

INSTITUTO FEDERAL

Catarinense

Campus Camboriú

AULA 7 (Subalgoritmos - parte II)

Professora: Lidiane Visintin

lidiane.visintin@ifc.edu.br

Professor: Rafael de Moura Speroni

rafael.speroni@ifc.edu.br

Para fixar:

Duas regras:

- Uma função deverá resolver apenas 1(um) problema;
- Quanto mais genérica for a sua solução, melhor ela será a longo prazo.

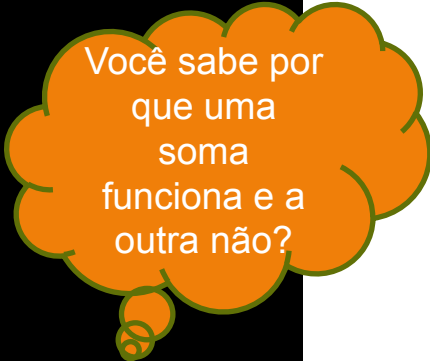
Dica: Se a função faz isso “e” aquilo já é um indicativo que talvez tenha que ser desmembrada.

Atenção as funções

```
# como não escrever uma função
```

```
def soma(L):  
    total = 0  
    x = 0  
    while x < 5:  
        total += L[x]  
        x += 1  
    return total
```

```
L = [1, 7, 2, 9, 15]  
print(soma(L))  
print(soma([7, 9, 12, 3, 100, 20, 4]))
```



Você sabe por
que uma
soma
funciona e a
outra não?

Atenção as funções

```
# soma dos valores de uma lista
```

```
def soma(L):
```

```
    total = 0
```

```
    x = 0
```

```
    while x < len(L):
```

```
        total += L[x]
```

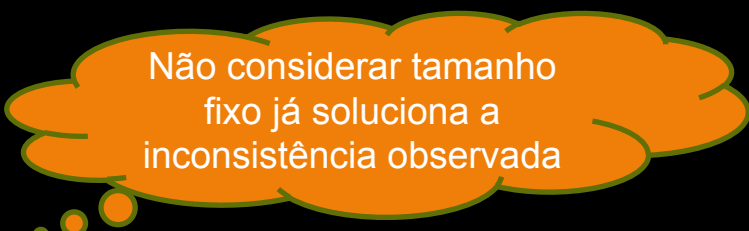
```
        x += 1
```

```
    return total
```

```
L = [1, 7, 2, 9, 15]
```

```
print(soma(L))
```

```
print(soma([7, 9, 12, 3, 100, 20, 4]))
```



Não considerar tamanho
fixo já soluciona a
inconsistência observada

Atenção as funções

Um problema clássico da programação é o cálculo do fatorial de um número:

- **Fatorial de 3** = $3 \times 2 \times 1$
= 6
- **Fatorial de 4** = $4 \times 3 \times 2$
 $\times 1$ = 24
- **Fatorial de 0** = 1

```
#calculo do fatorial
```

```
def fatorial(n):  
    fat = 1  
    while n > 1:  
        fat *= n  
        n -= 1  
    return fat
```

```
print(fatorial(0))  
print(fatorial(4))
```

Atenção as funções

Um problema clássico da programação é o cálculo do fatorial de um número:

- **Fatorial de 3** = $3 \times 2 \times 1$
= 6
- **Fatorial de 4** = $4 \times 3 \times 2$
 $\times 1$ = 24
- **Fatorial de 0** = 1

```
# calculo do fatorial  
#(outra forma)
```

```
def fatorial(n):  
    fat = 1  
    x = 1  
    while x <= n:  
        fat *= x  
        x += 1  
    return fat
```

```
print(fatorial(0))  
print(fatorial(4))
```

Variáveis locais

São variáveis **internas** a função, ou seja esta variável somente existirá enquanto a função está executando.

