

Out[159… `{1: 'apple', 2: 'banana', 4: 'orange'}`

In [160…
```
data[4]
```

Out[160… `'orange'`

In [162…
```
data[3] # we cannot access the value by index only it can possible by help c
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[162], line 1
----> 1 data[3] # we cannot access the value by index only it can possible b
y help of keys

KeyError: 3
```

In [164…
```
data.get(2) # if want to access item then use get()
```

Out[164… `'banana'`

In [165…
```
data.get(3)
```

In [166…
```
print(data.get(3))
```

```
None
```

In [169…
```
data.get(1,'Not Fount') # first it check is there 1 keys values presenet or
```

Out[169… `'apple'`

In [170…
```
data.get(3,'Not Found')
```

Out[170… `'Not Found'`

In [171…
```
data[5] = 'five' # here we updating the key value
```

In [172…
```
data
```

Out[172…
```
{1: 'apple', 2: 'banana', 4: 'orange', 5: 'five'}
```

In [173…
```
del data [5] # delete any value with the help of keys
```

In [174…
```
data
```

Out[174…
```
{1: 'apple', 2: 'banana', 4: 'orange'}
```

In [175…
```
#list in the dictionary
prog = {'python':['vscode', 'pycharm'], 'machine learning' : 'sklearn', 'dat
```

In [176…
```
prog
```

Out[176…
```
{'python': ['vscode', 'pycharm'],
 'machine learning': 'sklearn',
 'datascience': ['jupyter', 'spyder']}
```

In [177…
```
prog['python'] # retrive python keys value
```

Out[177…
```
['vscode', 'pycharm']
```

In [178…
```
prog['machine learning'] # here for machine learning
```

Out[178…
```
'sklearn'
```

In [179…
```
prog['datascience'] # here for data science
```

Out[179…
```
['jupyter', 'spyder']
```

How to create environment variable

- STEPS TO SET UP EXECUTE PYTHON IN SYSTEM CMD (TO CREATE ENVIRONMENT VARIABLE)
- Open cmd # python (You will get error when you execute 1st time)
- search with environment variable - system variable: (C:\Users\kdata\AppData\Local\Microsoft\WindowsApps)
- restart the cmd & type python in cmd it will work now

to find help

STEPS TO FIND HELP OPTION --> 1- help() | 2- topics | 3- search as per requirments | 4- quit if you want help on any command then help(list) || help(tuple)

In [180…
```
help() # help() use to take any help
```

```
Welcome to Python 3.12's help utility! If this is your first time using
Python, you should definitely check out the tutorial at
https://docs.python.org/3.12/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To get a list of available
modules, keywords, symbols, or topics, enter "modules", "keywords",
"symbols", or "topics".

Each module also comes with a one-line summary of what it does; to list
the modules whose name or summary contain a given string such as "spam",
enter "modules spam".

To quit this help utility and return to the interpreter,
enter "q" or "quit".

You are now leaving help and returning to the Python interpreter.
If you want to ask for help on a particular object directly from the
interpreter, you can type "help(object)".  Executing "help('string')"
has the same effect as typing a particular string at the help> prompt.
```

In [182…
```
help(list) # it give alll details of the list
```

```
Help on class list in module builtins:

class list(object)
 |  list(iterable=(), /)
 |
 |  Built-in mutable sequence.
 |
 |  If no argument is given, the constructor creates a new empty list.
 |  The argument must be an iterable if specified.
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return bool(key in self).
 |
 |  __delitem__(self, key, /)
 |      Delete self[key].
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(self, index, /)
 |      Return self[index].
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __iadd__(self, value, /)
 |      Implement self+=value.
 |
 |  __imul__(self, value, /)
 |      Implement self*=value.
 |
 |  __init__(self, /, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __len__(self, /)
 |      Return len(self).
 |
 |  __lt__(self, value, /)
 |      Return self<value.
```

```
|
|  __mul__(self, value, /)
|      Return self*value.
|
|  __ne__(self, value, /)
|      Return self!=value.
|
|  __repr__(self, /)
|      Return repr(self).
|
|  __reversed__(self, /)
|      Return a reverse iterator over the list.
|
|  __rmul__(self, value, /)
|      Return value*self.
|
|  __setitem__(self, key, value, /)
|      Set self[key] to value.
|
|  __sizeof__(self, /)
|      Return the size of the list in memory, in bytes.
|
|  append(self, object, /)
|      Append object to the end of the list.
|
|  clear(self, /)
|      Remove all items from list.
|
|  copy(self, /)
|      Return a shallow copy of the list.
|
|  count(self, value, /)
|      Return number of occurrences of value.
|
|  extend(self, iterable, /)
|      Extend list by appending elements from the iterable.
|
|  index(self, value, start=0, stop=9223372036854775807, /)
|      Return first index of value.
|
|      Raises ValueError if the value is not present.
|
|  insert(self, index, object, /)
|      Insert object before index.
|
|  pop(self, index=-1, /)
|      Remove and return item at index (default last).
|
|      Raises IndexError if list is empty or index is out of range.
|
|  remove(self, value, /)
|      Remove first occurrence of value.
|
|      Raises ValueError if the value is not present.
|
|  reverse(self, /)
```

```
 |      Reverse *IN PLACE*.
 |
 |  sort(self, /, *, key=None, reverse=False)
 |      Sort the list in ascending order and return None.
 |
 |      The sort is in-place (i.e. the list itself is modified) and stable
(i.e. the
 |      order of two equal elements is maintained).
 |
 |      If a key function is given, apply it once to each list item and sort
them,
 |      ascending or descending, according to their function values.
 |
 |      The reverse flag can be set to sort in descending order.
 |
 |  ----------------------------------------------------------------------
 |  Class methods defined here:
 |
 |  __class_getitem__(...)
 |      See PEP 585
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs)
 |      Create and return a new object.  See help(type) for accurate signatu
re.
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  __hash__ = None
```

