MAC2166 - Introdução à Computação

Prof. Dr. Helder Oliveira

Agenda

- **Módulos em Python**

O que é um módulo?

- Um módulo é um arquivo .py que contém código Python (funções, classes, variáveis).
- Permite organizar o código em partes reutilizáveis e facilita a manutenção.

Por que usar módulos?

- Reutilização de código.
- Melhor organização.
- Facilita testes e manutenção

Como usar modulos

- Colocar o módulo no mesmo diretório do arquivo que irá usa-lo.
- Para usar:
  - Voce deve importar o módulo criado.
  - import nome_do_modulo
  - nome_do_modulo.função
- Se estiver em diretórios diferentes:
  - Configure o PYTHONPATH
  - Use pacotes (__init__.py)

main.py | meu_modulo.py

```
1  import meu_modulo
2  meu_modulo.saudacao()
```

main.py | meu_modulo.py

```
1  #meu_modulo.py
2  def saudacao():
3      print("Hello World")
```

Como criar e usar módulos?

- Criar um arquivo com funções, por exemplo matematica.py
- Usar esse módulo em outro arquivo.

main.py | matematica.py

```
1  # matematica.py
2  def soma(a, b):
3      return a + b
4
5  def multiplica(a, b):
6      return a * b
```

main.py | matematica.py

```
1  # main.py
2  import matematica
3
4  print(matematica.soma(3, 5))
5  # Saída: 8
6  print(matematica.multiplica(4, 6))
7  # Saída: 24
```

Uso do from

- **O que é from na importação?**
  - O comando from permite importar **apenas partes específicas** de um módulo, em vez de importar o módulo inteiro.
  - Facilita o uso direto das funções, classes ou variáveis importadas sem precisar usar o prefixo do módulo.
- from nome_do_modulo import nome_da_funcao_ou_variável
- Exemplo:

```
1  from math import sqrt
2  print(sqrt(16))
3  # Saída: 4.0
```

Uso do from

- Importar múltiplos elementos:

```
from math import sin, cos, tan
```

- Importar tudo do módulo (não recomendado):

```
from math import *
```

- **Vantagens do from**
  - Código mais limpo e direto (sem modulo. antes do nome).
  - Evita carregar funcionalidades não utilizadas.
  - Ajuda a reduzir o uso de memória e melhora a legibilidade.

__name__

- O que é __name__?
  - __name__ é uma variável especial do Python.
  - Quando você **executa um arquivo diretamente**, __name__ vale "__main__".
  - Quando você **importa o arquivo como módulo**, __name__ vale o nome do arquivo (sem .py).
- **O que faz if __name__ == "__main__":?**
  - É um comando que verifica se o arquivo está sendo executado diretamente.
  - O código dentro desse bloco **só roda se o arquivo for o programa principal**.
  - Se o arquivo for importado, esse código não roda automaticamente.

Exemplo if __name__ == "__main__":

main.py

```
1 def main():
2     print("Executando main")
3
4 main()
```

- Se rodar o arquivo, a função main() executa.
- Se importar esse arquivo em outro, a função main() também executa automaticamente — o que geralmente não queremos.

main.py

```
1 def main():
2     print("Executando main")
3
4 if __name__ == "__main__":
5     main()
```

- Agora, se rodar o arquivo diretamente, main() executa.
- Se importar esse arquivo em outro, **main() não é executada automaticamente**

Exercício

- Crie um módulo operacoes.py com as funções:
  - adicionar(a, b) — retorna a soma de a e b.
  - subtrair(a, b) — retorna a subtração de b de a.
  - multiplicar(a, b) — retorna a multiplicação de a e b.
  - dividir(a, b) — retorna a divisão de a por b. Se b for zero, deve retornar None.

Exercício

- Crie um módulo operacoes.py:

main.py | operacoes.py

```python
# operacoes.py

def adicionar(a, b):
    return a + b

def subtrair(a, b):
    return a - b

def multiplicar(a, b):
    return a * b

def dividir(a, b):
    if b == 0:
        return None
    return a / b
```

main.py | operacoes.py

```python
# main.py
import operacoes

def main():
    print("Adição: 5 + 3 =", operacoes.adicionar(5, 3))
    print("Subtração: 10 - 7 =", operacoes.subtrair(10, 7))
    print("Multiplicação: 4 * 6 =", operacoes.multiplicar(4, 6))
    print("Divisão: 8 / 2 =", operacoes.dividir(8, 2))
    print("Divisão por zero: 5 / 0 =", operacoes.dividir(5, 0))
main()
```

Exercício

- Crie um módulo string_utils.py com as funções:
  - contar_vogais(texto) — retorna o número de vogais na string texto.
  - inverter_string(texto) — retorna a string invertida.
  - eh_palindromo(texto) — retorna True se a string for um palíndromo (ignorar maiúsculas e espaços), senão False.
- Crie um arquivo main.py para testar as funções.

