

Departamento de Engenharia de Sistemas e Computação

Organização e Módulos

Módulos em Python, reuso e estruturação de programas maiores

João Araujo

Algoritmos Computacionais I

Aula 12

Roteiro

- O que é um módulo? (Cap. 4.2)
- Formas de importar: `import`, `from`, `as`
- Criando seu próprio módulo
- `if __name__ == "__main__"`
- Estruturando um programa maior (4.6)

Módulos: bibliotecas de funções

- Um **módulo** é um arquivo `.py` com funções e constantes prontas
- Python vem com a *biblioteca padrão*: `math`, `random`, `statistics`, `datetime`, `os`...
- Milhares de módulos de terceiros: `numpy`, `pandas`, `matplotlib`...

Filosofia

"Não reinvente a roda": antes de escrever, procure se já existe no manual do Python.

Três formas de importar

```python
import math                          # usa math.sin(...)
print(math.sin(math.pi / 2))         # 1.0

from math import radians, sin        # traz só estes nomes
print(sin(radians(30)))              # 0.4999999999999994

import math as m                     # apelido curto
print(m.sqrt(16))                    # 4.0
```

- `import math`: o prefixo `math.` deixa claro *de onde* vem cada função
- `from ... import ...`: nomes diretos, mas risco de *colisão* com seus nomes
- Evite `from math import *`

Exemplo: tabela de senos e cossenos

```
from math import radians, sin, cos

print('{:>9}{:>9}{:>9}{:>9}'.format('Graus', 'Radianos',
                                   'Seno', 'Cosseno'))
for graus in range(0, 361, 30):
    rad = radians(graus)
    print('{:>9.2f}{:>9.2f}{:>9.2f}{:>9.2f}'.format(
        graus, rad, sin(rad), cos(rad)))
```

- Exercício 4.1: expandir a tabela para incluir a tangente (`tan`)
- Exercício 4.4: pesquise no manual outras funções de `math`

Fonte: `code/Cap4/prog.4.7_tabela_senos_cossenosimport.py`

Criando seu próprio módulo

Arquivo `geometria.py`:

```
import math

def area_circulo(raio):
    """Área do círculo."""
    return math.pi * raio ** 2

def distancia(x1, y1, x2, y2):
    """Distância entre dois pontos."""
    return math.hypot(x2 - x1, y2 - y1)
```

Em outro arquivo, na mesma pasta:

```
import geometria
print(geometria.area_circulo(5))
```

`__name__`: módulo ou programa?

```python
# fim de geometria.py
def main():
    print("Teste:", area_circulo(1), distancia(0, 0, 3, 4))

if __name__ == "__main__":
    main()
```

- Executando `python geometria.py`: `__name__` vale `"__main__"` e o teste roda
- Com `import geometria`: `__name__` vale `"geometria"` e o teste não roda
- Assim o mesmo arquivo serve como **programa** e como **biblioteca**

Estruturando programas maiores

Camadas sugeridas

1. **Entrada/saída**: lê do usuário/arquivos e imprime
2. **Regras** (cálculos): funções puras
3. **Dados**: listas, dicionários, arquivos

Organização de arquivos

```
projeto/
  principal.py
  calculos.py
  entrada_saida.py
  dados/
```

Cada módulo tem **uma responsabilidade**. Isso será essencial no projeto final.

Exemplo: π por Monte Carlo em módulos

montecarlo.py

```python
import random as r
import math as m

def gera_pontos():
    return r.random(), r.random()

def dentro(x, y):
    return m.hypot(x, y) < 1

def estima_pi(n):
    cont = 0
    for _ in range(n):
        x, y = gera_pontos()
        if dentro(x, y):
            cont += 1
    return 4 * cont / n
```

principal.py

```python
import montecarlo

n = int(input("Pontos: "))
pi = montecarlo.estima_pi(n)
print("pi aprox.", pi)
```

Os testes e a entrada/saída ficam fora do módulo de cálculo.

Fonte: code/Cap4/prog.4.23_calcula_pi_main.py

Onde procurar ajuda

- `help(math)` e `dir(math)` no interpretador
- Documentação oficial: `docs.python.org/pt-br/3/library/`
- Mensagem `ModuleNotFoundError`: o módulo não está instalado ou o arquivo não está na mesma pasta
- Instalar pacotes de terceiros: `pip install nome`

Atividade

Crie um módulo `conversoes.py` com as funções `celsius_para_fahrenheit`, `fahrenheit_para_celsius` e `km_para_milhas`. Importe-o em outro arquivo e imprima uma tabela.

Para praticar: exercícios do livro

Site do livro – Capítulo 4

- Módulos: **4.1** (tabela com tangente), **4.2** (raízes quadradas), **4.3** (quadrado e cubo), **4.4** (funções de `math`)

Enunciados e soluções comentadas em
`https://www.araujo.eng.uerj.br/livro/exercicios/cap4/`

Exercícios adicionais (sem solução): `https://www.araujo.eng.uerj.br/livro/extras/`

Resumo da aula

- Módulo = arquivo `.py` reutilizável; `import` o traz para o seu programa
- `import x`, `from x import y`, `import x as z`
- Todo arquivo pode ser módulo; `if __name__ == "__main__"` protege o código de teste
- Programas grandes: módulos com responsabilidades claras
- Próxima aula: `math`, `random` e tratamento de erros