Programação com Interfaces Gráfica

Mario Benevides e Paulo Roma

Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brasil

Módulos

Agenda

Nesta Aula: Módulos (Python)

- Introdução: Módulos em Python
- Onde Módulos são Buscados (Path)
- Variável $\_\_name\_\_$
- Pacotes

Introdução: Módulos

- Módulos são programas feitos para serem reaproveitados em outros programas
- Eles tipicamente contêm funções, variáveis, classes e objetos que provêm alguma funcionalidade comum
- Por exemplo, já vimos que o módulo math contém funções matemáticas como sin, exp, etc, além da constante π
- Toda a biblioteca padrão do Python é dividida em módulos e pacotes
- Alguns dos mais comuns são: sys, os, time, random, re, shelve

Introdução: Módulos

- Qualquer programa que você escreva e salve num arquivo pode ser importado como um módulo
- Por exemplo, se você salva um programa com o nome prog.py, ele pode ser importado usando o comando $import$ prog
- A a "importação" só ocorre uma vez.
- Python assume que o código do módulo serve meramente para inicializar variáveis e funções.

Introdução: Módulos

- Após a importação de um módulo, este é compilado, gerando um arquivo .pyc correspondente
- No exemplo, um arquivo prog.pyc será criado
- Python só recompila um programa se o arquivo .py for mais recente que o arquivo .pyc

Exemplo: Módulo

- Criamos um arquivo *teste_mod* contendo

```
def f():
        print ("Oi!")
f()
```

- Execute o *python*

```
$python
>>> import teste_mod
Oi!
>>> import teste_mod
>>> teste_mod.f()
Oi!
```

- Encerre o *python*

```
$dir
teste_mod.py   teste_mod.pyc   ...
```

Fazendo Módulos Disponíveis

Onde os módulos são buscados durante a importação?

- No diretório corrente
- Nos diretórios da lista sys.path

Mudar lugar onde os módulos residem

- Alterar diretamente a variável sys.path Não Recomendado;
- Alterar a variável de ambiente PYTHONPATH Recomendado.
- Não requer que o programa que importará o módulo seja alterado

Exemplo: Módulo - Path

- Criamos um diretório *meu_dir*;
- Movemos o programa *teste_mod* para lá;

```
$ mkdir meu_dir
$ mv teste_mod.py meu_dir/
$ more meu_dir/teste_mod.py
def f():
          print ("Oi!")
f()

$ export PYTHONPATH=~/meu_dir
```

- Execute o *python*

```
$python
>>> import teste_mod
Oi!
```

Variável __name__

Programa pode ser executado por si só ou importado dentro de outro, como distinguir as duas situações?

- A variável __name__ é definida para cada programa:
- Se é um módulo, retorna o nome do módulo
- Se é um programa sendo executado, retorna '__main__'
- Saber se o código está sendo executado como módulo, testar:
- If __name__ == '__main__': código
- Isto é útil em diversas circunstâncias
- Por exemplo, para colocar código de teste, código para instalação do módulo ou exemplos de utilização

Exemplo: Variável __name__

- Criamos um arquivo *teste_mod* contendo

```
def f():
          print ("Oi!")
if __name__ == '__main__':
    f()
```

- Execute o *python*

```
$ python teste_mod
Oi!
>>> print (__name__)
__main__
$python
....
>>> import teste_mod
>>> print (teste_mod.__name__)
teste_mod
```

Pacotes

São hierarquias de módulos

- É um diretório que contém um arquivo chamado

$$\_\_int\_\_.py$$

- O pacote deve estar em um dos diretórios nos quais o Python busca por módulos
- Para importar o pacote, use o nome do diretório
  - O programa correspondente ao pacote é $\_\_int\_\_.py$

Pacotes

São hierarquias de módulos

- Os demais arquivos e diretórios dentro do pacote são encarados recursivamente como módulos

$$\_\_int\_\_.py$$

- Por exemplo, se um pacote se chama p e contém um arquivo chamado m.py, então podemos importar
  - p (o arquivo p/$\_\_int\_\_.py$)
  - p.m (o arquivo p/m.py)
- Semelhantemente, p poderia ter um outro pacote sob a forma de outro diretório contendo um arquivo $\_\_int\_\_.py$

Exemplo: Pacote

- Criamos um diretório *meu_dir*;
  `$ export PYTHONPATH=~/meu_dir`
- Criamos um diretório dentro do *meu_dir* chamado *meu_pacote*;
- Programa *teste1* com: `print ("teste 1")`
- Prog. *teste2* dentro do diretório *meu_pacote*: `print ("teste 2")`
- Programa *__int__.py* dentro do diretório *meu_pacote*:

```
print ("pacote")
$dir meu_dir
meu_pacote   teste1
$dir meu_dir/meu_pacote
__int__.py   teste2
```

- Execute o *python*

```
>>> import  teste1
teste 1
>>> import  meu_pacote
pacote
>>> import  meu_pacote.teste2
teste 2
```