

Ed u c a ç ã o  
P r o f i s s i o n a l  
P a u l i s t a

Técnico em  
Ciência de  
Dados

# **Programação aplicada à Ciência de Dados**

## **Estrutura de controle de fluxo**

### **Aula 3**

**Código da aula: [DADOS]ANO1C2B2S10A3**



## Objetivo da aula

- Aplicar os conceitos de funções em Python.



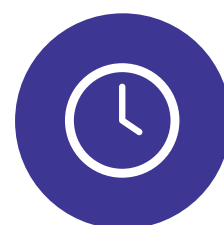
## Competências da unidade (técnicas e socioemocionais)

- Ser proficiente em linguagens de programação para manipular e analisar grandes conjuntos de dados;
- Usar técnicas para explorar e analisar dados, aplicar modelos estatísticos, identificar padrões, realizar inferências e tomar decisões baseadas em evidências.



## Recursos didáticos

- Recurso audiovisual para exibição de vídeos e imagens;
- Acesso ao laboratório de informática e/ou à internet;
- Software Anaconda/Jupyter Notebook instalado ou similar.



## Duração da aula

50 minutos.

## Função – Resumo

Uma função é um bloco de código reutilizável que realiza uma tarefa específica.

```
def nome_da_funcao(parametros):  
    # Corpo da função  
    # Realiza alguma tarefa  
    return resultado
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook

- Promove a **reutilização** de código.
- Contribui para a **organização** e a **modularidade**.
- Facilita a **compreensão** do código.

## Criando funções

Outros exemplos de como criar uma função:

```
1 def minha_primeira_funcao():  
2     print('Olá Mundo')
```

```
1 # chamar a função pelo nome para executá-la  
2 minha_primeira_funcao()
```

Olá Mundo

```
1 def bom_dia():  
2     print('Bom dia!')
```

```
1 bom_dia()
```

Bom dia!

```
1 def exemplo_funcao_simples():  
2     print('criando uma função com o "def"')
```

```
1 exemplo_funcao_simples()
```

criando uma função com o "def"

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

## Parâmetros

Será que conseguimos personalizar?

```
1 def funcao_ola(nome):  
2     print(f'Olá, meu nome é {nome}.')
```

```
1 funcao_ola("Alice")
```

Olá, meu nome é Alice.

```
1 funcao_ola("Renata")
```

Olá, meu nome é Renata.

```
1 funcao_ola("Tiago")
```

Olá, meu nome é Tiago.

```
1 def horario(hora):  
2     if hora < 12:  
3         print('Bom dia')  
4     elif hora < 18:  
5         print('Boa tarde')  
6     else:  
7         print('Boa noite')
```

```
1 horario(10)
```

Bom dia

```
1 horario(15)
```

Boa tarde

```
1 horario(19)
```

Boa noite



## Ordem dos parâmetros

A ordem dos parâmetros importa?

```
1 def exemplo(nome, idade):  
2     print(f"Nome: {nome}, Idade: {idade}")
```

```
1 exemplo('Alice', 25)
```

Nome: Alice, Idade: 25

```
1 exemplo(25, 'Alice')
```

Nome: 25, Idade: Alice

```
1 exemplo(nome="Alice", idade=25)
```

Nome: Alice, Idade: 25

```
1 exemplo(idade=25, nome="Alice")
```

Nome: Alice, Idade: 25

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

# Exposição

## Parâmetros e *return*

Qual é a diferença entre as funções?

