```
Last login: Tue Apr 28 17:07:25 on ttys000
➜  ~ ipython
Python 3.8.2 (default, Mar 11 2020, 00:29:50)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.13.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: """Python Operators"""
Out[1]: 'Python Operators'

In [2]: """Arithmetic operators"""
Out[2]: 'Arithmetic operators'

In [3]: """
   ...: + for addition
   ...: - for subtraction
   ...: * for multiplication
   ...: / for division, raise error if divided by 0
   ...: % for remainder/modulus
   ...: ** for exponent
   ...: // for floor division
   ...: """
Out[3]: '\n+ for addition\n- for subtraction\n* for multiplication\n/ for divisi
on, raise error if divided by 0\n% for remainder/modulus\n** for exponent\n// fo
r floor division\n'

In [4]: a = 9

In [5]: b = 7

In [6]: a + b
Out[6]: 16

In [7]: a - b
Out[7]: 2

In [8]: a * b
Out[8]: 63

In [9]: a / b
Out[9]: 1.2857142857142858

In [10]: a % b
Out[10]: 2

In [11]: a // b #This is the integer part of the division
Out[11]: 1

In [12]: # Like a / b = 1.2857142857142858, but a // b = 1

In [13]: # It is used when you only need the integer part of the division

In [14]: """Assignment Operators"""
Out[14]: 'Assignment Operators'

In [15]: """
    ...: =    x = 5    x = 5
    ...: +=   x += 3   x = x + 3
```

```
    ...: -=   x -= 3   x = x - 3
    ...: *=   x *= 3   x = x * 3
    ...: /=   x /= 3   x = x / 3
    ...: %=   x %= 3   x = x % 3
    ...: //=  x //= 3  x = x // 3
    ...: **=  x **= 3  x = x ** 3
    ...: &=   x &= 3   x = x & 3
    ...: |=   x |= 3   x = x | 3
    ...: ^=   x ^= 3   x = x ^ 3
    ...: >>=  x >>= 3  x = x >> 3
    ...: <<=  x <<= 3  x = x << 3
    ...:
    ...: These operators are used to assign the values in oython
    ...: """
Out[15]: '\n=    x = 5    x = 5\n+=   x += 3   x = x + 3\n-=   x -= 3   x = x -
3\n*=   x *= 3   x = x * 3\n/=   x /= 3   x = x / 3\n%=   x %= 3   x = x % 3\n//
=  x //= 3  x = x // 3\n**=  x **= 3  x = x ** 3\n&=   x &= 3   x = x & 3\n|=
x |= 3   x = x | 3\n^=   x ^= 3   x = x ^ 3\n>>=  x >>= 3  x = x >> 3\n<<=  x <<
= 3  x = x << 3\n\nThese operators are used to assign the values in oython\n'

In [16]: """Comparison Operators"""
Out[16]: 'Comparison Operators'

In [17]: """
    ...: These operators are used to compare between two objects
    ...:
    ...: ==   Equal      x == y
    ...: !=  Not equal   x!= y
    ...: >   Greater than   x > y
    ...: <   Less than    x < y
    ...: >=  Greater than or equal to    x >= y
    ...: <=  Less than or equal to       x <= y
    ...: """
Out[17]: '\nThese operators are used to compare between two objects\n\n==   Equa
l      x == y\n!=  Not equal   x!= y\n>   Greater than   x > y\n<   Less than
 x < y\n>=  Greater than or equal to    x >= y\n<=  Less than or equal to
x <= y\n'

In [18]: """Logical Operators"""
Out[18]: 'Logical Operators'

In [19]: """
    ...:
    ...: and     Returns true if both are true else return False
    ...: or      Returns true if any one of them is true, Once the first operand
    ...:  is found true it doesn't check for the other operand, $This is importa
    ...: nt to know
    ...: not     Reverse the result, returns False if the result is true
    ...: """
Out[19]: "\n\nand     Returns true if both are true else return False\nor      R
eturns true if any one of them is true, Once the first operand is found true it
