

# Estrutura de Dados

Arquivos

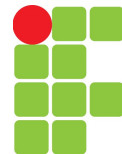
Prof. Dr. Danilo Barbosa



INSTITUTO FEDERAL DE  
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA  
PERNAMBUCO

# O que vamos ver nessa aula?

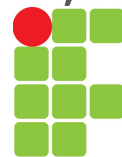
- ▶ Arquivo
  - ▶ Abrir arquivo
  - ▶ Lendo Arquivo
  - ▶ Escrevendo Arquivo
  - ▶ Fechando Arquivo



# Conceito de Arquivos

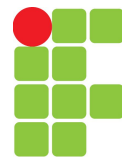
Um **arquivo** é uma área no disco onde gravamos e de onde lemos dados. Um arquivo pode ser de texto simples ou um arquivo binário;

- ▶ **Arquivos de texto** (plaintext) são uma sequência estruturada de linhas com sequência de caracteres;
- ▶ **Arquivos binários** é qualquer arquivo que não seja arquivo de texto padrão, como pdf, doc, imagens e etc.



# Abrindo arquivos em Python

- O primeiro processo natural que temos que realizar para fazer qualquer operação sobre algum arquivo é **abri-lo**;
- Para abrir um arquivo usando Python, usaremos a função **open()**. A função retorna um objeto de arquivo e é mais comumente usado com dois argumentos:




# Abrindo arquivos em Python

`open(filename, mode)`




## **Primeiro argumento(filename)**



É simplesmente o nome do arquivo que deseja abrir



## **Segundo argumento(mode)**



É uma string que indica como o arquivo vai ser aberto



# Abrindo arquivos em Python

- Exemplos de modos(modes) diferentes para abrir um arquivo são mostrados na tabela a seguir:

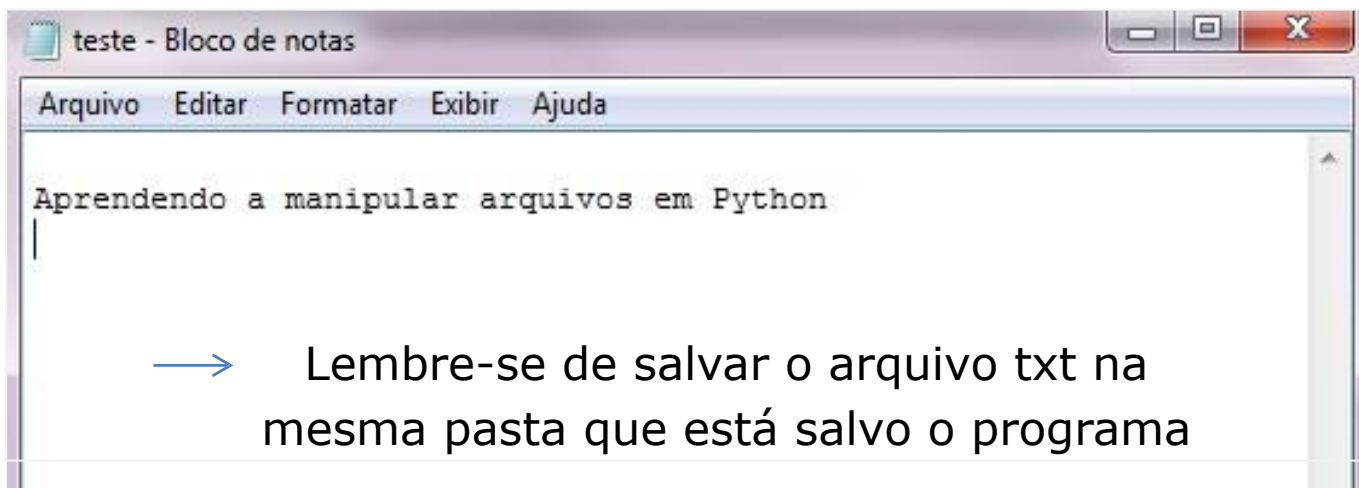
r	Abre o arquivo de texto para leitura.
r+	Abre para leitura e escrita.
w	Trunca o arquivo para zero ou cria um arquivo de texto para escrita.
w+	Abre para leitura e escrita. O arquivo é criado se ele não existir, caso contrário será sobrescrito.
a	Abre para escrita. O arquivo é criado caso não exista.
a+	Abre para leitura e escrita. O arquivo é criado se ele não existir.



# Abrindo arquivos em Python

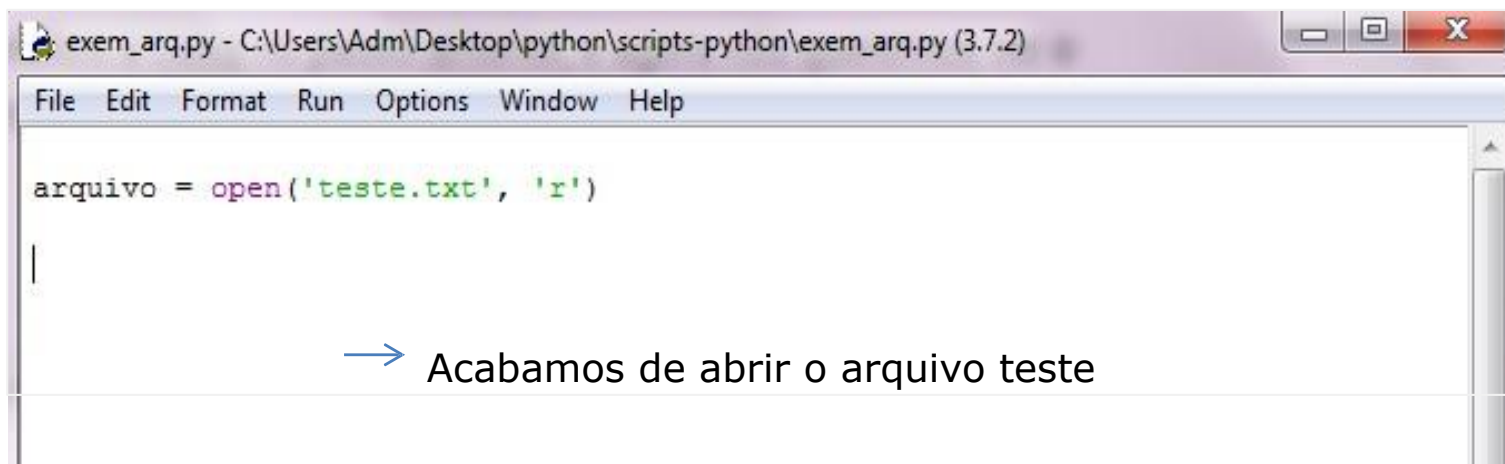
- Vamos exemplificar:

□ Vamos criar um arquivo chamado **teste**



# Abrindo arquivos em Python

- Agora abra o arquivo **teste.txt**, isso pode ser feito simplesmente com uma única linha em Python da seguinte maneira:



The screenshot shows a window titled "exem\_arq.py - C:\Users\Adm\Desktop\python\scripts-python\exem\_arq.py (3.7.2)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains the following Python code:

```
arquivo = open('teste.txt', 'r')  
|
```

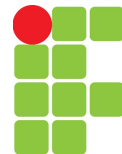
Below the code editor, there is a blue arrow pointing to the text "Acabamos de abrir o arquivo teste".



