Basic Python

In [24]:
```
2+2 #integers
```

Out[24]: 4

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

Out[2]: 1

In [3]:
```
2*6
```

Out[3]: 12

In [4]:
```
3/4
```

Out[4]: 0.75

In [5]:
```
3//4
```

Out[5]: 0

In [7]:
```
6+2-6
```

Out[7]: 2

In [8]:
```
2+4-
```

```
  Cell In[8], line 1
    2+4-
        ^
SyntaxError: invalid syntax
```

In [9]:
```
3+2*8
```

Out[9]: 19

In [11]:
```
(3+2)*8
```

Out[11]: 40

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

Out[12]: 32

In [13]:
```
2*4
```

Out[13]: 8

In [14]:
```
2**4
```

Out[14]: 16

In [15]:
```
13%5
```

Out[15]:
```
3
```

In [16]:
```
13%%4
```

```
  Cell In[16], line 1
    13%%4
       ^
SyntaxError: invalid syntax
```

In [41]:
```
-10//3
```

Out[41]:
```
-4
```

In [42]:
```
-11//3
```

Out[42]:
```
-4
```

In [19]:
```
3+'nit'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[19], line 1
----> 1 3+'nit'

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

In [20]:
```
3*'hi'
```

Out[20]:
```
'hihihi'
```

In [21]:
```
3*' hi '
```

Out[21]:
```
' hi  hi  hi '
```

In [22]:
```
a,b,c,d,e=13,2.7,'hi',1+2j,True
print(a)
print(b)
print(c)
print(d)
print(e)
```

```
13
2.7
hi
(1+2j)
True
```

In [23]:
```
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'>
```

Datatypes & Datastructures

1.String Basics

In [25]:
```
'Hello world'
```

Out[25]:
```
'Hello world'
```

In [26]:
```
print('Hello world')
```

```
Hello world
```

In [27]:
```
"Max IT Technology"
```

Out[27]:
```
'Max IT Technology'
```

In [28]:
```
s='Max IT Technology'
s
```

Out[28]:
```
'Max IT Technology'
```

In [29]:
```
a=2
b=4
a+b
```

Out[29]:
```
6
```

In [30]:
```
c=a+b
c
```

Out[30]:
```
6
```

In [31]:
```
a=4
b='hi'
c=a+b
print(c)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[31], line 3
      1 a=4
      2 b='hi'
----> 3 c=a+b
      4 print(c)

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

In [32]:
```
print('Max IT's"Technology"')
```

```
  Cell In[32], line 1
    print('Max IT's"Technology"')
                                ^
SyntaxError: unterminated string literal (detected at line 1)
```

In [33]:
```
print('Max IT\'s"Technology"')
```

```
Max IT's"Technology"
```

In [34]:
```
print('Max IT',"Technology")
```

```
Max IT Technology
```

In [35]:
```
print("Max IT','Technology")
```

```
Max IT','Technology
```

In [36]:
```
'hi'+'hi'
```

Out[36]: 'hihi'

In [38]:
```
'hi' ' hi'
```

Out[38]: 'hi hi'

In [39]:
```
4*'hi'
```

Out[39]: 'hihihihi'

In [40]:
```
4*' hi'
```

Out[40]: ' hi hi hi hi'

In [43]:
```
print('c:\nit')
```

```
c:
it
```

In [44]:
```
print(r'c:\nit')
```

```
c:\nit
```

Variable

In [1]:
```
4
```

Out[1]: 4

In [2]:
```
x=4
```

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

Out[3]: 7

In [12]:
```
y=2
y
```

Out[12]: 2

In [5]:
```
x+y
```

Out[5]: 6

In [6]:
```
x=5
x
```

Out[6]: 5

In [7]:
```
x+y
```

Out[7]: 7

In [16]:
```
x+5
```

Out[16]: 10

In [18]:
```
y
```

Out[18]: 2

In [20]:
```
_
```

Out[20]: '{"dataframes": [], "user": "DELL"}'

In [22]:
```
x=3
```

In [23]:
```
x
```

Out[23]: 3

In [25]:
```
name='hello'
name
```

Out[25]: 'hello'

In [26]:
```
name+'welcome'
```

Out[26]: 'hellowelcome'

In [27]:
```
name+' welcome'
```

Out[27]: 'hello welcome'

In [28]:
```
'a' 'b'
```

Out[28]: 'ab'

In [29]:
```
name 'technology'
```

```
  Cell In[29], line 1
    name 'technology'
         ^
SyntaxError: invalid syntax
```

