Язык программирования

Лекция № 9

Владимир Владимирович Руцкий
rutsky.vladimir@gmail.com

План занятия

- Interoperability с Python
- Django (продолжение)

Interoperability **c** Python

- Python простой, но мощный язык, удобный для написания высокоуровневой логики
  - при наличии реализованных низкоуровневых инструментов, библиотек
- Interoperability компонентов/программ — совместная работа компонентов/программ
- Пример interoperability с Python:

  "Оборачивание" библиотек в Python модули — рассматривали на примере SDL/pygame
  - Boost.Python, SWIG

Native code

- *native* code/binary/executable — код, скомпилированный в машинные коды, исполняемые на определённом процессоре (и специфичные для него)
  - Код на Python парсится в байткод (.pyc), который интерпретируется программой-интерпретатором — Python не "нативный"
- Примерами нативного кода являются скомпилированные из C/C++ библиотеки (DLL) и выполняемые файлы (exe)
- Для нативного кода существуют соглашения о вызове функций, передачи аргументов, возвращению значений и т.п. (*calling conventions*)
  - Например, в каком порядке аргументы функций кладутся на стек в памяти, кто очищает стек, в каком регистре процессора будет результат функции

Вызов нативного кода из Python

- Для вызова нативного кода создана библиотека *libffi* (*Foreign Function Interface*)
- Python предоставляет обертку над этой библиотекой — модуль **`ctypes`**
- **`ctypes`** позволяет создавать прозрачные обёртки над функциями из нативных библиотек:

```
>>> import ctypes
>>> # Загрузка MyLibrary.DLL для вызова по соглашению stdcall
... my_library = ctypes.windll.LoadLibrary('MyLibrary')
>>> # Обращение к атрибутам my_library прозрачно транслируется в создание
... # обёрток для вызова функций из DLL
... my_library.test(5, 6) # Вызов функции с именем "test"
11
>>>
```

Возможности ctypes

- Загрузка нативных библиотек, выполняемых файлов (*LoadLibrary*)
- Получение Python-обёрток над функциями из нативных модулей
- Вызов и передача примитивных типов, структур, указателей, блоков памяти внутрь функций (а также получение результата)
- Создание *callback* функций на Python и передача их внутрь нативных модулей

Как и любой нативный код — **очень** хрупкая вещь

- легко обратиться "мимо" нужной памяти

Соглашения о вызове функций в *ctypes*

- **`ctypes`** предоставляет несколько обёрток для загрузки модулей с разными соглашениями о вызове функций
  - *ctypes.cdll* — соглашение *cdecl*
  - *ctypes.windll* — соглашение *stdcall*
  - *ctypes.oledll* — соглашение *stdcall*, плюс функции возвращают *HRESULT*
  - *ctypes.pydll* — соглашение *cdecl*, но используется для библиотек Python

Стандартные модули ОС

- Для стандартных библиотек в разных ОС предоставляются лениво загружаемые модули
- В Windows:

```
import ctypes
import math
import time

set_cursor_pos_func = ctypes.windll.user32.SetCursorPos

while True:
    t = time.clock()
    x = 300 + 300 * math.sin(5 * t)
    y = 200 + 200 * math.cos(6 * t)
    set_cursor_pos_func(int(x), int(y))
```

Передача аргументов

- Большая часть типов Python должна быть обёрнута специальными типами *c_types* для передачи в функции
  - *c_bool*, *c_char*, *c_long*, *c_float*, *c_double* и т.п.
- В GNU/Linux:

```
import ctypes
import math

libc = ctypes.CDLL('libc.so.6')
x = ctypes.c_double(math.pi)
res = libc.printf(b'Hello, %s! x = %.4f\n', b'User', x)
print(res)
```

```
$ python linux_printf.py
Hello, User! x = 3.1416
24
$
```

Automation **в** Windows

- Для interoperability в Windows была разработана система компонентов: COM, *Component Object Model*
  - Низкоуровневый, но объектно-ориентированный интерфейс
  - Поддерживается в том или ином виде большим количеством продуктов Microsoft
  - Позволяет связывать программы на C++, .NET, Visual Basic и др.
  - Есть биндинги для COM в разных языках, включая Python
- В GNU/Linux в качестве альтернативы COM часто используется dbus

COM **в** Python

Пример замены текста в документе Word используя обёртку над COM *pywin32*

```
import win32com.client

def search_replace_all(word_file, find_str, replace_str):
    '''Заменить строку find_str на строку replace_str в word_file'''
    wdFindContinue = 1
    wdReplaceAll = 2

    # Создаём COM-объект
    app = win32com.client.DispatchEx("Word.Application")
    app.Visible = 0
    app.DisplayAlerts = 0
    app.Documents.Open(word_file)

    # expression.Execute(FindText, MatchCase, MatchWholeWord,
    #   MatchWildcards, MatchSoundsLike, MatchAllWordForms, Forward,
    #   Wrap, Format, ReplaceWith, Replace)
    app.Selection.Find.Execute(find_str, False, False, False, False, False, \
        True, wdFindContinue, False, replace_str, wdReplaceAll)
    app.ActiveDocument.Close(SaveChanges=True)
    app.Quit()

f = 'c:/path/to/my/word.doc'
search_replace_all(f, 'string_to_be_replaced', 'replacement_str')
```

Automation **в** OpenOffice

- В OpenOffice/LibreOffice используется своя технология *UNO* для вызова функций

```
import socket
import uno

# soffice "-accept=socket,host=localhost,port=2002;urp;"

# Получаем контекст UNO
localContext = uno.getComponentContext()

resolver = localContext.ServiceManager.createInstanceWithContext(
                                "com.sun.star.bridge.UnoUrlResolver", localContext)
# Соединяемся с OpenOffice
ctx = resolver.resolve("uno:socket,host=localhost,port=2002;urp;StarOffice.ComponentContext
smgr = ctx.ServiceManager
# Получаем главное окно
desktop = smgr.createInstanceWithContext("com.sun.star.frame.Desktop", ctx)
# Получаем текущий открытый документ
model = desktop.getCurrentComponent()
# Получаем текст текущего документа
text = model.Text
# Создаём курсор
cursor = text.createTextCursor()
# Вставляем по курсору текст
text.insertString(cursor, "Hello World", 0)
```

Встраивание Python **в приложения**

- Python — популярный язык для скриптования логики приложения
- Часто встраивается внутрь приложений: GIMP, Blender, Maya, ArcGIS
- Примеры на следующей лекции

Django (**продолжение**)

- Перевод документации и туториала на русский язык: http://djbook.ru/rel1.6/
- Задача: написать гостевую книгу
  - Гостевая книга: веб страница со списком сообщений и возможностью добавлять сообщения
  - Каждое сообщение: автор, дата, текст сообщения
- Дополнительные задания:
  - Показывать сообщения постранично
  - Сортировка сообщений по дате или автору
  - Удаление сообщений
  - Редактирование сообщений