In [ ]:
```
1 + 1
```

Out[ ]: 2

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

Out[2]: 1

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

Out[3]: 12

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

Out[ ]: 2.0

In [ ]:
```
8 / 5
```

Out[ ]: 1.6

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

Out[ ]: 2.0

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

Out[ ]: 2

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

Out[8]: 10

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

```
  Cell In[9], line 1
    8 + 8 -  #syntax error:
             ^
SyntaxError: invalid syntax
```

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

Out[10]: 30

In [ ]:
```
(5 + 5) * 5
```

Out[ ]: 50

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

Out[ ]: 32

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

Out[13]: 10

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

Out[14]: 32

In [15]:
```
3 ** 2
```

Out[15]: 9

In [16]:
```
15 / 3
```

Out[16]: 5.0

In [17]:
```
10 // 3
```

Out[17]: 3

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

Out[ ]: 1

In [19]:
```
10 % 2
```

Out[19]: 0

In [20]:
```
15 %% 2
```

```
  Cell In[20], line 1
    15 %% 2
       ^
SyntaxError: invalid syntax
```

In [21]:
```
-10//3
```

Out[21]: -4

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

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[22], line 1
----> 1 3 +

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

In [ ]:
```
3 + 'rainy'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[23], line 1
----> 1 3 + 'rainy'

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

In [ ]:
```
3 * 'nit'
```

Out[ ]: 'nitnitnit'

In [23]:
```
3 * ' nit '
```

Out[23]:
```
' nit  nit  nit '
```

In [24]:
```
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 [25]:
```
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 [26]:
```
type(c)
```

Out[26]:
```
str
```

In [27]:
```
'Naresh IT'
```

Out[27]:
```
'Naresh IT'
```

In [28]:
```
print('Max it')
```

```
Max it
```

In [29]:
```
"max it technology"
```

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

In [30]:
```
s1 = 'max it technology'
s1
```

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

In [31]:
```
a = 2
b = 3
a + b
```

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

In [32]:

```
c = a + b
c
```

Out[32]: 5

In [33]:
```
a = 3
b = 'hi'
c = a + b
print(c)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[33], line 3
      1 a = 3
      2 b = 'hi'
----> 3 c = a + b
      4 print(c)

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

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

```
  Cell In[34], line 1
    print('max it's"Technology"') # \ has some special meaning to ignore the error
                                ^
SyntaxError: unterminated string literal (detected at line 1)
```

In [ ]:
```
print('max it\'s"Technology"')
```

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

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

```
max it Technology
```

In [37]:
```
print("max it',  'Technology")
```

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

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

Out[ ]: 'nit nit'

In [39]:
```
'nit'  ' nit'
```

Out[39]: 'nit nit'

In [ ]:
```
5 * 'nit'
```

Out[ ]: 'nitnitnitnitnit'

In [ ]:
```
5*' nit'
```

Out[ ]: ' nit nit nit nit nit'

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

```
c:
it
```

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

```
c:\nit
```

In [44]:
```
2
```

Out[44]: 2

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

Out[ ]: 2

In [46]:
```
x + 3
```

Out[46]: 5

In [47]:
```
y = 3
y
```

Out[47]: 3

In [48]:
```
x + y
```

Out[48]: 5

In [49]:
```
x = 9
x
```

Out[49]: 9

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

Out[50]: 12

In [51]:
```
x + 10
```

Out[51]: 19

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

Out[ ]: 22

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

Out[53]: 25

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

Out[54]: 28

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

Out[55]: 31

In [56]:
```
y
```

Out[56]: 3

In [57]:
```
_ + y
```

Out[57]: 6

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

Out[58]: 9

In [59]:
```
_ + y
```

Out[59]: 12

In [ ]:
```
name = 'mit'
```

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

Out[61]: 'mit'

In [62]:
```
name + 'technology'
```

Out[62]: 'mittechnology'

In [63]:
```
name + ' technology'
```

Out[63]: 'mit technology'

In [64]:
```
'a' 'b'
```

Out[64]: 'ab'

