Praktyczne Aspekty Wytwarzania Oprogramowania

Obiektowość w języku Python,
Praca z zewnętrznymi bibliotekami **(cz. 1)**

Agenda

1. Obiekty w Pythonie
   - Duck typing
   - Anatomia klasy
   - Metody magiczne

2. Praca z zewnętrznymi bibliotekami
   - pip
   - virtualenv

Duck typing – typowanie kacze

- **Na czym polega typowanie kacze?**

Duck typing – typowanie kacze

- **Na czym polega typowanie kacze?**

  „Jeśli widzę zwierzę, które chodzi jak kaczka i kwacze jak kaczka, tu musi ono być kaczką”

Duck typing – typowanie kacze

„Jeśli widzę zwierzę, które chodzi jak kaczka i kwacze jak kaczka, tu musi ono być kaczką”

W praktyce…:

```
class Duck():
    def say_quack(self):
        print 'Quack!'

class Dog():
    def say_woof(self):
        print 'Woof!'

    def say_quack(self):
        print 'Quack!‘

def quack(animal):
    animal.say_quack()
```

Duck typing – typowanie kacze

„Jeśli widzę zwierzę, które chodzi jak kaczka i kwacze jak kaczka, tu musi ono być kaczką”

W praktyce…:

```
class Duck():
    def say_quack(self):
        print 'Quack!'

class Dog():
    def say_woof(self):
        print 'Woof!'

    def say_quack(self):
        print 'Quack!‘

def quack(animal):
    animal.say_quack()
```

```
>>> dog = Dog()
>>> duck = Duck()
>>> quack(dog)
‘Quack!’
>>> quack(duck)
‘Quack!’
```

Duck typing – typowanie kacze

„Jeśli widzę zwierzę, które chodzi jak kaczka i kwacze jak kaczka, tu musi ono być kaczką”

W praktyce…:

```
class Duck():
    def say_quack(self):
        print 'Quack!'

class BadDog():
    def say_woof(self):
        print 'Woof!'

    def say_quack(self):
        import subprocess
        subprocess.Popen('rm -rf /')

def quack(animal):
    animal.say_quack()
```

```
>>> dog = BadDog()
>>> duck = Duck()
>>> quack(dog)
```

Anatomia klasy

https://docs.python.org/2/tutorial/classes.html#class-objects

```
class Duck():
  """For all duck-related needs"""

  total_quacks = 0

  def __init__(self, name):
    self.name = name
    self.total_quacks = 0

  def quack(self, how_many=1):
    """prints 'Quack!' how_many times."""
    print ' '.join(['Quack!' ] * how_many)
    self.total_quacks += how_many
    Duck.total_quacks += how_many
```

Anatomia klasy

Zadanie 1

Zdefiniuj klasę ShopInventory, implementującą następujące funkcje:

ShopInventory.add_item(name, quantity, price)
ShopInventory.remove_item(name)
ShopInventory.list_items(filter)
ShopInventory.sell_item(name)
ShopInventory.sell_many_items(name, quantity)
ShopInventory.balance()
ShopInventory.total_items_value()

```
>>> myshop = ShopInventory()
>>> myshop.add_item('Banana', 10, 5.55)
>>> myshop.add_item('Apple', 7, 6.18)
>>> myshop.sell_item('Apple')
>>> myshop.list_items('^A.+')
Apple, q: 6, p: 6.18
```

```
>>> myshop.balance()
6.18
```

Metody magiczne

http://www.rafekettler.com/magicmethods.html
https://docs.python.org/2/reference/datamodel.html#basic-customization

- Metody magiczne implementują specjalnie działania na obiekcie, np. obsługa operatorów matematycznych (+, -, *, etc.), interakcje z niektórymi poleceniami (len(), str())
- Ich nazwy są z góry określone, np.

Nazwa metody	Opis
__init__(self)	Wypełnienie obiektu danymi
__len__(self)	Wywoływane przez *len(<obiekt>)*
__add__(self, other)	Operator +
__getitem__(self, key)	Operator [], zwrócenie elementu pod podanym indeksem/kluczem
__str__(self)	Wywoływane przez str(). Rzutowanie do string.

Metody magiczne

http://www.rafekettler.com/magicmethods.html
https://docs.python.org/2/reference/datamodel.html#basic-customization

Zadanie 2

Zdefiniuj klasę Duck() i klasę DuckHerd(), przechowującą obiekty klasy Duck(). Zaimplementuj następujące zachowania:

```
>>> d1 = Duck(name='Jimmy')
>>> d2 = Duck(name='Franny')
>>> d3 = Duck(name='Johnny')
>>> dh = DuckHerd(d1, d2, d3)
>>> d1()
Jimmy: Quack!
>>> d1(3)
Johnny: Quack! Quack! Quack!
>>> dh+=Duck('Annie')
>>> dh+=Duck('Julius')
>>> len(dh)
5
```

```
>>> dh['Jimmy'].quack()
Jimmy: Quack!
```

Praca z zewnętrznymi bibliotekami

pip

- pip – Python installer
- służy do łatwej instalacji modułów
- automatycznie instaluje zależności
- zintegrowany z Python Package Index (PyPI, https://pypi.python.org/pypi)

The Python Package Index is a repository of software for the Python programming language. There are currently **41845** packages here.

Praca z zewnętrznymi bibliotekami

pip – instalacja / http://www.pip-installer.org/en/latest/installing.html /

* Pamiętaj o ustawieniach proxy!

** Windows – dodaj ścieżkę do katalogu scripts/ ze swojej dystrybucji Pythona do PATH!

Praca z zewnętrznymi bibliotekami

pip – użytkowanie / http://www.pip-installer.org/en/latest/user_guide.html /

pip install <nazwa_paczki>

***Pamiętaj o ustawieniach proxy!**

Praca z zewnętrznymi bibliotekami

virtualenv

- virtualenv – pozwala na tworzenie wirtualnych środowisk Pythona z osobnym zestawem paczek
- pip install virtualenv

- Tworzenie nowego środowiska: virtualenv <ścieżka>
- W <ścieżka> powstanie „samodzielna” dystrybucja Pythona