Meeting 2 Live

December 1, 2025

[1]:
```
"foo" == 'foo'
```

[1]: True

[2]:
```
1 < 2
```

[2]: True

[3]:
```
2 < 1
```

[3]: False

[4]:
```
True and False
```

[4]: False

[5]:
```
True or False
```

[5]: True

[7]:
```
True or (False and False)
```

[7]: True

[8]:
```
(True or False) and False
```

[8]: False

[9]:
```
('foo' == "foo" and True and 1 < 2)
```

[9]: True

[10]:
```
('foo' == "foo" and True and 1 > 2)
```

[10]: False

[11]:
```
not True
```

[11]: False

[12]:
```
not False
```

[12]: True

[13]:
```
not not False
```

[13]: False

[14]:
```
def is_lecturer(name):
    if name == 'Ben':
        return True
    else:
        return False

is_lecturer("Simon")
```

[14]: False

[15]:
```
is_lecturer("Ben")
```

[15]: True

[16]:
```
def is_lecturer(name):
    if name == 'Ben':
        return True
    return False
```

[17]:
```
is_lecturer("Ben")
```

[17]: True

[18]:
```
is_lecturer("Simon")
```

[18]: False

[19]:
```
name = 'Ben'
```

[20]:
```
name == 'Ben'
```

[20]: True

[21]:
```
name == 'Simon'
```

[21]: False

[22]:
```
def is_lecturer(name):
    return name == 'Ben'
```

[23]:
```
is_lecturer("Ben")
```

[23]: True

[24]:
```
is_lecturer("Simon")
```

[24]: False

[25]:
```
def is_lecturer(name: str) -> bool:
    return name == 'Ben'
```

[26]:
```
is_lecturer(4)
```

[26]: False

[27]:
```
def welcome_with_time(name: str, time_of_day: str) -> None:
    if time_of_day == 'am':
        print(f"Good morning, {name}!")
    if time_of_day == 'pm':
        print(f"Good afternoon, {name}!")
```

[28]:
```
welcome_with_time("Ben", "am")
```

```
Good morning, Ben!
```

[29]:
```
welcome_with_time("Ben", "pm")
```

```
Good afternoon, Ben!
```

[30]:
```
def welcome_with_time(name: str, time_of_day: str) -> None:
    if time_of_day == 'am':
        print(f"Good morning, {name}!")
    elif time_of_day == 'pm':
        print(f"Good afternoon, {name}!")
    else:
        print(f"Good day, {name}!")
```

[31]:
```
welcome_with_time("Ben", "pm")
```

```
Good afternoon, Ben!
```

[32]:
```
welcome_with_time("Ben", "asdfasdf")
```

```
Good day, Ben!
```

[33]:
```
result = 1
i = 5
while i > 0:
    result *= i
    i -= 1
print(result)
```

```
120
```

[34]:
```
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

[36]:
```
result = 1
for i in range(6):
    result *= i
print(result)
```

```
0
```

[37]:
```
range?
```

```
Init signature: range(self, /, *args, **kwargs)
Docstring:
range(stop) -> range object
range(start, stop[, step]) -> range object

Return an object that produces a sequence of integers from start (inclusive)
to stop (exclusive) by step.  range(i, j) produces i, i+1, i+2, …, j-1.
start defaults to 0, and stop is omitted!  range(4) produces 0, 1, 2, 3.
These are exactly the valid indices for a list of 4 elements.
When step is given, it specifies the increment (or decrement).
Type:           type
Subclasses:
```

[38]:
```
result = 1
for i in range(1, 5 + 1):
    result *= i
print(result)
```

```
120
```

[39]:
```
for i in range(2, 6, 2):
    print(i)
```

```
2
4
```

[40]:
```
my_str = "Hello World"
```

[41]:
```
for c in my_str:
    print(c)
```

```
H
e
l
l
o

W
o
r
l
d
```

[42]:
```
for c in my_str:
    print(c)
    if c == "l":
        break
```

```
H
e
l
```

[43]:
```
for c in my_str:
    if c == "l":
        continue
    print(c)
```

```
H
e
o

W
o
r
d
```

[44]:
```
letters = ['A', 'B', 'C']
```

[45]:
```
letters[0]
```

[45]:
```
'A'
```

[46]:
```
letters[0] = 'a'
```

[47]:
```
letters
```

[47]:
```
['a', 'B', 'C']
```

[48]:
```
letters[-1]
```

[48]:
```
'C'
```

[49]:
```
letters[1:]
```

[49]:
```
['B', 'C']
```

[50]:
```
all_letters = list('abcdefgh')
```

[51]:
```
all_letters
```

[51]:
```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']
```

[52]:
```
all_letters[::2]
```

[52]:
```
['a', 'c', 'e', 'g']
```

[53]:
```
all_letters.append('i')
```

[54]:
```
all_letters
```

[54]:
```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']
```

[55]:
```
all_letters = all_letters + ['j', 'k', 'l']
```

[56]:
```
all_letters
```

[56]:
```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l']
```

[57]:
```
all_letters += ['m', 'n']
```

[58]:
```
all_letters
```

[58]:
```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n']
```

[59]:
```
dir(all_letters)
```

