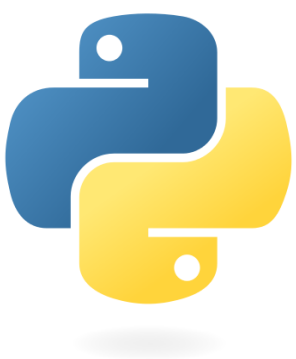


python™

Astronomiczny Kurs Studencki
cz. 2 - Napisy, listy i pokrewne. Importowanie.

Paweł Biernacki

8 marca 2012 r.

Plan prezentacji

```
>>> S = 'Spam'
>>> len(S) # Długość (length)
4
>>> S[0] # Pierwszy element S, indeksowanie od 0
'S'
>>> S[1] # Drugi od początku/lewej
'p'
>>> S[-1] # Pierwszy od końca
'm'
>>> S[-2] # Drugi od końca
'a'
>>> S # Pełny, 4-literowy napis
'Spam'
>>> S[1:3] # Wycinek od 1 do 2 (bez 3)
'pa'
```

```
>>> S
'Spam'
>>> S + 'xyz' # Łączenie
'Spamxyz'
>>> S # S jest niezmienione
'Spam'
>>> S * 8 # Powtórzenie
'SpamSpamSpamSpamSpamSpamSpamSpam'
```

Niezmienniczość napisów

```
>>> S
'Spam'
>>> S[0] = 'z'
# No i mamy błąd:
TypeError: 'str' object does not support item assignment
>>> S = 'z' + S[1:]
>>> S
'zpam'
```

Metody specyficzne dla typów

```
>>> S.find('pa') # Znajdź offset podnapisu
1
>>> S.replace('pa', 'XYZ') # Zastąpienie występowania
'SXYZm'
>>> S
'Spam'

>>> line = 'aaa,bbb,ccccc,dd'
>>> line.split(',') # Podziel przy przecinku na podnapisy
['aaa', 'bbb', 'ccccc', 'dd']
>>> S = 'spam'
>>> S.upper() # Konwersja wielkości
'SPAM'
```

Formatowanie napisów

```
>>> '%s, eggs, and %s' % ('spam', 'SPAM!')
# Wszystkie wersje pythona
'spam, eggs, and SPAM!'
>>> '{0}, eggs, and {1}'.format('spam', 'SPAM!')
# Python >= 2.6
'spam, eggs, and SPAM!'
```

Listy są bardzo podobne do napisów.

```
>>> L = [123, 'spam', 1.23]
>>> len(L)
3
>>> L.append('NI') # Dopisywanie elementu
>>> L
[123, 'spam', 1.23, 'NI']
>>> L.pop(2) # Usuwanie elementu
1.23
>>> L
[123, 'spam', 'NI']
```

Zagnieżdżanie

```
>>> M = [[1, 2, 3], # Macierz 3 x 3 jako zagnieżdżona lista
[4, 5, 6], # Kod może być ujęty w kilku liniach
[7, 8, 9]]
>>> M
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]

>>> M[1] # Rząd 2
[4, 5, 6]
>>> M[1][2] # Rząd 2, element 3
6
```

Zasięgi

```
>>> col2 = [row[1] for row in M]
>>> col2
[2, 5, 8]
```

Ale też bardziej skomplikowanie

```
>>> [row[1] for row in M if row[1] % 2 == 0]
# Wyrzuć nieparzyste
[2, 8]
>>> diag = [M[i][i] for i in [0, 1, 2]]
>>> diag
[1, 5, 9]
```

Podstawowe operacje

```
>>> f = open('data.txt', 'w')
# otwarcie nowego pliku do zapisu
>>> f.write('Hello\n')
6
>>> f.write('world\n')
# zwraca liczbę bitów zapiasnych (python 3.0+)
6
>>> f.close() # Zamyka plik i opróżnia bufor
```

Zawartość

```
>>> f = open('data.txt') # 'r' jest trybem domyślnym
>>> text = f.read() # wczytaj cały do zmiennej text
>>> text
'Hello\nworld\n'
>>> print(text) # print interpretuje zapis
Hello
world
>>> text.split() # Zawartość pliku to ZAWSZE NAPIS!!!
['Hello', 'world']
```