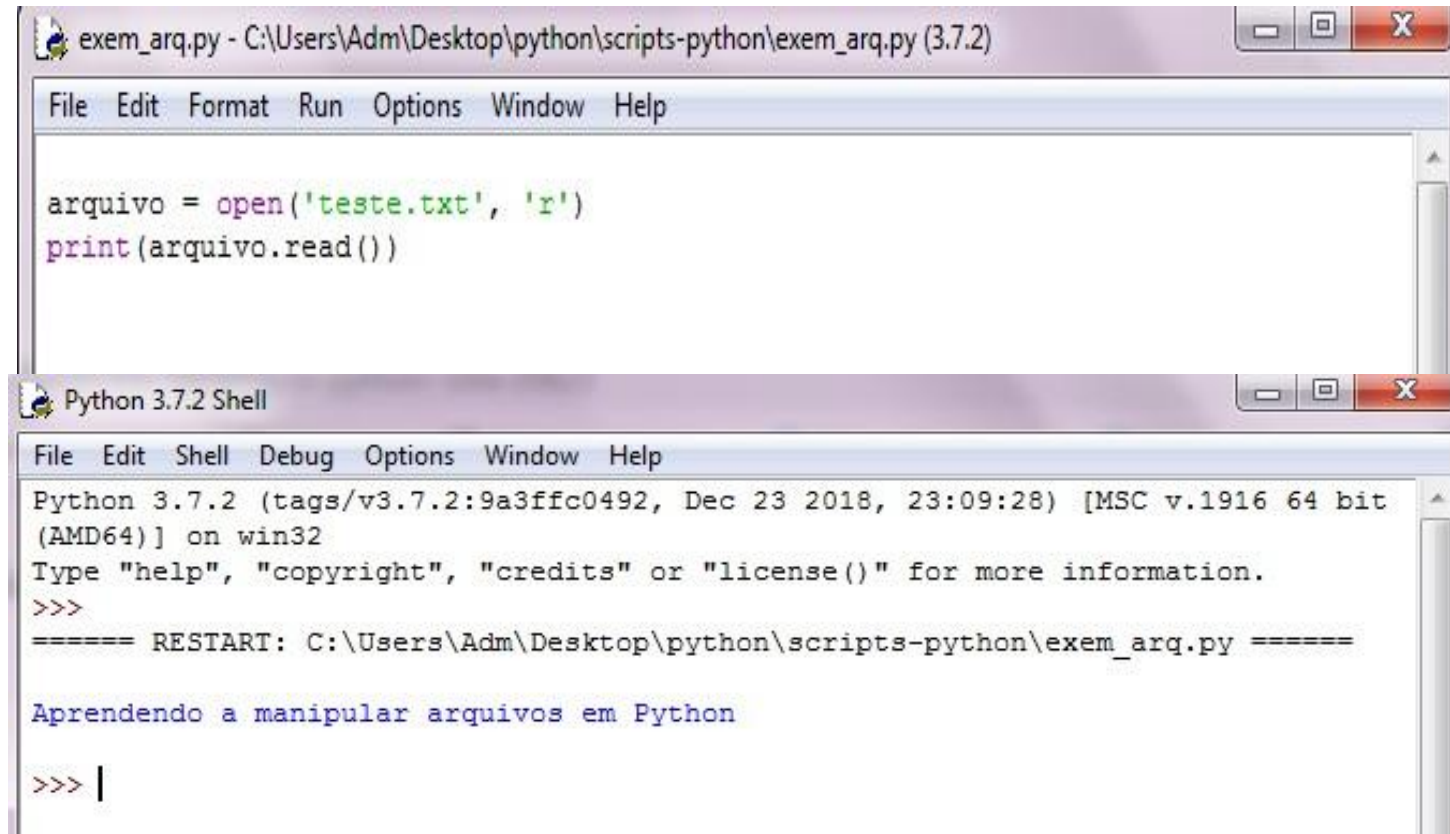


# Lendo arquivos em Python

- O arquivo é como uma caixa secreta. Nós Abrimos a caixa no passo anterior, e agora nós queremos ver o que há dentro. Ler um arquivo simples em Python pode ser feito usando o método **read()**.
- A função **read()** lê o arquivo todo de uma vez para uma variável. Isso ocorre em casos em que eu quero apenas exibir a variável.



# Lendo arquivos em Python



The image shows a screenshot of a Python IDE with two windows. The top window, titled 'exem\_arq.py - C:\Users\Adm\Desktop\python\scripts-python\exem\_arq.py (3.7.2)', contains the following Python code:

```
arquivo = open('teste.txt', 'r')
print(arquivo.read())
```

The bottom window, titled 'Python 3.7.2 Shell', shows the output of running the script. It displays the Python version and build information, followed by a restart message and the content of the file 'teste.txt'.

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Adm\Desktop\python\scripts-python\exem_arq.py =====
Aprendendo a manipular arquivos em Python
>>> |
```



# Lendo arquivos em Python Linha por linha

- Além do método `read()`, visto anteriormente, também podemos ler o conteúdo de um arquivo usando os métodos **`readline()`** e **`readlines()`**.

## Readline()

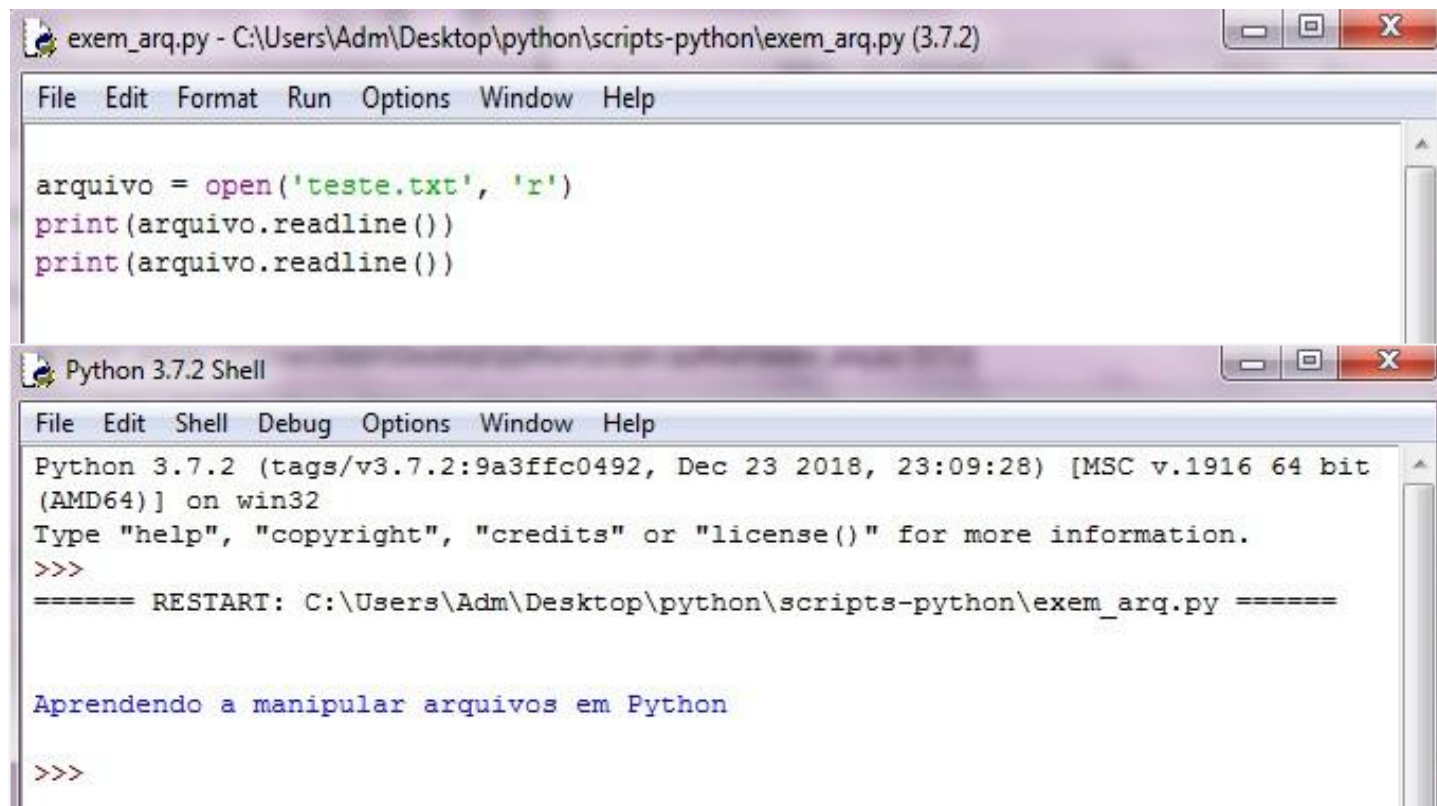
Retorna uma linha do texto a cada chamada, na ordem em que aparecem no arquivo

## Readlines()

Retorna uma lista de valores de string do arquivo, sendo que cada string corresponde a uma linha do texto.



# Lendo arquivos em Python Linha por linha



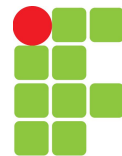
The image shows two windows from a Windows operating system. The top window is a text editor titled 'exem\_arq.py - C:\Users\Adm\Desktop\python\scripts-python\exem\_arq.py (3.7.2)'. It contains the following Python code:

