

Atividade S3 A2

Nome: Adalberto Caldeira Brant Filho

Repositório GitHub: <https://github.com/adalbertobrant/lipai>

Código das Videoaulas

A função open() em python é utilizada para abrir arquivos em diretórios uma das formas que pode ser escrita é:

```
# abre arquivo , nome do arquivo, modo de leitura, codificação do arquivo  
# (opcional)  
open("caminho_do_arquivo.extensão", "modo_de_abertura_do_arquivo",  
enconding='tipo_de_codificação_do_arquivo')  
# resto do programa
```

Tabela com modos de abertura de arquivos em python

Símbolo	Explicação
r	leitura
a	append / incrementar
w	escrita
x	criar arquivo
r+	leitura e escrita

```
""" Aula 01 - Manipulação de Arquivos """  
  
# abertura do arquivo  
  
arquivo = open("src/06-arquivos/teste.txt", "r", encoding='utf-8')  
  
# verifica se o arquivo tem condições de ser lido  
print(arquivo.readable())  
  
# fechar o arquivo  
arquivo.close()
```

A função readable() verifica se o arquivo pode ser lido ou não, ou seja se o mesmo foi aberto com autorização de leitura, caso tenha sido escrito na forma:

```
arquivo = open('teste.txt', 'w', enconding='utf-8')  
print(arquivo.readable())
```

O resultado esperado é False, pois o arquivo tem apenas a propriedade de escrita que é 'w'. Podemos ler todo um arquivo usando a função read():

```
# abertura do arquivo

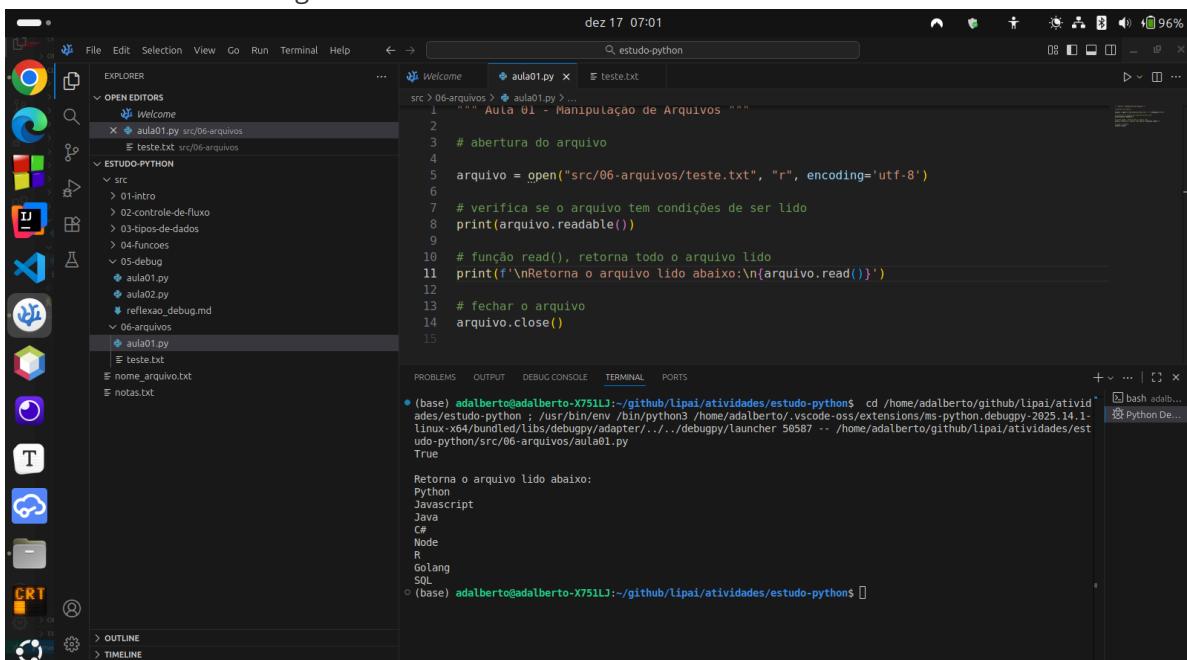
arquivo = open("src/06-arquivos/teste.txt", "r", encoding='utf-8')

# verifica se o arquivo tem condições de ser lido
print(arquivo.readable())

# função read(), retorna todo o arquivo lido
print(f'\nRetorna o arquivo lido abaixo:\n{arquivo.read()}\n')

# fechar o arquivo
arquivo.close()
```