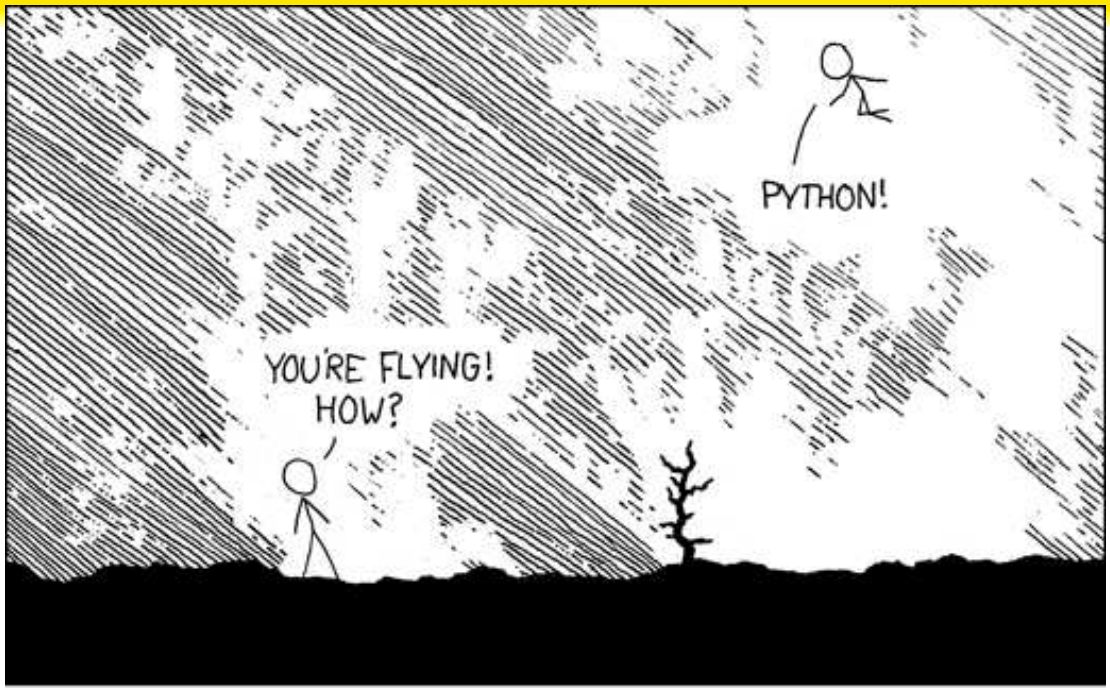
Własna klasa

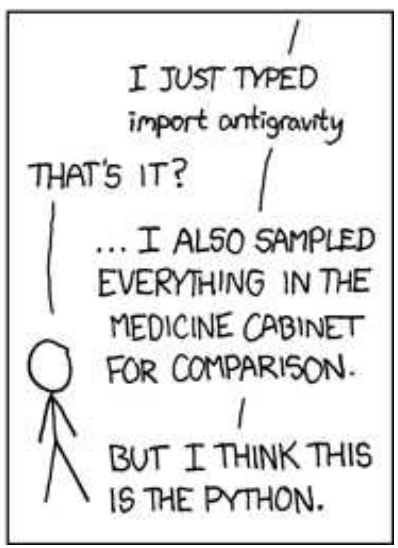
```
>>> class Worker:
def __init__(self, name, pay): # Zainicjalizuj gdy zdefiniu
self.name = name # self to nowy obiekt
self.pay = pay
def lastName(self):
return self.name.split()[-1] # Podziel na pustych
def giveRaise(self, percent):
self.pay *= (1.0 + percent) # Uaktualnij pensję
```

Przykład działania

```
>>> bob = Worker('Bob Smith', 50000)
>>> sue = Worker('Sue Jones', 60000)
>>> bob.lastName()
'Smith'
>>> sue.lastName()
'Jones'
>>> sue.giveRaise(.10)
>>> sue.pay
66000.0
```

Przypomnijmy o co chodzi...





Python dostracza wielu pakietów rozszerzających jego działanie. Do najpopularniejszych należą:

- math
- string
- random
- Numpy
- SciPy
- Matplotlib
- PyQt/PySide
- ...

Część z nich należy do samej dystrybucji pythona, a część trzeba doinstalować samodzielnie.

Jak korzystać z funkcji z pakietów?

```
1 import numpy
2 # uzycie funkcji: numpy.funkcja()
3
4 from random import *
5 # uzycie funkcji: funkcja_z_random()
6
7 from math import pi as pie
8 # uzycie: pie
```

Wszystkie powyższe importy powinny zawierać się między shebangiem a pierwszą z funkcji/definicji/etc.