```
arquivo = open('teste.txt', 'r')
print(arquivo.readline())
print(arquivo.readline())
```

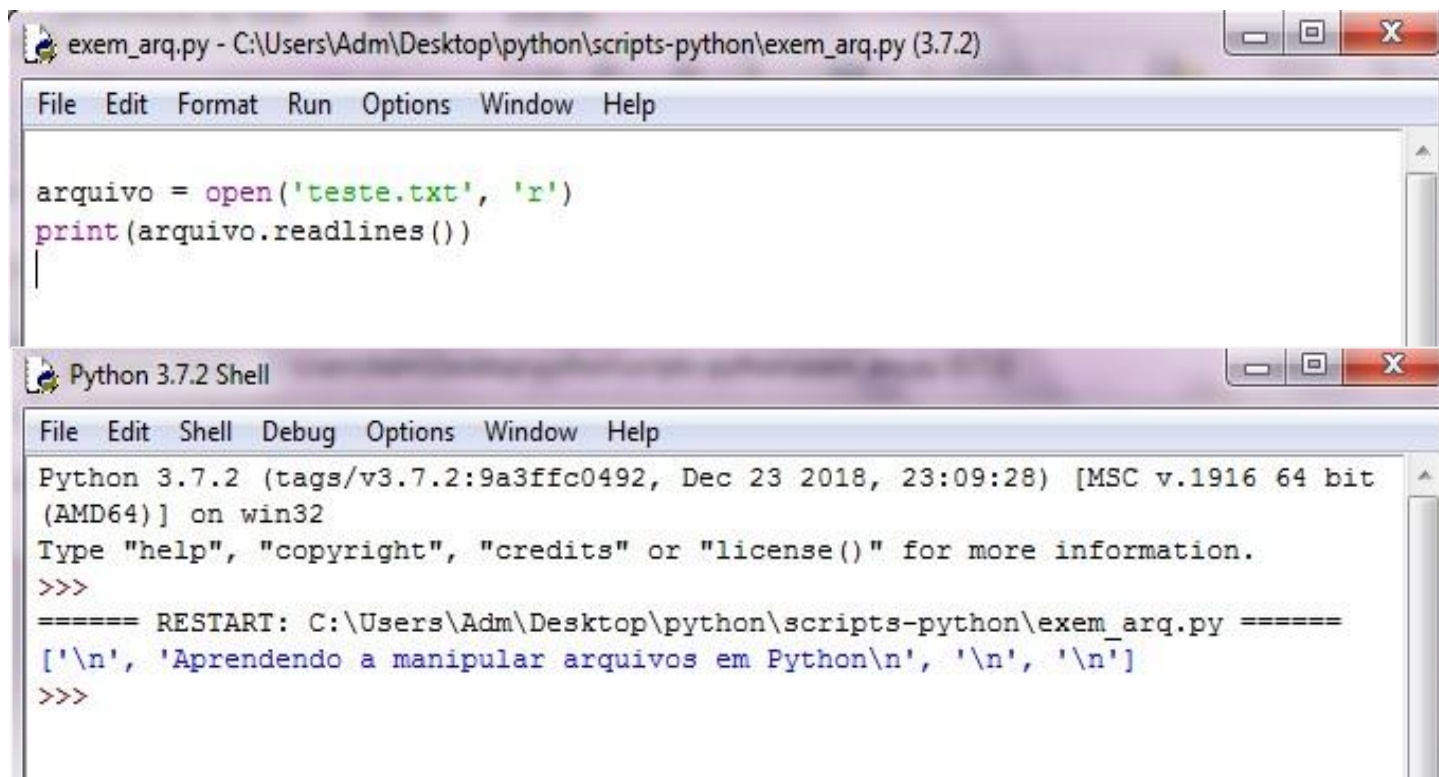
The bottom window is a 'Python 3.7.2 Shell' with a menu bar (File, Edit, Shell, Debug, Options, Window, Help). It displays the output of running the script:

```
Python 3.7.2 (tags/v3.7.2:9a3fffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Adm\Desktop\python\scripts-python\exem_arq.py =====

Aprendendo a manipular arquivos em Python
>>>
```



# Lendo arquivos em Python Linha por linha

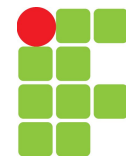


The image shows a screenshot of a Python IDE window titled "exem\_arq.py - C:\Users\Adm\Desktop\python\scripts-python\exem\_arq.py (3.7.2)". The code in the editor is:

```
arquivo = open('teste.txt', 'r')
print(arquivo.readlines())
```

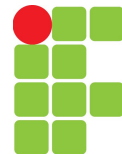
Below the editor is a "Python 3.7.2 Shell" window showing the execution output:

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Adm\Desktop\python\scripts-python\exem_arq.py =====
['\n', 'Aprendendo a manipular arquivos em Python\n', '\n', '\n']
>>>
```