O resultado desse código é demonstrado abaixo:

A screenshot of the Visual Studio Code interface. The left sidebar shows a file tree with a project named 'ESTUDO-PYTHON' containing several subfolders and files. In the center, there's a code editor window with the file 'aula01.py' open, displaying the code provided in the previous block. Below the code editor is a terminal window showing the command line output of the Python script. The terminal output includes the code itself, followed by the message 'Retorna o arquivo lido abaixo:', and then the contents of the file 'teste.txt' which is 'Aula 01 - Manipulação de Arquivos'.

```
dez 17 07:01
File Edit Selection View Go Run Terminal Help ← → estudo.python
src > 06-arquivos > aula01.py > ...
1 """ Aula 01 - Manipulação de Arquivos """
2
3 # abertura do arquivo
4
5 arquivo = open("src/06-arquivos/teste.txt", "r", encoding='utf-8')
6
7 # verifica se o arquivo tem condições de ser lido
8 print(arquivo.readable())
9
10 # função read(), retorna todo o arquivo lido
11 print(f'\nRetorna o arquivo lido abaixo:\n{arquivo.read()}\n')
12
13 # fechar o arquivo
14 arquivo.close()
15

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
• (base) adalberto@adalberto-X751L:~/github/lipai/atividades/estudo-python$ cd /home/adalberto/github/lipai/atividades/estudo/python ; /usr/bin/env /bin/python3 /home/adalberto/.vscode-oss/extensions/ms-python.debugpy-2025.14.1-linux-x64/bundled/Libs/debugpy/adapter/.../debugpy/launcher 50587 -- /home/adalberto/github/lipai/atividades/estudo-python/src/06-arquivos/aula01.py
True
Retorna o arquivo lido abaixo:
Python
Javascript
Java
C++
Node
R
Golang
SQL
• (base) adalberto@adalberto-X751L:~/github/lipai/atividades/estudo-python$
```

Ao executar o tutorial do vídeo explicativo, como não comentei as linhas a próxima função a ser estudada que é a readline(), estava retornando linha vazia, pois ao executar a função read() o python leu já todo o arquivo e está no final dele.

