Kurs rozszerzony języka Python

Inne języki

Marcin Młotkowski

26 stycznia 2018

Plan wykładu

Plan wykładu

Problemy łączenia dwóch języków

Zagadnienia

- problemy z różnymi typami danych (listy, kolekcje, napisy);
- przekazywanie argumentów i zwracanie wartości;
- tworzenie nowych wartości;
- obsługa wyjątków;
- zarządzanie pamięcią.

Dodanie do Pythona nowej funkcji

Zadanie

Moduł z funkcją obliczającą średnią arytmetyczną elementów listy.

Dodanie do Pythona nowej funkcji

Zadanie

Moduł z funkcją obliczającą średnią arytmetyczną elementów listy.

Elementy implementacji:

- plik nagłówkowy <Python.h>;
- implementacja funkcji;
- odwzorowanie funkcji w C na nazwę udostępnioną w Pythonie;
- funkcja inicjalizująca o nazwie *initnazwa_modułu*.

Implementacja funkcji

```
extern PyObject * mean(PyObject *, PyObject *);

PyObject * mean(PyObject * self, PyObject * args)
{
    int suma = 0, n, i;
    PyObject * res;
    PyObject * item;
    PyObject * lista;
    lista = PySequence_ITEM(args, 0);
    if (!PyList_Check(lista)) printf("To nie jest lista!\n");
    n = PyList_Size(lista);
```

Implementacja, cd.

cd. funkcji

```
    for (i = 0; i < n; i++) {
        item = PyList_GetItem(lista, i);
        if (!PyInt_Check(item)) continue;
        suma += PyInt_AsLong(item);
    }
    res = Py_BuildValue("i", suma/n);
    Py_INCREF(res);
    return res;
}
```

Opakowanie funkcji

```
#include <python3.5/Python.h>
```

```
extern PyObject * mean(PyObject *, PyObject *);

PyObject * mean(PyObject * self, PyObject * args)
{ ... }
```

Opakowanie funkcji

```
#include <python3.5/Python.h>
```

```
extern PyObject * mean(PyObject *, PyObject *);

PyObject * mean(PyObject * self, PyObject * args)
{ ... }
```

```
static PyMethodDef modulik[] = {
    { "mean", mean, METH_VARARGS, "Pierwsza funkcja" },
    { NULL, NULL, 0, NULL }
};

PyMODINIT_FUNC
initmodulik(void) {
    Py_InitModule("modulik", modulik);
```

Kompilacja i instalacja

setup.py

```
from distutils.core import setup, Extension
module1 = Extension('modulik', sources = ['test.c'])
setup(name = 'MyPackage',
        version = '1.0',
        description = 'Pakiet demonstracyjny',
        ext_modules = [module1])
```

Kompilacja i instalacja

```
$ python setup.py build
$ python setup.py install
```

Typy danych w Pythonie

Wszystko w Pythonie jest obiektem

Nowe wartości

Tworzenie nowych wartości Pythonowych

```
PyObject* PyInt_FromLong(long ival)
PyObject* Py_False
PyObject* Py_True
PyObject* PyList_New(Py_ssize_t len)
PyObject* PyString_FromString(const char *v)
```

```
PyObject *Py_BuildValue(char *format, ...);
```

```
Py_BuildValue()                                  None
Py_BuildValue("ss", "hello", "world")            ('hello', 'world')
Py_BuildValue("[i,i]", 123, 456)                 [123, 456]
Py_BuildValue("{s:i,s:i}",                       {'abc': 123, 'def': 456}
        "abc", 123, "def", 456)
```

Listy

dostęp do list

PyObject* PyList_GetItem(PyObject *list, Py_ssize_t index)
int PyList_SetItem(PyObject *list, Py_ssize_t index, PyObject *item)

Obiekty

Makra dostępu do pól obiektów

PyObject* PyObject_GetItem(PyObject *o, PyObject *key)

int PyObject_SetItem(PyObject *o, PyObject *key, PyObject *v)

Standardowe funkcje wieloargumentowe

Nagłowek funkcji

```
PyObject * foo(PyObject * self, PyObject * args)
```

Parsowanie argumentów

```
int PyArg_ParseTuple(PyObject *args, const char *format, ...)
```

Wpis w tabeli funkcji

```
{ "mean", mean, METH_VARARGS, "Pierwsza funkcja" },
```

Zarządzanie pamięcią

Mechanizm zarządzania pamięcią

- Każdy obiekt ma licznik odwołań zwiększany za każdym przypisaniem.
- Jeśli licznik jest równy zero obiekt jest usuwany z pamięci.
- W programach w C trzeba dbać o aktualizację licznika.

