ООП в Python

Иванов Сергей (317)

Практикум на ЭВМ

07.11.2016

Питон любит ООП

Python изначально задумывался объектно-ориентированным языком

Всё есть объект (first-class object), а значит, со всем можно работать по одинаковым принципам

Питон любит лентяйничать

```
class SomeClass:
    pass
```

С этим уже можно работать:

```
a = SomeClass()
b = SomeClass()
c = a     #just reference

>>> a == b
False
>>> a == c
True
```

(!) Конструктор копирования за нас не сделают.
Пользуйтесь модулем copy

Питон любит атрибуты

Объектом являются как экземпляры класса, так и сам класс:

```
A.field = 179

>>> a.field
179
```

Кстати, у функций тоже есть атрибуты!

```
def f():
    return 179

f.spanish_inquisition = "unexpected?"
```

Питон любит словари

Классы и экземпляры классов по внутреннему устройству похожи на словари:

```
class Student:
    pass
s = Student()
s.name = 'Sergey'
Student.fears = ['exams']

>>> Student.__dict__
mappingproxy({..., 'fears': ['exams']})

>>> s.__dict__
{'name': 'Sergey'}
```

Питон любит крякать

Питон следует принципам "утиной типизации"(Duck Typing):

"If it looks like a duck, swims like a duck and quacks like a duck, then it probably is a duck."

```
def test(duck):
    duck.swim()
    duck.quack()
    return "It is a duck!"

>> test(Human())
"It is a duck!"
```

Питон любит инкапсуляцию

x	public	свободный доступ
_x	protected	предостережение пользователям (!) от использования вне производных классов
__x	private	атрибут *невидим* вне класса

Но если очень попросить, питон всё равно позволит:

```
class Student:
    def __tellSecret(self):
        print("I learn nothing")

Sergey = Student()
>>> Sergey._Student__tellSecret()
I learn nothing
```

Питон любит наследование

```
class CoolGuy:
    pass
class Student:
    def learn(self):
        print("I am learning")

class Sergey(CoolGuy, Student):
    pass

>>> S = Sergey()
>>> S.learn()
I am learning
```

Сначала питон пытается заставить учиться класс Sergey, потом - CoolGuy, и только после этих двух досадных промахов - Student.

Питон любит магию

__init__(self, [...)	конструктор - но он вызывается только когда экземпляр объекта уже создан
__new__(cls, [...)	процесс создания экземпляра (так называемый *классовый метод*)
__del__(self)	деструктор - но помните об автоматическом сборщике мусора!

Всё это так называемые "магические методы"(magic methods)

Питон любит переопределять операторы!.. при помощи магии

- Арифметические операторы
- Операторы сравнения
- Унарные операции и функции (например, округление)
- Операторы присваивания
- Операторы преобразования типов
- Операторы представления:

__str__(self)	как выглядит объект в строковом представлении
__hash__(self)	целочисленный хэш объектов класса
__bool__(self)	тру ли объект или фолс

- И много чего ещё

Любите питон!