In [183…

```
help(tuple) # it gives all details about the tuple
```

```
Help on class tuple in module builtins:

class tuple(object)
 |  tuple(iterable=(), /)
 |
 |  Built-in immutable sequence.
 |
 |  If no argument is given, the constructor returns an empty tuple.
 |  If iterable is specified the tuple is initialized from iterable's items.
 |
 |  If the argument is a tuple, the return value is the same object.
 |
 |  Built-in subclasses:
 |      asyncgen_hooks
 |      UnraisableHookArgs
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return bool(key in self).
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(self, key, /)
 |      Return self[key].
 |
 |  __getnewargs__(self, /)
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __hash__(self, /)
 |      Return hash(self).
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __len__(self, /)
 |      Return len(self).
 |
 |  __lt__(self, value, /)
 |      Return self<value.
 |
```

```
|  __mul__(self, value, /)
|      Return self*value.
|
|  __ne__(self, value, /)
|      Return self!=value.
|
|  __repr__(self, /)
|      Return repr(self).
|
|  __rmul__(self, value, /)
|      Return value*self.
|
|  count(self, value, /)
|      Return number of occurrences of value.
|
|  index(self, value, start=0, stop=9223372036854775807, /)
|      Return first index of value.
|
|      Raises ValueError if the value is not present.
|
|  ----------------------------------------------------------------------
|  Class methods defined here:
|
|  __class_getitem__(...)
|      See PEP 585
|
|  ----------------------------------------------------------------------
|  Static methods defined here:
|
|  __new__(*args, **kwargs)
|      Create and return a new object.  See help(type) for accurate signatu
re.
```

introduce to ID()

- id() use to determine the address

In [184…
```
# variable address
num = 5
id(num)
```

Out[184…
```
140734827608632
```

In [185…
```
name = 'nit'
id(name) #Address will be different for both
```

Out[185…
```
2722937518480
```

In [186…
```
a = 10
id(a)
```

Out[186…
```
140734827608792
```

In [187…
```
b = a #thats why python is more memory efficient
```

In [188…
```
id(b)
```

Out[188… 140734827608792

In [189…
```
id(10)
```

Out[189… 140734827608792

In [190…
```
k = 10
id(k)
```

Out[190… 140734827608792

In [191…
```
a = 20  # as we change the value of a then address will change
id(a)
```

Out[191… 140734827609112

In [192…
```
id(b)
```

Out[192… 140734827608792

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 [193…
```
PI = 3.14  #in math this is alway constant but python we can chang
PI
```

Out[193… 3.14

In [194…
```
PI = 3.18
PI
```

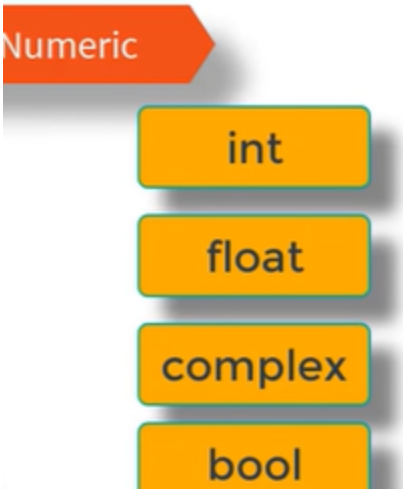
Out[194… 3.18

In [195…
```
type(PI)
```

Out[195… float

DATA TYPES & DATA STRUCTURES-->

1- NUMERIC || 2-LIST || 3-TUPLE || 4-SET || 5-STRING || 6-RANGE || 7-DICTIONARY

1- NUMERIC :- 

INT || FLOAT || COMPLEX || BOOL

In [196…
```
w = 2.5 # float data type
type(w)
```

Out[196…
```
float
```

In [197…
```
a
```

Out[197…
```
20
```

In [198…
```
(a)
```

Out[198…
```
20
```

In [199…
```
w2 = 2 + 3j #so hear j is represent as root of -1
type(w2) # complex data type
```

Out[199…
```
complex
```

In [200…
```
#convert flot to integer
a = 5.6
b = int(a) # conversion from float to int
```

In [201…
```
b
```

Out[201…
```
5
```

In [202…
```
type(b)
```

Out[202…
```
int
```

In [203…
```
type(a)
```

Out[203…
```
float
```

In [204…
```
k = float(b)
```

In [205…
```
k
```

Out[205…
```
5.0
```

In [206…
```
print(a)
print(b)
print(k)
```

```
5.6
5
5.0
```

In [207…
```
k1 = complex(b,k)
```

In [208…
```
print(k1)
type(k1)
```

```
(5+5j)
```

Out[208…
```
complex
```

In [209…
```
b < k
```

Out[209…
```
False
```

In [210…
```
condition = b<k
condition
```

Out[210…
```
False
```

In [212…
```
type(condition) # type is boolean
```
Out[212…
```
bool
```
In [213…
```
int(True)
```
Out[213…
```
1
```
In [214…
```
int(False) # True = 1, False = 0
```
Out[214…
```
0
```
In [219…
```
l = [1,2,3,4] # list type
print(l)
type(l)
```
```
[1, 2, 3, 4]
```
Out[219…
```
list
```
In [220…
```
s = {1,2,3,4} # set type
s
```
Out[220…
```
{1, 2, 3, 4}
```
In [217…
```
type(s)
```
Out[217…
```
set
```
In [221…
```
s1 = {1,2,3,4,4,3,11} #duplicates are not allowed in set
s1
```
Out[221…
```
{1, 2, 3, 4, 11}
```
In [223…
```
t = (10,20,30) # tuple type
t
```
Out[223…
```
(10, 20, 30)
```
In [224…
```
type(t)
```
Out[224…
```
tuple
```
In [225…
```
str = 'nit' #we dont have character in python
type(str)
```
Out[225…
```
str
```
In [226…
```
st = 'n'
type(st)
```
Out[226…
```
str
```

range()

In [227…
```
r = range(0,10) # range function work to print number sequence , here 0 is s
r
```

Out[227…
```
range(0, 10)
```

In [228…
```
type(r)
```

Out[228…
```
range
```

In [229…
```
# if you want to print the range
list(range(0,10))
```

Out[229…
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [230…
```
r1 = list(r)
r1
```

Out[230…
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [232…
```
#if you want to print even number
even_number = list(range(2,10,2)) # here 2 is starting index 10 is end index
even_number
```

Out[232…
```
[2, 4, 6, 8]
```

In [233…
```
d= {1:'one', 2:'two', 3:'three'}
d
```

Out[233…
```
{1: 'one', 2: 'two', 3: 'three'}
```

In [234…
```
type(d)
```

Out[234…
```
dict
```

In [ ]:
```
# print the keys
d.keys()
```

In [235…
```
d.values() # to print all the values from the dict
```

Out[235…
```
dict_values(['one', 'two', 'three'])
```

In [236…
```
# how to get particular value
d[2]
```

Out[236…
```
'two'
```

In [237…
```
# other way to get value as
d.get(2)
```

Out[237… 'two'

operator's in python

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

Arithmetic operator

In [238…
```
x1, y1 = 10, 5 # asign multiple variable to multiple value respectively, her
```