In [30]:
```
name
```

Out[30]:
```
'hello'
```

In [31]:
```
len(name)
```

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

In [32]:
```
name[0]
```

Out[32]:
```
'h'
```

In [33]:
```
name[4]
```

Out[33]:
```
'o'
```

In [34]:
```
name[-1]
```

Out[34]:
```
'o'
```

In [35]:
```
name[-3]
```

Out[35]:
```
'l'
```

slicing

In [36]:
```
name
```

Out[36]:
```
'hello'
```

In [37]:
```
name[0:1]
```

Out[37]:
```
'h'
```

In [39]:
```
name[0:2]
```

Out[39]:
```
'he'
```

In [40]:
```
name[1:4]
```

Out[40]:
```
'ell'
```

In [41]:
```
name[1:]
```

Out[41]:
```
'ello'
```

In [42]:
```
name[:4]
```

Out[42]:
```
'hell'
```

In [44]:
```
name[6:9]
```

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

In [45]:
```
name1='hat'
name1
```

Out[45]:
```
'hat'
```

In [46]:
```
name1[0:1]
```

Out[46]:
```
'h'
```

In [48]:
```
name1[0:1]='m'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[48], line 1
----> 1 name1[0:1]='m'

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

In [49]:
```
name1[0]='m'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[49], line 1
----> 1 name1[0]='m'

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

In [50]:
```
name1
```

Out[50]:
```
'hat'
```

In [51]:
```
name1[1:]
```

Out[51]:
```
'at'
```

In [52]:
```
'm'+name1[1:]
```

Out[52]:
```
'mat'
```

In [53]:
```
len(name1)
```

Out[53]:
```
3
```

2.List

In [54]:
```
l=[]
```

In [55]:
```
nums=[10,20,30]
nums
```

Out[55]:
```
[10, 20, 30]
```

In [56]:
```
nums[0]
```

Out[56]: 10

In [57]:
```
nums[-1]
```

Out[57]: 30

In [58]:
```
nums[:1]
```

Out[58]: [10]

In [59]:
```
num1=['hi','hello']
num1
```

Out[59]: ['hi', 'hello']

In [61]:
```
num2=['hi',3.4,56]
num2
```

Out[61]: ['hi', 3.4, 56]

In [62]:
```
num3=[nums,num1]
num3
```

Out[62]: [[10, 20, 30], ['hi', 'hello']]

In [64]:
```
num4=[nums,num1,num2]
num4
```

Out[64]: [[10, 20, 30], ['hi', 'hello'], ['hi', 3.4, 56]]

In [66]:
```
nums.append(45)
nums
```

Out[66]: [10, 20, 30, 45, 45]

In [67]:
```
nums.remove(45)
nums
```

Out[67]: [10, 20, 30, 45]

In [68]:
```
nums.pop(1)
nums
```

Out[68]: [10, 30, 45]

In [69]:
```
nums.pop()
nums
```

Out[69]: [10, 30]

In [70]:
```
num1
```

Out[70]: ['hi', 'hello']

In [71]:
```
num1.insert(2,'world')
num1
```

Out[71]: ['hi', 'hello', 'world']

In [72]:
```
num1.insert(0,1)
num1
```

Out[72]: [1, 'hi', 'hello', 'world']

In [73]:
```
num2
```

Out[73]: ['hi', 3.4, 56]

In [74]:
```
del num2[1:]
num2
```

Out[74]: ['hi']

In [75]:
```
num2.extend([29,34,50])
num2
```

Out[75]: ['hi', 29, 34, 50]

In [76]:
```
num3
```

Out[76]: [[10, 30], [1, 'hi', 'hello', 'world']]

In [77]:
```
num3.extend(['a',4,8.9])
num3
```

Out[77]: [[10, 30], [1, 'hi', 'hello', 'world'], 'a', 4, 8.9]

In [78]:
```
nums
```

Out[78]: [10, 30]

In [79]:
```
min(nums)
```

Out[79]: 10

In [80]:
```
max(nums)
```

Out[80]: 30

In [81]:
```
num1
```

Out[81]: [1, 'hi', 'hello', 'world']

In [82]:
```
min(num1)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[82], line 1
----> 1 min(num1)

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

In [83]:
```
sum(nums)
```

Out[83]: 40

In [85]:
```
nums.sort()
nums
```

Out[85]: [10, 30]

In [86]:
```
l=[1,2,3]
l
```

Out[86]: [1, 2, 3]

In [87]:
```
l[0]=100
l
```

Out[87]: [100, 2, 3]

3.Tuple

In [88]:
```
t=(12,34,56)
t
```

Out[88]: (12, 34, 56)

In [89]:
```
t[0]
```

Out[89]: 12

In [90]:
```
t[0]=30
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[90], line 1
----> 1 t[0]=30