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

```
  Cell In[65], line 1
    name 'technology'
         ^
SyntaxError: invalid syntax
```

In [66]:
```
name
```

Out[66]: 'mit'

In [67]:
```
len(name)
```

Out[67]: 3

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

Out[ ]: 'm'

In [69]:
```
name
```

Out[69]: 'mit'

```
name[5]
```

In [70]:

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[70], line 1
----> 1 name[5]

IndexError: string index out of range
```

In [71]:
```
name[7]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[71], line 1
----> 1 name[7]

IndexError: string index out of range
```

In [72]:
```
name[-1]
```

Out[72]: 't'

In [73]:
```
name[-2]
```

Out[73]: 'i'

In [74]:
```
name[-6]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[74], line 1
----> 1 name[-6]

IndexError: string index out of range
```

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

Out[75]: 'mit'

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

Out[ ]: 'm'

In [77]:
```
name[0:2]
```

Out[77]: 'mi'

In [78]:
```
name[1:4]
```

Out[78]: 'it'

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

Out[79]: 'mit'

In [80]:
```
name[1:]
```

Out[80]: 'it'

In [81]:
```
name[:4]
```

Out[81]: 'mit'

In [82]:
```
name
```

Out[82]: 'mit'

In [83]:
```
name[3:9]
```

Out[83]: ''

In [84]:
```
name
```

Out[84]: 'mit'

In [ ]:
```
name1 = 'fine'
name1
```

Out[ ]: 'fine'

In [86]:
```
name1[0:1]
```

Out[86]: 'f'

In [ ]:
```
name1[0:1] = 'd'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[87], 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 [ ]:
```
name1[0] = 'd'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[88], line 1
----> 1 name1[0] = 'd' #strings in python are immutable

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

In [89]:
```
name1
```

Out[89]: 'fine'

In [90]:
```
name1[1:]
```

Out[90]: 'ine'

In [ ]:
```
'd' + name1[1:]
```

Out[ ]: 'dine'

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

Out[ ]: 4

In [93]:
```
l = []
```

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

Out[ ]: [10, 20, 30]

In [95]:
```
nums[0]
```

Out[95]: 10

In [96]:
```
nums[-1]
```

Out[96]: 30

In [97]:
```
nums[1:]
```

Out[97]: [20, 30]

In [98]:
```
nums[:1]
```

Out[98]: [10]

In [99]:
```
num1 = ['hi', 'hallo']
```

In [100…
```
num1
```

Out[100… ['hi', 'hallo']

In [ ]:
```
num2 = ['hi', 8.9, 34]
num2
```

Out[ ]: ['hi', 8.9, 34]

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

In [103…
```
num3
```

Out[103… [[10, 20, 30], ['hi', 'hallo']]

In [104…
```
num4 = [nums, num1, num2]
```

In [105…
```
num4
```

Out[105… [[10, 20, 30], ['hi', 'hallo'], ['hi', 8.9, 34]]

In [106…
```
nums
```

Out[106… [10, 20, 30]

In [107…
```
nums.append(45)
```

```
nums
```

In [108…

Out[108… [10, 20, 30, 45]

In [109…
```
nums.remove(45)
```

In [110…
```
nums
```

Out[110… [10, 20, 30]

In [111…
```
nums.pop(1)
```

Out[111… 20

In [112…
```
nums
```

Out[112… [10, 30]

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

Out[ ]: 30

In [114…
```
nums
```

Out[114… [10]

In [115…
```
num1
```

Out[115… ['hi', 'hallo']

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

In [117…
```
num1
```

Out[117… ['hi', 'hallo', 'nit']

In [118…
```
num1.insert(0, 1)
```

In [119…
```
num1
```

Out[119… [1, 'hi', 'hallo', 'nit']

In [ ]:
```
num2
```

Out[ ]: ['hi', 8.9, 34]

In [121…
```
del num2[2:]
```

In [122…
```
num2
```

Out[122… ['hi', 8.9]

In [ ]:
```
num2.extend([29,15,20])
```

In [124…
```
num2
```

Out[124… ['hi', 8.9, 29, 15, 20]

In [125…
```
num3
```

Out[125… [[10], [1, 'hi', 'hallo', 'nit']]

In [126…
```
num3.extend(['a', 5, 6.7])
```

In [127…
```
num3
```

Out[127… [[10], [1, 'hi', 'hallo', 'nit'], 'a', 5, 6.7]

In [128…
```
nums
```

Out[128… [10]

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

Out[ ]: 10

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

Out[ ]: 10

In [131…
```
num1
```

Out[131… [1, 'hi', 'hallo', 'nit']

In [132…
```
min(num1)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[132], line 1
----> 1 min(num1)

