NOKIA

Nokia Networks

Praktyczne Aspekty Wytwarzania Oprogramowania

PYTHON

Maciej Chmiel <maciej.chmiel@nsn.com>
Maciej Ogrodniczek <maciej.ogrodniczek@nsn.com>

Czym jest Python?

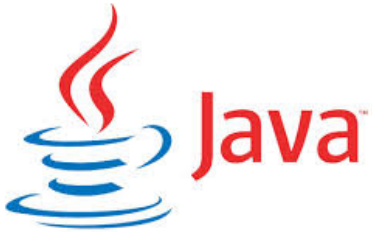
- */ˈpaɪθ(ə)n/ (paj-ton)*
- *interpretowany język programowania wysokiego poziomu*
- *Powszechnie używane wersje: 2.x i 3.x*

Nokia Siemens Networks

Python jest crossplatformowy

- *Czyli: dostępny dla wielu platform sprzętowych i programowych*

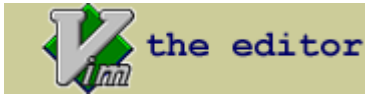
- *OS: Linux/Unix, OS/2, Mac OS, Amiga, Windows, Symbian, …*
- *VM: .NET, Java*



Nokia Siemens Networks

Środowiska programistyczne

- *Eclipse (PyDev)*
- *Netbeans IDE for Python*
- *Idle*
- *Visual Studio (dla implementacji IronPython)*
- *PyCharm*
- *Geany*
- *Komodo IDE*
- *NINJA-IDE*
- *VIM*
- *Notepad++*



Nokia Siemens Networks

Główne cechy języka Python

- ***Interpretowany (Kod źródłowy -> byte code -> VM)***
- ***Dynamicznie typowany***
- ***Wszystko jest obiektem – nawet funkcje i moduły***
- ***Automatyczne zarządzanie pamięcią (Garbage Collector)***
- ***Duży nacisk położony na czytelność i elegancję kodu***

- ***Bogata biblioteka standardowa i biblioteki zewnętrzne.***
  - PyPi – ponad 29 tys. dostępnych pakietów
    https://pypi.python.org/pypi

- ***Specyficzna „filozofia Pythona”.***

Filozofia Pythona

- ***Piękne jest lepsze niż brzydkie***
- ***Proste jest lepsze niż złożone***
- ***Czytelność się liczy***

PEP-0008
Style Guide for Python Code

- ***częściej czytamy kod źródłowy niż go piszemy***
- ***składnia języka powinna być jak najprostsza***

```
def bubblesort(lst):
    "Sorts lst in place and returns it."
    for passesLeft in range(len(lst)-1, 0, -1):
        for index in range(passesLeft):
            if lst[index] > lst[index + 1]:
                lst[index], lst[index + 1] = lst[index + 1], lst[index]
    return lst
```

Python

Perl

```
sub swap {
  @_[0, 1] = @_[1, 0];
}

sub bubble_sort {
  for ($i=$[; $i < $#_; ++$i) {
    for ($j=$[; $j < $#_; ++$j) {
      ($_[$j] > $ _[$j+1]) and swap($_[$j], $_[$j+1]);
    }
  }
}
```

Nokia Siemens Networks

Filozofia Pythona

- *Powinien być jeden – i najlepiej tylko jeden – oczywisty sposób na zrobienie danej rzeczy*
  - *Po co nam operator <>, skoro możemy != ?*
  - *Po co nam case, skoro możemy if .. elif .. else ?*
  - *[Python 3.x] Dlaczego print ma być wywoływany inaczej niż pozostałe funkcje?*
- *Jeśli rozwiązanie jest trudno wyjaśnić, to jest ono złym pomysłem*
- *Jeśli rozwiązanie jest łatwo wyjaśnić, to może ono być dobrym pomysłem*