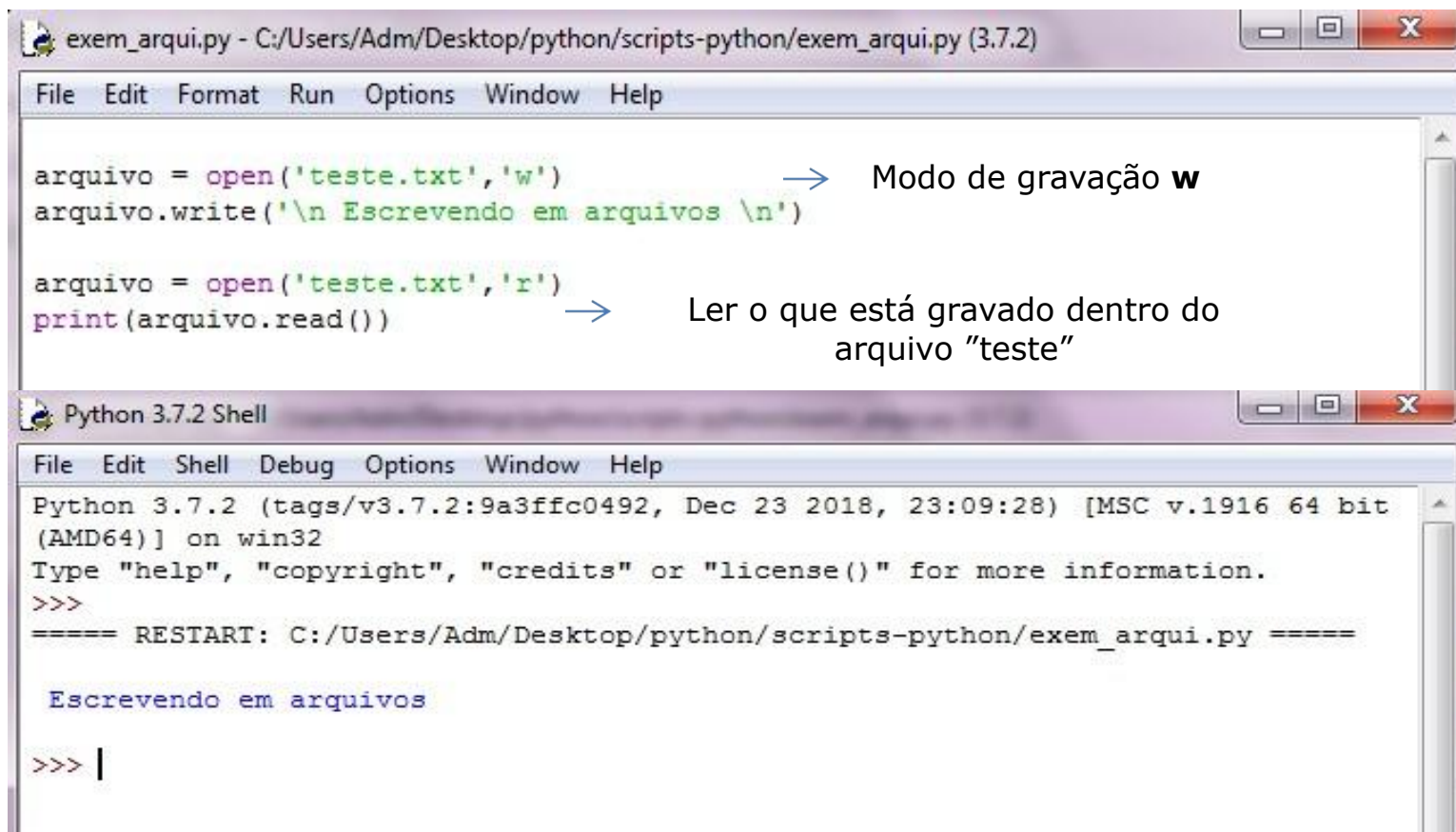


# Escrevendo arquivos em Python

- Talvez você queira escrever outra frase ou parágrafo no arquivo que já lemos. Digamos que queria adicionar a seguinte frase: **Escrevendo em arquivos**. Isso pode ser feito em Python usando o método **write()**;
- A função **write()** utiliza como argumento a **string** que desejamos gravar no nosso arquivo.



# Escrevendo arquivos em Python



The image shows a screenshot of a Python IDE window titled "exem\_arqui.py - C:/Users/Adm/Desktop/python/scripts-python/exem\_arqui.py (3.7.2)". The window contains the following Python code:

```
arquivo = open('teste.txt', 'w')
arquivo.write('\n Escrevendo em arquivos \n')

arquivo = open('teste.txt', 'r')
print(arquivo.read())
```

Annotations with arrows point to the mode letters in the code:

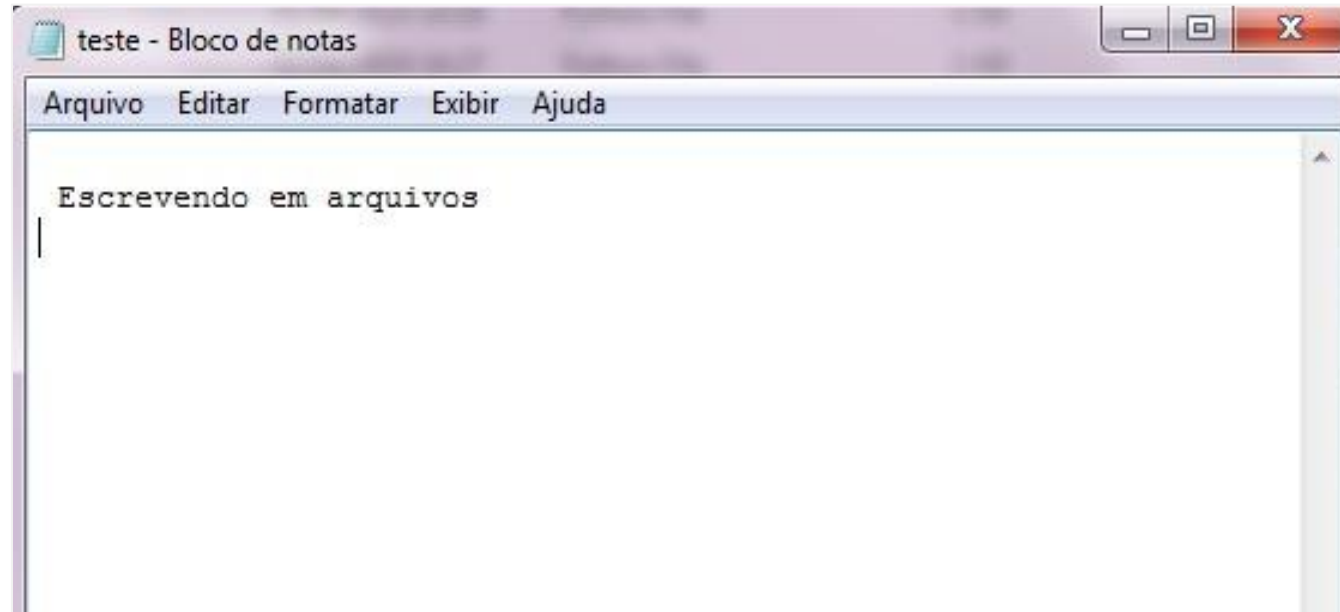
- An arrow points to the `'w'` in `open('teste.txt', 'w')` with the text "Modo de gravação **w**".
- An arrow points to the `'r'` in `open('teste.txt', 'r')` with the text "Ler o que está gravado dentro do arquivo 'teste'".