Não podemos acessar o seu valor fora da função que a criou e por isso passamos por parâmetros que retornam valores através das funções, de forma a viabilizar a troca de dados.

```
# calculo do fatorial  
#(outra forma)
```

Variável
local

```
def fatorial(n):
```

```
    fat = 1
```

```
    x = 1
```

```
    while x <= n:
```

```
        fat *= x
```

```
        x += 1
```

```
    return fat
```

```
print(fatorial(0))
```

```
print(fatorial(4))
```

Variáveis Globais

São variáveis **externas as funções**, que todos podem ter acesso.

```
#exemplo variável global
```

Variável Global

```
EMPRESA = 'Unidos Venceremos LTDA'
```

```
def imprime_cabecalho():
```

```
    print(EMPRESA)
```

```
    print("-" * len(EMPRESA))
```

```
imprime_cabecalho()
```

Unidos Venceremos LTDA

Variáveis Globais e Locais

Exemplos:

```
a = 5
def muda_e_imprime():
    a = 7
    print(f"A de dentro da função:{a}")
print(f"A antes de mudar:{a}")
muda_e_imprime()
print(f"A depois de mudar:{a}")
```

Variável Local

Resultado:

```
A antes de mudar:5
A de dentro da função:7
A depois de mudar:5
```

Variáveis Globais e Locais

Se quisermos modificar uma variável global dentro de uma função, devemos informar que estamos utilizando a variável global.

```
a = 5
def muda_e_imprime():
    global a
    a = 7
    print(f"A de dentro da função:{a}")
print(f"A antes de mudar:{a}")
muda_e_imprime()
print(f"A depois de mudar:{a}")
```

Variável global

Resultado:

```
A antes de mudar:5
A de dentro da função:7
A depois de mudar:7
```

Recursividade

- Uma função que chama a si mesma. Quando isso ocorre, temos uma função recursiva. Para isso utilizaremos o exemplo do fatorial;

$$\text{Fatorial}(n) = \begin{cases} 1 & 0 \leq n \leq 1 \\ n \times \text{fatorial}(n-1) & \end{cases}$$

Recursividade

```
#Função recursiva do fatorial
```

```
def fatorial(n):  
    if n == 0 or n == 1:  
        return 1  
    else:  
        return n * fatorial(n-1)
```

```
a = int(input('informe um valor para obter o fatorial: '))  
print("O resultado do fatorial é:",fatorial(a))
```

Recursividade - Exemplo

- Fatorial de 0
- Fatorial de 3

```
#Função recursiva do fatorial

def fatorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * fatorial(n-1)

a = int(input('informe um valor
para obter o fatorial: '))
print("O resultado do fatorial
é:",fatorial(a))
```

```
#Função recursiva do fatorial

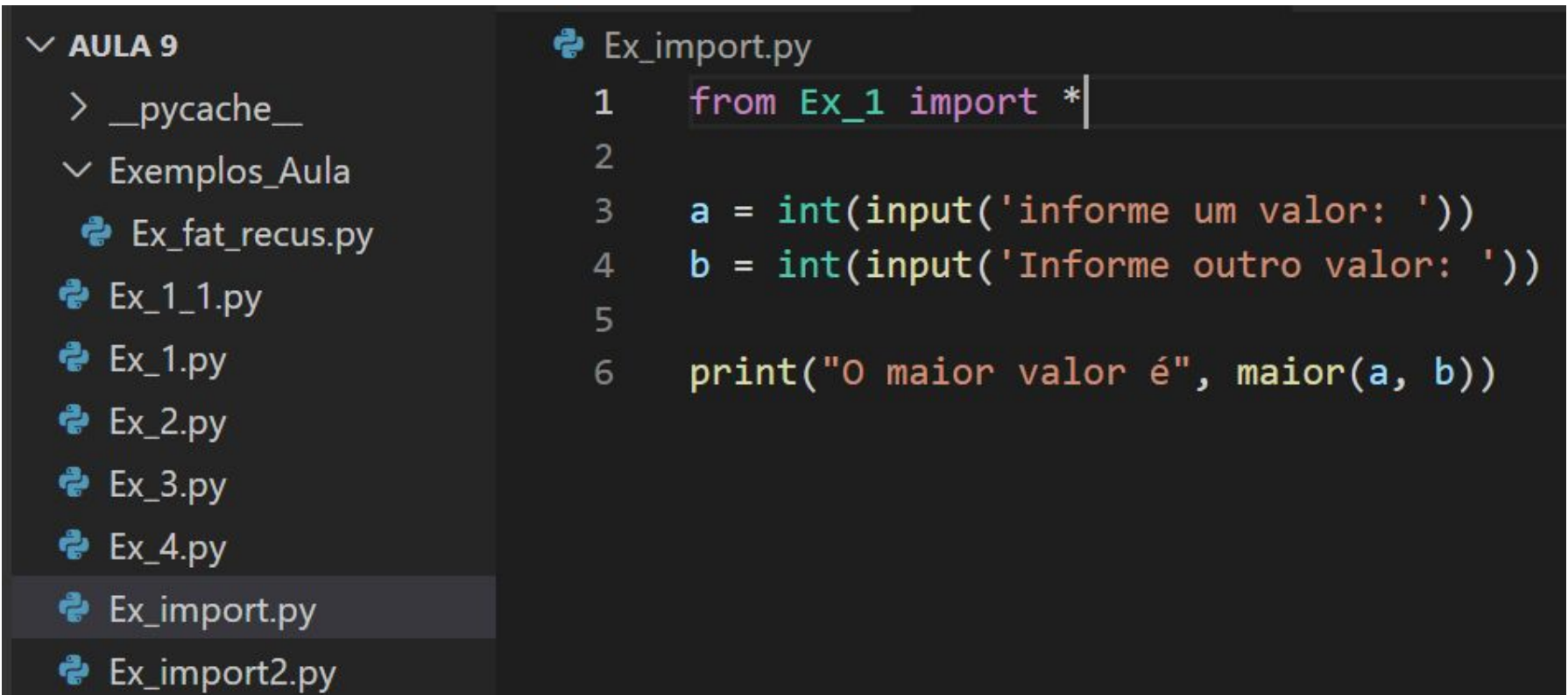
def fatorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * fatorial(n-1)
```

```
#Função recursiva do fatorial

def fatorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * fatorial(n-1)
```