In [239…
```
x1  + y1  # sum of two value
```

Out[239… 15

In [240…
```
x1 - y1 # subtracted value of two number
```

Out[240… 5

In [241…
```
x1 * y1 # multiplication of two value
```

Out[241… 50

In [242…
```
x1 / y1 # division of two value in output we get float number
```

Out[242… 2.0

In [243…
```
x1 // y1 # devision of two value in output get integer value
```
Out[243… 2

In [244…
```
x1 % y1 # it give modul of the two number(remainder)
```
Out[244… 0

In [246…
```
x1 ** y1 # here x1 is  power of  y1
```
Out[246… 100000

In [247…
```
2 ** 3 # 2 power of 3
```
Out[247… 8

Assignment operator

In [248…
```
x =  2 # the value 2 is asign to the variable x
```
In [249…
```
x = x + 2
```
In [250…
```
x
```
Out[250… 4

In [251…
```
x += 2 #here first operation done is adding then asign  to variable
```
In [252…
```
x
```
Out[252… 6

In [253…
```
x += 2 # previous x = 6, then first add 2 the 8 will asign to x
```
In [254…
```
x
```
Out[254… 8

In [255…
```
x *= 2 # first multiply the asign
```
In [256…
```
x
```
Out[256… 16

In [257…
```
x -= 2 # first subtract then asign
```
In [258…
```
x
```
Out[258… 14

In [259…
```
x /= 2 # first devision then asign
```

In [260…
```
x
```

Out[260… 7.0

In [261…
```
a, b = 5,6
```

In [262…
```
a
```

Out[262… 5

In [263…
```
b
```

Out[263… 6

unary operator

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 [264…
```
n = 7 #negattion
```

In [265…
```
m = -(n)
```

In [266…
```
m
```

Out[266… -7

In [267…
```
n
```

Out[267… 7

In [268…
```
-n
```

Out[268… -7

Relational operator

- we are using this operator for comparing.
- it always return value as boolean
- < , > ,<= , >=, ==, !=

In [269…
```
a = 5
b = 7
```

In [270…
```
a == b
```

Out[270… False

In [271…
```
a<b
```

Out[271… True

In [272…
```
a>b
```

Out[272… False

In [273…
```
# a = b # we cannot use = operatro that means it is assigning
```

In [274…
```
a == b
```

Out[274… False

In [275…
```
a = 10
```

In [276…
```
a != b
```

Out[276… True

In [277…
```
# hear if i change b = 6
b = 10
```

In [278…
```
a == b
```

Out[278… True

In [279…
```
a >= b
```

Out[279… True

In [280…
```
a <= b
```

Out[280… True

In [281…
```
a < b
```

Out[281… False

In [282…
```
a>b
```

Out[282… False

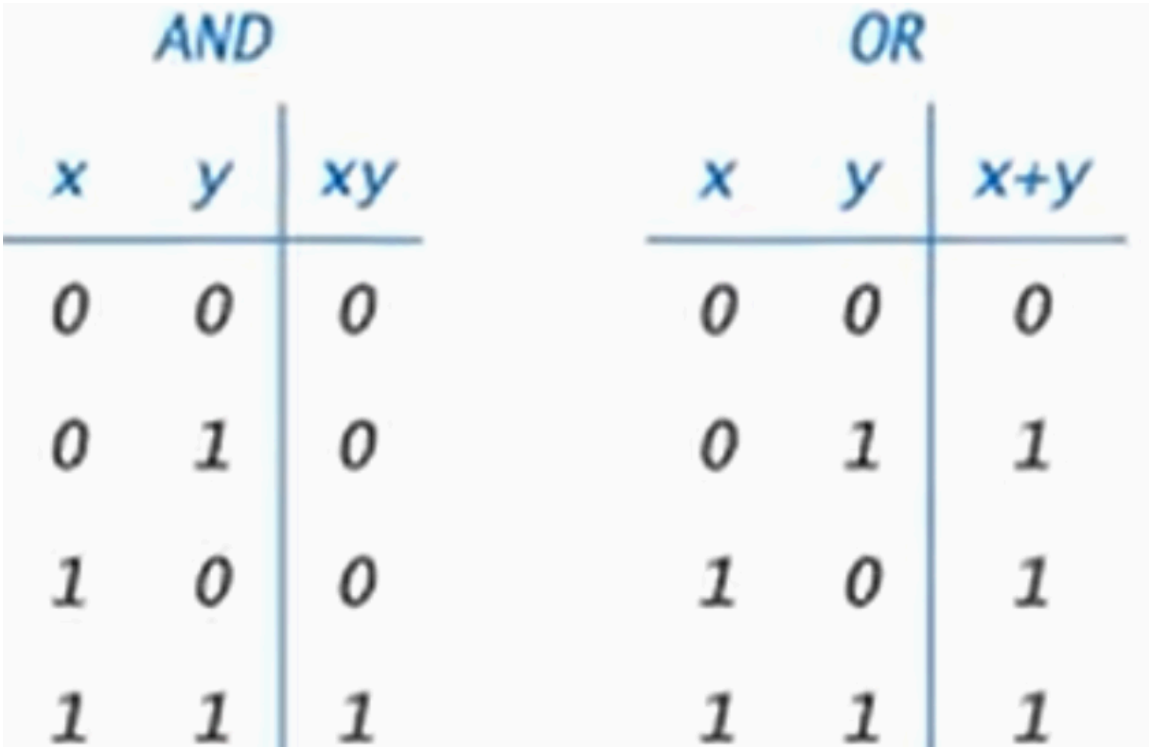
In [283…
```
b = 7
```

In [284…
```
a != b
```

Out[284… True

LOGICAL OPERATOR

AND(), OR, NOT



AND

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

In [285…
```
a = 5
b = 4
```

In [286…
```
a < 8 and b < 5 #if both condition are tru then it true otherwise its not
```

Out[286… True

In [287…
```
a < 8 and b < 2
```

Out[287… False

In [288…
```
a < 8 or b < 2 # if atleast one condition is true then its true other wise r
```