Below the code editor is a "Python 3.7.2 Shell" window. It displays the following output:

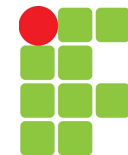
```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/Adm/Desktop/python/scripts-python/exem_arqui.py =====
Escrevendo em arquivos
>>> |
```



# Escrevendo arquivos em Python



Antes o nosso arquivo **"teste"** continha a frase **"Aprendendo a manipular arquivos em Python"** agora o nosso arquivo gravou a nova frase adicionada pelo comando **write**.

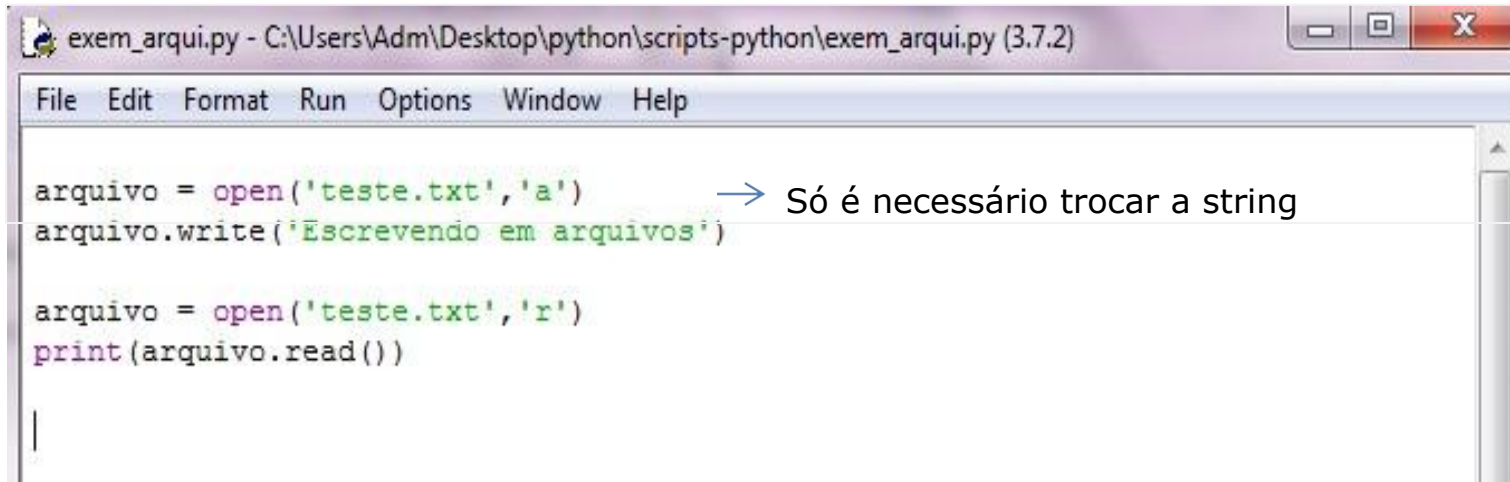




# Escrevendo arquivos em Python

Se você está interessado em manter o texto original do documento, você pode usar o modo '**a**':

:



```
exem_arqui.py - C:\Users\Adm\Desktop\python\scripts-python\exem_arqui.py (3.7.2)
File Edit Format Run Options Window Help

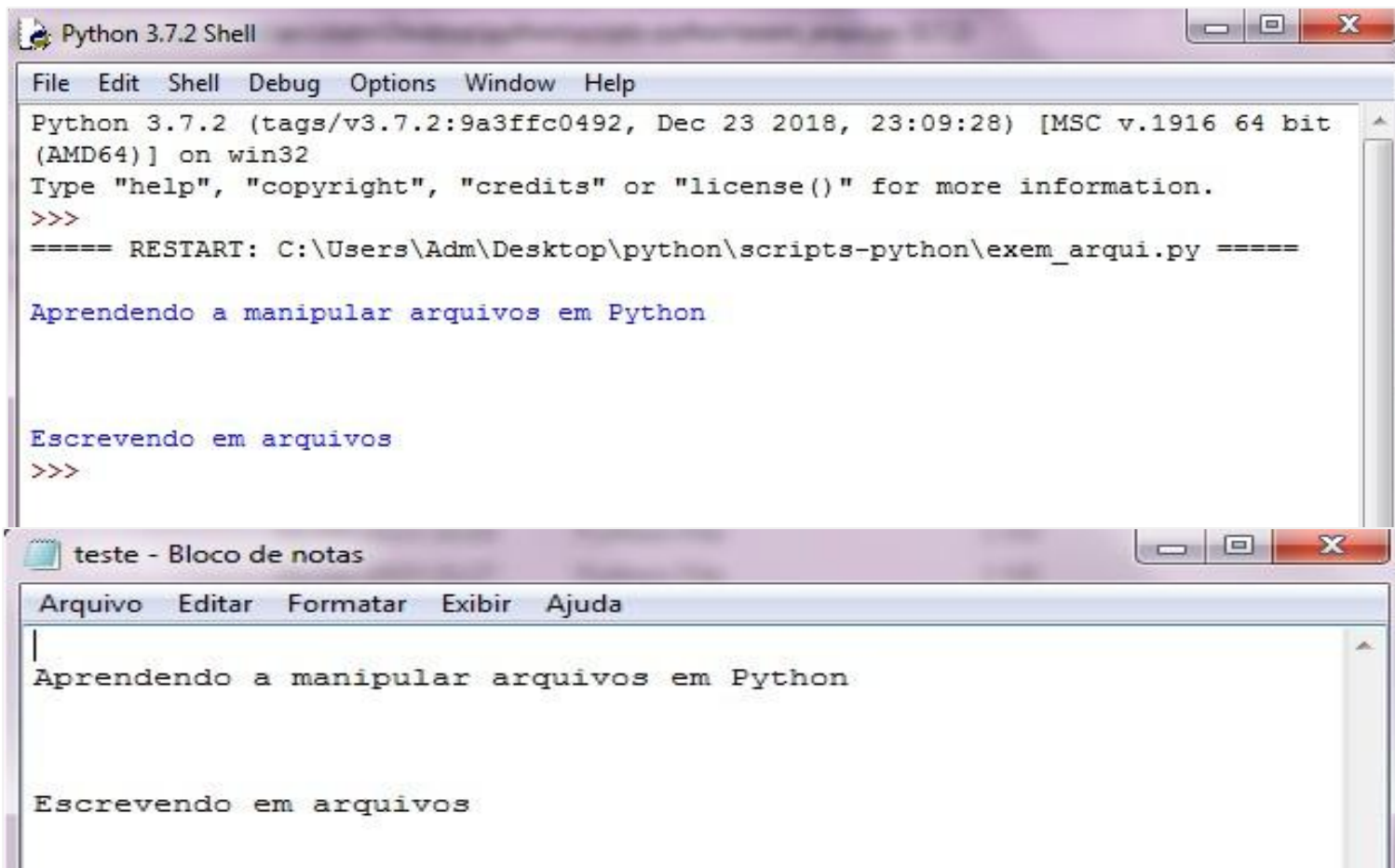
arquivo = open('teste.txt','a')
arquivo.write('Escrevendo em arquivos')

arquivo = open('teste.txt','r')
print(arquivo.read())

|
```