```
1 def soma(a, b):  
2     print(a + b)
```

```
1 soma(2, 9)
```

11

```
1 soma(7, 8)
```

15

```
1 soma(10, 15)
```

25

```
1 resultado = soma(10, 15)  
2 print('O resultado é: ', resultado)
```

25

O resultado é: None

```
1 def soma(a, b):  
2     return a + b
```

```
1 soma(2, 9)
```

11

```
1 soma(7, 8)
```

15

```
1 soma(10, 15)
```

25

```
1 resultado = soma(10, 15)  
2 print('O resultado é: ', resultado)
```

O resultado é: 25

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.



## Parâmetros e *return*

```
1 def soma(a, b):  
2     resultado = a + b  
3     print("print do resultado dentro da função: ", resultado) # apenas mostra na tela o valor de uma variável  
4     return resultado # retorna um resultado/saída da função  
5  
6  
7 resultado = soma(2, 3)  
8  
9 media = resultado / 2  
10 print(f'A média é {media}')
```

```
print do resultado dentro da função: 5  
A média é 2.5
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

## Chamar uma função como argumento

```
1 def soma_dois_valores(a, b):  
2     resultado = a + b  
3     return resultado  
4  
5 res = soma_dois_valores(2, 3)  
6  
7 def media_dois_valores(resultado):  
8     media = resultado / 2  
9     return media  
10  
11 print(media_dois_valores(res))
```

2.5

```
1 media_dois_valores(soma_dois_valores(2,3))
```

2.5

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

## Exercícios

1. Faça uma função que recebe um número e imprima seu **dobro**.
2. Faça uma função que recebe o valor do raio de um **círculo** e retorna o valor do comprimento de sua circunferência  **$C = 2 * \pi * r$** .
3. Crie uma função chamada **concatenar\_palavras**, que receba duas *strings* como parâmetros e retorna a concatenação dessas duas *strings*, separadas por um espaço.

## Exercícios

4. Crie uma função chamada **verificar\_par** que receba um número como parâmetro e retorne **True**, se o número for par, e **False**, em caso contrário.
5. Crie uma função chamada **calcular\_media** que receba três números como parâmetros e retorne a **média aritmética** desses números.
6. Faça uma função para cada **operação matemática básica** (soma, subtração, multiplicação e divisão). As funções devem receber dois números e retornar o resultado da operação.



© Getty Images

O que nós  
**aprendemos  
hoje?**

## Hoje desenvolvemos:

- 1** Aplicações práticas de funções em Python.
- 2** Resoluções de exercícios sobre funções.





# Saiba mais

## Entenda mais sobre *built-in functions* e funções:

ALURA. *Python para Data Science*: trabalhando com funções, estruturas de dados e exceções. Disponível em:  
<https://cursos.alura.com.br/course/python-data-science-funcoes-estruturas-dados-excecoes/task/125896>. Acesso em: 28 fev. 2024.



# Referências da aula

Identidade visual: Imagens © Getty Images.

MENEZES, N. N. C. *Introdução à programação com Python: algoritmos e lógica de programação para iniciantes*. São Paulo: Novatec, 2019.

Ed u c a ç ã o  
P r o f i s s i o n a l  
P a u l i s t a

Técnico em  
Ciência de  
Dados

**Educação  
Profissional  
Paulista**

Técnico em  
**Ciência de  
Dados**

# Variáveis e tipos de dados

## Dicionários

### Aula 2

**[DADOS]ANO1C2B2S12A2**

# Exposição



## Objetivos da Aula

- Introduzir conceito de operação de adição, remoção e atualização em dicionários Python.



## Competências da Unidade (Técnicas e Socioemocionais)

- Ser proficiente em linguagens de programação para manipular e analisar grandes conjuntos de dados;
- Usar técnicas para explorar e analisar dados, aplicar modelos estatísticos, identificar padrões, realizar inferências e tomar decisões baseadas em evidências;
- Colaborar, efetivamente, com outros profissionais, como cientistas de dados e engenheiros de dados;
- Trabalhar em equipes multifuncionais colaborando com colegas, gestores e clientes.



## Recursos Didáticos

- Recurso audiovisual para exibição de vídeos e imagens;
- Acesso ao laboratório de informática e/ou internet;
- Software Anaconda/Jupyter Notebook instalado, ou similar.