Out[288… True

In [289…
```
a>8 or b<2
```

Out[289… False

In [290…
```
x = False
x
```

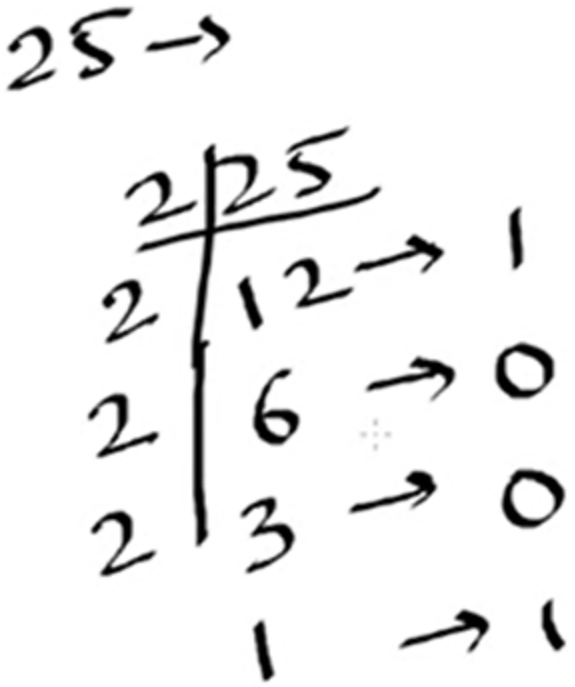
Out[290… False

In [291…
```
not x  # it take only one condition if its true return false and vise-versa
```
Out[291…
```
True
```
In [292…
```
x = not x
x
```
Out[292…
```
True
```
In [293…
```
x
```
Out[293…
```
True
```
In [294…
```
not x
```
Out[294…
```
False
```

Number system coverstion (bit-binary digit)

- binary : base (0-1)(0b) --> please divide 15/2 & count in reverse order
- octal : base (0-7)(0o)
- hexadecimal : base (0-9 & then a-f)(0x)
- when you check ipaddress you will these format --> cmd - ipconfig

In [298…
```
25
```
Out[298…
```
25
```
In [299…
```
bin(25) # it gives the binary numbr of 25
```
Out[299…
```
'0b11001'
```
In [300…
```
0b11001 # it convert the binary number to decimal number,
```
Out[300…
```
25
```



In [301…
```
int(0b11001)
```
Out[301… 25

In [302…
```
bin(35) # it gives the binary of 35
```
Out[302… '0b100011'

In [303…
```
int(0b100011)
```
Out[303… 35

In [304…
```
bin(20)
```
Out[304… '0b10100'

In [305…
```
int(0b10100)
```
Out[305… 20

In [306…
```
0b1111
```
Out[306… 15

In [307…
```
oct(15) # it gives octal of the number 15
```
Out[307… '0o17'

In [308…
```
0o17 # here 17 number convert to decimal
```

Out[308… 15

In [309…
```
hex(9) # it gives hexadecimal value of the numer
```

Out[309… '0x9'

In [310…
```
0xf
```

Out[310… 15

In [311…
```
hex(10)
```

Out[311… '0xa'

In [312…
```
0xa
```

Out[312… 10

In [313…
```
hex(25)
```

Out[313… '0x19'

In [314…
```
0x19
```

Out[314… 25

In [315…
```
0x15
```

Out[315… 21

swap variable in python

(a,b = 5,6) After swap we should get ==> (a, b = 6,5)

In [316…
```
a = 5
b = 6
```

In [317…
```
a = b
b = a
```

In [318…
```
a,b = b,a
```

In [319…
```
print(a)
print(b)
```

```
6
6
```

swaping number using 3rd variable

In [320…
```
# in above scenario we lost the value 5
a1 = 7
b1 = 8
```

In [321…
```
temp = a1
a1 = b1
b1 = temp
```

In [322…
```
print(a1)
print(b1)
```

```
8
7
```

swapimg number without using 3rd variable

In [324…
```
a2 = 5
b2 = 6
```

In [326…
```
#swap variable formulas
a2 = a2 + b2
b2 = a2 - b2
a2 = a2 - b2
```

In [327…
```
print(a2)
print(b2)
```

```
5
6
```

In [328…
```
print(0b101) # 101 is 3 bit
print(0b110) # 110 also 3bit
```

```
5
6
```

In [329…
```
#but when we use a2 + b2 then we get 11 that means we will get 4 bit which i
print(bin(11))
print(0b1011)
```

```
0b1011
11
```

swaping of two number using XOR

In [330…
```
#there is other way to work using swap variable also which is XOR because it
a2 = a2 ^ b2
b2 = a2 ^ b2
a2 = a2 ^ b2
```

In [331…
```
print(a2)
print(b2)
```

```
6
5
```

In [332…
```
print(a2)
print(b2)
```

```
6
5
```

sswaping using multiple variable

In [333…
```
a2 , b2 = b2, a2 # here b2 value asign to a2, and a2 value asign to b2 respe
```

In [334…
```
print(a2)
print(b2)
```

```
5
6
```

BITWISE OPERATOR

- WE HAVE 6 OPERATORS
- COMPLEMENT (~)
- AND (&)
- OR (|)
- XOR (^)
- LEFT SHIFT (<<)
- RIGHT SHIFT (>>)

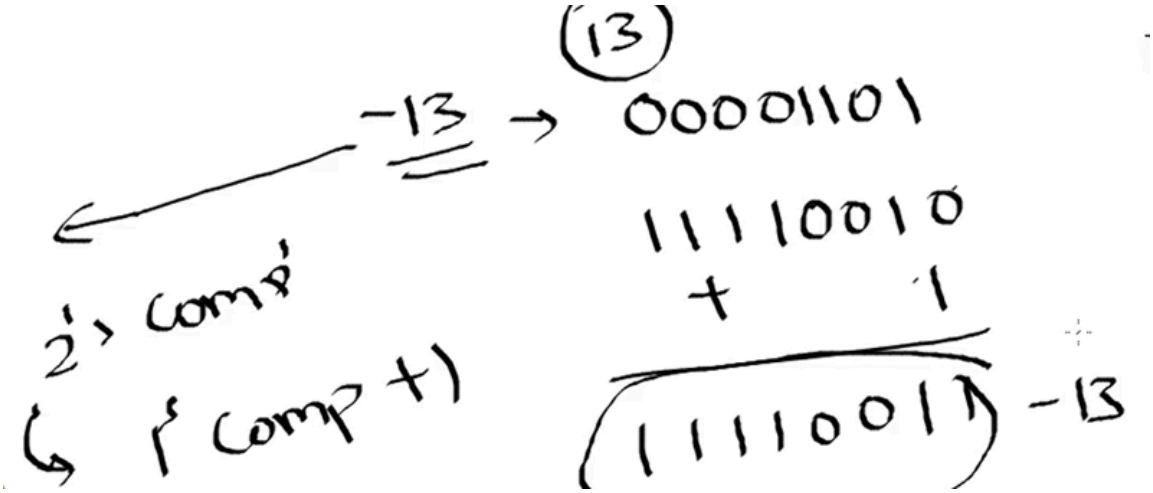
In [ ]:
```
print(bin(12))
print(bin(13))
```