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

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

Out[ ]: 10

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

In [135…
```
nums
```

Out[135… [10]

In [136…
```
l = [1,2,3]
l
```

Out[136… [1, 2, 3]

In [137…
```
l[0] = 100
l
```

Out[137… [100, 2, 3]

In [ ]:
```
tup = (15,25,35)
tup
```

Out[ ]: (15, 25, 35)

In [139…
```
tup[0]
```

Out[139… 15

In [140…
```
tup[0] = 10
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[140], line 1
----> 1 tup[0] = 10

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

In [ ]:
```
S = {}
```

In [142…
```
s1 = {21,6,34,58,5}
```

In [143…
```
s1
```

Out[143… {5, 6, 21, 34, 58}

In [144…
```
s3= {50,35,53,'nit', 53}
```

In [145…
```
s3
```

Out[145… {35, 50, 53, 'nit'}

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

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[146], line 1
----> 1 s1[1] #as we dont have proper sequencing thats why indexing not subscriptable

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

In [ ]:
```
data = {1:'apple', 2:'banana',4:'orange'}
data
```

Out[ ]: {1: 'apple', 2: 'banana', 4: 'orange'}

In [148…
```
data[4]
```

Out[148… 'orange'

In [149…
```
data[3]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[149], line 1
----> 1 data[3]

KeyError: 3
```

In [150…
```
data.get(2)
```

Out[150…
```
'banana'
```

In [151…
```
data.get(3)
```

In [152…
```
print(data.get(3))
```

```
None
```

In [153…
```
data.get(1,'Not Fount')
```

Out[153…
```
'apple'
```

In [154…
```
data.get(3,'Not Found')
```

Out[154…
```
'Not Found'
```

In [155…
```
data[5] = 'five'
```

In [156…
```
data
```

Out[156…
```
{1: 'apple', 2: 'banana', 4: 'orange', 5: 'five'}
```

In [157…
```
del data [5]
```

In [158…
```
data
```

Out[158…
```
{1: 'apple', 2: 'banana', 4: 'orange'}
```

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

In [160…
```
prog
```

Out[160…
```
{'python': ['vscode', 'pycharm'],
 'machine learning': 'sklearn',
 'datascience': ['jupyter', 'spyder']}
```

In [161…
```
prog['python']
```

Out[161…
```
['vscode', 'pycharm']
```

In [162…
```
prog['machine learning']
```

Out[162…
```
'sklearn'
```

In [163…
```
prog['datascience']
```

Out[163…
```
['jupyter', 'spyder']
```

In [164…
```
help()
```

```
Welcome to Python 3.11's help utility! If this is your first time using
Python, you should definitely check out the tutorial at
https://docs.python.org/3.11/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 [166…
```
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 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__(...)
 |      x.__getitem__(y) <==> x[y]
 |
 |  __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 signature.
|
|  ----------------------------------------------------------------------
|  Data and other attributes defined here:
|
|  __hash__ = None
```

In [167…
```
2 + 3
```

Out[167…
```
5
```

In [168…
```
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
 |      UnraisableHookArgs
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return 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 signature.
```