## Duração da Aula

50 minutos

## Relembrando

**Dicionário** é uma estrutura de dados que armazena diferentes tipos de dados e funciona como um mapeamento, ou seja, temos uma chave e um valor associado a essa chave. Para encontrar algo no dicionário, basta procurar pela chave que você quer.

**dicionario = {chave: valor}**

A **chave**, no geral, é **string**; já o valor é qualquer tipo de dado.



## Operações: adição e remoção de elementos

Dicionários em Python permitem adicionar novos pares chave-valor e remover elementos existentes.

```
# Criando um dicionário inicial
meu_dicionario = {'nome': 'João', 'idade': 25, 'cidade': 'São Paulo'}

# Acesso a elementos
nome_da_pessoa = meu_dicionario['nome']
print(f"Nome da pessoa: {nome_da_pessoa}")

# Adição de um novo par chave-valor
meu_dicionario['profissao'] = 'Programador'

# Remoção de um par chave-valor
del meu_dicionario['cidade']

# Exibindo o dicionário após as operações
print("Dicionário atualizado:", meu_dicionario)
```

Nome da pessoa: João  
Dicionário atualizado: {'nome': 'João', 'idade': 25, 'profissao': 'Programador'}

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

## Operações: atualização

Os valores associados às chaves podem ser atualizados, alterando-se diretamente o valor correspondente.

```
# Criando um dicionário inicial  
meu_dicionario = {'nome': 'João', 'idade': 25, 'cidade': 'São Paulo'}
```

```
# Acesso a elementos  
nome_da_pessoa = meu_dicionario['nome']  
print(f"Nome da pessoa: {nome_da_pessoa}")
```

```
# Adição de um novo par chave-valor  
meu_dicionario['profissao'] = 'Programador'
```

```
# Atualização de um valor existente  
meu_dicionario['nome'] = 'Carlos'  
meu_dicionario['profissao'] = 'Estudante'  
meu_dicionario['idade'] = 21
```

```
# Exibindo o dicionário após as operações  
print("Dicionário atualizado:", meu_dicionario)
```

Nome da pessoa: João

Dicionário atualizado: {'nome': 'Carlos', 'idade': 21, 'cidade': 'São Paulo', 'profissao': 'Estudante'}

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

## Exposição

# Métodos úteis

Já vimos que o Python fornece métodos como `keys()`, `values()`, e `items()` para obter listas de chaves, valores e pares chave-valor, respectivamente.

```
# Criando um dicionário
meu_dicionario = {'nome': 'Carlos', 'idade': 25, 'cidade': 'São Paulo', 'profissao': 'Estudante'}

# Exibindo todas as chaves e valores
print("Chaves e Valores:")
for chave, valor in meu_dicionario.items():
    print(f"{chave}: {valor}")

# Exibindo apenas as chaves
print("\nChaves:")
for chave in meu_dicionario.keys():
    print(chave)

# Exibindo apenas os valores
print("\nValores:")
for valor in meu_dicionario.values():
    print(valor)
```

Chaves e Valores:  
nome: Carlos  
idade: 25  
cidade: São Paulo  
profissao: Estudante

Chaves:  
nome  
idade  
cidade  
profissao

Valores:  
Carlos  
25  
São Paulo  
Estudante

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

## Exemplos

Como criar um dicionário que armazena informações de contato?

```
1 # Dicionário representando informações de um contato
2 contato = {
3     'nome': 'Ana Silva',
4     'telefone': '123-456-7890',
5     'email': 'ana@email.com',
6     'idade': 30,
7     'cidade': 'São Paulo'
8 }
9
10 # Acesso às informações do contato
11 print(f"Nome: {contato['nome']}")
12 print(f"Telefone: {contato['telefone']}")
13 print(f"Email: {contato['email']}")
14 print(f"Idade: {contato['idade']}")
15 print(f"Cidade: {contato['cidade']}")
16
```

Nome: Ana Silva  
Telefone: 123-456-7890  
Email: ana@email.com  
Idade: 30  
Cidade: São Paulo