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

4.Set

In [92]:
```
s={}
```

In [93]:
```
s1={21,34,5,67,9}
s1
```

Out[93]: {5, 9, 21, 34, 67}

In [94]:
```
s2={23,45,6,'hi',46}
s2
```

Out[94]: {23, 45, 46, 6, 'hi'}

In [95]:
```
s1[1]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[95], line 1
----> 1 s1[1]

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

5.Dictionary

In [96]:
```
d={1:'apple',2:'banana',3:'chico'}
d
```

Out[96]:
```
{1: 'apple', 2: 'banana', 3: 'chico'}
```

In [97]:
```
d[2]
```

Out[97]:
```
'banana'
```

In [98]:
```
d[3]
```

Out[98]:
```
'chico'
```

In [99]:
```
d[4]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[99], line 1
----> 1 d[4]

KeyError: 4
```

In [100…
```
d.get(2) #get() usage
```

Out[100…
```
'banana'
```

In [101…
```
d.get(1)
```

Out[101…
```
'apple'
```

In [102…
```
print(d.get(1))
```

```
apple
```

In [103…
```
d.get(1,'Not Found')
```

Out[103…
```
'apple'
```

In [104…
```
d.get(3,'Not Found')
```

Out[104…
```
'chico'
```

In [105…
```
d[5]='fruit'
```

In [106…
```
d
```

Out[106… {1: 'apple', 2: 'banana', 3: 'chico', 5: 'fruit'}

In [107…
```
del d[5]
```

In [108…
```
d
```

Out[108… {1: 'apple', 2: 'banana', 3: 'chico'}

In [109…
```
prog = {'python':['vscode', 'pycharm'], 'machine learning' : 'sklearn', 'datasci
```

In [110…
```
prog
```

Out[110…
```
{'python': ['vscode', 'pycharm'],
 'machine learning': 'sklearn',
 'datascience': ['jupyter', 'spyder']}
```

In [111…
```
prog['python']
```

Out[111… ['vscode', 'pycharm']

In [112…
```
prog['machine learning']
```

Out[112… 'sklearn'

In [114…
```
prog['datascience']
```

Out[114… ['jupyter', 'spyder']

In [115…
```
help() #to find help
```

```
Welcome to Python 3.13's help utility! If this is your first time using
Python, you should definitely check out the tutorial at
https://docs.python.org/3.13/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", "quit" or "exit".

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 [ ]:
```
help()
```

In [116…
```
help(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 the
m,
 |      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__(object, /)
 |      See PEP 585
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs)
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  __hash__ = None
```

In [117…

```
help(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
 |      MonthDayNano
 |      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__(object, /)
|      See PEP 585
|
|  ----------------------------------------------------------------------
|  Static methods defined here:
|
|  __new__(*args, **kwargs)
|      Create and return a new object.  See help(type) for accurate signature.
```

In [119…
```
num=3 #id()
id(num)
```

Out[119…
```
140717966341096
```

In [120…
```
name='hi'
id(name)
```

Out[120…
```
140717966385024
```

In [123…
```
a=10
id(a)
```

Out[123…
```
140717966341320
```

In [124…
```
b=a
```

In [125…
```
id(b)
```

Out[125…
```
140717966341320
```

In [126…
```
id(10)
```

Out[126…
```
140717966341320
```

In [127…
```
k=12
id(k)
```

Out[127… 140717966341384

In [128…
```
a=30
id(a)
```

Out[128… 140717966341960

In [129…
```
id(b)
```

Out[129… 140717966341320

In [130…
```
pi=3.14
pi
```

Out[130… 3.14

In [131…
```
pi=3.15
pi
```

Out[131… 3.15

In [132…
```
type(pi)
```

Out[132… float

6.Numeric(int,float,complex,boolean)