Reutilização de funções

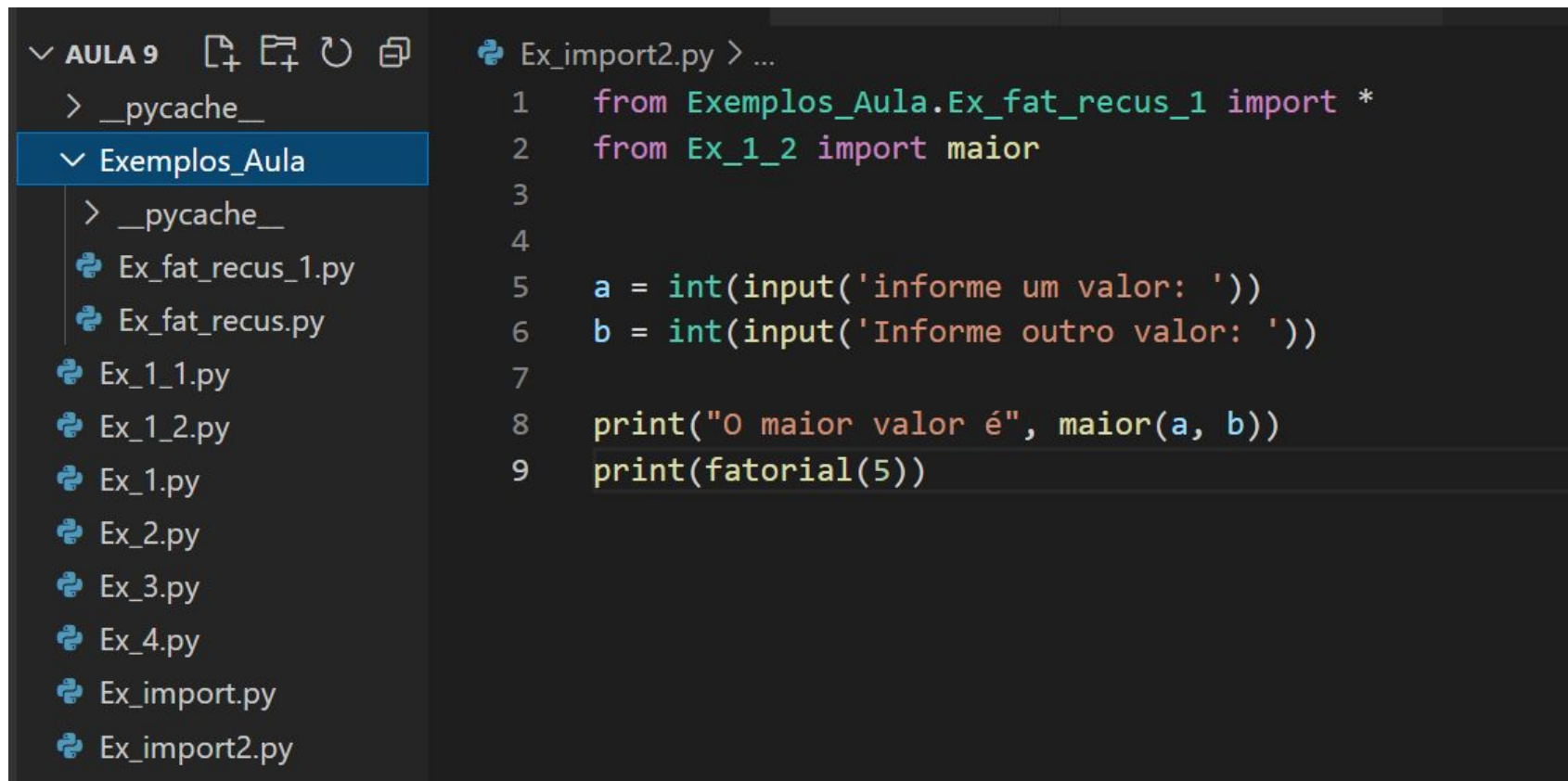
- Na mesma pasta:



```
Ex_import.py
1  from Ex_1 import *
2
3  a = int(input('informe um valor: '))
4  b = int(input('Informe outro valor: '))
5
6  print("O maior valor é", maior(a, b))
```