In [ ]:
```
num = 5
id(num)
```

Out[ ]:
```
140733999903656
```

In [ ]:
```
name = 'nit'
id(name)
```

Out[ ]:
```
2032155896496
```

In [171…
```
a = 10
id(a)
```

Out[171…
```
140733999903816
```

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

In [173…
```
id(b)
```

Out[173…
```
140733999903816
```

In [174…
```
id(10)
```

Out[174…
```
140733999903816
```

In [175…
```
k = 10
id(k)
```

Out[175…
```
140733999903816
```

In [ ]:
```
a = 20
id(a)
```

Out[ ]:
```
140733999904136
```

In [177…
```
id(b)
```

Out[177…
```
140733999903816
```

In [ ]:
```
PI = 3.14
PI
```

Out[ ]:
```
3.14
```

In [179…
```
PI = 3.15
PI
```

Out[179…
```
3.15
```

In [180…
```
type(PI)
```

Out[180…
```
float
```

In [181…
```
w = 2.5
type(w)
```

Out[181…
```
float
```

In [182…
```
(a)
```

Out[182…
```
20
```

In [ ]:
```
w2 = 2 + 3j
type(w2)
```

Out[ ]:
```
complex
```

In [ ]:
```
a = 5.6
b = int(a)
```

In [185…
```
b
```

Out[185…
```
5
```

In [186…
```
type(b)
```

Out[186…
```
int
```

In [187…
```
type(a)
```

Out[187…
```
float
```

In [188…
```
k = float(b)
```

In [189…
```
k
```

Out[189… 5.0

In [190…
```
print(a)
print(b)
print(k)
```

```
5.6
5
5.0
```

In [191…
```
k1 = complex(b,k)
```

In [192…
```
print(k1)
type(k1)
```

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

Out[192… complex

In [193…
```
b < k
```

Out[193… False

In [194…
```
condition = b<k
condition
```

Out[194… False

In [195…
```
type(condition)
```

Out[195… bool

In [196…
```
int(True)
```

Out[196… 1

In [197…
```
int(False)
```

Out[197… 0

In [198…
```
l = [1,2,3,4]
print(l)
type(l)
```

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

Out[198… list

In [199…
```
s = {1,2,3,4}
s
```

Out[199… {1, 2, 3, 4}

In [200…
```
type(s)
```

Out[200… set

In [ ]:

```
s1 = {1,2,3,4,4,3,11}
s1
```

Out[ ]: {1, 2, 3, 4, 11}

In [202…
```
t = (10,20,30)
t
```

Out[202… (10, 20, 30)

In [203…
```
type(t)
```

Out[203… tuple

In [ ]:
```
str = 'nit'
type(str)
```

Out[ ]: str

In [205…
```
st = 'n'
type(st)
```

Out[205… str

range()

In [206…
```
r = range(0,10)
r
```

Out[206… range(0, 10)

In [207…
```
type(r)
```

Out[207… range

In [ ]:
```
list(range(10,20))
```

Out[ ]: [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]

In [209…
```
r1 = list(r)
r1
```

Out[209… [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [ ]:
```
even_number = list(range(2,10,2))
even_number
```

Out[ ]: [2, 4, 6, 8]

In [211…
```
d= {1:'one', 2:'two', 3:'three'}
d
```

Out[211… {1: 'one', 2: 'two', 3: 'three'}

In [212…
```
type(d)
```

Out[212… dict

In [ ]:
```
d.keys()
```

Out[ ]: dict_keys([1, 2, 3])

In [214…
```
d.values()
```

Out[214… dict_values(['one', 'two', 'three'])

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

Out[ ]: 'two'

In [ ]:
```
d.get(2)
```

Out[ ]: 'two'