# Escrevendo arquivos em Python



The image shows two overlapping windows. The top window is titled 'Python 3.7.2 Shell' and displays the Python interpreter's startup sequence, including version information and a restart command for a file named 'exem\_arqui.py'. Below this, the text 'Aprendendo a manipular arquivos em Python' and 'Escrevendo em arquivos' is shown, followed by a prompt '>>>'. The bottom window is titled 'teste - Bloco de notas' and contains the same two lines of text: 'Aprendendo a manipular arquivos em Python' and 'Escrevendo em arquivos'.

```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Adm\Desktop\python\scripts-python\exem_arqui.py =====

Aprendendo a manipular arquivos em Python

Escrevendo em arquivos
>>>
```

```
teste - Bloco de notas
Arquivo Editar Formatar Exibir Ajuda

Aprendendo a manipular arquivos em Python

Escrevendo em arquivos
```

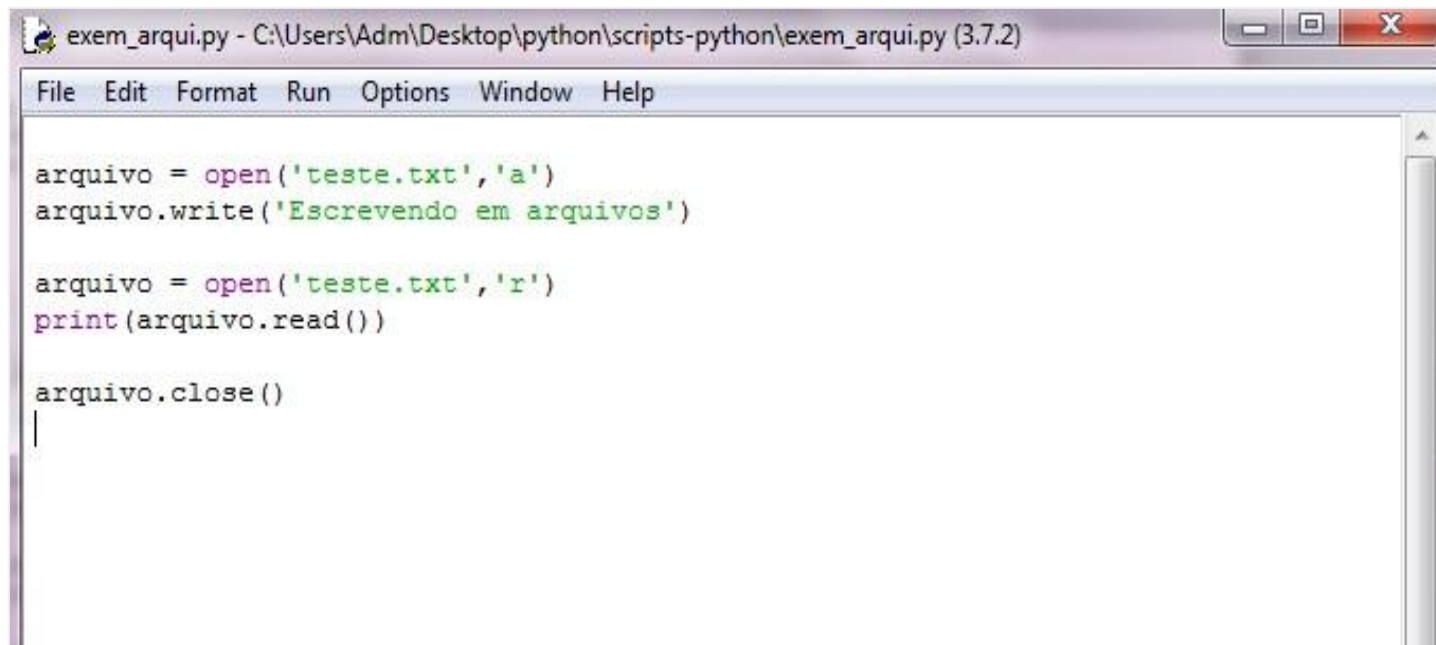


# Fechando arquivos em Python

- Ter o hábito de fechar um arquivo após a leitura ou escrita permitirá que você libere memória. Sim, o Python fecha automaticamente os arquivos depois que o script termina, mas se você não fizer isso de antemão, todos os arquivos abertos ocuparão espaço que o Python poderia usar. →
- Fechar um arquivo pode ser facilmente realizado usando o método **close()**.



# Fechando arquivos em Python

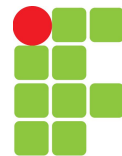


```
exem_arqui.py - C:\Users\Adm\Desktop\python\scripts-python\exem_arqui.py (3.7.2)
File Edit Format Run Options Window Help

arquivo = open('teste.txt','a')
arquivo.write('Escrevendo em arquivos')

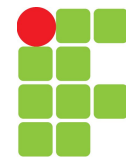
arquivo = open('teste.txt','r')
print(arquivo.read())

arquivo.close()
|
```

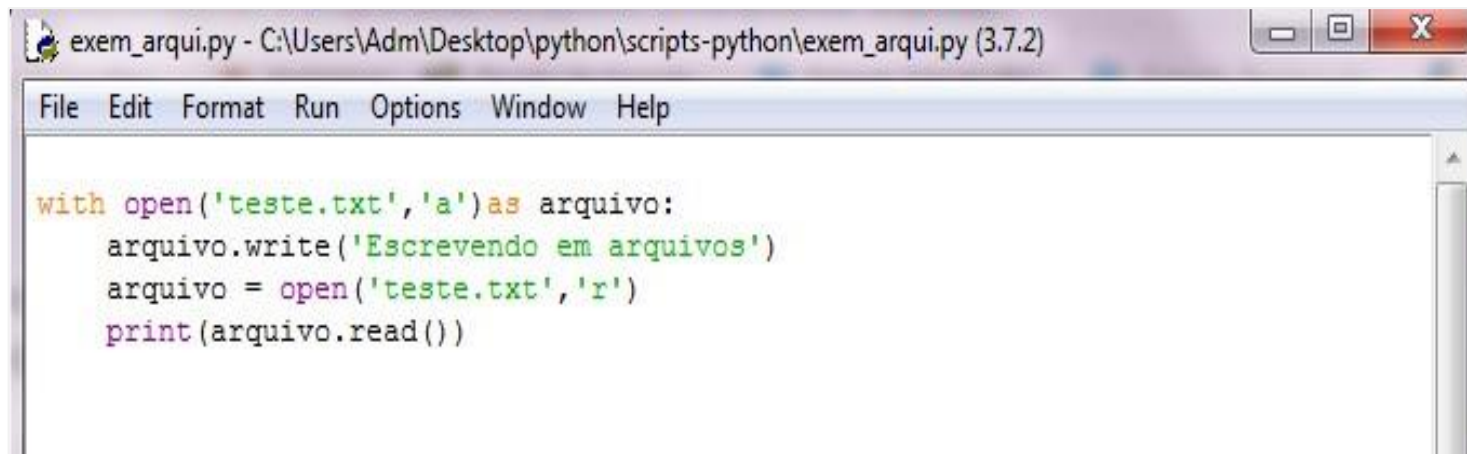


# Fechando arquivos em Python

- Outra forma de lembrarmos de fechar o arquivo é utilizando o comando **with()**;
- O comando **with()** fecha o arquivo automaticamente assim que saímos de sua suite, ele nos permite → abrir o arquivo, processá-lo e certificar-se de que está fechado.