Reutilização de funções

- Em outra pasta, mas no mesmo projeto



```

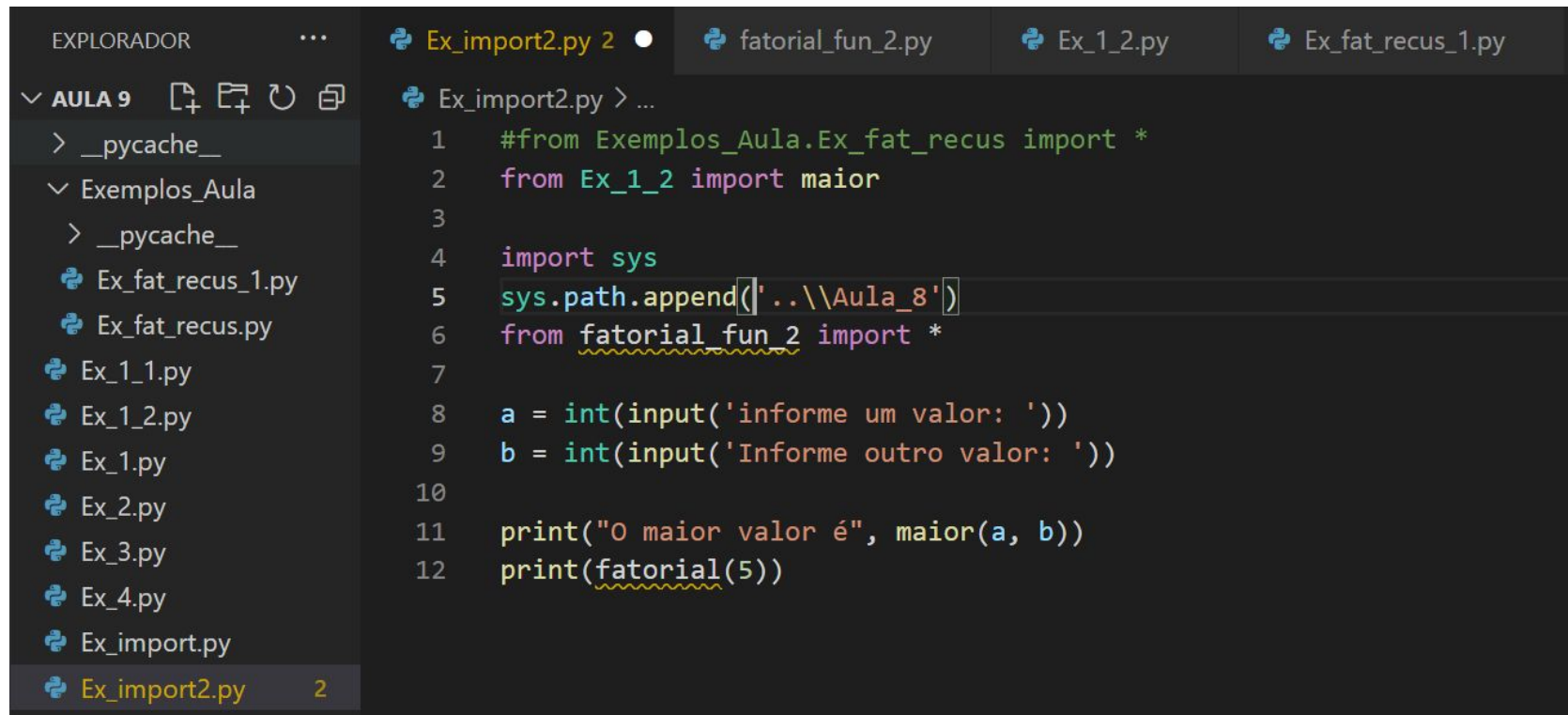
AULA 9
├── __pycache__
└── Exemplos_Aula
    ├── __pycache__
    ├── Ex_fat_recus_1.py
    ├── Ex_fat_recus.py
    ├── Ex_1_1.py
    ├── Ex_1_2.py
    ├── Ex_1.py
    ├── Ex_2.py
    ├── Ex_3.py
    ├── Ex_4.py
    ├── Ex_import.py
    └── Ex_import2.py

Ex_import2.py > ...
1  from Exemplos_Aula.Ex_fat_recus_1 import *
2  from Ex_1_2 import maior
3
4
5  a = int(input('informe um valor: '))
6  b = int(input('Informe outro valor: '))
7
8  print("O maior valor é", maior(a, b))
9  print(fatorial(5))