Para que o python volte ao ponto inicial do arquivo é necessário indicar isso através do comando seek():

```
""" Aula 01 - Manipulação de Arquivos """

# abertura do arquivo

arquivo = open("teste.txt", "r", encoding='utf-8')

# verifica se o arquivo tem condições de ser lido
print(arquivo.readable())

# função read(), retorna todo o arquivo lido
print(f'\nRetorna o arquivo lido abaixo:\n{arquivo.read()}\n')

# função para retornar o cursor de leitura para o início do arquivo
```

```

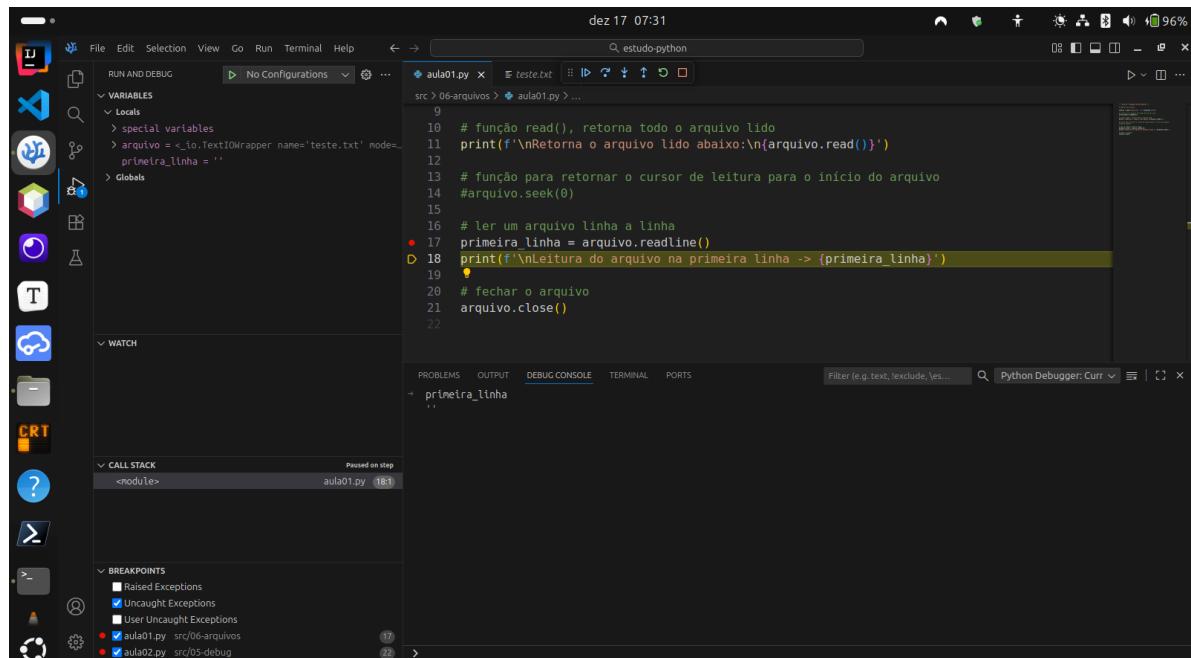
arquivo.seek(0)

# ler um arquivo linha a linha
primeira_linha = arquivo.readline()
print(f'\nLeitura do arquivo na primeira linha -> {primeira_linha}')

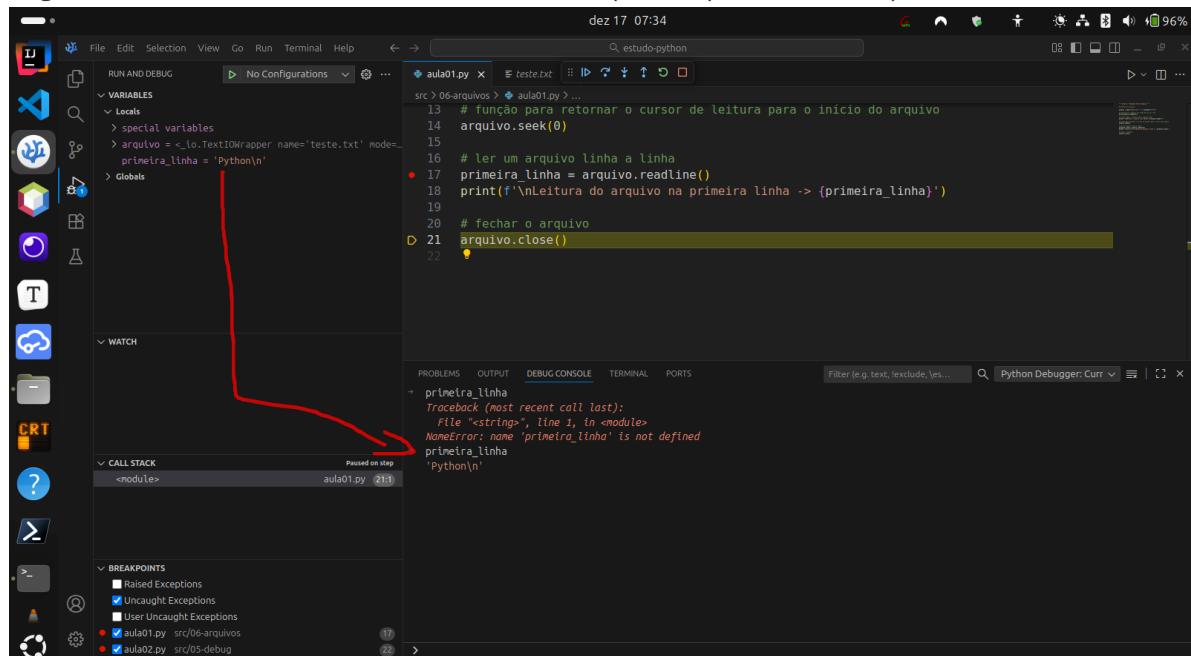
# fechar o arquivo
arquivo.close()