In [134…
```
w=1.4
type(w)
```

Out[134… float

In [135…
```
(a)
```

Out[135… 30

In [136…
```
w2=1+3J
type(w2)
```

Out[136… complex

In [137…
```
a=5.3
b=int(a)
```

In [138…
```
b
```

Out[138… 5

In [139…
```
type(b)
```

Out[139… int

In [140…
```
type(a)
```

Out[140…
```
float
```

In [141…
```
k=float(b)
k
```

Out[141…
```
5.0
```

In [142…
```
print(a)
print(b)
print(k)
```

```
5.3
5
5.0
```

In [143…
```
k1=complex(b,k)
```

In [144…
```
print(k1)
```

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

In [145…
```
type(k1)
```

Out[145…
```
complex
```

In [146…
```
b<k
```

Out[146…
```
False
```

In [147…
```
condition=b<k
```

In [148…
```
condition
```

Out[148…
```
False
```

In [149…
```
type(condition)
```

Out[149…
```
bool
```

In [150…
```
int(True)
```

Out[150…
```
1
```

In [151…
```
int(False)
```

Out[151…
```
0
```

In [153…
```
l=[1,2,3,4]
print(l)
type(l)
```

```
[1, 2, 3, 4]
```

Out[153…
```
list
```

In [155…
```
s={1,3,4,5}
s
```

Out[155…
```
{1, 3, 4, 5}
```

In [156…
```
type(s)
```

Out[156…
```
set
```

In [158…
```
s1={1,3,4,4,3,5}
s1
```

Out[158…
```
{1, 3, 4, 5}
```

In [159…
```
t=(10,40,30)
t
```

Out[159…
```
(10, 40, 30)
```

In [160…
```
type(t)
```

Out[160…
```
tuple
```

In [161…
```
str='hi'
type(str)
```

Out[161…
```
str
```

In [162…
```
st='g'
type(st)
```

Out[162…
```
str
```

7.Range

In [163…
```
r=range(10,20)
r
```

Out[163…
```
range(10, 20)
```

In [164…
```
type(r)
```

Out[164…
```
range
```

In [165…
```
list(range(10,20))
```

Out[165…
```
[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

In [166…
```
r1=list(r)
r1
```

Out[166…
```
[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

In [167…
```
even_num=list(range(3,20,2))
even_num
```

Out[167…
```
[3, 5, 7, 9, 11, 13, 15, 17, 19]
```

In [168…
```
d={1:'one',2:'two',3:'three'}
d
```

Out[168…
```
{1: 'one', 2: 'two', 3: 'three'}
```

In [169…
```
type(d)
```

Out[169…
```
dict
```

In [170…
```
d.keys()
```

Out[170…
```
dict_keys([1, 2, 3])
```

In [171…
```
d.values()
```

Out[171…
```
dict_values(['one', 'two', 'three'])
```

In [172…
```
d[2]
```

Out[172…
```
'two'
```

In [173…
```
d.get(2)
```

Out[173…
```
'two'
```

Operator

1.Arithmetic operator

In [175…
```
x,y=24,34
```

In [176…
```
x+y
```

Out[176…
```
58
```

In [177…
```
x-y
```

Out[177…
```
-10
```

In [178…
```
x*y
```

Out[178…
```
816
```

In [179…
```
x/y
```

Out[179…
```
0.7058823529411765
```

In [180…
```
x//y
```

Out[180…
```
0
```

In [181…
```
x%y
```

Out[181…
```
24
```

In [182…
```
x**y
```

Out[182…
```
845633575667902767630322901978300862909130997 76
```

In [183…
```
x1=4
y1=5
x1**y1
```

Out[183…
```
1024
```

2.Assignment operator

In [184…
```
x=4
```

In [189…
```
x=x+4
```

In [190…
```
x
```

Out[190…
```
15
```

In [188…
```
x+=2
x
```

Out[188…
```
11
```

In [191…
```
x*=3
x
```

Out[191…
```
45
```

In [192…
```
x-=14
x
```

Out[192…
```
31
```

In [193…
```
x/=4
x
```

Out[193…
```
7.75
```

In [194…
```
x//=2
x
```

Out[194…
```
3.0
```

In [195…
```
a,d=4,2
print(a)
print(d)
```

```
4
2
```

In [196…
```
a
```

Out[196… 4

In [197…
```
d
```

Out[197… 2

3.Unary operator

In [198…
```
u=5
u
```

Out[198… 5

In [199…
```
v=-(u)
v
```

Out[199… -5

In [200…
```
u
```

Out[200… 5

In [201…
```
-u
```

Out[201… -5

4.Relational operator

In [202…
```
a=2
b=3
```

In [203…
```
a<b
```

Out[203… True

In [204…
```
a>b
```

Out[204… False

In [205…
```
a==b
```

Out[205… False

In [206…
```
a!=b
```
Out[206…
```
True
```
In [207…
```
b=4
```
In [208…
```
a==b
```
Out[208…
```
False
```
In [209…
```
a
```
Out[209…
```
2
```
In [210…
```
b
```
Out[210…
```
4
```
In [211…
```
a>b
```
Out[211…
```
False
```
In [212…
```
a>=b
```
Out[212…
```
False
```
In [213…
```
a<b
```
Out[213…
```
True
```
In [214…
```
a>b
```
Out[214…
```
False
```
In [215…
```
a<=b
```
Out[215…
```
True
```
In [216…
```
b=5
```
In [217…
```
a!=b
```
Out[217…
```
True
```