doesn't check for the other operand, $This is important to know\nnot     Reverse
 the result, returns False if the result is true\n"

In [20]: a = True

In [21]: True and True
```

```
Out[21]: True

In [22]: True and False
Out[22]: False

In [23]: False and True
Out[23]: False

In [24]: False and False
Out[24]: False

In [25]:

In [25]: True or True
Out[25]: True

In [26]: True or False
Out[26]: True

In [27]: False or True
Out[27]: True

In [28]: False or False
Out[28]: False

In [29]: 5 == 5 or 5/0 == 5
Out[29]: True

In [30]: # this example shows it doesn't check for the other operand once the fi
    ...: rst operand is found True, else this statement would have raised an err
    ...: or

In [31]: 5/0 == 5 or 5 == 5
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-31-7db2813b9524> in <module>
----> 1 5/0 == 5 or 5 == 5

ZeroDivisionError: division by zero

In [32]:

In [32]:

In [32]: not True
Out[32]: False

In [33]: not False
Out[33]: True

In [34]:

In [34]:

In [34]: """Identity Operators"""
Out[34]: 'Identity Operators'
```

```
In [35]: """
    ...: is     Returns True if both variables are the same object
    ...: is not Returns True if both variables are not the same object
    ...: """
Out[35]: '\nis     Returns True if both variables are the same object\nis not Re
turns True if both variables are not the same object\n'

In [36]: a = 5

In [37]: b = 5

In [38]: a is b
Out[38]: True

In [39]: a = [5]

In [40]: b = {5}

In [41]: a is b
Out[41]: False

In [42]: a is not b
Out[42]: True

In [43]: a = {7}

In [44]: b = {9}

In [45]: a is b
Out[45]: False

In [46]: a = [8]

In [47]: b = [8]

In [48]: a is b
Out[48]: False

In [49]: c = {8}

In [50]: a is not c
Out[50]: True

In [51]:

In [51]:

In [51]: """Membership Operators"""
Out[51]: 'Membership Operators'

In [52]: """
    ...: in Returns True if a sequence with the specified value is present in th
    ...: e object
    ...:
    ...: not in  Returns True if a sequence with the specified value is not pres
    ...: ent in the object
    ...: """
```

```
Out[52]: '\nin Returns True if a sequence with the specified value is present in
 the object\n\nnot in  Returns True if a sequence with the specified value is no
t present in the object\n'

In [53]: a = [1,2,3,4,5]

In [54]: 3 in a
Out[54]: True

In [55]: 0 in a
Out[55]: False

In [56]: 0 not in a
Out[56]: True

In [57]:

In [57]: """Bitwise Operators"""
Out[57]: 'Bitwise Operators'

In [58]: # & : AND Sets each bit to 1 if both bits are 1

In [59]: # | : OR  Sets each bit to 1 if one of two bits is 1

In [60]: # ^ : XOR Sets each bit to 1 if only one of two bits is 1

In [61]: # ~ : NOT Inverts all the bits

In [62]: # << : Zero fill left shift : Shift left by pushing zeros in from the r
    ...: ight and let the leftmost bits fall off

In [63]: # >> : Signed right shift : Shift right by pushing copies of the leftmo
    ...: st bit in from the left, and let the rightmost bits fall off

In [64]: a = 17

In [65]: bin(a)
Out[65]: '0b10001'

In [66]: b = 4

In [67]: bin(b)
Out[67]: '0b100'

In [68]: bin(a & b)
Out[68]: '0b0'

In [69]: bin(a | b)
Out[69]: '0b10101'

In [70]: bin(a ^ b)
Out[70]: '0b10101'

In [71]: bin( ~ a)
Out[71]: '-0b10010'

In [72]: bin( ~b)
```

```
Out[72]: '-0b101'

In [73]: bin( a << 1)
Out[73]: '0b100010'

In [74]: bin( a >> 1)
Out[74]: '0b1000'

In [75]:

In [75]:
```