Zmiana licznika odwołań

Zwiększenie licznika

void Py_INCREF(PyObject *o)

Zmniejszenie licznika

void Py_DECREF(PyObject *o)

Zmiana licznika odwołań

Zwiększenie licznika

void Py_INCREF(PyObject *o)

Zmniejszenie licznika

void Py_DECREF(PyObject *o)

Przypomnienie

```
res = Py_BuildValue("i", suma/n);
Py_INCREF(res);
return res;
```

Trochę łatwiej

Biblioteka Boost:

+ łączenie Pythona z C++
+ łatwiejsza od C API
  - czasem nie da się ominąć C API (ale się rozwija)

Plan wykładu

Wykonanie programów Pythonowych

```
Py_Initialize();

PyRun_SimpleString("i = 2");
PyRun_SimpleString("i = i*i\nprint i");

Py_Finalize();
```

Wykonanie programów w pliku

```
Py_Initialize();
FILE * f = fopen("test.py", "r");
PyRun_SimpleFile(f, "test.py");
Py_Finalize();
```

Kompilacja

```
gcc -lpython3.5 test.c
```

Bezpośrednie wywoływanie funkcji Pythonowych

Deklaracja zmiennych

```
PyObject *pName, *pModule, *pArgs, *pFunc, *pValue;
```

Import modułu Pythonowego

```
Py_Initialize();
pName = PyString_FromString("modulik");
pModule = PyImport_Import(pName);
```

Pobranie funkcji z modułu

```
pFunc = PyObject_GetAttrString(pModule, "foo");
```

Wywołanie funkcji

```
pValue = PyObject_CallObject(pFunc, pArgs);
```

Plan wykładu

Kanoniczna implementacja

CPython

Podstawowa implementacja języka Python w C.

PyPy

- jit compilation;
- wysoka zgodność z Pythonem 2.7 i 3.5;
- możliwość dołączania własnego odśmiecacza pamięci;
- nieco inne zarządzanie pamięcią.

Stackless Python

- interpreter oparty na mikrowątkach realizowanych przez interpreter, nie przez kernel;
- dostępny w CPythonie jako *greenlet*.

Jython

Cechy Jythona

- implementacja Pythona na maszynę wirtualną Javy;
- kompilacja do plików `.class`;
- dostęp do bibliotek Javy;
- pełna zgodność z językiem Python (te same testy regresyjne).

IronPython

- Implementacja Pythona w środowisku Mono i .NET;
- zgodny z Pythonem 2.7.2, choć są niezgodności.

Python for S60

Implementacja Nokii na tefony komórkowe z systemem Symbian 60

- implementuje Python 2.2.2;
- dostęp do sprzętu (SMS'y, siła sygnału, nagrywanie video, wykonywanie i odbieranie połączeń);
- wsparcie dla GPRS i Bluetooth;
- dostęp do 2D API i OpenGL.

Plan wykładu

Lokalne środowisko Pythonowe

virtualenv

Tworzy w lokalnym katalogu pełną wersję środowsika pythonowego, którą można modyfikować niezależnie od głównej instalacji. Można mieć wiele takich wirtualnych środowisk.

Lokalne środowisko Pythonowe

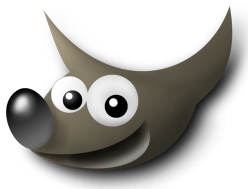
virtualenv

Tworzy w lokalnym katalogu pełną wersję środowsika pythonowego, którą można modyfikować niezależnie od głównej instalacji. Można mieć wiele takich wirtualnych środowisk.

```
$ virtualenv --system-site-packages $HOME/mojesrodowisko
$ cd $HOME/mojesrodowisko/
$ source bin/activate
```

Plan wykładu

Rozwinięcie skrótu



GNU Image Manipulation Program

Python w GIMP'ie

Gimp-Python

Jest to wrapper do biblioteki `libgimp`, umożliwiający implementację skryptów manipulujących obrazami.

Schemat skryptu

```
from gimpfu import *
def funkcja(argumenty, np. obraz):
    ...
register(nazwa, autor, )
main()
```

Instalacja i użycie

Instalacja

skrypty instaluje się w $HOME/gimp-xyz/plug-ins

Instalacja i użycie

Instalacja

skrypty instaluje się w $HOME/gimp-xyz/plug-ins

Wywołanie skryptu z linii poleceń

```
gimp --no-interface --batch
    '(python-fu-moja-funkcja RUN-NONINTERACTIVE
    2.71 3.1416) '(gimp-quit 1)'
```

gdzie `moja_funkcja` jest zarejestrowaną nazwą funkcji,

Budowa OpenOffice

UNO — Universal Network Object

Model komponentów obiektowych używany w OpenOffice, umożliwiający komunikację między obiektami przez sieć.

Python w OpenOffice

```
import uno
```

Przykład

```
local = uno.getComponentContext()
resolver = local.ServiceManager. createInstanceWithContext
    (''com.sun.star.bridge.UnoUrlResolver'', local)
context = resolver.resolve
    (''uno:socket,host=localhost,port=2002;urp;StarOffice.ComponentCo
desktop = context.ServiceManager.createInstanceWithContext
    (''com.sun.star.frame.Desktop'', context)
document = desktop.loadComponentFromURL
    (''file:///home/user/szekspir.odt'' ,''_blank'', 0, ())
```

Modyfikacja dokumentu

```
cursor = document.Text.createTextCursor()
document.Text.insertString
    (cursor, '"Być albo nie być, oto jest pytanie."', 0)
document.store()
```

Gdzie są skrypty

OpenOffice 2.0

windows:
C:\Documents and Settings\<current-user>\Application Data\OpenOffice.org 2.0\user\Scripts\python
unix:
$HOME/.openoffice.org.2.0/user/Scripts/python, /.openoffice.org.2.0/user/Scripts/python

OpenOffice 3.0

windows:
C:\Users\<current-user>\AppData\Roaming\OpenOffice.org\3\user\Scripts\python
unix:
$HOME/.openoffice.org/3/user/Scripts/python

Plan wykładu