

Python: classes

`budnyjj@pirates.by`

- ▶ ООП
 - ▶ Инкапсуляция
 - ▶ Наследование
 - ▶ Полиморфизм
- ▶ Работа с исключениями

Объект — сущность, состоящая из

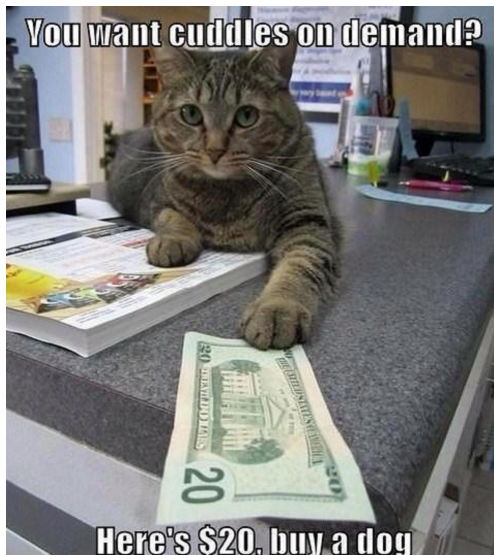
- ▶ данных (полей, атрибутов)
- ▶ методов обработки этих данных (методов объекта).

Кроме прочих своих достоинств, кот

- ▶ демонстрирует характерное поведение,
- ▶ реагирует на сообщения,
- ▶ наделён унаследованными реакциями,
- ▶ управляет своим, вполне независимым, внутренним состоянием.¹

¹Roger King, My cat is object-oriented

Пример объекта



Область применения ООП

Объектно-ориентированный подход хорош там,
где проект подразумевает развитие.

Объявление класса [dataClass.py]

```
1 class exampleDataClass:
2     '''A data example class'''
3
4     cls_var = 'cls_data'
5
6     def __init__(self):
7         self.obj_var = 'obj_data'
8
9     def f(self):
10         return self.obj_var
11
```

Атрибуты объекта [dataExample1.py]

```
1 import dataClass as dc
2
3 o1 = dc.exampleDataClass();
4 o2 = dc.exampleDataClass();
5
6 print('obj_var is object data member:')
7 print('BEFORE:')
8 print('o1.obj_var: ', o1.obj_var)
9 print('o2.obj_var:', o2.obj_var)
10
11 o1.obj_var = 'new_object_data'
12
13 print('AFTER:')
14 print('o1.obj_var:', o1.obj_var)
15 print('o2.obj_var:', o2.obj_var)
16
```

```
class exampleDataClass:
    cls_var = 'cls_data'

    def __init__(self):
        self.obj_var =
            'obj_data'

    def f(self):
        return self.obj_var
```


Атрибуты класса [dataExample2.py]

```
1 import dataClass as dc
2
3 o1 = dc.exampleDataClass();
4 o2 = dc.exampleDataClass();
5
6 print('Data is object data member:')
7 print('BEFORE:')
8 print('o1.cls_var:', o1.cls_var)
9 print('o2.cls_var:', o2.cls_var)
10
11 dc.exampleDataClass.cls_var = \
12     'new_object_data'
13
14 print('AFTER:')
15 print('o1.cls_var: ', o1.cls_var)
16 print('o2.cls_var:', o2.cls_var)
17
```

```
class exampleDataClass:
    cls_var = 'cls_data'

    def __init__(self):
        self.obj_var =
            'obj_data'

    def f(self):
        return self.obj_var
```

Изменение атрибутов объекта

```
getattr(object, name)  
setattr(object, name, value)  
delattr(object, name)  
hasattr(object, name)
```

```
var = object.name  
object.name = value  
del object.name
```

implemented by calling `getattr(object, name)`
and seeing whether it raises an exception or not

Изменение атрибутов объекта [dataExample3.py]

```
1 import dataClass as dc
2
3 o = dc.exampleDataClass()
4 print(dir(o))
5
6 del o.obj_var
7 o.new_data = 'some_data'
8 o.f2 = lambda x: x * 3
9
10 print(dir(o))
11 print(o.new_data)
12 print(o.f2('M'))
13
```

```
class exampleDataClass:
    cls_var = 'cls_data'

    def __init__(self):
        self.obj_var =
            'obj_data'

    def f(self):
        return self.obj_var
```

Инкапсуляция

```
class Simple:
    '''Simple class with private attribute'''
    def __init__(self):
        self.__private_attr = 20
        print(self.__private_attr)

s = Simple()
print(s.__private_attr)
```

Наследование

Синтаксис:

```
class Derived(Base):
```

```
    pass
```

```
class Derived(module_name.Base):
```

```
    pass
```

```
class Derived(Base1, Base2, Base3):
```

```
    pass
```

Особенности:

- ▶ Все методы — виртуальные
- ▶ Вызов метода базового класса: `Base.method()`
- ▶ Порядок поиска атрибута:
 1. Derived
 2. Base1, затем рекурсивно в базовых классах
 3. Base2, затем рекурсивно в базовых классах ...