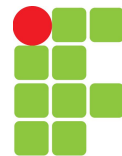


# Fechando arquivos em Python

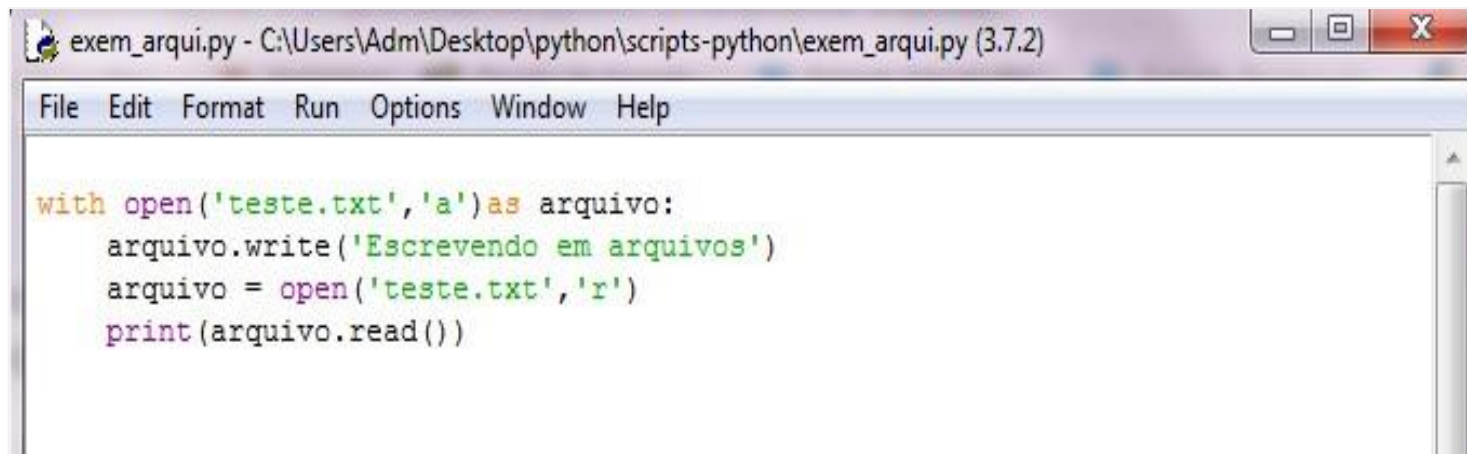


```
exem_arqui.py - C:\Users\Adm\Desktop\python\scripts-python\exem_arqui.py (3.7.2)
File Edit Format Run Options Window Help

with open('teste.txt','a') as arquivo:
    arquivo.write('Escrevendo em arquivos')
arquivo = open('teste.txt','r')
print(arquivo.read())
```

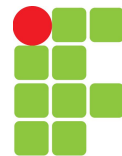


# Fechando arquivos em Python



```
exem_arqui.py - C:\Users\Adm\Desktop\python\scripts-python\exem_arqui.py (3.7.2)
File Edit Format Run Options Window Help

with open('teste.txt','a') as arquivo:
    arquivo.write('Escrevendo em arquivos')
arquivo = open('teste.txt','r')
print(arquivo.read())
```



# Exercício 1

- ✓ Crie um arquivo em Python chamado "**novo.txt**" e salve seu nome no arquivo e logo depois leia o arquivo.





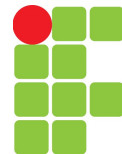
## Exercício 2

- ✓ Com o arquivo criado “**novo.txt**” acrescente a data de nascimento, cidade que nasceu e sua idade e depois leia o arquivo normalmente.



## Exercício 3

✓ Ainda utilizando o arquivo criado “**novo.txt**” vamos ler o que está dentro do arquivo mas agora utilizando um laço for.



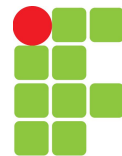
## Exercício 4

✓ Continuando o exercício anterior, vamos agora contar quantas linhas tem no arquivo **novo.txt**.



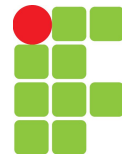
## Exercício 5

- ✓ Crie um novo arquivo chamado **frases.txt** com as seguintes frases: “**Estudando Python, programando em Python e manipulação de arquivos**” e agora crie um programa que retorne somente as linhas que possuem a palavra **Python** .



## Exercício 6

- ✓ Seguindo o exercício anterior, agora vamos criar um programa que retorne somente as linhas que contém uma palavra específica a onde nós iremos dizer qual será a palavra de busca.



# Crud em python com arquivo

# Função para adicionar um novo usuário ao arquivo

**def criar\_usuario(nome):**

# Abre o arquivo 'usuarios.txt' no modo de adição ('a'). Isso permite adicionar dados ao final do arquivo.

**with open('usuarios.txt', 'a') as f:**

# Gera o ID do usuário com base no número de linhas do arquivo.

# Cada linha representa um usuário, então o ID será o número de linhas + 1.

**id\_usuario = len(open('usuarios.txt').readlines()) + 1**

# Escreve o novo usuário no arquivo no formato "ID,Nome", seguido de uma nova linha.

**f.write(f'{id\_usuario},{nome}\n')**

# Exibe uma mensagem informando que o usuário foi adicionado com sucesso.

**print(f"Usuário {nome} adicionado com sucesso.")**



# Crud em python com arquivo

# Função para listar todos os usuários armazenados no arquivo

**def listar\_usuarios():**

**try:**

# Abre o arquivo 'usuarios.txt' no modo de leitura ('r') para obter os dados.

**with open('usuarios.txt', 'r') as f:**

# Lê todas as linhas do arquivo e as armazena em uma lista.

**usuarios = f.readlines()**

# Verifica se o arquivo está vazio. Se não houver usuários, exibe uma mensagem.

**if not usuarios:**

**print("Nenhum usuário encontrado.")**

**else:**

# Itera sobre as linhas do arquivo, que representam os usuários.

**for usuario in usuarios:**

# Separa cada linha em ID e Nome, baseando-se na vírgula.

**id\_usuario, nome = usuario.strip().split(',')**

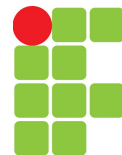
# Exibe o ID e o nome do usuário no formato adequado.