Exercício

- Crie um módulo string_utils.py e crie um arquivo main.py para testar as funções.

main.py | string_utils.py

```python
# main.py
import string_utils

def main():
    texto = "A man a plan a canal Panama"
    print(f"Texto: '{texto}'")
    print("Número de vogais:", string_utils.contar_vogais(texto))
    print("Texto invertido:", string_utils.inverter_string(texto))
    print("É palíndromo?", string_utils.eh_palindromo(texto))
main()
```

main.py | string_utils.py

```python
# string_utils.py
def contar_vogais(texto):
    texto = texto.lower()
    vogais = 'aeiou'
    return sum(1 for letra in texto if letra in vogais)

def inverter_string(texto):
    return texto[::-1]

def eh_palindromo(texto):
    texto_limpo = ''.join(c.lower() for c in texto if c.isalnum())
    return texto_limpo == texto_limpo[::-1]
```

Exercício

- Crie um módulo listas.py com as funções:
  - encontrar_maior(lista) — retorna o maior número da lista.
  - remover_repetidos(lista) — retorna uma lista sem elementos duplicados, mantendo a ordem.
  - media(lista) — retorna a média dos números da lista. Se lista vazia, retorna 0.
- Crie um arquivo main.py para testar as funções com diferentes casos.

Exercício

- Crie um módulo listas.py e crie um arquivo main.py para testar as funções com diferentes casos.

main.py | listas.py

```python
# listas.py
def encontrar_maior(lista):
    if not lista:
        return None
    return max(lista)

def remover_repetidos(lista):
    resultado = []
    vistos = set()
    for item in lista:
        if item not in vistos:
            resultado.append(item)
            vistos.add(item)
    return resultado

def media(lista):
    if not lista:
        return 0
    return sum(lista) / len(lista)
```

main.py | listas.py

```python
# main.py
import listas

def main():
    lista = [1, 2, 2, 3, 4, 4, 5]
    print("Lista original:", lista)
    print("Maior número:", listas.encontrar_maior(lista))
    print("Lista sem repetidos:", listas.remover_repetidos(lista))
    print("Média:", listas.media(lista))
main()
```

Exercício

- Crie um módulo datas.py com a função:
  - eh_ano_bissexto(ano) — retorna True se o ano for bissexto, senão False.
- Crie um arquivo main.py para testar anos bissextos e não bissextos.

Exercício

- Crie um módulo datas.py com a função:
  - eh_ano_bissexto(ano) — retorna True se o ano for bissexto, senão False.
- Crie um arquivo main.py para testar anos bissextos e não bissextos.

main.py datas.py

```python
# datas.py

def eh_ano_bissexto(ano):
    return ano % 400 == 0 or (ano % 4 == 0 and ano % 100 != 0)
```

main.py datas.py

```python
# main.py
import datas

def main():
    anos = [1900, 2000, 2021, 2024]
    for ano in anos:
        print(f"O ano {ano} é bissexto? {datas.eh_ano_bissexto(ano)}")
main()
```

Exercício

- Crie um módulo matematica.py com a função:
  - fatorial(n) — retorna o fatorial de n (assumir n >= 0). Se n for zero, retorna 1.
- Crie um arquivo main.py para testar fatorial de vários números, inclusive 0.

Exercício

- Crie um módulo matematica.py com a função:
  - fatorial(n) — retorna o fatorial de n (assumir n >= 0). Se n for zero, retorna 1.
- Crie um arquivo main para testar fatorial de vários números, inclusive 0.

main.py matematica.py

```python
# matematica.py

def fatorial(n):
    if n == 0:
        return 1
    resultado = 1
    for i in range(1, n+1):
        resultado *= i
    return resultado
```

main.py matematica.py

```python
# main.py
import matematica

def main():
    for n in range(6):
        print(f"Fatorial de {n} é {matematica.fatorial(n)}")
main()
```

Exercício

- Você deve criar dois módulos e um programa principal para trabalhar com nome e idade de uma pessoa.
- O primeiro módulo deve ter uma função que recebe nome e idade e retorna uma frase com essas informações.
- O segundo módulo deve ter duas funções que recebem a idade:
  - Uma que converte a idade para meses.
  - Outra que calcula o quadrado da idade.
- O programa principal deve:
  - Importar os dois módulos.
  - Definir nome e idade fixos.
  - Usar as funções dos módulos para obter as informações e imprimir tudo no console.
- Exemplo de saída esperada:

  Nome: Carlos, Idade: 25 anos

  Idade em meses: 300

  Idade ao quadrado: 625

Exercício

main.py | calculadora.py | pessoas.py

```python
#main.py
import pessoas
import calculadora

def main():
    nome = "Carlos"
    idade = 25
    info = pessoas.mostrar_nome_idade(nome, idade)
    print(info)
    meses = calculadora.idade_em_meses(idade)
    print("Idade em meses:", meses)
    quadrado = calculadora.idade_ao_quadrado(idade)
    print("Idade ao quadrado:", quadrado)
main()
```

main.py | calculadora.py | pessoas.py

```python
#pessoas.py
def mostrar_nome_idade(nome, idade):
    return f"Nome: {nome}, Idade: {idade} anos"
```

main.py | calculadora.py | pessoas.py

```python
#calculadora.py
def idade_em_meses(idade):
    return idade * 12
def idade_ao_quadrado(idade)
    return idade ** 2
```

Dúvidas