```

Reutilização de funções

- Em outra pasta, mas no mesmo projeto



The screenshot shows a code editor interface with a dark theme. On the left, the 'EXPLORADOR' (Explorer) pane shows a project structure under 'AULA 9'. The files listed are: `__pycache__`, `Exemplos_Aula` (containing `__pycache__`, `Ex_fat_recus_1.py`, and `Ex_fat_recus.py`), `Ex_1_1.py`, `Ex_1_2.py`, `Ex_1.py`, `Ex_2.py`, `Ex_3.py`, `Ex_4.py`, `Ex_import.py`, and `Ex_import2.py` (which is selected and has a '2' next to it). The main editor area shows the code for `Ex_import2.py`. The code imports functions from `Exemplos_Aula.Ex_fat_recus` and `Ex_1_2`, imports `sys`, and appends the directory `..\Aula_8` to the `sys.path`. It then imports `fatorial_fun_2` and uses its `fatorial` function. The code also takes user input for two values, `a` and `b`, and prints the result of `maior(a, b)` and `fatorial(5)`.

```
1  #from Exemplos_Aula.Ex_fat_recus import *
2  from Ex_1_2 import maior
3
4  import sys
5  sys.path.append(['..\Aula_8'])
6  from fatorial_fun_2 import *
7
8  a = int(input('informe um valor: '))
9  b = int(input('Informe outro valor: '))
10
11 print("O maior valor é", maior(a, b))
12 print(fatorial(5))
```


Referências



Referências Básicas

FORBELLONE, André Luiz Villar; EBERSPÄCHER, Henri Frederico. **Lógica de programação: a construção de algoritmos e estruturas de dados**. 3. ed. Pearson Prentice Hall. 2005

MANZANO, José Augusto N. G; OLIVEIRA, Jayr Figueiredo de. **Algoritmos: lógica para desenvolvimento de programação de computadores**. 27. ed.. Érica. 2014

Referências Complementares

MENEZES, Nilo Ney de Coutinho. **Introdução a programação com Python**. 3ª Ed. Novatec. 2019.

CORMEN, Thomas H et al. **Algoritmos: teoria e prática**. 2. ed. Elsevier, Campus,. 2002

Referências na Internet

<https://docs.python.org/3/>

<https://www.w3schools.com/python/default.asp>

<https://panda.ime.usp.br/pensepy/static/pensepy/05-Funcoes/funcoes.html>