Nokia Siemens Networks

Do czego można użyć

- *przetwarzanie tekstu (wyrażenia regularne, XML)*
- *zbieranie i przetwarzanie danych*
- *usługi internetowe (XML-RPC, SMTP, HTTP, …)*
- *operacje na plikach, socketach*
- *prototypowanie algorytmów*
- *zastosowania obliczeniowe i naukowe (SciPy, NumPy, Matplotlib)*
- *aplikacje internetowe (Django, Pylons, Flask)*
- *GUI (PyQt, PyGtk, WxPython)*
- *integracja z istniejącym oprogramowaniem*

Przykładowy kod

```
1  def dirdiff(l_dir, r_dir):
2      """
3      Compare contents of l_dir to contents of r_dir. Returns list of items
4      that differ by filenames.
5      """
6      l_paths = set()
7      r_paths = set()
8
9      for root, dirnames, filenames in os.walk(l_dir):
10         root = os.path.relpath(root, l_dir)
11         for name in dirnames + filenames:
12             l_paths.add(os.path.join(root, name))
13
14     for root, dirnames, filenames in os.walk(r_dir):
15         root = os.path.relpath(root, r_dir)
16         for name in dirnames + filenames:
17             r_paths.add(os.path.join(root, name))
18
19     return [os.path.join(l_dir, p) for p in l_paths.difference(r_paths)]
```


Nokia Siemens Networks

Interpreter

```
Python 2.7.3 (default, Apr 10 2013, 05:46:21)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

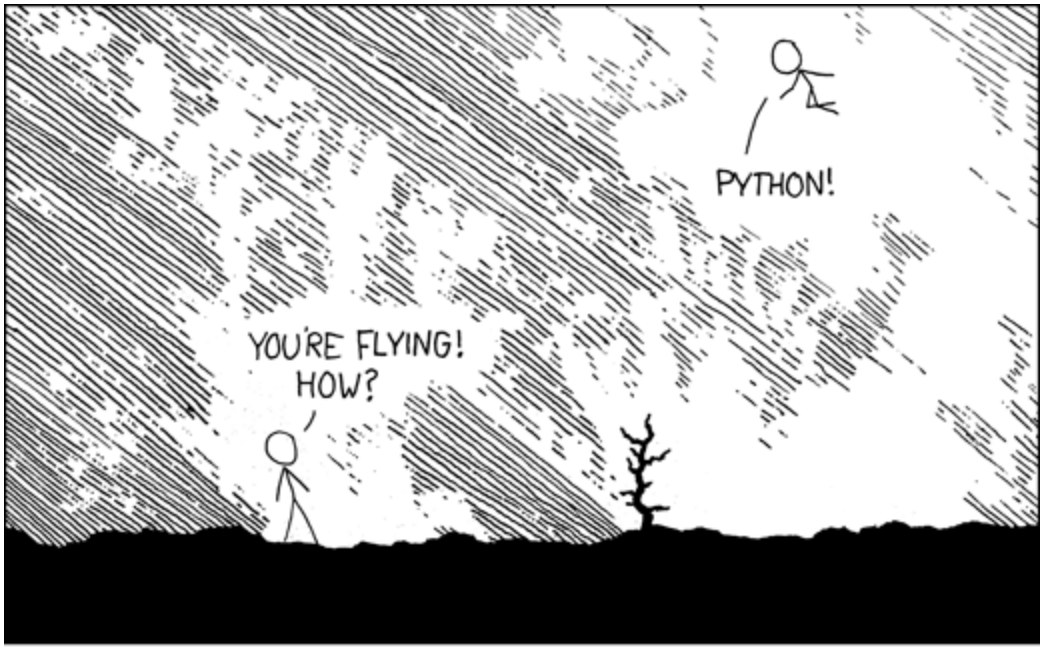
Wbudowane funkcje języka Python

		Built-in Functions		
abs()	divmod()	input()	open()	staticmethod()
all()	enumerate()	int()	ord()	str()
any()	eval()	isinstance()	pow()	sum()
basestring()	execfile()	issubclass()	print()	super()
bin()	file()	iter()	property()	tuple()
bool()	filter()	len()	range()	type()
bytearray()	float()	list()	raw_input()	unichr()
callable()	format()	locals()	reduce()	unicode()
chr()	frozenset()	long()	reload()	vars()
classmethod()	getattr()	map()	repr()	xrange()
cmp()	globals()	max()	reversed()	zip()
compile()	hasattr()	memoryview()	round()	__import__()
complex()	hash()	min()	set()	apply()
delattr()	help()	next()	setattr()	buffer()
dict()	hex()	object()	slice()	coerce()
dir()	id()	oct()	sorted()	intern()

Nokia Siemens Networks

Co teraz?

- *help()*
- *dir()*
- *Programujemy!*



http://xkcd.com/353/

Co teraz?

- *help()*
- *dir()*
- *Programujemy!*