**print(f"ID: {id\_usuario}, Nome: {nome}")**

**except FileNotFoundError:**

# Caso o arquivo não exista, exibe uma mensagem de erro.

**print("Arquivo de usuários não encontrado.")**



# Crud em python com arquivo

```
def atualizar_usuario(id_usuario, novo_nome):
```

```
    try:
```

```
        # Abre o arquivo 'usuarios.txt' no modo de leitura ('r') para carregar os dados.
```

```
        with open('usuarios.txt', 'r') as f:
```

```
            usuarios = f.readlines()
```

```
        # Abre o arquivo 'usuarios.txt' no modo de escrita ('w') para sobrescrever o arquivo com as alterações
```

```
        with open('usuarios.txt', 'w') as f:
```

```
            # Itera sobre a lista de usuários carregada.
```

```
            for usuario in usuarios:
```

```
                # Separa cada linha em ID e Nome, baseado na vírgula.
```

```
                id_atual, nome = usuario.strip().split(',')
```

```
                # Verifica se o ID atual corresponde ao ID fornecido.
```

```
                if int(id_atual) == id_usuario:
```

```
                    # Se encontrar o ID, escreve o novo nome para aquele usuário.
```

```
                    f.write(f'{id_usuario},{novo_nome}\n')
```

```
                    print(f"Usuário {id_usuario} atualizado para {novo_nome}.")
```

```
                else:
```

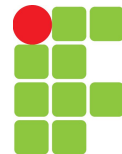
```
                    # Se o ID não corresponder, escreve a linha original (sem alterações).
```

```
                    f.write(usuario)
```

```
    except FileNotFoundError:
```

```
        # Caso o arquivo não exista, exibe uma mensagem de erro.
```

```
        print("Arquivo de usuários não encontrado.")
```





# Crud em python com arquivo

```
def deletar_usuario(id_usuario):
```

```
    try:
```

```
        # Abre o arquivo 'usuarios.txt' no modo de leitura ('r') para carregar os dados.
```

```
        with open('usuarios.txt', 'r') as f:
```

```
            usuarios = f.readlines()
```

```
        # Abre o arquivo 'usuarios.txt' no modo de escrita ('w') para sobrescrever o arquivo com as alterações
```

```
        with open('usuarios.txt', 'w') as f:
```

```
            # Itera sobre a lista de usuários carregada.
```

```
            for usuario in usuarios:
```

```
                # Separa cada linha em ID e Nome, baseado na vírgula.
```

```
                id_atual, nome = usuario.strip().split(',')
```

```
                # Verifica se o ID não corresponde ao ID fornecido.
```

```
                if int(id_atual) != id_usuario:
```

```
                    # Se o ID não for o do usuário a ser deletado, escreve a linha no arquivo.
```

```
                    f.write(usuario)
```

```
            else:
```

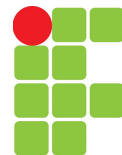
```
                # Se encontrar o usuário a ser deletado, exibe uma mensagem de confirmação.
```

```
                print(f"Usuário {id_usuario} deletado.")
```

```
except FileNotFoundError:
```

```
    # Caso o arquivo não exista, exibe uma mensagem de erro.
```

```
    print("Arquivo de usuários não encontrado.")
```



# Crud em python com arquivo

```
def deletar_usuario(id_usuario):
```

```
    try:
```

```
        # Abre o arquivo 'usuarios.txt' no modo de leitura ('r') para carregar os dados.
```

```
        with open('usuarios.txt', 'r') as f:
```

```
            usuarios = f.readlines()
```

```
        # Abre o arquivo 'usuarios.txt' no modo de escrita ('w') para sobrescrever o arquivo com as alterações
```

```
        with open('usuarios.txt', 'w') as f:
```

```
            # Itera sobre a lista de usuários carregada.
```

```
            for usuario in usuarios:
```

```
                # Separa cada linha em ID e Nome, baseado na vírgula.
```

```
                id_atual, nome = usuario.strip().split(',')
```

```
                # Verifica se o ID não corresponde ao ID fornecido.
```

```
                if int(id_atual) != id_usuario:
```

```
                    # Se o ID não for o do usuário a ser deletado, escreve a linha no arquivo.
```

```
                    f.write(usuario)
```

```
            else:
```

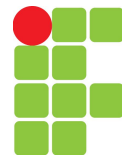
```
                # Se encontrar o usuário a ser deletado, exibe uma mensagem de confirmação.
```

```
                print(f"Usuário {id_usuario} deletado.")
```

```
except FileNotFoundError:
```

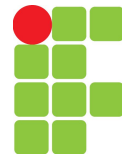
```
    # Caso o arquivo não exista, exibe uma mensagem de erro.
```

```
    print("Arquivo de usuários não encontrado.")
```



# Crud em python com arquivo

```
def menu():  
    while True:  
        # Exibe o menu com as opções disponíveis.  
        print("\n1. Criar Usuário")  
        print("2. Listar Usuários")  
        print("3. Atualizar Usuário")  
        print("4. Deletar Usuário")  
        print("5. Sair")  
  
        # Pede ao usuário para escolher uma opção.  
        opcao = input("Escolha uma opção: ")
```



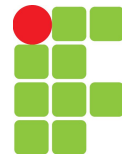
# Crud em python com arquivo

```
# Executa a ação correspondente à opção escolhida.  
if opcao == '1':  
    # Se a opção for 1, solicita o nome do usuário e chama a função  
    para criar o usuário.  
    nome = input("Digite o nome do usuário: ")  
    criar_usuario(nome)  
elif opcao == '2':  
    # Se a opção for 2, chama a função para listar todos os usuários.  
    listar_usuarios()  
elif opcao == '3':  
    # Se a opção for 3, solicita o ID do usuário e o novo nome, e  
    chama a função para atualizar o usuário.  
    id_usuario = int(input("Digite o ID do usuário a ser atualizado: "))  
    novo_nome = input("Digite o novo nome do usuário: ")  
    atualizar_usuario(id_usuario, novo_nome)
```



# Crud em python com arquivo

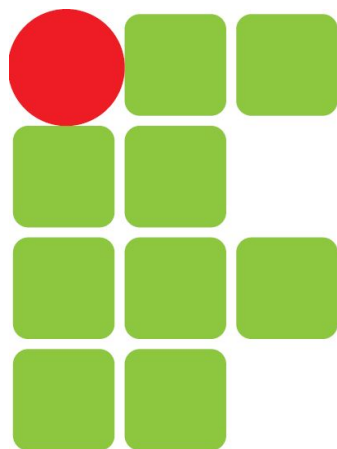
```
elif opcao == '4':  
    # Se a opção for 4, solicita o ID do usuário a ser deletado e chama a  
    função para deletar o usuário.  
    id_usuario = int(input("Digite o ID do usuário a ser deletado: "))  
    deletar_usuario(id_usuario)  
elif opcao == '5':  
    # Se a opção for 5, o loop é interrompido e o programa é encerrado.  
    break  
else:  
    # Se a opção for inválida, exibe uma mensagem de erro.  
    print("Opção inválida, tente novamente.")  
  
# Inicia o programa chamando a função 'menu', que exibe as opções e permite  
a interação com o usuário.  
menu()
```



# Referência

- ▶ CELES FILHO, W.; CERQUEIRA, R.; RANGEL, J. L. Introdução da Estrutura de Dados: com técnicas de programação em C. Rio de Janeiro: Elsevier, 2014.
- ▶ LEITE, Thiago et al. Estruturas de Dados: Domine as práticas essenciais em C, Java, C#, Python e JavaScript. Casa do Código, 2023.
- ▶ FORBELLONE, André LV; EBERSPÄCHER, Henri F. Lógica de Programação-: A Construção de Algoritmos e Estruturas de Dados com Aplicações em Python. Bookman Editora, 2022.





**INSTITUTO FEDERAL DE  
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA**  
**PERNAMBUCO**