```
[59]: ['__add__',
       '__class__',
       '__class_getitem__',
       '__contains__',
       '__delattr__',
       '__delitem__',
       '__dir__',
       '__doc__',
       '__eq__',
       '__format__',
       '__ge__',
       '__getattribute__',
       '__getitem__',
       '__gt__',
       '__hash__',
       '__iadd__',
       '__imul__',
       '__init__',
       '__init_subclass__',
       '__iter__',
       '__le__',
       '__len__',
       '__lt__',
       '__mul__',
       '__ne__',
       '__new__',
       '__reduce__',
       '__reduce_ex__',
       '__repr__',
       '__reversed__',
       '__rmul__',
       '__setattr__',
       '__setitem__',
       '__sizeof__',
       '__str__',
       '__subclasshook__',
       'append',
       'clear',
       'copy',
       'count',
       'extend',
       'index',
       'insert',
       'pop',
       'remove',
       'reverse',
       'sort']
```

[60]:
```
all_letters.sort?
```

```
Signature: all_letters.sort(*, key=None, reverse=False)
Docstring:
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.
Type:      builtin_function_or_method
```

[61]:
```
all_letters
```

[61]:
```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n']
```

[62]:
```
"<sep>".join(all_letters)
```

[62]:
```
'a<sep>b<sep>c<sep>d<sep>e<sep>f<sep>g<sep>h<sep>i<sep>j<sep>k<sep>l<sep>m<sep>n
'
```

[63]:
```
"".join(all_letters)
```

[63]:
```
'abcdefghijklmn'
```

[65]:
```
students_pets = ['cat', 'dog', 'cat', 'dog', 'bunny']
```

[66]:
```
team_pets = ['cat', 'horse', 'elephant']
```

[67]:
```
both_pets = []
for sp in students_pets:
    if sp in team_pets:
        both_pets.append(sp)
print(both_pets)
```

```
['cat', 'cat']
```

[68]:
```
both_pets = []
for sp in students_pets:
    if sp in team_pets and sp not in both_pets:
        both_pets.append(sp)
print(both_pets)
```

```
['cat']
```

[70]:
```
set(students_pets)
```

[70]:
```
{'bunny', 'cat', 'dog'}
```

[71]:
```
set(students_pets) & set(team_pets)
```

[71]:
```
{'cat'}
```

[72]:
```
set(students_pets) | set(team_pets)
```

[72]:
```
{'bunny', 'cat', 'dog', 'elephant', 'horse'}
```

[73]:
```
s = set(students_pets)
t = set(team_pets)
```

[74]:
```
t - s
```

[74]:
```
{'elephant', 'horse'}
```

[75]:
```
s - t
```

[75]:
```
{'bunny', 'dog'}
```

[76]:
```
t ^ s
```

[76]:
```
{'bunny', 'dog', 'elephant', 'horse'}
```

[77]:
```
for animal in s - t:
    print(animal)
```

```
bunny
dog
```

[78]:
```
all_letters
```

[78]:
```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n']
```

[80]:
```
set(all_letters[::-1])
```

[80]:
```
{'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n'}
```

[81]:
```
foo = set()
```

[82]:
```
foo.add("Simon")
```

[83]:
```
foo.add("Ben")
```

[84]:
```
foo
```

[84]:
```
{'Ben', 'Simon'}
```

[85]:
```
set(all_letters[::-1])[::-1]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[85], line 1
----> 1 set(all_letters[::-1])[::-1]

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

[86]:
```
emails = {'ben': 'stock@cispa.de', 'simon': 'simon@simon.de'}
```

[87]:
```
emails['ben']
```

[87]:
```
'stock@cispa.de'
```

[88]:
```
emails['sebi']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[88], line 1
----> 1 emails['sebi']

KeyError: 'sebi'
```

[89]:
```
'sebi' in emails
```

[89]:
```
False
```

[90]:
```
'ben' in emails
```

[90]:
```
True
```

[91]:
```
'stock@cispa.de' in emails
```

[91]:
```
False
```

[92]:
```
for entry in emails:
    print(entry)
```

```
ben
simon
```

[93]:
```
for entry in emails.items():
    print(entry)
```

```
('ben', 'stock@cispa.de')
('simon', 'simon@simon.de')
```

[94]:
```
for entry in emails.items():
    name = entry[0]
    email = entry[1]
    print(name, email)
```

```
ben stock@cispa.de
simon simon@simon.de
```

[95]:
```
for entry in emails.items():
    name, email = entry
    print(name, email)
```

```
ben stock@cispa.de
simon simon@simon.de
```

[96]:
```
for name, email in emails.items():
    print(name, email)
```

```
ben stock@cispa.de
simon simon@simon.de
```

[97]:
```
"foo" in "foobar"
```

[97]: True

[ ]:
```

```