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

## Exemplos

Como criar uma lista com nomes e notas dos alunos?

```
# Dicionário de notas dos alunos  
notas_alunos = {'Alice': 85, 'Bob': 92, 'Charlie': 78, 'Diana': 95}  
  
# Utilizando um loop para exibir as notas de cada aluno  
for aluno, nota in notas_alunos.items():  
    print(f"{aluno}: {nota} pontos")
```

```
Alice: 85 pontos  
Bob: 92 pontos  
Charlie: 78 pontos  
Diana: 95 pontos
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

## Exemplos

Será que conseguimos colocar um dicionário dentro do dicionário?

```
# Dicionário aninhado representando informações sobre livros
biblioteca = {
    'livro1': {'titulo': 'Aventuras Fantásticas', 'autor': 'João Silva'},
    'livro2': {'titulo': 'Código Mestre', 'autor': 'Maria Oliveira'},
    'livro3': {'titulo': 'Noite Sombria', 'autor': 'Carlos Souza'}
}

# Acesso às informações de um livro específico
livro_id = 'livro2'
print(f"Título: {biblioteca[livro_id]['titulo']}")
print(f"Autor: {biblioteca[livro_id]['autor']}")
```

```
Título: Código Mestre
Autor: Maria Oliveira
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.





Vamos  
fazer uma  
**atividade**

## Atividade: Times de futebol

Confira as orientações para a atividade ao lado:

 **25 minutos**

 **Em grupo**

- 1** Faça uma **pesquisa, na sala de aula**, com todos os estudantes anotando o **time de futebol** de preferência.
- 2** Crie um **dicionário** com o nome **times\_futebol**, usando, como chave, o nome do time de futebol e, como valor, a quantidade de estudantes que torcem por aquele time.
- 3** Agora, crie um dicionário **quantidade\_times**, cuja chave é a quantidade e o valor, o nome do time.
- 4** Acesse o time **“Palmeiras”** nos dois dicionários.

Ao finalizar a atividade, envie pelo AVA (Ambiente Virtual de Aprendizagem) arquivo com extensão .ipynb.



© Getty Images

O que nós  
**aprendemos  
hoje?**

## Hoje desenvolvemos:

- 1** Conhecimento sobre operações com dicionário em Python, tais como: adição, remoção e atualização.
- 2** Compreensão sobre métodos como `keys()`, `values()`, e `items()` que o Python oferece para obter listas de chaves, valores e pares chave-valor.
- 3** Exercício com aplicação prática dos conceitos estudados sobre operações com dicionário em Python



# Saiba mais

Confira outras **operações de dicionários** acessando o link abaixo:

SILVEIRA, G. *Python collections parte 2: conjuntos e dicionários. Mais operações de dicionários*. Alura, 2023. Disponível em:  
<https://cursos.alura.com.br/course/python-collections-conjuntos-e-dicionarios/task/53515>. Acesso em: 13 mar. 2024.

Caso queira, também é possível **rever o conceito** de dicionário neste outro link:

COSTA, M. *Python para data science: primeiros passos. Dicionário*. Alura, 2024. Disponível em:  
<https://cursos.alura.com.br/course/python-data-science-primeiros-passos/task/122400>. Acesso em: 13 mar. 2024.

# Referências da aula

COSTA, M. *Python para data science: primeiros passos*. Dicionário. Alura, 2024. Disponível em: <https://cursos.alura.com.br/course/python-data-science-primeiros-passos/task/122400>. Acesso em: 13 mar. 2024.

MENEZES, N. N. C. *Introdução à programação com Python: algoritmos e lógica de programação para iniciantes*. São Paulo: Novatec, 2019.

SILVEIRA, G. *Python collections parte 2: conjuntos e dicionários*. Mais operações de dicionários. Alura, 2023. Disponível em: <https://cursos.alura.com.br/course/python-collections-conjuntos-e-dicionarios/task/53515>. Acesso em: 13 mar. 2024.