complement --> you will get this key below esc character

12 ==> 1100 || first thing we need to understand what is mean by complement. complement means it will do reverse of the binary format i.e. - ~0 it will give you 1 ~1 it will give 0 12 binary format is 00001100 (complement of ~00001100 reverse the number - 11110011 which is (-13)

but the question is why we got -13 to understand this concept (we have concept of 2's complement 2's complement mean (1's complement + 1) in the system we can store +Ve number but how to store -ve number

lets understand binary form of 13 - 00001101 + 1



In [335…
```
bin(12)
```

Out[335…
```
'0b1100'
```

In [336…
```
0b0011+1
```

Out[336…
```
4
```

In [337…
```
# COMPLEMENT (~) (TILDE  OR TILD)
~12 # why we get -13 . first we understand what is complment means (reversr
```

Out[337…
```
-13
```

In [338…
```
~45
```

Out[338…
```
-46
```

In [339…
```
~6
```

Out[339…
```
-7
```

In [340…
```
~-6
```

Out[340…
```
5
```

In [341…
```
~-1
```

Out[341…
```
0
```

bit wise and operator

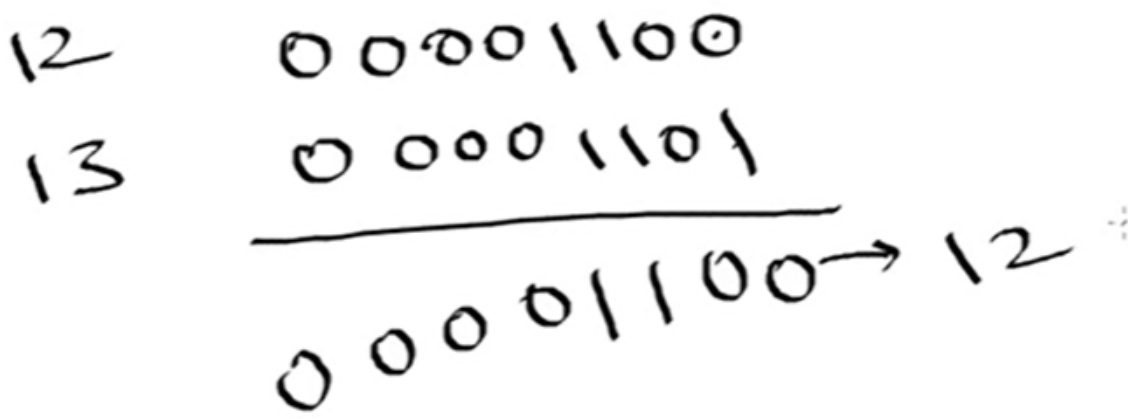
AND - LOGICAL OPERATOR ||| & - BITWISE AND OPERATOR
(we know that 1 & 1 is 1) 12 - 00001100 13 - 00001101 when we are add both then then outut we will get as 12