```

Note pelas figuras a seguir que sem o comando seek() o retorno da variável primeira_linha é uma string vazia:



Agora com o comando seek() temos o retorno esperado para a variável primeira_linha:



Também podemos ler todas as linhas de um arquivo e armazenar em uma lista usando a função `readlines()`:

```
# lendo todas as linhas do arquivo e guardando em uma lista

lista_arquivo = arquivo.readlines()

# imprime a lista
print(f'\n\nImprime a lista \n{lista_arquivo}'')
```

```
aula01.py x teste.txt
src > 06-arquivos > aula01.py > ...
21 arquivo.seek(0)
22
23 # lendo várias linhas ao mesmo tempo e guardando em uma lista
24
25 lista_arquivo = arquivo.readlines()
26
27 # imprime a lista
28 print(f'\n\nImprime a lista \n{lista_arquivo}')
29
30 # fechar o arquivo
31 arquivo.close()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Filter (e.g. text, exclude, |es... | Python Debugger: Curr x
listar_arquivo
> ['Python\n', 'Javascript\n', 'Java\n', 'C#\n', 'Node\n', 'R\n', 'Golang\n', 'SQL']
```

Note que a saída do arquivo `lista_arquivo` tem a linha completa inclusive com o `\n` separador de linha.

Podemos também no nosso arquivo ' teste.txt ' adicionar mais texto, no caso linguagens de programação, para isso devemos fechar o arquivo e então modificar o arquivo de 'r' read para 'a' que é append, se colocarmos 'w' podemos sobreescriver todo o nosso arquivo, então devemos ter esse cuidado em arquivos que já existem é necessário utilizar o comando append.

```
""" Aula 01 - Manipulação de Arquivos """

# abertura do arquivo

arquivo = open("teste.txt", "r", encoding='utf-8')

# verifica se o arquivo tem condições de ser lido
print(arquivo.readable())

# função read(), retorna todo o arquivo lido
print(f'\nRetorna o arquivo lido abaixo:\n{arquivo.read()}')

# função para retornar o cursor de leitura para o início do arquivo
arquivo.seek(0)

# ler um arquivo linha a linha
primeira_linha = arquivo.readline()
print(f'\nLeitura do arquivo na primeira linha -> {primeira_linha}')

# voltar o arquivo para o início do arquivo
arquivo.seek(0)
```

```
# lendo várias linhas ao mesmo tempo e guardando em uma lista

lista_arquivo = arquivo.readlines()

# imprime a lista
print(f'\n\nImprime a lista \n{lista_arquivo}')

# fechar o arquivo
arquivo.close()

# abrir o arquivo para modo de adicionar no final append

arquivo = open("teste.txt", "a", encoding='utf-8')

# adicionar texto
ADICIONAR_TEXTO = '\nC\nRust\n' # pois o anterior estava na mesma linha

# verificar se o arquivo aceita escrita
if (arquivo.writable()):
    # adicionar texto no arquivo
    arquivo.write(ADICIONAR_TEXTO)
    arquivo.close()

# abrindo novamente o arquivo agora no modo leitura
arquivo = open('teste.txt', 'r', encoding='utf-8')

# variável lista recebe lista do conteúdo de arquivo
lista = arquivo.readlines()

# imprimir lista
print(f'\nNova lista gerada\n{lista}')

# fechar o arquivo
arquivo.close()
```

```
aula01.py
40 # verificar se o arquivo aceita escrita
41 if arquivo.writable():
42     # adicionar texto no arquivo
43     arquivo.write(ADICIONAR_TEXTO)
44     arquivo.close()
45
46 # abrindo novamente o arquivo agora no modo leitura
47 arquivo = open('teste.txt', 'r', encoding='utf-8')
48
49 # variável lista recebe lista do conteúdo do arquivo
50 lista = arquivo.readlines()
51
52 # imprimir lista
```