5.Logical operator

In [218…
```
a=3
b=2
```
In [219…
```
a<4 and b<5
```
Out[219…
```
True
```

In [220…
```
a<4 and b<2
```
Out[220…
```
False
```
In [221…
```
a<4 or a<2
```
Out[221…
```
True
```
In [222…
```
a>4 or b<1
```
Out[222…
```
False
```
In [223…
```
x=False
x
```
Out[223…
```
False
```
In [224…
```
not x
```
Out[224…
```
True
```
In [225…
```
x
```
Out[225…
```
False
```
In [226…
```
not x
```
Out[226…
```
True
```

Number system conversion

In [227…
```
12
```
Out[227…
```
12
```
In [228…
```
bin(12)
```
Out[228…
```
'0b1100'
```
In [229…
```
int(0b1100)
```
Out[229…
```
12
```
In [230…
```
bin(23)
```
Out[230…
```
'0b10111'
```
In [231…
```
int(0b10111)
```
Out[231…
```
23
```
In [232…
```
oct(12)
```

Out[232… '0o14'

In [233…
```
int(0o14)
```

Out[233… 12

In [234…
```
int(0b11001)
```

Out[234… 25

In [235…
```
bin(5)
```

Out[235… '0b101'

In [236…
```
oct(25)
```

Out[236… '0o31'

In [237…
```
hex(25)
```

Out[237… '0x19'

In [238…
```
hex(15)
```

Out[238… '0xf'

In [239…
```
0x1
```

Out[239… 1

In [240…
```
0xa
```

Out[240… 10

In [241…
```
0xb
```

Out[241… 11

In [242…
```
0xf
```

Out[242… 15

In [243…
```
0xg
```

```
  Cell In[243], line 1
    0xg
     ^
SyntaxError: invalid hexadecimal literal
```

In [244…
```
hex(1)
```

Out[244… '0x1'

In [245…
```
0x16
```

Out[245… 22

In [246…
```
0x25
```

Out[246… 37

Swap two variables

In [247…
```
a=4
b=3
```

In [248…
```
a=b
b=a
```

In [249…
```
print(a)
print(b)
```

```
3
3
```

In [250…
```
a1=2
b1=5
```

In [258…
```
temp=a1 #swapping using third variable
a1=b1
b1=temp
```

In [252…
```
print(a1)
print(b1)
```

```
5
2
```

In [254…
```
a2=7
b2=9
```

In [259…
```
a2=a2+b2 #swapping without using third variable
b2=a2-b2
a2=a2-b2
```

In [256…
```
print(a2)
print(b2)
```

```
9
7
```

In [260…
```
a2=a2^b2 #swapping using xor
b2=a2^b2
a2=a2^b2
```

In [261…
```
print(a2)
print(b2)
```

```
7
9
```

In [265…
```
a2, b2 #swapping using rot_two()
```

Out[265… (9, 7)

In [263…
```
a2,b2=b2,a2
```

In [264…
```
print(a2)
print(b2)
```

```
9
7
```

Bitwise operator

In [266…
```
print(bin(12))
print(bin(14))
```

```
0b1100
0b1110
```

Bitwise complement operator(~)

In [272…
```
~12
```

Out[272… -13

In [268…
```
~23
```

Out[268… -24

In [269…
```
~10
```

Out[269… -11

In [270…
```
~67
```

Out[270… -68

Bitwise AND operator(&)

In [273…
```
12&13
```

Out[273… 12

In [274…
```
1&0
```

Out[274… 0

In [275…
```
35 & 40
```

Out[275… 32

In [276…
```
1100 & 1101
```

Out[276… 1100

Bitwise OR operator(|)

In [277…
```
12^12
```

Out[277…
```
0
```

In [278…
```
13^14
```

Out[278…
```
3
```

In [279…
```
23^35
```

Out[279…
```
52
```

Bitwise LEFT SHIFT operator(<<)

In [280…
```
23<<1
```

Out[280…
```
46
```

In [281…
```
1<<2
```

Out[281…
```
4
```

In [282…
```
10<<2
```

Out[282…
```
40
```

In [284…
```
10<<3
```

Out[284…
```
80
```

Bitwise RIGHT SHIFT operator(>>)

In [285…
```
10>>1
```

Out[285…
```
5
```

In [286…
```
10>>2
```

Out[286…
```
2
```

In [287…
```
10>>3
```

Out[287…
```
1
```

In [ ]: