Meeting 4 Live

January 5, 2026

```
[1]: solved_tasks = {'ben': [1, 2, 3], 'simon': [3, 4, 5]}
```

```
[2]: solved_tasks['sebastian'].append(2)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[2], line 1
----> 1 solved_tasks['sebastian'].append(2)

KeyError: 'sebastian'
```

```
[3]: if 'sebastian' not in solved_tasks:
         solved_tasks['sebastian'] = []
     solved_tasks['sebastian'].append(2)
```

```
[4]: solved_tasks
```

```
[4]: {'ben': [1, 2, 3], 'simon': [3, 4, 5], 'sebastian': [2]}
```

```
[5]: from collections import defaultdict
```

```
[6]: solved_tasks = defaultdict(list)
```

```
[7]: solved_tasks['sebastian'].append(2)
```

```
[8]: solved_tasks
```

```
[8]: defaultdict(list, {'sebastian': [2]})
```

```
[9]: import random
```

```
[10]: random.randint(1, 100)
```

```
[10]: 76
```

```
[11]: random.randint(1, 100)
```

```
[11]: 41
```

```
[12]: import string
```

```
[13]: string.ascii_letters
```

```
[13]: 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

```
[14]: random.choice(string.ascii_letters)
```

```
[14]: 'w'
```

```
[15]: random.choice(string.ascii_letters)
```

```
[15]: 'J'
```

```
[16]: random.choices(string.ascii_letters, k=10)
```

```
[16]: ['t', 'C', 'Z', 'w', 'z', 'H', 'W', 'g', 'C', 'd']
```

```
[17]: "".join(random.choices(string.ascii_letters, k=10))
```

```
[17]: 'XulxeUEojp'
```

```
[18]: random.seed("foobar")
      print("".join(random.choices(string.ascii_letters, k=10)))
```

```
nsiCQQOaQE
```

```
[19]: random.seed("foobar")
      print("".join(random.choices(string.ascii_letters, k=10)))
```

```
nsiCQQOaQE
```

```
[20]: random.choices?
```

```
Signature: random.choices(population, weights=None, *, cum_weights=None, k=1)
Docstring:
Return a k sized list of population elements chosen with replacement.

If the relative weights or cumulative weights are not specified,
the selections are made with equal probability.
File:      /opt/conda/lib/python3.10/random.py
Type:      method
```

```
[21]: import requests
```

```
[22]: resp = requests.get("https://httpbin.org/ip")
```

```
[23]: resp
```

```
[23]: <Response [200]>
```

```
[24]: resp.text
```

```
[24]: '{\n  "origin": "134.96.225.205"\n}\n'
```

```
[25]: json.loads(resp.text)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[25], line 1
----> 1 json.loads(resp.text)

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

```
[26]: resp.json()
```

```
[26]: {'origin': '134.96.225.205'}
```

```
[27]: sess = requests.Session()
```

```
[28]: sess.get("https://httpbin.org/cookies/set/test_cookie/test_value")
```

```
[28]: <Response [200]>
```

```
[29]: sess.cookies
```

```
[29]: <RequestsCookieJar[Cookie(version=0, name='test_cookie', value='test_value',
port=None, port_specified=False, domain='httpbin.org', domain_specified=False,
domain_initial_dot=False, path='/', path_specified=True, secure=False,
expires=None, discard=True, comment=None, comment_url=None, rest={},
rfc2109=False)]>
```

```
[33]: sess.get("https://httpbin.org/cookies").json()
```

```
[33]: {'cookies': {'test_cookie': 'test_value'}}
```

```
[34]: requests.get("https://httpbin.org/cookies").json()
```

```
[34]: {'cookies': {}}
```

```
[39]: sess.get("https://executer.python-course.de/demo/json_post").json()
```

[39]:
```
{'nonce': 'OyTdbCYDar',
 'description': "Please POST this value back to the same URL, using JSON, as the
key 'nonce'"}
```

[40]:
```
class Vehicle:
    def __init__(self, wheelcount):
        self.wheelcount = wheelcount
        self.name = "Vehicle"

    def wheels(self):
        print(f"{self.name} has {self.wheelcount} wheels")
```

[41]:
```
veh = Vehicle(4)
veh.wheels()
```

```
Vehicle has 4 wheels
```

[42]:
```
class Bike(Vehicle):
    def __init__(self):
        super().__init__(2)
        self.name = "Bike"
```

[43]:
```
bik = Bike()
```

[44]:
```
bik.wheels()
```

```
Bike has 2 wheels
```

[45]:
```
class Car(Vehicle):
    def __init__(self):
        super().__init__(4)
        self.name = "Car"

    def honk(self):
        print("Hoooooooonk!")
```

[47]:
```
car = Car()
car.honk()
car.wheels()
```

```
Hoooooooonk!
Car has 4 wheels
```

[48]:
```
bik.honk()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[48], line 1
```

```
----> 1 bik.honk()

AttributeError: 'Bike' object has no attribute 'honk'
```

[49]:
```
class Boat(Vehicle):
    def __init__(self):
        super().__init__(0)
        self.name = "Boat"

    def wheels(self):
        print("Boats don't have wheels!")

boa = Boat()
boa.wheels()
```

```
Boats don't have wheels!
```

[ ]:
```
```