

# Kurs języka Python

## Wykład 14.

Marcin Młotkowski

25 stycznia 2010

- 1 Aplikacje WWW w Pythonie
  - Python i Apache
  - Pythonowe platformy aplikacyjne
- 2 Kącik porad
  - Dystrybucja aplikacji w U\*IX

# Python w aplikacjach serwerowych

- CGI;
- FastCGI, SCGI: ulepszone CGI;
- WSGI: Web Server Gateway Interface
- mod\_python

# Moduł mod\_python

## Moduł do serwera Apache

- Wykonywanie samodzielnych programów (publisher handler);
- interpretacja programów osadzonych w stronach html (PSP handler);
- szybki (szybszy niż CGI).

# Inne własności mod\_python

Co oferuje mod\_python

- Dostęp do żądań http wywołujących funkcje
- Obsługa formularzy
- Upload plików
- Ciasteczka
- Wsparcie dla sesji

# Przykład działania publishera

```
~/public_html/index.py
```

```
def index():  
    return "<html><body>A kuku</body></html>"  
  
def foo():  
    return "<html><body>foo</body></html>"
```

## Wywołanie funkcji `index`

```
http://.../  
http://.../index  
http://.../index.py
```

## Wywołanie funkcji `foo`

```
http://.../index/foo  
http://.../index.py/foo  
http://.../foo
```

# Widzialność i niewidzialność

## Widzialna zmienna, plik `main.py`

```
widoczne = """<html><body>Zmienna s</body></html>"""  
_niewidoczne = "<div>Niewidoczne</div>"
```

## Odwołania

```
http://.../main.py/widoczne  
http://.../main/widoczne  
http://.../main/_niewidoczne
```

# Osadzanie Pythona w HTML'u: Python Server Pages

## Wstawienie wyrażenia, plik `index.psp`

```
<html><body> <h2><%= 'A kuku!' %></h2>  
</html></body>
```

## Wstawienie instrukcji

```
<%  
import time  
weekday = time.strftime('%A', time.localtime(time.time()))  
message = 'Witaj! Mamy piękny dzień %s.' % weekday  
%>  
<html><body>  
<h2><%= message %></h2>  
</html></body>
```

## Dodatkowe informacje

- Każde wywołanie uruchamia odrębny *subinterpreter* Pythona z własną tablicą zmiennych;
- dostęp do wewnętrznej struktury Apache
- `mod_python` współpracuje z Django

# Spis

## Platformy do tworzenia serwisów webowych

- Django
- Cherry
- TurboGears
- web2py
- Pylons
- Google App Engine

# Spis

## Platformy do tworzenia serwisów webowych

- Django
- Cherry
- TurboGears
- web2py
- Pylons
- Google App Engine

## Z Object Publishing Environment — Zope

- serwer aplikacyjny
- silny nacisk na obiektowość, referencje odpowiadają obiektom, nie plikom
- własna obiektowa baza danych Zope Object Database
- framework do CMS'a Plone

# Typowa aplikacja

```
main.py lib/*.pyc moduly/*.pyo img/*
```

# Jak to zrobić

- Python umożliwia uruchamianie modułów z archiwów \*.zip
- Potrzebujemy kombinacji skryptu i archiwum \*.zip

# Krok pierwszy

## Kompilacja pliku main.py

```
$ python -O *.py
```

Uruchamia plik, kompilacja jest przy okazji.

## Kompilacja

```
$ python -O -c "import main"
```

# Krok pierwszy

## Kompilacja pliku main.py

```
$ python -O *.py
```

Uruchamia plik, kompilacja jest przy okazji.

## Kompilacja

```
$ python -O -c "import main"
```

## Krok drugi

### Utworzenie archiwum

```
$ zip aplikacja *.pyo lib/*.pyo
```

# Krok trzeci

## Skrypt run.uix

```
#!/bin/bash
exec python - $O $@ << END_START
import main
main.main()
END_START
```

## Utworzenie pliku

```
$ cat run.uix dist.zip > myapp
$ chmod +x myapp
```

## Uruchomienie

```
$ ./myapp
```

# A jak zrobić \*.exe

Moduł distutils i dodatek py2exe

setup.py

```
from distutils.core import setup
import sys, os, py2exe
name = sys.argv[1]
sys.argv[1] = 'py2exe'
sys.path.append(os.path.dirname(os.path.abspath(name)))
setup(name=name[:-3], scripts=[name])
```