```
Дескрипторы
In [1]:
         class Descriptor:
             def __get__(self, obj, obj_type):
                 print('get')
             def __set__(self, obj, value):
                 print('set')
             def __delete__(self, obj):
                 print('delete')
          class Class:
             attr = Descriptor()
         instance = Class()
In [2]: instance.attr
         get
         instance.attr = 10
         set
         del instance.attr
         delete
 In [5]:
         class Value:
             def __init__(self):
                 self.value = None
             @staticmethod
             def _prepare_value(value):
                 return value * 10
             def __get__(self, obj, obj_type):
                 return self.value
             def __set__(self, obj, value):
                 self.value = self._prepare_value(value)
 In [6]:
         class Class:
             attr = Value()
         instance = Class()
         instance.attr = 10
         print(instance.attr)
         100
         Функции и методы
 In [7]:
         class Class:
             def method(self):
                 pass
         obj = Class()
         print(obj.method)
         print(Class.method)
         <bound method Class.method of <__main__.Class object at 0x10ee77278>>
         <function Class.method at 0x10ee3bea0>
 In [8]:
         class User:
             def __init__(self, first_name, last_name):
                 self.first_name = first_name
                 self.last_name = last_name
             @property
             def full_name(self):
                 return f'{self.first_name} {self.last_name}'
         amy = User('Amy', 'Jones')
         print(amy.full_name)
         print(User.full_name)
         Amy Jones
         object at 0x10ee7b598>
 In [9]:
         class Property:
             def __init__(self, getter):
                 self.getter = getter
             def __get__(self, obj, obj_type=None):
                 if obj is None:
                     return self
                 return self.getter(obj)
In [10]:
         class Class:
             @property
             def original(self):
                 return 'original'
             @Property
             def custom_sugar(self):
                 return 'custom sugar'
             def custom_pure(self):
                 return 'custom pure'
             custom_pure = Property(custom_pure)
In [11]: obj = Class()
         print(obj.original)
         print(obj.custom_sugar)
         print(obj.custom_pure)
         original
         custom sugar
         custom pure
In [12]:
         class StaticMethod:
             def __init__(self, func):
                 self.func = func
             def __get__(self, obj, obj_type=None):
                 return self.func
In [13]:
         class ClassMethod:
             def __init__(self, func):
                 self.func = func
             def __get__(self, obj, obj_type=None):
                 if obj_type is None:
                     obj_type = type(obj)
                 def new_func(*args, **kwargs):
                     return self.func(obj_type, *args, **kwargs)
                 return new_func
          __slots__
In [14]:
         class Class:
             __slots__ = ['anakin']
             def __init__(self):
                 self.anakin = 'the chosen one'
         obj = Class()
         obj.luke = 'the chosen too'
                                                  Traceback (most recent call last)
         AttributeError
         <ipython-input-14-66c0c798df1f> in <module>()
               8 obj = Class()
         ---> 10 obj.luke = 'the chosen too'
         AttributeError: 'Class' object has no attribute 'luke'
```