



escola
britânica de
artes criativas
& tecnologia

Analista de Dados

Python: Arquivos & Funções

Módulo | Python: Arquivos & Funções

Caderno de Aula

Professor [André Perez](#)

Tópicos

1. Leitura;
 2. Escrita;
 3. Funções;
 4. Escopo.
-

Aulas

1. Leitura

1.1. Configuração inicial

Vamos utilizar uma função do **Google Colab** para criar arquivos para esse módulo. **Nota:** esse código **não** é do Python e sim da ferramenta.

Arquivo CSV: banco.csv

```
In [ ]: %%writefile banco.csv
age,job,marital,education,default,balance,housing,loan
30,unemployed,married,primary,no,1787,no,no
33,services,married,secondary,no,4789,yes,yes
35,management,single,tertiary,no,1350,yes,no
30,management,married,tertiary,no,1476,yes,yes
59,blue-collar,married,secondary,no,0,yes,no
35,management,single,tertiary,no,747,no,no
36,self-employed,married,tertiary,no,307,yes,no
39,technician,married,secondary,no,147,yes,no
41,entrepreneur,married,tertiary,no,221,yes,no
43,services,married,primary,no,-88,yes,yes
```

1.2. with / open

Comando para ler arquivos.

```
with open(file='<caminho do arquivo>', mode='<modo de leitura>',
encoding='<decodificador>') as <apelido>:
    bloco de código
```

Os modos de leitura são:

- **r**: Abrir o arquivo para leitura (padrão).

1.3. read

Comando para ler todo o conteúdo de um arquivo.

```
In [ ]: conteudo = None

with open(file='./banco.csv', mode='r', encoding='utf8')
as arquivo:
    conteudo = arquivo.read()

print(conteudo)
```

1.4. readline

Comando para ler o conteúdo de um arquivo uma linha por vez.

```
In [ ]: conteudo = []

with open(file='./banco.csv', mode='r', encoding='utf8')
as arquivo:
    linha = arquivo.readline() # lê a primeira linha
    while linha:
        conteudo.append(linha)
        linha = arquivo.readline()
        # lê uma nova linha,
        # se a linha não existir, salva o valor None

print(conteudo)
```

```
In [ ]: for linha in conteudo:
        print(linha)
```

Exemplo: Extraindo os valores da primeira coluna (idade).

```
In [ ]: idades = []

with open(file='./banco.csv', mode='r', encoding='utf8')
as arquivo:
    linha = arquivo.readline() # lê o cabeçalho
    linha = arquivo.readline() # lê a primeira linha
    while linha:
        linha_separada = linha.split(sep=',')
        # quebra a string nas virgulas e
        # salva os resultados em uma lista
        idade = linha_separada[0]
        # seleciona o primeiro elemento da lista
        idade = int(idade)
        # converte o valor de string para integer (inteiro)
        idades.append(idade)
        # salva o valor na lista de idades
        linha = arquivo.readline()
        # lê uma nova linha, se a linha não existir,
        # salva o valor None

print(idades)
```

2. Escrita

2.1. with / open

Comando para ler/escrever arquivos.

```
with open(file='<caminho do arquivo do arquivo>', mode='<modo de
leitura/escrita>', encoding='<decodificador>') as <apelido>:
    bloco de código
```

Os modos de leitura são:

- **r**: Abrir o arquivo para leitura (padrão);
- **w**: Abrir o arquivo para escrita (sobrescreve o arquivo original).
- **a**: Abrir o arquivo para acrescentar (não sobrescreve o arquivo original)

2.2. write

Comando para escrever em um arquivo, se o arquivo não existir, ele será criado.

- Modo de escrita (w).

```
In [ ]: with open(file='idades.csv', mode='w', encoding='utf8')
as fp:
    linha = 'idade' + '\n'
    fp.write(linha)
    for idade in idades:
        linha = str(idade) + '\n'
        fp.write(linha)
```

- Modo de acréscimo (a).

```
In [ ]: with open(file='idades.csv', mode='a', encoding='utf8')
as fp:
    for idade in idades:
        linha = str(idade + 100) + '\n'
        fp.write(linha)
```

Exemplo: Copiando um arquivo com uma extensão diferente.

```
In [ ]: %%writefile banco-texto.txt
age,job,marital,education,default,balance,housing,loan
30,unemployed,married,primary,no,1787,no,no
33,services,married,secondary,no,4789,yes,yes
35,management,single,tertiary,no,1350,yes,no
30,management,married,tertiary,no,1476,yes,yes
59,blue-collar,married,secondary,no,0,yes,no
35,management,single,tertiary,no,747,no,no
36,self-employed,married,tertiary,no,307,yes,no
39,technician,married,secondary,no,147,yes,no
41,entrepreneur,married,tertiary,no,221,yes,no
43,services,married,primary,no,-88,yes,yes
```

```
In [ ]: with open(file='./banco-texto.txt', mode='r', encoding='utf8')
as leitura:
    with open(file='./banco-csv.csv', mode='w', encoding='utf8')
    as escrita:
        linha = leitura.readline()
        while linha:
            escrita.write(linha)
            linha = leitura.readline()
```