Identidade Visual: imagens © Getty Images



**Educação  
Profissional  
Paulista**

Técnico em  
**Ciência de  
Dados**

Ed u c a ç ã o  
P r o f i s s i o n a l  
P a u l i s t a

Técnico em  
Ciência de  
Dados



# **Programação aplicada à ciência de dados**

## **Loops "for" aplicado**

**Aula 2**

**[DADOS]ANO1C2B2S9A2**



## Objetivos da aula

Aplicar conceitos de loop, condicional, lista e string em um único exercício.



## Competências da unidade (técnicas e socioemocionais)

- Ser proficiente em linguagens de programação para manipular e analisar grandes conjuntos de dados;
- Usar técnicas para explorar e analisar dados, aplicar modelos estatísticos, identificar padrões, realizar inferências e tomar decisões baseadas em evidências;
- Colaborar efetivamente com outros profissionais, como cientistas de dados e engenheiros de dados;
- Trabalhar em equipes multifuncionais colaborando com colegas, gestores e clientes.



## Recursos didáticos

- Recurso audiovisual para exibição de vídeos e imagens;
- Acesso ao laboratório de informática e/ou à internet.
- Software Anaconda/Jupyter Notebook instalado.



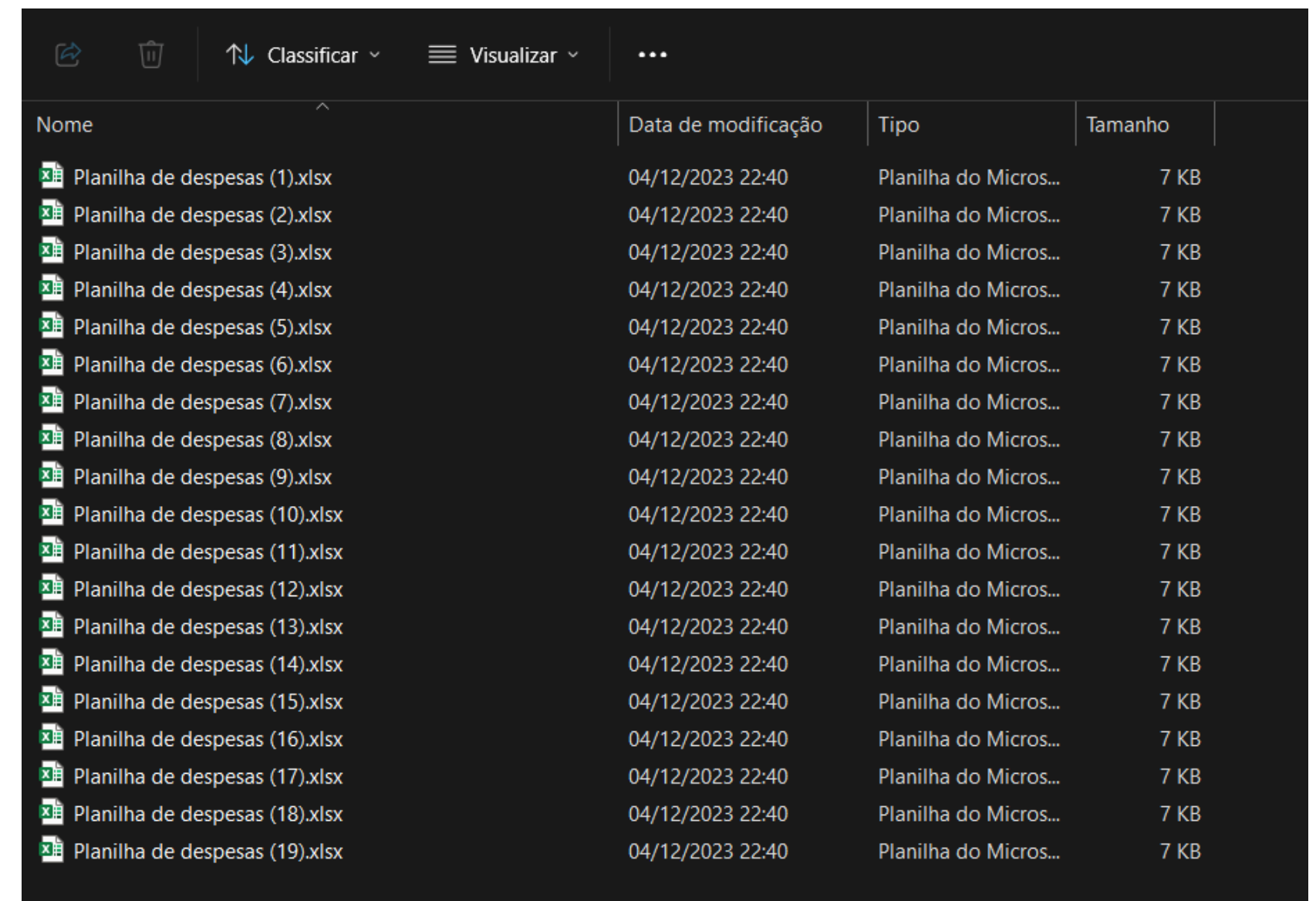
## Duração da aula

50 minutos

## Exposição

# Tarefa que pode ser automatizada com Python

Supomos que você precise renomear mais de mil arquivos dentro de uma pasta/diretório.



Nome	Data de modificação	Tipo	Tamanho
Planilha de despesas (1).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
Planilha de despesas (2).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
Planilha de despesas (3).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
Planilha de despesas (4).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
Planilha de despesas (5).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
Planilha de despesas (6).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
Planilha de despesas (7).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
Planilha de despesas (8).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
Planilha de despesas (9).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
Planilha de despesas (10).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
Planilha de despesas (11).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
Planilha de despesas (12).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
Planilha de despesas (13).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
Planilha de despesas (14).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
Planilha de despesas (15).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
Planilha de despesas (16).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
Planilha de despesas (17).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
Planilha de despesas (18).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
Planilha de despesas (19).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB

Elaborado especialmente para o curso



## Exposição

# Tarefa que pode ser automatizada com Python

Antes do nome de cada arquivo, deve aparecer a palavra “novo”. Por exemplo:

**Planilha de despesa (1).xlsx**

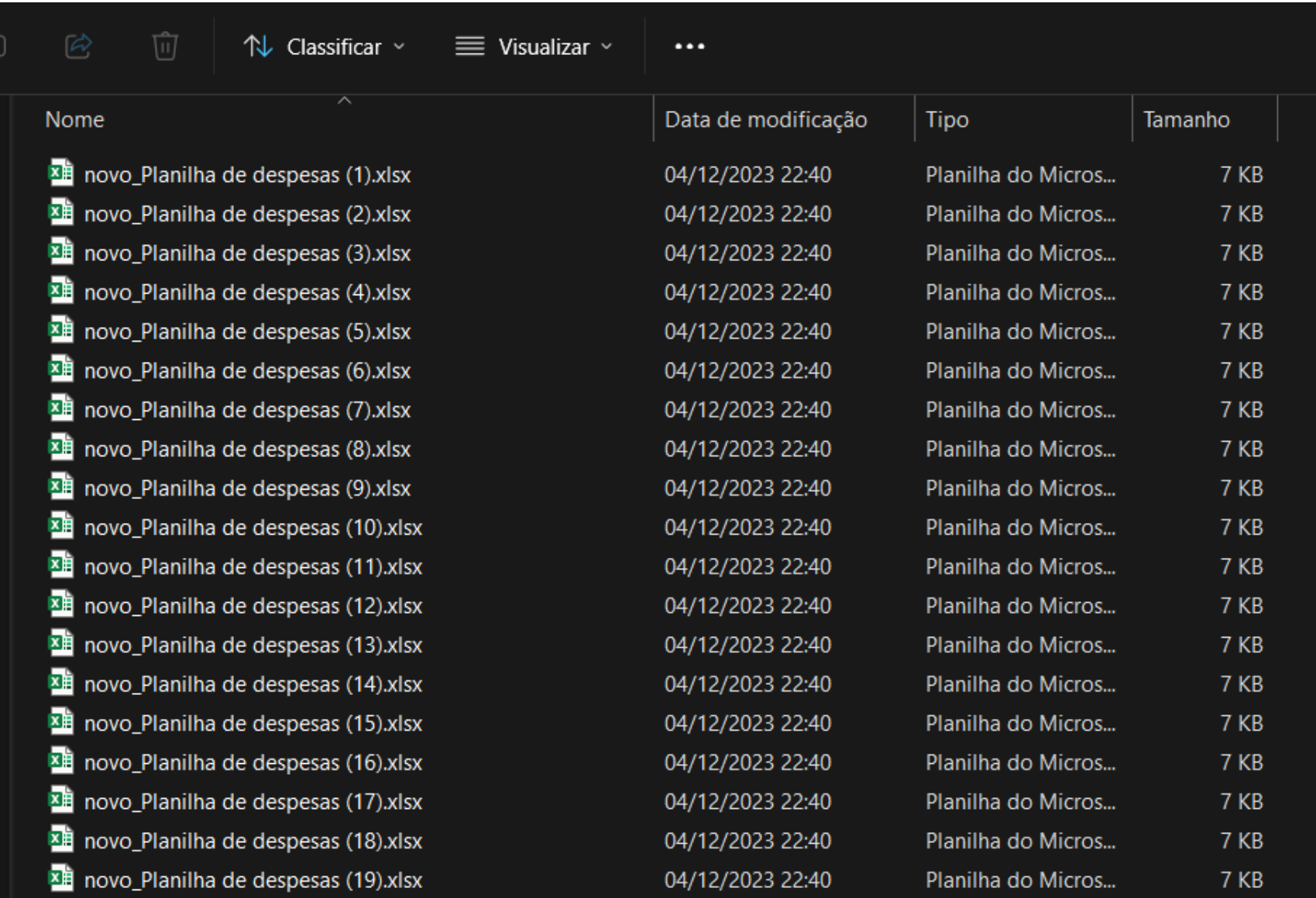
deve ser renomeada para

**novo\_Planilha de despesa (1).xlsx**



### Reflita

Como você resolveria essa questão?



Nome	Data de modificação	Tipo	Tamanho
novo_Planilha de despesas (1).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
novo_Planilha de despesas (2).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
novo_Planilha de despesas (3).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
novo_Planilha de despesas (4).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
novo_Planilha de despesas (5).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
novo_Planilha de despesas (6).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
novo_Planilha de despesas (7).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
novo_Planilha de despesas (8).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
novo_Planilha de despesas (9).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
novo_Planilha de despesas (10).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
novo_Planilha de despesas (11).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
novo_Planilha de despesas (12).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
novo_Planilha de despesas (13).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
novo_Planilha de despesas (14).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
novo_Planilha de despesas (15).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
novo_Planilha de despesas (16).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
novo_Planilha de despesas (17).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
novo_Planilha de despesas (18).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB
novo_Planilha de despesas (19).xlsx	04/12/2023 22:40	Planilha do Micros...	7 KB

Elaborado especialmente para o curso.

Exposição

## Tarefa que pode ser automatizada com Python



Refleta

Faz sentido?

```
: 1 import os
  2
  3 # Caminho para a pasta com os arquivos
  4 caminho_pasta = "C:/Users/RMM/exemplo"
  5
  6 # Prefixo a ser adicionado aos nomes dos arquivos
  7 prefixo = "novo_"
  8
  9 # Lista todos os arquivos na pasta
 10 arquivos = os.listdir(caminho_pasta)
 11
 12 # Itera sobre cada arquivo na lista e renomeia
 13 for nome_arquivo in arquivos:
 14     caminho_antigo = os.path.join(caminho_pasta, nome_arquivo)
 15     novo_nome = os.path.join(caminho_pasta, prefixo + nome_arquivo)
 16     os.rename(caminho_antigo, novo_nome)
 17
 18 print("Renomeação concluída com sucesso!")
 19
```

Renomeação concluída com sucesso!