AND

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1



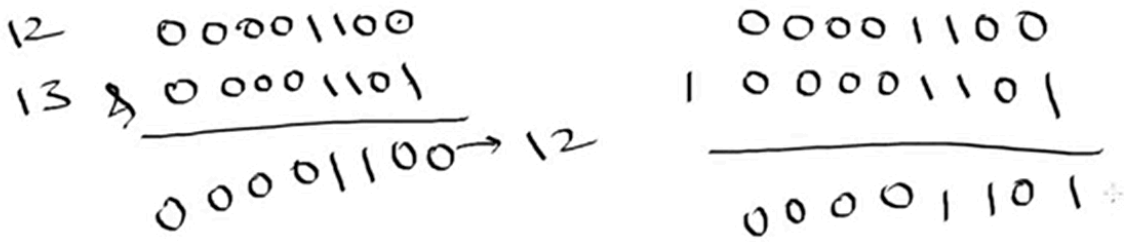
In [342…
```
12 & 13
```
Out[342… 12

In [343…
```
1 & 1
```
Out[343… 1

In [344…
```
1 | 0
```
Out[344… 1

In [345…
```
1 & 0
```
Out[345… 0

In [346…
```
12 | 13
```
Out[346… 13

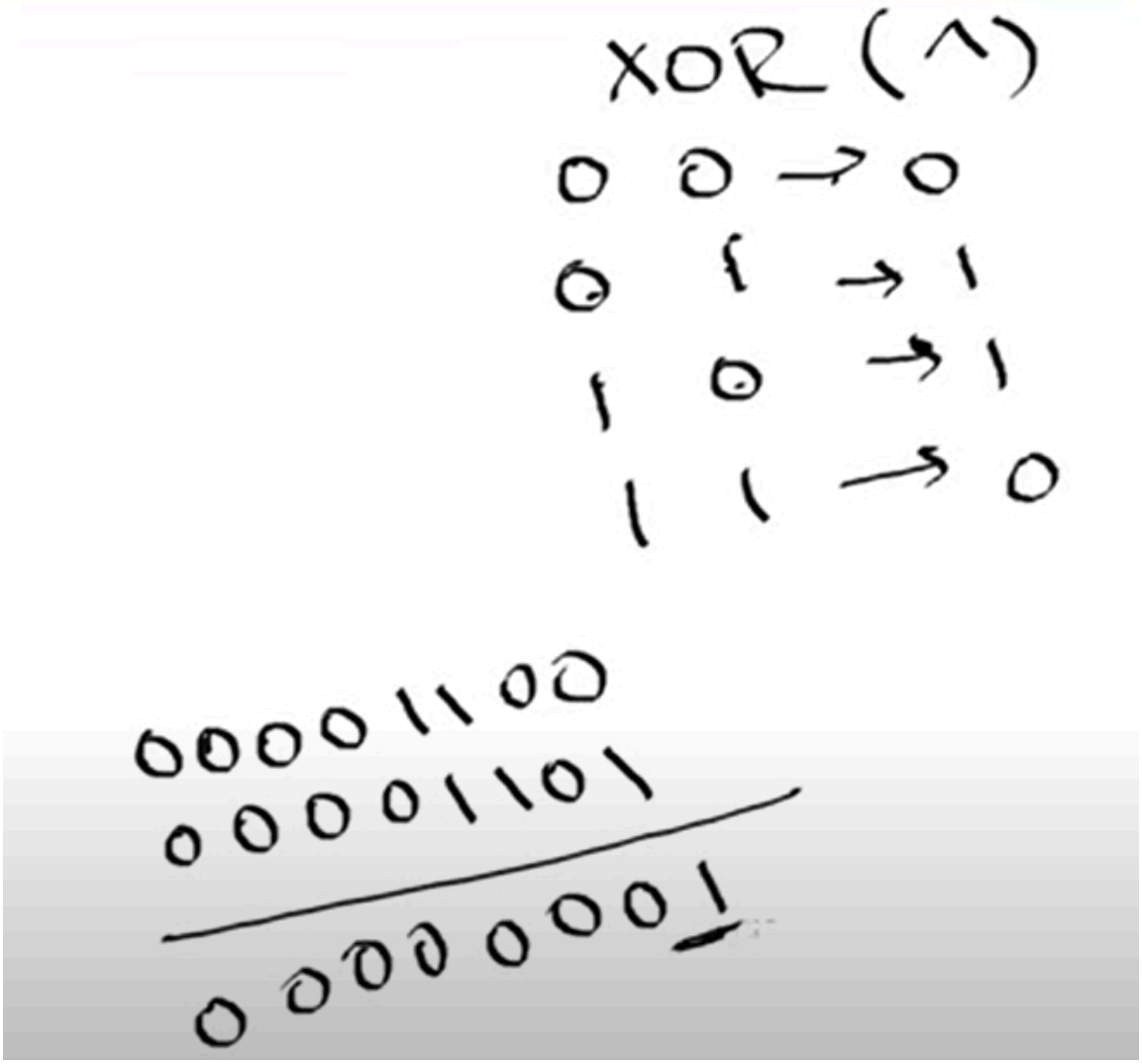


In [347…
```
35 & 40 #please do the homework conververt 35,40 to binary format
```

Out[347…
```
32
```

In [348…
```
35 | 40
```

Out[348…
```
43
```



In [349…
```
# in XOR if the both number are different then we will get 1 or else we will

12 ^ 13
```

Out[349… 1

In [350…
```
25 ^ 30
```

Out[350… 7

In [351…
```
bin(25)
```

Out[351… '0b11001'

In [352…
```
bin(30)
```

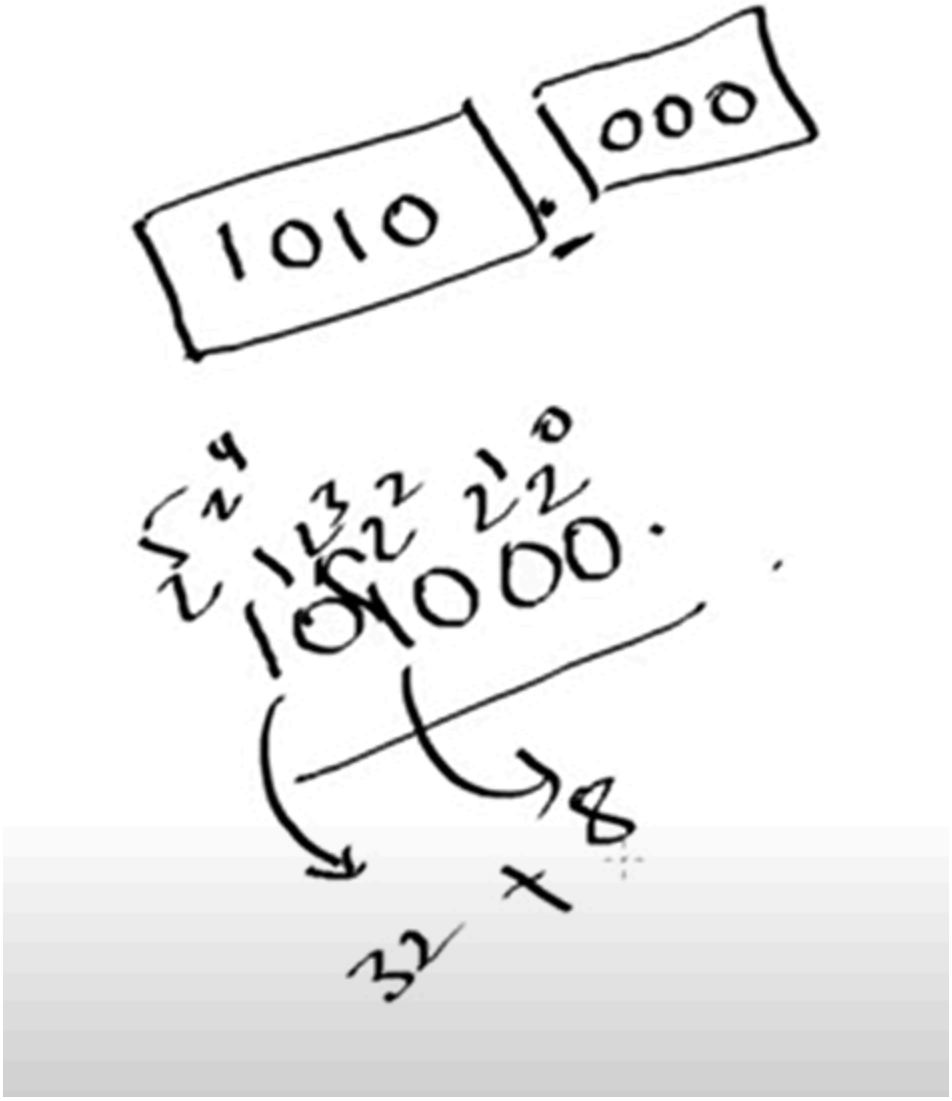
Out[352… '0b11110'

In [353…
```
int(0b000111)
```

Out[353… 7

In [354…
```
# BIT WISE LEFT OPERATOR
#bit wise left operator bydefault you will take 2 zeros ( )
#10 binary operator is 1010 | also i can say 1010
10<<2
```