In [217…
```
x1, y1 = 10, 5
```

In [219…
```
x1  + y1
```

Out[219… 15

In [220…
```
x1 - y1
```

Out[220… 5

In [221…
```
x1 * y1
```

Out[221… 50

In [222…
```
x1 / y1
```

Out[222… 2.0

In [223…
```
x1 // y1
```

Out[223… 2

In [224…
```
x1 % y1
```

Out[224… 0

In [225…
```
x1 ** y1
```

Out[225… 100000

In [226…
```
x2 = 3
y2 = 3

x2 ** y2
```

Out[226… 27

In [227…
```
x = 2
```

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

In [229…
```
x
```

Out[229…
```
4
```

In [230…
```
x += 2
x
```

Out[230…
```
6
```

In [231…
```
x += 2
x
```

Out[231…
```
8
```

In [232…
```
x *= 2
```

In [233…
```
x
```

Out[233…
```
16
```

In [234…
```
x -= 2
```

In [235…
```
x
```

Out[235…
```
14
```

In [236…
```
x /= 2
x
```

Out[236…
```
7.0
```

In [237…
```
x //= 2
x
```

Out[237…
```
3.0
```

In [ ]:
```
a, b = 5,6
print(a)
print(b)
```

```
5
6
```

In [239…
```
a = 5
b = 6
print(a)
print(b)
```

```
5
6
```

In [240…
```
a
```

Out[240… 5

In [241…
```
b
```

Out[241… 6

In [ ]:
```
n = 7
n
```

Out[ ]: 7

In [243…
```
m = -(n)
m
```

Out[243… -7

In [244…
```
n
```

Out[244… 7

In [245…
```
-n
```

Out[245… -7

In [246…
```
a = 5
b = 6
```

In [247…
```
a<b
```

Out[247… True

In [248…
```
a>b
```

Out[248… False

In [250…
```
a == b
```

Out[250… False

In [251…
```
a != b
```

Out[251… True

In [ ]:
```
b = 5
```

In [253…
```
a == b
```

Out[253… True

In [254…
```
a
```

Out[254… 5

In [255…
```
b
```

Out[255… 5

In [256…
```
a > b
```

Out[256… False

In [257…
```
a >= b
```

Out[257… True

In [258…
```
a <= b
```

Out[258… True

In [259…
```
a < b
```

Out[259… False

In [260…
```
a>b
```

Out[260… False

In [261…
```
b = 7
```

In [262…
```
a != b
```

Out[262… True

In [263…
```
a = 5
b = 4
```

In [ ]:
```
a < 8 and b < 5
```

Out[ ]: True

In [265…
```
a < 8 and b < 2
```

Out[265… False

In [266…
```
a < 8 or b < 2
```

Out[266… True

In [267…
```
a>8 or b<2
```

Out[267… False

In [268…
```
x = False
x
```

Out[268… False

In [ ]:
```
not x
```

Out[ ]: True

In [270…
```
x = not x
x
```

Out[270… True

In [271…
```
x
```

Out[271… True

In [272…
```
not x
```

Out[272… False

In [273…
```
25
```

Out[273… 25

In [274…
```
bin(25)
```

Out[274… '0b11001'

In [275…
```
int(0b11001)
```

Out[275… 25

In [276…
```
bin(30)
```

Out[276… '0b11110'

In [277…
```
int(0b11110)
```

Out[277… 30

In [278…
```
int(0b11001)
```

Out[278… 25

In [279…
```
oct(25)
```

Out[279… '0o31'

In [280…
```
int(0o31)
```

Out[280… 25

In [281…
```
int(0b11110)
```

Out[281… 30

In [282…
```
0o31
```

Out[282… 25

In [283…
```
0b11001
```

```
Out[283…  25

In [284…  int(0b11001)

Out[284…  25

In [285…  bin(7)

Out[285…  '0b111'

In [286…  oct(25)

Out[286…  '0o31'

In [287…  0o31

Out[287…  25

In [288…  int(0o31)

Out[288…  25

In [289…  hex(25)

Out[289…  '0x19'

In [290…  0x19

Out[290…  25

In [291…  hex(16)

Out[291…  '0x10'

In [292…  0xa

Out[292…  10

In [293…  0xb

Out[293…  11

In [294…  hex(1)

Out[294…  '0x1'

In [295…  hex(25)

Out[295…  '0x19'

In [296…  0x19

Out[296…  25

In [297…  0x15
```

