📘 Aula 05 — Trabalhando com Módulos no Python

1. O que são módulos?

- **Módulo** = arquivo Python (`.py`) que contém funções, classes ou variáveis que podem ser reutilizadas em outros programas.
- Serve para **organizar o código** e **reaproveitar recursos**.
- Python já vem com uma **biblioteca padrão** (standard library) — centenas de módulos prontos.
- Também podemos:
    1. Criar nossos próprios módulos.
    2. Instalar módulos externos (via **PyPI**).

---

2. Formas de importar módulos

Importando o módulo inteiro

```
import math

print(math.sqrt(25))  # 5.0
```

Importando funções ou classes específicas

```
from math import sqrt, ceil

print(sqrt(9))   # 3.0
print(ceil(4.2)) # 5
```

Dando apelido (alias)

```
import math as m
import random as r

print(m.factorial(5))  # 120
print(r.randint(1, 10))
```

Importando tudo (*)

```
from math import *
print(sqrt(16))  # 4.0
```

⚠ **Cuidado!** Isso polui o namespace e pode causar conflitos de nomes.

---

3. Usando módulos da biblioteca padrão (built-in)

📐 `math`

- `ceil(x)` → arredonda para cima.
- `floor(x)` → arredonda para baixo.
- `trunc(x)` → remove casas decimais.
- `pow(x, y)` → potência (`x**y`).
- `sqrt(x)` → raiz quadrada.
- `factorial(x)` → fatorial.

```
import math
print(math.sqrt(81))    # 9.0
print(math.factorial(4))  # 24
```

🎲 `random`

- `randint(a, b)` → número inteiro aleatório entre `a` e `b`.
- `random()` → número entre 0 e 1.
- `choice(lista)` → escolhe item aleatório.
- `shuffle(lista)` → embaralha lista.

```
import random
print(random.randint(1, 100))
print(random.choice(["maçã", "banana", "uva"]))
```

🗓️ `datetime`

Trabalha com data e hora.

```
import datetime

hoje = datetime.date.today()
agora = datetime.datetime.now()
print("Data de hoje:", hoje)
print("Data e hora:", agora)
```

🔑 `os` e `sys`

- `os` → interagir com o sistema operacional.
- `sys` → informações sobre o interpretador.

```
import os, sys
```

```
print(os.getcwd())      # diretório atual
print(sys.version)      # versão do Python
```

---

4. Criando seu próprio módulo

Crie um arquivo chamado **meu_modulo.py**:

```
def saudacao(nome):
    return f"Olá, {nome}!"
```

Agora use em outro arquivo:

```
import meu_modulo
print(meu_modulo.saudacao("Marcelo"))
```

Ou:

```
from meu_modulo import saudacao
print(saudacao("Ana"))
```

---

5. Pacotes (packages)

- Pacote = **pasta** que contém vários módulos.
- Para ser reconhecida como pacote, precisa ter um arquivo `__init__.py`.

Estrutura:

```
meu_pacote/
    __init__.py
    modulo1.py
    modulo2.py
```

Uso:

```
from meu_pacote import modulo1
```

---

6. Módulos externos (PyPI)

- O **PyPI (Python Package Index)** é o repositório oficial de pacotes Python.
- Instalação com `pip`:

```
pip install nome_do_pacote
```

Exemplo com **emoji**:

```
import emoji
print(emoji.emojize("Olá, Mundo! :earth_americas:", language="alias"))
```

---

7. Boas práticas ao usar módulos

✅ Use **alias** quando o nome for muito longo (`import numpy as np`).
✅ Evite `from modulo import *`.
✅ Agrupe imports no início do arquivo (por convenção PEP8):

```
# Imports padrão
import os
import sys

# Imports de terceiros
import numpy as np

# Imports locais
import meu_modulo
```

8. Como importar módulos?

Importando o módulo inteiro

```
import math

numero = 9
raiz = math.sqrt(numero)
print(f"A raiz quadrada de {numero} é {raiz}")
```

➡ Aqui precisamos usar `math.sqrt`, `math.ceil`, etc.

Importando apenas o que precisamos

```
from math import sqrt, floor

numero = 25
print(sqrt(numero))   # 5.0
print(floor(4.9))     # 4
```

➡ Assim chamamos direto a função sem precisar escrever `math.`.

Importando com apelido (alias)

```
import math as m

print(m.factorial(5))  # 120
```

➡ Muito usado com bibliotecas grandes (ex.: `import numpy as np`).

---

9. Funções úteis do módulo `math`

- `ceil(x)` → arredonda para cima.
- `floor(x)` → arredonda para baixo.

- `trunc(x)` → corta a parte decimal (sem arredondar).
- `pow(x, y)` → potência (`x**y`).
- `sqrt(x)` → raiz quadrada.
- `factorial(x)` → fatorial de um número inteiro.

Exemplo:

```
import math

print(math.ceil(7.2))     # 8
print(math.floor(7.8))    # 7
print(math.trunc(7.9))    # 7
print(math.pow(2, 5))     # 32.0
print(math.sqrt(16))      # 4.0
print(math.factorial(5))  # 120
```

---

10. Módulo `random` — valores aleatórios

```
import random

print(random.randint(1, 10))              # número inteiro entre 1 e 10
print(random.choice(["maçã", "banana"]))  # escolhe item aleatório
```

---

11. Usando módulos externos (PyPI)

- Nem tudo já vem no Python.
- Podemos instalar pacotes extras do **PyPI** (Python Package Index).
- Exemplo: biblioteca `emoji`.

Instalação (no terminal):

```
pip install emoji
```

Uso:

```
import emoji

print(emoji.emojize("Olá, Mundo! :earth_americas:", language="alias"))
```

---

12. Exercícios práticos

1. Peça um número ao usuário e mostre:
    - a raiz quadrada (`sqrt`),
    - o arredondado pra cima (`ceil`),

    - o arredondado pra baixo (`floor`).

2. Sorteie um número de 1 a 20 e peça para o usuário adivinhar. Diga se ele precisa tentar **maior** ou **menor** até acertar.
3. Peça uma lista de 5 nomes e sorteie um vencedor com `random.choice()`.
4. Instale a biblioteca `emoji` e faça um programa que mostre uma mensagem de boas-vindas com um emoji (👋, 🌎, etc).