3. Funções

3.1. Motivação

Você trabalha na bolsa de valores e precisa simular o retorno de um investimento para diversos cenários:

```
In [ ]: valor_inicial, taxa_juros_anual, anos = 1000.00, 0.05, 10

valor_final = valor_inicial
for ano in range(1, anos+1):
    valor_final = valor_final * (1 + taxa_juros_anual)
valor_final = round(valor_final, 2)
print(
    f'Para um valor inicial de R$ {valor_inicial} ' +
    ' e uma taxa de juros anual de {taxa_juros_anual}, ' +
    'em {anos} anos você terá R$ {valor_final}'
)

valor_inicial, taxa_juros_anual, anos = 1020.00, 0.03, 10

valor_final = valor_inicial
for ano in range(1, anos+1):
    valor_final = valor_final * (1 + taxa_juros_anual)
valor_final = round(valor_final, 2)
print(
    f'Para um valor inicial de R$ {valor_inicial} ' +
    'e uma taxa de juros anual de {taxa_juros_anual}, ' +
    'em {anos} anos você terá R$ {valor_final}'
)
```

Como podemos fazer para reaproveitar o código e evitar repetições?

3.2. Definição

Um bloco de código que só executado quando chamado.

```
def <nome>(<param 1>, <param 2>, ...):
    bloco de código
    return <valor de retorno>

var = <nome da funcao>(<param 1>, <param 2>, ...)
```

```
In [ ]: def imprime(mensagem: str):
        print(mensagem)
```

```
In [ ]: texto = 'Fala pessoal, meu nome é André Perez!'
```

```
In [ ]: imprime(mensagem='Fala pessoal, meu nome é André Perez!')
```

3.3. Retorno

Toda função retorna pelo menos um valor, se não especificado, retorna o valor nulo.

```
In [ ]: def maiusculo(texto: str) -> str:
        text_maiusculo = texto.upper()
        return text_maiusculo
```

```
In [ ]: nome = 'André Perez'
        print(nome)

        nome_maiusculo = maiusculo(texto=nome)
        print(nome_maiusculo)
```

```
In [ ]: def extrair_usuario_email_provedor(email: str) -> (str, str):
        email_separado = email.split(sep='@')
        usuario = email_separado[0]
        provedor = email_separado[1]
        return usuario, provedor
```

```
In [ ]: email = 'andre.perez@gmail.com'
        usuario, provedor = extrair_usuario_email_provedor(email=email)
        print(usuario)
        print(provedor)
```

3.3. Parâmetros

Parâmetros são os valores que a passamos na chamada da função.

- Função sem parâmetro:

```
In [ ]: def pi() -> float:
        return 3.14159265359
```

```
In [ ]: pi = pi()
        print(pi)
```

```
In [ ]: def imprime_pi() -> None:
        print(3.14159265359)
        return None
```

```
In [ ]: imprime_pi()
```

- Função com parâmetro:

In []:

```
def escreve_arquivo_csv(
    nome: str,
    cabecalho: str,
    conteudos: list
) -> bool:

    try:

        with open(
            file=nome,
            mode='w',
            encoding='utf8'
        ) as fp:
            linha = cabecalho + '\n'
            fp.write(linha)
            for conteudo in conteudos:
                linha = str(conteudo) + '\n'
                fp.write(linha)

    except Exception as exc:

        print(exc)
        return False

    return True
```

In []:

```
nome = 'idades-funcao-erro.csv'
cabecalho = 'idade'
# conteudos = [
# 30, 33, 35, 30, 59, 35, 36, 39, 41, 43
# ]
conteudos = 10

escreveu_com_sucesso = escreve_arquivo_csv(
    nome=nome,
    cabecalho=cabecalho,
    conteudos=conteudos
)
print(escreveu_com_sucesso)
```

3.4. Revisitando a motivação

In []:

```
def juros_compostos_anual(
    valor_inicial: float,
    taxa_juros_anual: float,
    anos: int) -> float:
    valor_final = valor_inicial
    for ano in range(1, anos+1):
        valor_final =
            valor_final * (
                1 + taxa_juros_anual
            )
    valor_final = round(valor_final, 2)
    print(
        f'Para um valor inicial de R$ {valor_inicial} ' +
        'e uma taxa de juros anual de {taxa_juros_anual}, ' +
        'em {anos} anos você terá R$ {valor_final}')
    return valor_final

valor_inicial, taxa_juros_anual, anos = 1000.00, 0.05, 10
valor_final = juros_compostos_anual(
    valor_inicial=valor_inicial,
    taxa_juros_anual=taxa_juros_anual,
    anos=anos
)

valor_inicial, taxa_juros_anual, anos = 1020.00, 0.03, 10
valor_final = juros_compostos_anual(
    valor_inicial=valor_inicial,
    taxa_juros_anual=taxa_juros_anual,
    anos=anos
)
```

4. Escopo

4.1. Definição

Define o ciclo de vida de uma variável.

- Escopo de função.

In []:

```
def soma_lista(numeros: list) -> int:
    s = 0
    for numero in numeros:
        s = s + numero
    return s
```

In []:

```
soma = soma_lista(numeros=[2] * 20)
print(soma)
```

In []:

```
print(s)
```

- Escopo de estrutura condicional / repetição.

In []:

```
if True:
    x = 100
else:
    w = 50

print(x)
```