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Vamos  
fazer uma  
**atividade**

# Renomeando arquivos de mídia para um projeto especial!

Você está participando de um projeto especial de organização de mídia, em que a renomeação criativa dos arquivos é crucial para a padronização e eficiência. Nesse laboratório, você será desafiado a utilizar Python para renomear uma lista de arquivos de mídia em uma pasta, adicionando extensões específicas com base no conteúdo do nome do arquivo.

## Problema:

Os arquivos na pasta de mídia apresentam nomes variados e não seguem um padrão consistente. Seu objetivo é converter esses nomes para maiúsculas e adicionar extensões específicas com base no conteúdo do nome do arquivo. A extensão deve ser determinada da seguinte forma:

- Se o nome do arquivo contiver "ENTREVISTA", a extensão será ".mp3";
  - Se o nome do arquivo contiver "TRAILER", a extensão será ".mp4";
  - Para todos os outros casos, a extensão será ".jpg".
- 
- `arquivos_na_pasta = ["trailer_fantasia", "entrevista_celebridade", "foto_perfil", "ensaio_fotografico"]`



Vamos  
fazer uma  
**atividade**

## Roteiro para a atividade

Confira o passo a passo para realizar a atividade proposta. Não é preciso fazer entrega nesta aula, pois haverá continuação na próxima.

 **45 minutos**

 **Em grupo**

### 1 Listagem dos arquivos:

Observe a lista `arquivos_na_pasta`, que conta com variados nomes de arquivos de mídia.

```
arquivos_na_pasta = ["trailer_fantasia",  
"entrevista_celebridade", "foto_perfil", "ensaio_fotografico"]
```

### 2 Padronização para maiúsculas:

Utilize um método de string para converter cada nome de arquivo para maiúsculas e imprima os novos nomes.

### 3 Verificação e renomeação:

- Itere sobre cada nome de arquivo na lista;
- Verifique o conteúdo do nome do arquivo para determinar a extensão associada;
- Adicione a extensão ao nome do arquivo e imprima o novo nome, simulando a renomeação.

### 4 Resultado final:

Após a renomeação, imprima uma mensagem indicando que a organização criativa do projeto de mídia foi aprimorada com sucesso.



© Getty Images

O que nós  
**aprendemos  
hoje?**

## Hoje desenvolvemos:

- 1** Aplicação de Python em tarefas do cotidiano.
- 2** Aplicações dos conceitos de loop, condicional, lista e string em um único exercício.



# Saiba mais

## Quer aprender mais sobre o módulo OS?

PYTHON SOFTWARE FOUNDATION. Diversas formas de sistema operacional. Disponível em: <https://docs.python.org/pt-br/3/library/os.html>. Acesso em: 15 fev. 2024.

## Veja mais sobre arquivos e diretórios:

MENEZES, N. N. C. Introdução à programação com Python: algoritmos e lógica de programação para iniciantes. São Paulo: Novatec, 2019.

# Referências da aula

Identidade visual: Imagens © Getty Images.

MENEZES, N. N. C. *Introdução à programação com Python: algoritmos e lógica de programação para iniciantes*. São Paulo: Novatec, 2019.

PYTHON SOFTWARE FOUNDATION. *Diversas formas de sistema operacional*. Disponível em: <https://docs.python.org/pt-br/3/library/os.html>. Acesso em: 15 fev. 2024.

Ed u c a ç ã o  
P r o f i s s i o n a l  
P a u l i s t a

Técnico em  
Ciência de  
Dados