- ▶ `type(object)`
- ▶ `isinstance(object, classinfo)`
- ▶ `issubclass(class, classinfo)`

Встроенные методы классов

- ▶ **__doc__**
- ▶ **__name__**
- ▶ **__module__**
- ▶ **__bases__**

Встроенные методы объектов

- ▶ **`__init__`**
- ▶ **`__dict__`**
- ▶ **`__class__`**
- ▶ **`__del__`**
- ▶ **`__cmp__`**
- ▶ **`__hash__`**
- ▶ **`__getattr__`**
- ▶ **`__setattr__`**
- ▶ **`__delattr__`**
- ▶ **`__call__`**

Эмуляция последовательностей

- ▶ `__len__`
- ▶ `__getitem__`
- ▶ `__setitem__`
- ▶ `__delitem__`
- ▶ `__getslice__`
- ▶ `__setslice__`
- ▶ `__delslice__`
- ▶ `__contains__`

```
import logging

class LoggingDict(dict):
    def __setitem__(self, key, value):
        logging.info(
            'Setting {k} to {v}'.\
            format(k=key, v=value))
    return super(LoggingDict, self).\
        __setitem__(key, value)

logging.basicConfig(level=logging.INFO)

ld = LoggingDict()
ld['a'] = 123
```

Приведение к базовым типам

- ▶ `__repr__`
- ▶ `__str__`
- ▶ `__oct__`
- ▶ `__hex__`
- ▶ `__complex__`
- ▶ `__int__`
- ▶ `__long__`
- ▶ `__float__`

Пример перегрузки оператора

```
class SignableMatrix:
    def __init__(self, array=[]):
        self.array = dc(array)
        dim_size = len(self.array)
        self.__row_signs = [False for i in range(dim_size)]
        self.__col_signs = [False for i in range(dim_size)]

    def __repr__(self):
        repr_str = 'Matrix: [\n'
        for row in self.array:
            repr_str += '    {}\n'.format(row)
        repr_str += ']\n'
        repr_str += 'Row signs:\n    {}'.format(self.__row_signs)
        repr_str += 'Column signs:\n    {}'.format(self.__col_signs)

        return repr_str


init_graph = [[1, 2, 3], [1, 2, 3], [1, 2, 3]]
init_matrix = SignableMatrix(init_graph)

print(init_matrix)
```

Остальные магические методы

- ▶ rafekettler.com/magicmethods.html
- ▶ docs.python.org/2/reference/datamodel.html#special-method-names

Обработка исключительных ситуаций — механизм языков программирования, предназначенный для описания реакции программы на возможные проблемы (исключения), которые могут возникнуть при выполнении программы и приводят к невозможности (бессмысленности) дальнейшей отработки программой её базового алгоритма. ²

²http://en.wikipedia.org/wiki/Exception_handling 

Пример: открытие файла

```
for arg in sys.argv[1:]:  
    try:  
        f = open(arg, 'r')  
    except IOError:  
        print('cannot open', arg)  
    else:  
        print(arg, 'has',  
              len(f.readlines()), 'lines')  
        f.close()
```

Пример: безопасное деление

```
1 def divide(x, y):  
2     try:  
3         result = x / y  
4     except ZeroDivisionError:  
5         print('division by zero!')  
6     else:  
7         print('result is', result)  
8     finally:  
9         print('executing finally clause')  
10
```

Как сгенерировать исключение

```
1 try:
2     raise Exception('spam', 'eggs')
3 except Exception as inst:
4     print(type(inst))
5     print(inst.args)
6     print(inst)
7     x, y = inst.args
8     print('x =', x)
9     print('y =', y)
10
```


Стандартные исключения

- ▶ Exception
- ▶ ImportError
- ▶ IOError
- ▶ AttributeError
- ▶ ValueError
- ▶ SyntaxError
- ▶ ZeroDivisionError
- ▶ ...

- ▶ docs.python.org/3/tutorial/classes.html
- ▶ docs.python.org/3/library/exceptions.html
- ▶ rafekettler.com/magicmethods.html
- ▶ docs.python.org/3/reference/datamodel.html#special-method-names
- ▶ www.ibm.com/developerworks/ru/library/l-python_part_6/
- ▶ www.ibm.com/developerworks/ru/library/l-python_part_7/

Спасибо за внимание!

- ▶ https://github.com/budnyjj/courses_python
- ▶ <https://vk.com/budnyjj>
- ▶ budnyjj@gmail.com