(base) adalberto@adalberto-X751L:~/github/lipai/atividades/estudo-python\$ cd /home/adalberto/github/lipai/atividades/estudo-python ; /usr/bin/env /bin/python3 /home/adalberto/.vscode-oss/extensions/ms-python.debugpy-2025.14.1-linux-x64/bundled/libs/debugpy/adapter/../../debugpy/launcher 57611 -- /home/adalberto/github/lipai/atividades/estudo-python/src/06-arquivos/aula01.py
Retorna o arquivo lido abaixo:
Python
Javascrip
Java
C#
Node
R
Golang
SQL

Leitura do arquivo na primeira linha -> Python

Imprime a lista
['Python\n', 'Javascript\n', 'Java\n', 'C#\n', 'Node\n', 'R\n', 'Golang\n', 'SQL\n', 'C\n', 'Rust\n']

(base) adalberto@adalberto-X751L:~/github/lipai/atividades/estudo-python\$

Podemos também utilizar a função writable() para verificar se o arquivo pode ser escrito e só depois escrever o mesmo, vomo foi feito no código acima.

No python não é possível depois de usar open sem antes fechar o mesmo, alterar a característica do mesmo, se para leitura , escrita ou adição, veja o que acontece se fizermos um programa para dar dois open's e só depois um close():

```
""" teste para ver se o open funciona ao ter um arquivo já usando ele """
# abre arquivo para escrita
arquivo = open('teste_arquivo.txt', 'w', encoding='utf-8')

# escreve no arquivo
arquivo.write('Escrevendo algo no arquivo\n')

# testa se o arquivo pode ser lido
if arquivo.readable():
    print('O arquivo pode ser lido')
else:
    print('Arquivo não pode ser lido')

# tentativa de abrir o arquivo para leitura
arquivo_leitura = open('teste_arquivo.txt', 'r', encoding='utf-8')

# testa se o arquivo foi lido e aberto
if arquivo_leitura.readable():
    print(arquivo_leitura.readlines())
```

```
dez 17 10:39
File Edit Selection View Go Run Terminal Help < > estudo-python
OPEN EDITORS
ESTUDO-PYTHON
src
01-intro
aula01.py
aula02.py
aula03.py
aula04.py
aula05.py
aula06.py
aula07.py
02-controle-de-fluxo
03-tipos-de-dados
04-funcoes
05-debug
06-arquivos
aula01.py
teste.py
teste_arquivo.txt
notas.txt
teste_arquivo.txt
teste.txt
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(base) adalberto@adalberto-X751LJ:~/github/lipai/atividades/estudo-python$ cd /home/adalberto/github/lipai/atividades/estudo-python ; /usr/bin/env /bin/python3 /home/adalberto/.vscode-oss/extensions/ms-python.debugpy-2025.14.1-linux-x64/bundled/Libs/debugpy/adapter/../debugpy/launcher 46871 -- /home/adalberto/github/lipai/atividades/estudo-python/src/06-arquivos/teste.py
Arquivo não pode ser lido
[]
(base) adalberto@adalberto-X751LJ:~/github/lipai/atividades/estudo-python$
```

Notamos que apesar de termos dado um novo nome a variável arquivo_leitura ela abre o arquivo do mesmo local mas não consegue proceder com a leitura e retorna vazia, isso acontece no python pois o arquivo ainda está aberto para escrita.

Podemos também importar a biblioteca os do python para manipularmos arquivos usando o sistema operacional.

