Магические методы

In [1]:
```python
class User:
    def __init__(self, name, email):
        self.name = name
        self.email = email

    def get_email_data(self):
        return {
            'name': self.name,
            'email': self.email
        }


jane = User('Jane Doe', 'janedoe@example.com')

print(jane.get_email_data())
```

{'name': 'Jane Doe', 'email': 'janedoe@example.com'}

In [2]:
```python
class Singleton:
    instance = None

    def __new__(cls):
        if cls.instance is None:
            cls.instance = super().__new__(cls)

        return cls.instance


a = Singleton()
b = Singleton()

a is b
```

Out[2]: True

__str__

In [3]:
```python
class User:
    def __init__(self, name, email):
        self.name = name
        self.email = email


    def __str__(self):
        return '{} <{}>'.format(self.name, self.email)



jane = User('Jane Doe', 'janedoe@example.com')


print(jane)
```

Jane Doe <janedoe@example.com>

__hash__, __eq__

In [4]:
```python
class User:
    def __init__(self, name, email):
        self.name = name
        self.email = email

    def __hash__(self):
        return hash(self.email)

    def __eq__(self, obj):
        return self.email == obj.email


jane = User('Jane Doe', 'jdoe@example.com')
joe = User('Joe Doe', 'jdoe@example.com')


print(jane == joe)
```

True

```
In [5]:  print(hash(jane))
         print(hash(joe))
```

```
7885430882792781082
7885430882792781082
```

```
In [6]:  user_email_map = {user: user.name for user in [jane, joe]}

         print(user_email_map)
```

```
{<__main__.User object at 0x107415908>: 'Joe Doe'}
```

__getattr__, __getattribute__, __setattr__, __delattr__

```
In [7]:  class Researcher:
             def __getattr__(self, name):
                 return 'Nothing found :('

             def __getattribute__(self, name):
                 return 'nope'


         obj = Researcher()

         print(obj.attr)
         print(obj.method)
         print(obj.DFG2H3J00KLL)
```

```
nope
nope
nope
```

```
In [8]:  class Researcher:
             def __getattr__(self, name):
                 return 'Nothing found :()\n'

             def __getattribute__(self, name):
                 print('Looking for {}'.format(name))
                 return object.__getattribute__(self, name)


         obj = Researcher()

         print(obj.attr)
         print(obj.method)
         print(obj.DFG2H3J00KLL)
```

```
Looking for attr
Nothing found :()

Looking for method
Nothing found :()

Looking for DFG2H3J00KLL
Nothing found :()
```

```
In [9]:  class Ignorant:
             def __setattr__(self, name, value):
                 print('Not gonna set {}!'.format(name))


         obj  = Ignorant()
         obj.math = True
```

```
Not gonna set math!
```

```
In [10]:    print(obj.math)
```

```
        ----------------------------------------------------------
        ------------------
        AttributeError                          Traceback (most
        recent call last)
        <ipython-input-10-677c3efbe80d> in <module>()
        ----> 1 print(obj.math)

        AttributeError: 'Ignorant' object has no attribute 'math'
```

```
In [11]:    class Polite:
                def __delattr__(self, name):
                    value = getattr(self, name)
                    print(f'Goodbye {name}, you were {value}!')

                    object.__delattr__(self, name)


            obj = Polite()

            obj.attr = 10
            del obj.attr
```

```
        Goodbye attr, you were 10!
```

__call__

```
In [12]:   class Logger:
               def __init__(self, filename):
                   self.filename = filename

               def __call__(self, func):
                   with open(self.filename, 'w') as f:
                       f.write('Oh Danny boy...')
                   return func


           logger = Logger('log.txt')

           @logger
           def completely_useless_function():
               pass
```

```
In [13]:   completely_useless_function()

           with open('log.txt') as f:
               print(f.read())
```

Oh Danny boy...

__add__

```
In [14]:   import random


           class NoisyInt:
               def __init__(self, value):
                   self.value = value

               def __add__(self, obj):
                   noise = random.uniform(-1, 1)
                   return self.value + obj.value + noise


           a = NoisyInt(10)
           b = NoisyInt(20)
```

```
In [15]:  for _ in range(3):
              print(a + b)
```

30.605646527205856
30.170967742734117
29.071231797981817

Написать свой контейнер с помощью __getitem__, __setitem__

```
In [16]:  class PascalList:
              def __init__(self, original_list=None):
                  self.container = original_list or []

              def __getitem__(self, index):
                  return self.container[index - 1]

              def __setitem__(self, index, value):
                  self.container[index - 1] = value

              def __str__(self):
                  return self.container.__str__()


          numbers = PascalList([1, 2, 3, 4, 5])

          print(numbers[1])
```

1

```
In [17]:  numbers[5] = 25

          print(numbers)
```

[1, 2, 3, 4, 25]