Out[354… 40



In [355…
```
20<<4 #can we do this
```

Out[355… 320

BITWISE RIGHTSHIFT OPERATOR

In [356…
```
10>>2
```

Out[356…
```
2
```

In [357…
```
bin(20)
```

Out[357…
```
'0b10100'
```

In [358…
```
20>>4
```

Out[358…
```
1
```

import math module

https://docs.python.org/3/library/math.html

In [359…
```
x = sqrt(25) #sqrt is inbuild function
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[359], line 1
----> 1 x = sqrt(25) #sqrt is inbuild function

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

In [360…
```
import math # math is module
```

In [361…
```
x = math.sqrt(25)
x
```

Out[361…
```
5.0
```

In [362…
```
x1 = math.sqrt(15)
x1
```

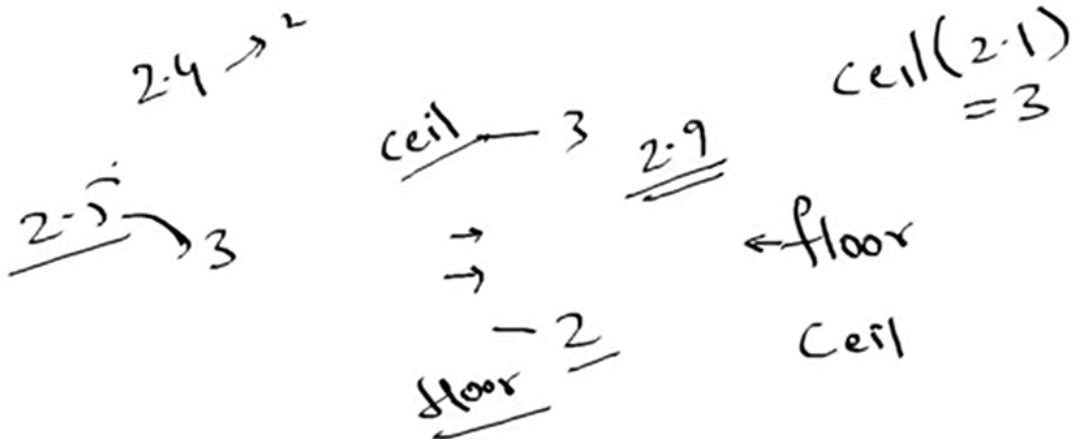
Out[362…
```
3.872983346207417
```

In [363…
```
print(math.floor(2.9)) #floor - minimum or least value
```

2

In [364…
```
print(math.ceil(2.9)) #ceil - maximum or highest value
```

3



In [366…
```
print(math.pow(3,2)) # pow is a funcition use to calculate the power of numb
```

9.0

In [367…
```
print(math.pi) #these are constant
```

3.141592653589793

In [368…
```
pi=3.44
pi
```

Out[368… 3.44

In [369…
```
math.pi
```

Out[369… 3.141592653589793

In [370…
```
print(math.e) #these are constant
```

2.718281828459045

In [391…
```
import math as m
m.sqrt(10)
```

Out[391… 3.1622776601683795

In [372…
```
from math import sqrt,pow # math has many function if you want to call speci
pow(2,3)
```

Out[372… 8.0

In [373…
```
round(pow(2,3))
```

Out[373…
```
8
```

In [374…
```
help(math)
```

```
Help on built-in module math:

NAME
    math

DESCRIPTION
    This module provides access to the mathematical functions
    defined by the C standard.

FUNCTIONS
    acos(x, /)
        Return the arc cosine (measured in radians) of x.

        The result is between 0 and pi.

    acosh(x, /)
        Return the inverse hyperbolic cosine of x.

    asin(x, /)
        Return the arc sine (measured in radians) of x.

        The result is between -pi/2 and pi/2.

    asinh(x, /)
        Return the inverse hyperbolic sine of x.

    atan(x, /)
        Return the arc tangent (measured in radians) of x.

        The result is between -pi/2 and pi/2.

    atan2(y, x, /)
        Return the arc tangent (measured in radians) of y/x.

        Unlike atan(y/x), the signs of both x and y are considered.

    atanh(x, /)
        Return the inverse hyperbolic tangent of x.

    cbrt(x, /)
        Return the cube root of x.

    ceil(x, /)
        Return the ceiling of x as an Integral.

        This is the smallest integer >= x.

    comb(n, k, /)
        Number of ways to choose k items from n items without repetition and
without order.

        Evaluates to n! / (k! * (n - k)!) when k <= n and evaluates
        to zero when k > n.

        Also called the binomial coefficient because it is equivalent
        to the coefficient of k-th term in polynomial expansion of the
```

```
        expression (1 + x)**n.

        Raises TypeError if either of the arguments are not integers.
        Raises ValueError if either of the arguments are negative.

    copysign(x, y, /)
        Return a float with the magnitude (absolute value) of x but the sign
of y.

        On platforms that support signed zeros, copysign(1.0, -0.0)
        returns -1.0.

    cos(x, /)
        Return the cosine of x (measured in radians).

    cosh(x, /)
        Return the hyperbolic cosine of x.

    degrees(x, /)
        Convert angle x from radians to degrees.

    dist(p, q, /)
        Return the Euclidean distance between two points p and q.

        The points should be specified as sequences (or iterables) of
        coordinates.  Both inputs must have the same dimension.

        Roughly equivalent to:
            sqrt(sum((px - qx) ** 2.0 for px, qx in zip(p, q)))

    erf(x, /)
        Error function at x.

    erfc(x, /)
        Complementary error function at x.

    exp(x, /)
        Return e raised to the power of x.

    exp2(x, /)
        Return 2 raised to the power of x.

    expm1(x, /)
        Return exp(x)-1.

        This function avoids the loss of precision involved in the direct ev
aluation of exp(x)-1 for small x.

    fabs(x, /)
        Return the absolute value of the float x.

    factorial(n, /)
        Find n!.

        Raise a ValueError if x is negative or non-integral.
```

```
    floor(x, /)
        Return the floor of x as an Integral.

        This is the largest integer <= x.

    fmod(x, y, /)
        Return fmod(x, y), according to platform C.

        x % y may differ.

    frexp(x, /)
        Return the mantissa and exponent of x, as pair (m, e).

        m is a float and e is an int, such that x = m * 2.**e.
        If x is 0, m and e are both 0.  Else 0.5 <= abs(m) < 1.0.

    fsum(seq, /)
        Return an accurate floating point sum of values in the iterable seq.