Com isso podemos remover diretórios usando o os.rmdir(nome_diretório), desde que o mesmo esteja vazio e também removermos arquivos desde que os mesmos tenham sido fechados antes:

```
# removendo arquivo
# os.remove('teste_arquivo.txt')

# verificando se o arquivo pode ser removido ou não
if os.path.exists('teste_arquivo.txt'):
    os.remove('teste_arquivo.txt')
else:
    print('Arquivo não existe')

# removendo uma pasta
os.rmdir('src/06-arquivos/nova_pasta')
```

```

aula01.py x E teste_arquivo.txt  E aula01.py  E teste.txt
src > 06-arquivos > aula01.py ...
55 if os.path.exists('teste_arquivo.txt'):
56     os.remove('teste_arquivo.txt')
57 else:
58     print('Arquivo não existe')
59
60 # removendo uma pasta
61 os.rmdir('src/06-arquivos/nova_pasta')
62
63
64
65
66
67
68
69
70
71
72

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(base) adalberto@adalberto-X751L:~/github/lipai/atividades/estudo-python$ cd /home/adalberto/github/lipai/atividades/estudo-python ; /usr/bin/env /bin/python3 /home/adalberto/.vscode-oss/extensions/ms-python.debugpy-2025.14.1-linux-x64/bundled/Libs/debugpy/adapter/../debugpy/launcher 44245 -- /home/adalberto/github/lipai/atividades/estudo-python/src/06-arquivos/aula01.py
bash adal... Python De...
C Rust
C Rust
Leitura do arquivo na primeira linha -> Python

Imprime a lista
['Python\n', 'Javascript\n', 'Java\n', 'C#\n', 'Node\n', 'R\n', 'Golang\n', 'SQL\n', 'C\n', 'Rust\n', '\n', 'C\n', 'Rust\n', '\n', 'C\n', 'Rust\n']

Nova lista gerada
['Python\n', 'Javascript\n', 'Java\n', 'C#\n', 'Node\n', 'R\n', 'Golang\n', 'SQL\n', 'C\n', 'Rust\n', '\n', 'C\n', 'Rust\n', '\n', 'C\n', 'Rust\n']
Arquivo não existe
(base) adalberto@adalberto-X751L:~/github/lipai/atividades/estudo-python$ 

```

Aula 02 - Manipulando arquivos

Imagen ilustrando que o arquivo foi criado e tem a instância de uma classe '`<io.TextIOWrapper name='arquivo.txt' mode='w' encoding='utf-8'>`'

```

aula02.py x
src > 06-arquivos > aula02.py ...
1 """ Aula 02 - Manipulação arquivos python """
2
3 # abrindo um arquivo
4 arg = open('arquivo_.txt', 'w', encoding='utf-8')
5
6 print(arg)
7

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(base) adalberto@adalberto-X751L:~/github/lipai/atividades/estudo-python$ cd /home/adalberto/github/lipai/atividades/estudo-python ; /usr/bin/env /bin/python3 /home/adalberto/.vscode-oss/extensions/ms-python.debugpy-2025.14.1-linux-x64/bundled/Libs/debugpy/adapter/../debugpy/launcher 40109 -- /home/adalberto/github/lipai/atividades/estudo-python/src/06-arquivos/aula02.py
<_io.TextIOWrapper name='arquivo_.txt' mode='w' encoding='utf-8'>
(base) adalberto@adalberto-X751L:~/github/lipai/atividades/estudo-python$ 

```

```

""" Aula 02 - Manipulação arquivos python """

# abrindo um arquivo
arg = open('arquivo_.txt', 'w', encoding='utf-8')
# imprime o conteúdo
print(arg)
# escreve no arquivo uma string
arg.write('escreve uma única string\nque pode ser barra n \n\n')
# iterável de strings para gravação usando o writelines()
lista = ['nom', 'nome', 'nomer', 'homer']
arg.writelines(lista)

# forma mais correta de abrir arquivos em python
with open('arquivo_texto.txt', 'w', encoding='utf-8') as arg:

```

```

arq.write('teste\ntestando\ntestador\n')

# ao sair da subrotina with open, temos que o arquivo aberto para leitura já está
fechado

# modo append
with open('arquivo_texto.txt', 'a', encoding='utf-8') as arq:
    arq.write('adicionando linha no final do arquivo')

# modo leitura
with open('arquivo_texto.txt', 'r', encoding='utf-8') as arq:
    x = arq.read()
    print(x)
    print(type(x))
    # ir para o início do arquivo
    arq.seek(0)
    # lê o arq e transforma em uma lista
    x = arq.readlines()
    print(x)
    print(type(x))

# leitura modo binário (rb)
with open('arquivo_texto.txt', 'rb') as arq:
    x = arq.read()
    print(x)
    print(type(x))
    # para decodificar de binário para string podemos usar o método decode()
    print(x.decode())
    print(type(x.decode()))

# gerador iterável arq
with open('arquivo_texto.txt', 'r', encoding='utf-8') as arq:
    for i in arq:
        print(i)