```
Python 2.7.3 (default, Apr 10 2013, 05:46:21)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print 'Hello World!'
Hello World!
>>>
```

Nokia Siemens Networks

Podstawowe typy danych

Typy, które dziś poznamy:

- ***str – ‘ ‘ – string, czyli tekst***
- ***int – integer, czyli liczba całkowita***
- ***float – czyli liczba zmiennoprzecinkowa***
- ***list – [ ] – lista, uporządkowany zbiór obiektów***
- ***tuple – ( ) – krotka, uporządkowany zbiór obiektów (!)***

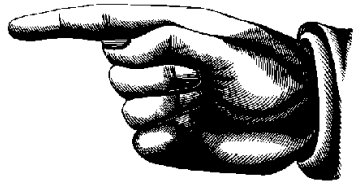
Typy, które zostaną przedstawione na następnych zajęciach

- ***dict – {} – słownik, zbiór nazwanych obiektów (pary klucz-wartość)***
- ***set – {} – zbiór obiektów***

O czym należy pamiętać programując w Pythonie

- ***Wszystko jest obiektem.***
- ***Dynamiczne typowanie – tak zwany duck typing.***
- ***Nazwy zmiennych są tylko etykietą przypisaną obiektom w pamięci (referencja do miejsca w pamięci).***

  Nazwa zmiennej = wielki paluch.



- ***Typy danych dzielimy na dwa rodzaje:***
  - Mutowalne (ang. mutable) - zmiana zawartości (stanu obiektu) jest możliwa po utworzeniu tego obiektu. Przykład: **list**.
  - Niemutowalne (ang. immutable) – zmiana zawartości (stanu obiektu) nie jest możliwa po jego utworzeniu. Przykład: **int**, **float**, **str**.

Podstawowe typy danych - ponownie

Typy, które dziś poznamy:

- ***str – ‘ ‘ – string, czyli tekst***
- ***int – integer, czyli liczba całkowita***
- ***float – czyli liczba zmiennoprzecinkowa***
- ***list – [ ] – lista, uporządkowany zbiór obiektów***
- ***tuple – ( ) – krotka, uporządkowany zbiór obiektów (!)***

Typy, które zostaną przedstawione na następnych zajęciach

- ***dict – {} – słownik, zbiór nazwanych obiektów (pary klucz-wartość)***
- ***set – {} – zbiór obiektów***

Definiowanie funkcji

```
def <nazwa_funkcji>(<parametry>):
    <ciało funkcji>
```


Nokia Siemens Networks

Definiowanie funkcji

```
def <nazwa_funkcji>(<parametry>):
    <ciało funkcji>
    return <obiekt>
```


Nokia Siemens Networks

Definiowanie funkcji

```
def <nazwa_funkcji>(<parametry>):
    """<docstring>"""
    <ciało funkcji>
    return <obiekt>
```


Nokia Siemens Networks

Instrukcje sterujące

Instrukcje sterujące służą sterowaniu kolejnością wykonania instrukcji zawartych w programie.

- ***if … elif … else***
- ***for … else***
- ***while …***

Nokia Siemens Networks