

Módulo 04 | Python: Arquivos & Funções

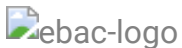
Caderno de **Aula**

Desenvolvedor [Enzo Schitini](#)

▼ 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

```
%%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
```


Comando para ler o conteúdo de um arquivo uma linha por vez. Recomendado para arquivos grandes, maiores que 100Mb

```
conteudo = []

with open(file='./banco.csv', mode='r', encoding='utf8') as arquivo:
    linha = arquivo.readline() # lê a primeira linha
    while linha: # Enquanto uma condição for verdadeira
        conteudo.append(linha)
        linha = arquivo.readline() # lê uma nova linha, se a linha não existir, salva o valor

print(conteudo)
```

```
['age,job,marital,education,default,balance,housing,loan\n', '30,unemployed,married,
```



```
for linha in conteudo:
    print(linha)

    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
```

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

```
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 vírgulas e salva os resultados
        idade = linha_separada[0] # seleciona o primeiro elemento da lista
```

```
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.

```
%%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
```

Writing banco-texto.txt

```
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:

```
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_ju
```

▼ 3.3. Retorno

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

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

```
# "-> str" tipo do retorno
```

```
nome = 'André Perez'
print(nome)
```

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

```
André Perez
ANDRÉ PEREZ
```

```
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
```

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

```
andre.perez
gmail.com
```

▼ 3.3. Parâmetros

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

- Função sem parâmetro:

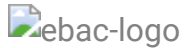
```
def pi() -> float:
    return 3.14159265359
```

```
pi = pi()
print(pi)
```

```
valor_inicial, taxa_juros_anual, anos = 1000.00, 0.05, 10
valor_final = juros_compostos_anual(valor_inicial=valor_inicial, taxa_juros_anual=taxa_jur

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

Para um valor inicial de R\$ 1000.0 e uma taxa de juros anual de 0.05, em 10 anos voc
Para um valor inicial de R\$ 1020.0 e uma taxa de juros anual de 0.03, em 10 anos voc



▼ 4. Escopo

▼ 4.1. Definição

Define o ciclo de vida de uma variável.

- Escopo de função.

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

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

40

```
#print(s) # Quando a função terminou a variavel "S" ficou vazia
```

- Escopo de estrutura condicional / repetição.

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

```
print(x)
```

100