```

Também podemos usar o método next() para iterar sobre o arq desde que em modo leitura

```

with open('arquivo_texto.txt', 'r', encoding='utf-8') as arq:
    print(next(arq)) # mostra a primeira linha
    print(next(arq)) # mostra a segunda linha e assim por diante

```

Resolução Exercícios

Ex 01

```

""" Ex 01 - Carregar dados de alunos """

def carregar_dados_alunos(nome_arquivo):
    """ retorna uma tupla de dicionários com os dados do arquivo """
    lista_dicionario = list()
    with open(nome_arquivo, 'r', encoding='utf-8') as arq:
        linha = arq.read()
        linha = linha.split('\n')

```

```

        for i in range(len(linha)):
            prontuario, nome, email = linha[i].split(',')
            lista_dicionario.append(
                {
                    'prontuario': prontuario,
                    'nome': nome,
                    'email': email
                }
            )

        return tuple(lista_dicionario)

ARQUIVO = 'arquivo_ex01.txt'

lista_dados = carregar_dados_alunos(ARQUIVO)

print(type(lista_dados))

print(lista_dados)

```

Essa não é a melhor forma de resolver esse exercício, pois se o arquivo for grande vai travar devido ao impacto na memória.

Ex 02

```

""" Ex 02 - Carregar dados de projeto """

def carregar_dados_projetos(nome_arquivo):
    """Retorna uma tupla de dicionários com dados de projetos."""
    with open(nome_arquivo, 'r', encoding='utf-8') as arq:
        projeto = []
        for linha in arq:
            codigo, titulo, responsavel = linha.strip().replace('\n', '').split(',')
            projeto.append(
                {
                    'codigo': codigo,
                    'titulo': titulo,
                    'responsavel': responsavel
                }
            )

        return tuple(projeto)

ARQUIVO = 'arquivo_ex02.txt'

lista_dados = carregar_dados_projetos(ARQUIVO)

print(type(lista_dados))

print(lista_dados)

```

EX 03

```
""" Ex 03 - Convertendo uma linha em dicionário genérico """

def linha_para_dict(frase, lista_chaves):
    """Recebe uma linha e uma lista de chaves e retorna um dicionário."""
    # cria uma lista com separador na vírgula para cada elemento da lista
    lista_linha = frase.strip().replace(" ", "").split(',')
    if len(lista_linha) != len(lista_chaves):
        return f"{frase} e {lista_chaves} devem ter a mesma quantidade de itens"
    else:
        dicionario = {}
        for i in range(len(lista_chaves)):
            key = lista_chaves[i]
            valor = lista_linha[i]
            dicionario[key] = valor

    return dicionario

# casos testes

# caso teste 1
LINHA = "SP000001,Maria da Silva,maria@email.com"
chaves = ['prontuario', 'nome', 'email']
print(linha_para_dict(LINHA, chaves))

# caso teste 2
LINHA2 = "banana,3"
chaves2 = ['item', 'quantidade']
print(linha_para_dict(LINHA2, chaves2))

# caso teste 3
TEXTO = 'pentium IV, 8gb, 120gb, False'
itens = ['cpu', 'memoria', 'hd', 'cd-room']
print(linha_para_dict(TEXTO, itens))

# caso teste 4
DATA = 'FM 104.5, canal 8, faixa cidadão'
dados = ['radio', 'tv']
print(linha_para_dict(DATA, dados))
```

Reflexão

- Qual a vantagem de transformar cada linha do arquivo em dicionários em vez de trabalhar apenas com strings?

R. Melhora o entendimento dos valores a serem encontrados, pois ao utilizar uma chave e um valor não preciso ficar manipulando strings e contando até onde vai cada substring para "achar" o valor.

- Em que situações pode ser útil retornar uma tupla de registros (como nos exercícios ex01 e ex02) em vez de apenas uma lista de linhas?

R. Quando queremos garantir que os dados na tupla não serão alterados.

- O que você achou mais desafiador: ler o arquivo ou transformar as linhas em estruturas de dados (dicionários)?

R. Transformar as linhas em estruturas de dados é mais desafiador visto que cada dado pode ter suas particularidades que devem ser tratadas de uma maneira personalíssima, já a abertura de um arquivo segue uma norma já estabelecida.