Out[297… 21

In [298…
```
a = 5
b = 6
```

In [299…
```
a = b
b = a
```

In [300…
```
print(a)
print(b)
```

```
6
6
```

In [ ]:
```
a1 = 7
b1 = 8
```

In [302…
```
temp = a1
a1 = b1
b1 = temp
```

In [303…
```
print(a1)
print(b1)
```

```
8
7
```

In [304…
```
a2 = 5
b2 = 6
```

In [ ]:
```
a2 = a2 + b2
b2 = a2 - b2
a2 = a2 - b2
```

In [306…
```
print(a2)
print(b2)
```

```
6
5
```

In [307…
```
0b110
```

Out[307… 6

In [308…
```
0b101
```

Out[308… 5

In [309…
```
print(0b110)
print(0b101)
```

```
6
5
```

In [310…
```
print(0b101)
print(0b110)
```

```
5
6
```

In [ ]:
```
print(bin(11))
print(0b1011)
```

```
0b1011
11
```

In [312…
```
print(a2)
print(b2)
```

```
6
5
```

In [ ]:
```
a2 = a2 ^ b2
b2 = a2 ^ b2
a2 = a2 ^ b2
```

In [314…
```
print(a2)
print(b2)
```

```
5
6
```

In [315…
```
a2, b2
```

Out[315…
```
(5, 6)
```

In [ ]:
```
a2 , b2 = b2, a2
```

In [317…
```
print(a2)
print(b2)
```

```
6
5
```

In [318…
```
print(bin(12))
print(bin(13))
```

```
0b1100
0b1101
```

In [319…
```
0b1101
```

Out[319…
```
13
```

In [320…
```
0b1100
```

Out[320…
```
12
```

In [ ]:
```
~12
```

Out[ ]:
```
-13
```

In [322…
```
~46
```

Out[322…
```
-47
```

In [323…
```
~54
```

Out[323…
```
-55
```

In [324…
```
~10
```
Out[324…
```
-11
```
In [325…
```
12 & 13
```
Out[325…
```
12
```
In [326…
```
12 | 13
```
Out[326…
```
13
```
In [327…
```
1 & 0
```
Out[327…
```
0
```
In [328…
```
1 | 0
```
Out[328…
```
1
```
In [329…
```
bin(13)
```
Out[329…
```
'0b1101'
```
In [330…
```
print(bin(35))
print(bin(40))
```
```
0b100011
0b101000
```
In [ ]:
```
35 & 40
```
Out[ ]:
```
32
```
In [332…
```
35 | 40
```
Out[332…
```
43
```
In [ ]:
```
12 ^ 13
```
Out[ ]:
```
1
```
In [334…
```
print(bin(25))
print(bin(30))
```
```
0b11001
0b11110
```
In [335…
```
25^30
```
Out[335…
```
7
```
In [336…
```
bin(7)
```
Out[336…
```
'0b111'
```
```
bin(25)
```

In [337…

Out[337… '0b11001'

In [338…
```
bin(30)
```

Out[338… '0b11110'

In [339…
```
0b00111
```

Out[339… 7

In [340…
```
bin(10)
```

Out[340… '0b1010'

In [341…
```
10<<1
```

Out[341… 20

In [342…
```
10<<2
```

Out[342… 40

In [343…
```
bin(10)
```

Out[343… '0b1010'

In [344…
```
10<<1
```

Out[344… 20

In [345…
```
10<<2
```

Out[345… 40

In [ ]:
```
10<<2
```

Out[ ]: 40

In [347…
```
10<<3
```

Out[347… 80

In [348…
```
bin(20)
```

Out[348… '0b10100'

In [ ]:
```
20<<4
```

Out[ ]: 320

In [350…
```
bin(10)
```

Out[350… '0b1010'

In [351…
```
10 >> 1
```
Out[351…
```
5
```
In [352…
```
10>>2
```
Out[352…
```
2
```
In [353…
```
10>>3
```
Out[353…
```
1
```
In [354…
```
bin(20)
```
Out[354…
```
'0b10100'
```