        Assumes IEEE-754 floating point arithmetic.

    gamma(x, /)
        Gamma function at x.

    gcd(*integers)
        Greatest Common Divisor.

    hypot(...)
        hypot(*coordinates) -> value

        Multidimensional Euclidean distance from the origin to a point.

        Roughly equivalent to:
            sqrt(sum(x**2 for x in coordinates))

        For a two dimensional point (x, y), gives the hypotenuse
        using the Pythagorean theorem:  sqrt(x*x + y*y).

        For example, the hypotenuse of a 3/4/5 right triangle is:

            >>> hypot(3.0, 4.0)
            5.0

    isclose(a, b, *, rel_tol=1e-09, abs_tol=0.0)
        Determine whether two floating point numbers are close in value.

          rel_tol
            maximum difference for being considered "close", relative to the
            magnitude of the input values
          abs_tol
            maximum difference for being considered "close", regardless of t
he
            magnitude of the input values

        Return True if a is close in value to b, and False otherwise.
```

```
        For the values to be considered close, the difference between them
        must be smaller than at least one of the tolerances.

        -inf, inf and NaN behave similarly to the IEEE 754 Standard.  That
        is, NaN is not close to anything, even itself.  inf and -inf are
        only close to themselves.

    isfinite(x, /)
        Return True if x is neither an infinity nor a NaN, and False otherwi
se.

    isinf(x, /)
        Return True if x is a positive or negative infinity, and False other
wise.

    isnan(x, /)
        Return True if x is a NaN (not a number), and False otherwise.

    isqrt(n, /)
        Return the integer part of the square root of the input.

    lcm(*integers)
        Least Common Multiple.

    ldexp(x, i, /)
        Return x * (2**i).

        This is essentially the inverse of frexp().

    lgamma(x, /)
        Natural logarithm of absolute value of Gamma function at x.

    log(...)
        log(x, [base=math.e])
        Return the logarithm of x to the given base.

        If the base is not specified, returns the natural logarithm (base e)
of x.

    log10(x, /)
        Return the base 10 logarithm of x.

    log1p(x, /)
        Return the natural logarithm of 1+x (base e).

        The result is computed in a way which is accurate for x near zero.

    log2(x, /)
        Return the base 2 logarithm of x.

    modf(x, /)
        Return the fractional and integer parts of x.

        Both results carry the sign of x and are floats.

    nextafter(x, y, /, *, steps=None)
```

```
        Return the floating-point value the given number of steps after x to
wards y.

        If steps is not specified or is None, it defaults to 1.

        Raises a TypeError, if x or y is not a double, or if steps is not an
integer.
        Raises ValueError if steps is negative.

    perm(n, k=None, /)
        Number of ways to choose k items from n items without repetition and
with order.

        Evaluates to n! / (n - k)! when k <= n and evaluates
        to zero when k > n.

        If k is not specified or is None, then k defaults to n
        and the function returns n!.

        Raises TypeError if either of the arguments are not integers.
        Raises ValueError if either of the arguments are negative.

    pow(x, y, /)
        Return x**y (x to the power of y).

    prod(iterable, /, *, start=1)
        Calculate the product of all the elements in the input iterable.

        The default start value for the product is 1.

        When the iterable is empty, return the start value.  This function i
s
        intended specifically for use with numeric values and may reject
        non-numeric types.

    radians(x, /)
        Convert angle x from degrees to radians.

    remainder(x, y, /)
        Difference between x and the closest integer multiple of y.

        Return x - n*y where n*y is the closest integer multiple of y.
        In the case where x is exactly halfway between two multiples of
        y, the nearest even value of n is used. The result is always exact.

    sin(x, /)
        Return the sine of x (measured in radians).

    sinh(x, /)
        Return the hyperbolic sine of x.

    sqrt(x, /)
        Return the square root of x.

    sumprod(p, q, /)
        Return the sum of products of values from two iterables p and q.
```

```
            Roughly equivalent to:

                sum(itertools.starmap(operator.mul, zip(p, q, strict=True)))

            For float and mixed int/float inputs, the intermediate products
            and sums are computed with extended precision.

        tan(x, /)
            Return the tangent of x (measured in radians).

        tanh(x, /)
            Return the hyperbolic tangent of x.

        trunc(x, /)
            Truncates the Real x to the nearest Integral toward 0.

            Uses the __trunc__ magic method.

        ulp(x, /)
            Return the value of the least significant bit of the float x.

DATA
    e = 2.718281828459045
    inf = inf
    nan = nan
    pi = 3.141592653589793
    tau = 6.283185307179586

FILE
    (built-in)
```

user input function in python || command line input

In [375…
```
x = input()
y = input()
z = x + y
print(z) # console is waiting for user to enter input
# also if you work in idle
```

```
678
```

In [376…
```
x1 = input('Enter the 1st number') #whenevery you works in input function it
y1 = input('Enter the 2nd number') # it wont understand as arithmetic operat
z1 = x1 + y1
print(z1)
```

```
66609
```

In [377…
```
type(x1)
type(y1)
```

Out[377… str

In [378…
```
x1 = input('Enter the 1st number') #whenevery you works in input function it
a1 = int(x1)
y1 = input('Enter the 2nd number') # it wont understand as arithmetic operat
b1 = int(y1)
z1 = a1 + b1
print(z1)
```

```
62
```

for the above code notice we are using many lines because fo that wasting some memory spaces as well

In [379…
```
x2 = int(input('Enter the 1st number'))
y2 = int(input('Enter the 2nd number'))
z2 = x2 + y2
z2
```

Out[379…
```
277
```

lets take input from the user in char format, but we dont have char format in python

In [382…
```
ch = input('enter a char')
print(ch)
```

```
Krishna
```

In [383…
```
print(ch[0])
```

```
K
```

In [384…
```
print(ch[1])
```

```
r
```

In [385…
```
print(ch[-1])
```

```
a
```

In [386…
```
ch = input('enter a char')[0]
print(ch)
```

```
r
```

In [387…
```
ch = input('enter a char')[1:3]
print(ch)
```

```
ar
```

In [388…
```
ch = input('enter a char')
print(ch) # if you enter as 2 + 6 -1 we get output as 2 + 6-1 only
```

```
Ram Ram
```

EVAL function using input

In [389…

```
result = eval(input('enter an expr')) # its useto evaluate the expression
print(result)
```

```
-464
```

Basic python completed

In [ ]: