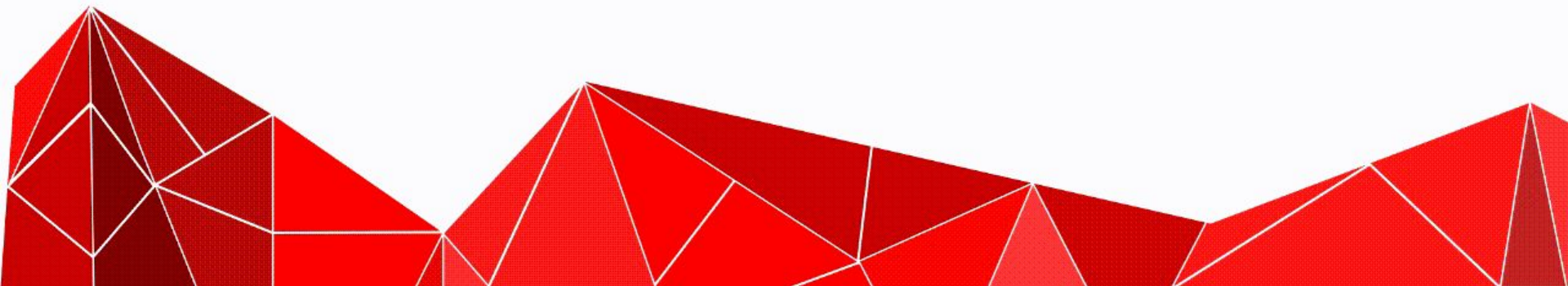




CODING ARQUIVOS

Press -> to continue

Powered by PET computação UFC





INTRODUÇÃO

Introdução

- As variáveis são apagadas da memória quando o programa fecha
- Precisamos de um jeito para guardar informações “para sempre”
- Uma solução simples é guardar a informação em arquivos
- Podemos ler e escrever informações “para sempre”
- Vamos ver arquivo em formato txt
- Mas existem muitos outros

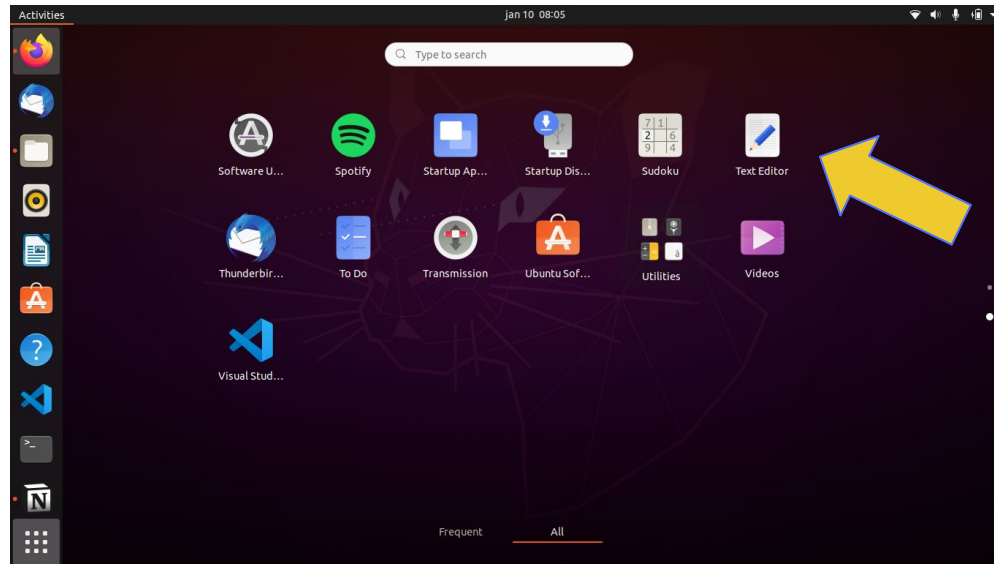




LENDO ARQUIVOS

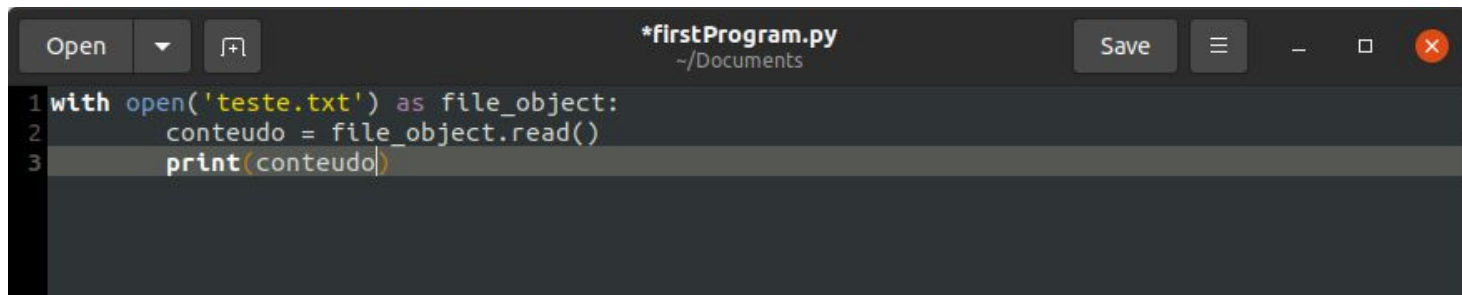
Lendo arquivos

- Primeiro vamos criar um arquivo de texto manualmente
- Abra o editor de texto, crie um arquivo
- Escreva algo
- Salve na pasta do seu programa, com a extensão “.txt”



Lendo arquivos

- Agora vamos escrever nosso programa
- Vamos ler todo o arquivo de uma vez só
- Para isso, escrevemos o seguinte :



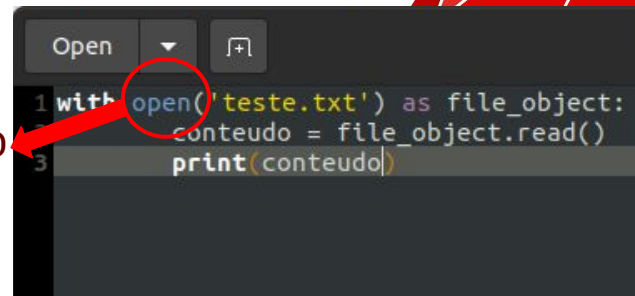
The screenshot shows a code editor window with a dark theme. The title bar indicates the file is named `*firstProgram.py` and is located in the `~/Documents` directory. The window includes standard buttons for 'Open', 'Save', and window controls. The code is as follows:

```
1 with open('teste.txt') as file_object:
2     conteudo = file_object.read()
3     print(conteudo)
```



Lendo arquivos

- Para fazer qualquer coisa precisamos abrir o arquivo
- Usamos a função open()
- Retorna um objeto representando o arquivo
- Python armazena o objeto na variável file_object
- É com ela que vamos trabalhar
- Existe também a função close()
 - É importante para evitar corromper os dados ao fechar o programa

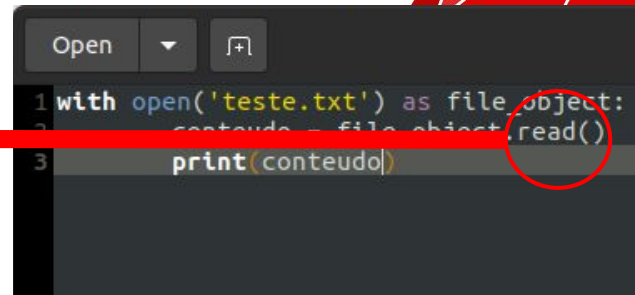


```
Open ▼ [icon]  
1 with open('teste.txt') as file_object:  
2     conteudo = file_object.read()  
3     print(conteudo)
```



Lendo arquivos

- No caso queremos ler todo o arquivo de uma vez
- Para isso, usamos o método read()
- Ele lê o arquivo inteiro
- E guarda o conteúdo em forma de string
- Na variável que nós definimos
- Por fim, usamos print() para imprimir essa string



```
Open ▼ [+]  
1 with open('teste.txt') as file_object:  
2     conteudo = file_object.read()  
3     print(conteudo)
```



Lendo arquivos

- O resultado é:

```
rafael@rafael-Lenovo-ideapad-330-15IKB: ~/Documents
rafael@rafael-Lenovo-ideapad-330-15IKB:~/Documents$ python3
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exec(open('firstProgram.py').read())
lista de compras:
Tomate
goiaba
banana
granola
mel
alface
cereal
pêra
carne
espinafre
melão
>>> 
```



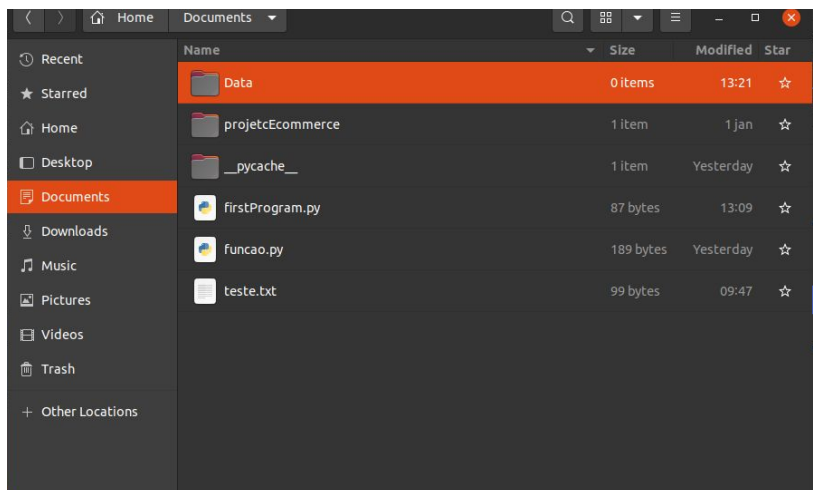
O caminho dos arquivos

- Um nome simples na função `open()`
 - Indica que o arquivo está na mesma pasta do programa
- Mas podemos colocar o arquivo em outra pasta
- Assim, passamos o caminho do arquivo para a função `open()`
- Usamos o caminho relativo
 - Diz que ao python que a pasta estamos procurando por algo no diretório onde o programa está rodando
- Vamos ver um exemplo



O caminho dos arquivos

- Primeiro vamos criar uma nova pasta para colocar nosso arquivo
- Aperte o botão direito do mouse e selecione “nova pasta”
- Depois mova o arquivo de texto para a pasta criada

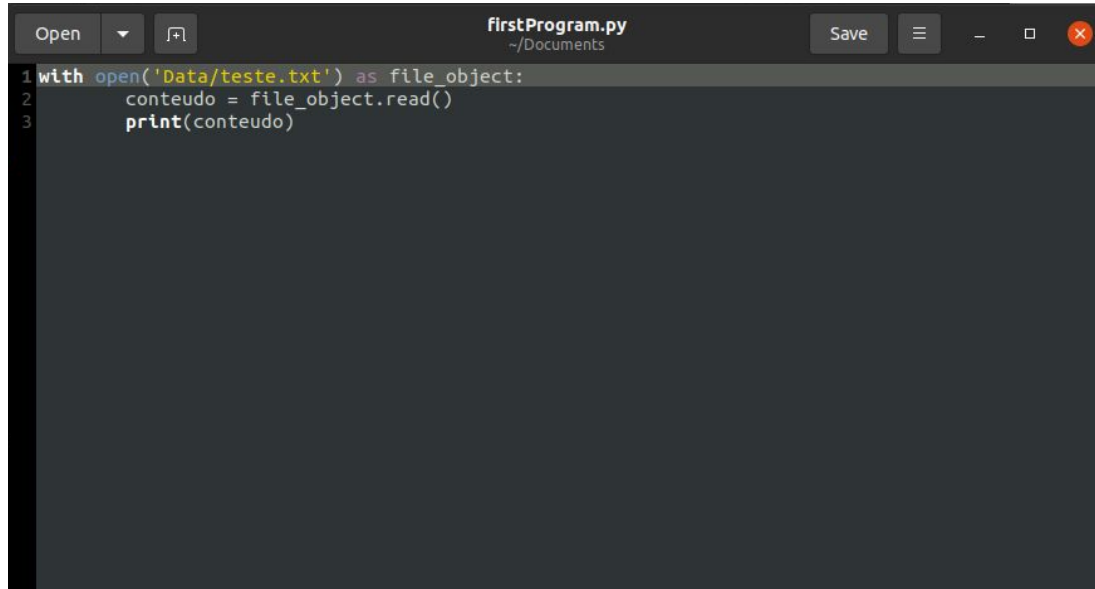


Basta arrastar o arquivo com o mouse



O caminho dos arquivos

- Agora precisamos mudar o caminho na função `open()`
- Usando o caminho relativo, ficará assim:



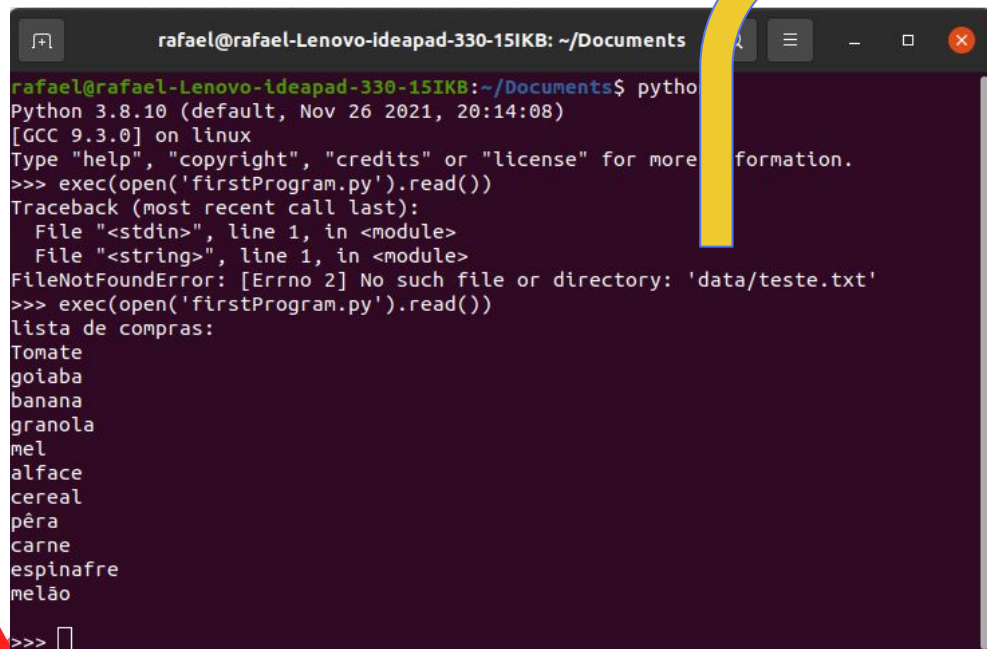
```
1 with open('Data/teste.txt') as file_object:
2     conteudo = file_object.read()
3     print(conteudo)
```

The screenshot shows a code editor window with the title 'firstProgram.py' and a subtitle '~ / Documents'. The code is written in Python and uses a relative path 'Data/teste.txt' within a `with open()` statement. The code is as follows:



O caminho dos arquivos

- E o resultado é o mesmo do anterior



```
rafael@rafael-Lenovo-ideapad-330-15IKB: ~/Documents
rafael@rafael-Lenovo-ideapad-330-15IKB:~/Documents$ python3 firstProgram.py
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more
>>> exec(open('firstProgram.py').read())
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    File "<string>", line 1, in <module>
FileNotFoundError: [Errno 2] No such file or directory: 'data/teste.txt'
>>> exec(open('firstProgram.py').read())
lista de compras:
Tomate
goiaba
banana
granola
mel
alface
cereal
pêra
carne
espinafre
melão
>>>
```

Veja que aqui cometi um erro, pois coloquei o nome da pasta com letra minúscula, mas eu criei com maiúscula.

Tome cuidado para não cometer o mesmo erro.



Lendo linha por linha

- Se você quiser/precisar fazer uma análise mais minuciosa do arquivo
- Ou modificar uma certa informação
- Pode ser interessante ler o arquivo linha por linha
- Para isso, escrevemos o seguinte programa:

```
*firstProgram.py
~/Documents

Open  ▾  [icon]  Save  [icon]  [icon]  [icon]

1 with open('Data/teste.txt') as file_object:
2     for linha in file_object:
3         print(linha)
```



Lendo linha por linha

- O resultado é o seguinte
- Ele adiciona uma linha branca no fim de cada linha
- Lembre-se, podemos tirar com o método `rstrip()`

```
rafael@rafael-Lenovo-ideapad-330-151KB: ~/Documents
Type "help", "copyright", "credits" or "license" for more information.
>>> exec(open('firstProgram.py').read())
lista de compras:

Tomate

goiaba

banana

granola

mel

alface

cereal

pêra

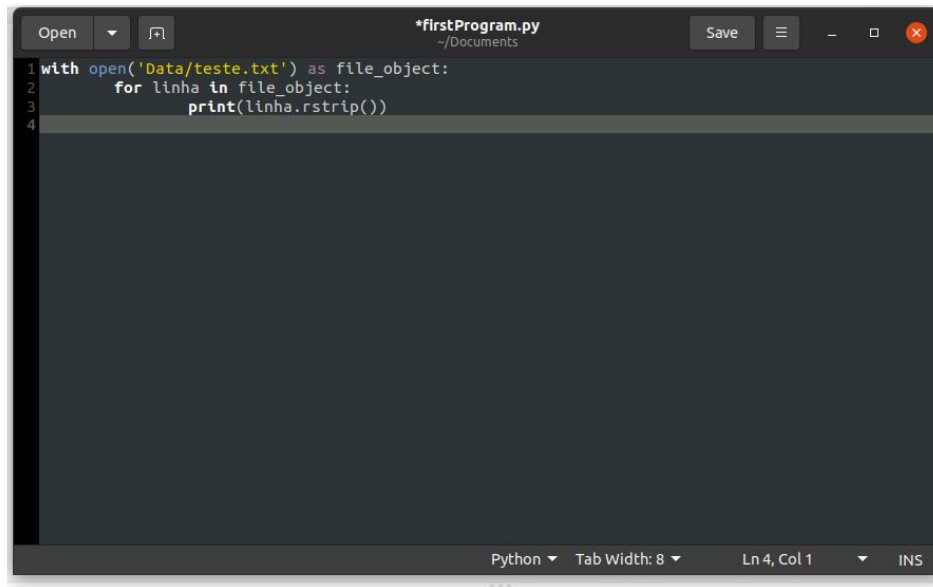
carne

espinafre
```



Lendo linha por linha

- Usando o método
- O código ficará assim :



```
*firstProgram.py
~/Documents
Open [icon] Save [icon] [icon] [icon] [icon]
1 with open('Data/teste.txt') as file_object:
2     for linha in file_object:
3         print(linha.rstrip())
4
Python Tab Width: 8 Ln 4, Col 1 INS
```



Lendo linha por linha

- E o resultado será esse :

```
rafael@rafael-Lenovo-ideapad-330-15IKB: ~/Documents
rafael@rafael-Lenovo-ideapad-330-15IKB:~/Documents$ python3
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exec(open('firstProgram.py').read())
lista de compras:
Tomate
goiaba
banana
granola
mel
alface
cereal
pêra
carne
espinafre
melão
>>> 
```



Criando um array das linhas

- Podemos criar uma lista com as informações de cada linha
- Para isso podemos usar o método `readlines()`
- Ele lê cada linha do arquivo e armazena em um posição da lista
- O código ficará assim :

```
*firstProgram.py
~/Documents

Open  Save  [Menu]  [Window]  [Close]

1 with open('Data/teste.txt') as file_object:
2     linhas = file_object.readlines()
3
4 for linha in linhas:
5     print(linha.rstrip())
6
7
```



Criando um array das linhas

- Podemos criar uma lista com as informações de cada linha
- Para isso podemos usar o método `readlines()`
- Ele lê cada linha do arquivo e armazena em um posição da lista
- O código ficará assim :

```
*firstProgram.py
~/Documents

Open  Save  [Menu]  [Window]  [Close]

1 with open('Data/teste.txt') as file_object:
2     linhas = file_object.readlines()
3
4 for linha in linhas:
5     print(linha.rstrip())
6
7
```



Criando um array das linhas

- Na hora de printar, o resultado é o mesmo
- Mas lembre-se, as aplicações podem ser diferentes

```
rafael@rafael-Lenovo-ideapad-330-15IKB: ~/Documents
rafael@rafael-Lenovo-ideapad-330-15IKB:~/Documents$ python3
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exec(open('firstProgram.py').read())
lista de compras:
Tomate
goiaba
banana
granola
mel
alface
cereal
pêra
carne
espinafre
melão
>>> 
```



Criando um array das linhas

- Escrevendo `print(linhas)` no arquivo, podemos ver a lista inteira
- Observamos também as quebras de linha ao fim de cada linha

```
rafael@rafael-Lenovo-ideapad-330-15IKB: ~/Documents
rafael@rafael-Lenovo-ideapad-330-15IKB:~/Documents$ python3
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exec(open('firstProgram.py').read())
['lista de compras:\n', 'Tomate\n', 'goiaba \n', 'banana\n', 'granola\n', 'mel\n',
'alface\n', 'cereal \n', 'pêra \n', 'carne \n', 'espinafre\n', 'melão \n']
>>>
```





Escrevendo arquivos



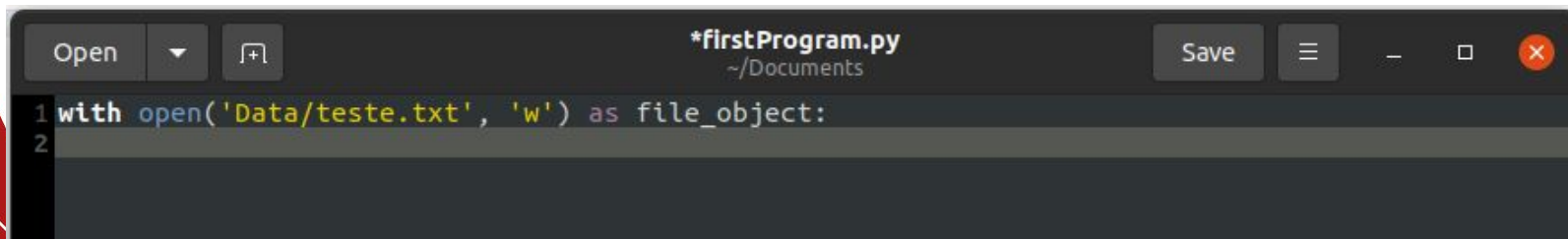
Escrevendo no arquivo

- Para escrevermos em um arquivo precisamos abrir de uma maneira um pouco diferente
- Usamos a mesma função `open()`, mas com um parâmetro a mais
 - O qual é modo que desejamos abrir o arquivo
 - Existem várias modos de abertura
 - Os principais são :
 - "a" - Escreve ao final do arquivo; se este não existir, é criado
 - "r" - Abre o arquivo para a leitura, se não existir, lançar um erro de `IOError`
 - "r+" - Abra um arquivo para leitura e escrita. Se não existe, lança um erro `IOError`
 - "w" - Abre um arquivo para escrita. Se existe, ele apaga tudo e escreve por cima. Se não existir o arquivo, ele cria
 - "w+" - Abre para leitura e escrita. Se existe, apaga todo conteúdo e escreve por cima. Se não existir o arquivo, ele cria



Escrevendo no arquivo

- Para especificar o modo é muito simples
- Escrevemos o nome ou caminho do arquivo
- E após a vírgula o modo como queremos abrir o arquivo
- Exemplo, vamos usar o modo 'w':



```
*firstProgram.py
~/Documents

Open [icon] Save [icon] [icon] [icon] [icon]

1 with open('Data/teste.txt', 'w') as file_object:
2
```



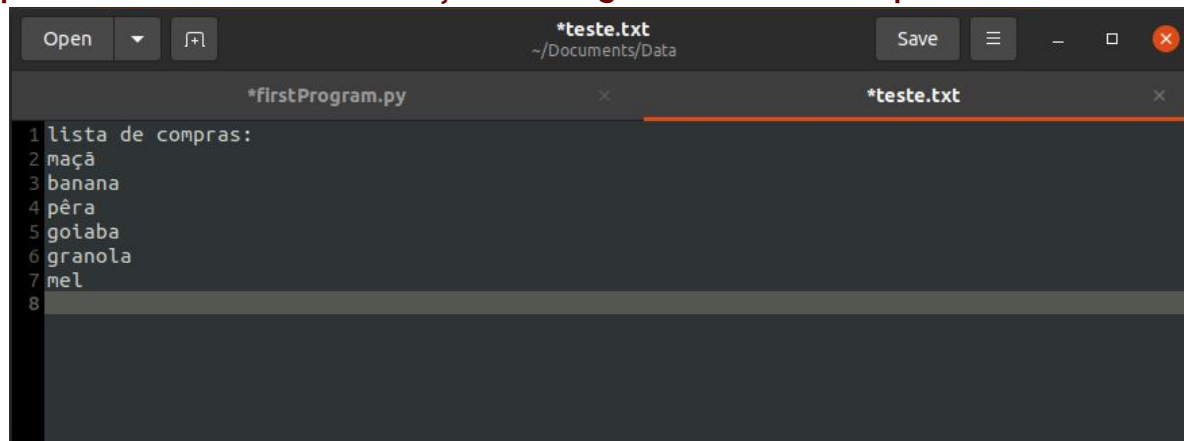
Escrevendo no arquivo

- Usamos o método write() para escrever informações no arquivo
- Nele passamos a informação que queremos usar

• Exemplo

```
1 with open('Data/teste.txt', 'w') as file_object:
2     file_object.write("abacate")
3
4
```

- Veja que ainda temos informações antigas no nosso arquivo :



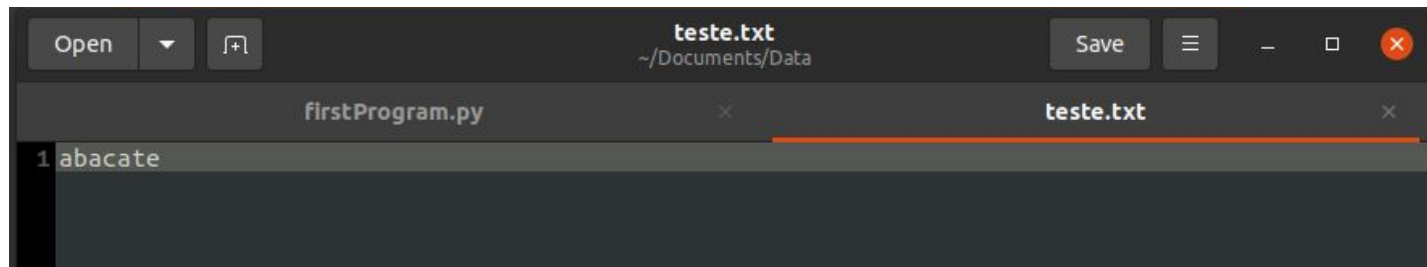
The screenshot shows a code editor window with two tabs: '*firstProgram.py' and '*teste.txt'. The '*teste.txt' tab is active and displays a shopping list. The list is as follows:

```
1 lista de compras:
2 maçã
3 banana
4 pêra
5 goiaba
6 granola
7 mel
8
```



Escrevendo no arquivo

- Ao executar o programa todo o conteúdo será apagado
- O resultado é como esperado:



The screenshot shows a code editor window with a dark theme. The title bar indicates the file is 'teste.txt' located at '~/Documents/Data'. The editor has two tabs: 'firstProgram.py' and 'teste.txt'. The 'teste.txt' tab is active and shows a single line of text: '1 abacate'.

- Por isso usamos o modo 'w' apenas se:
 - O arquivo estiver vazio
 - Ou se realmente temos a intenção de apagar e sobrescrever as informações



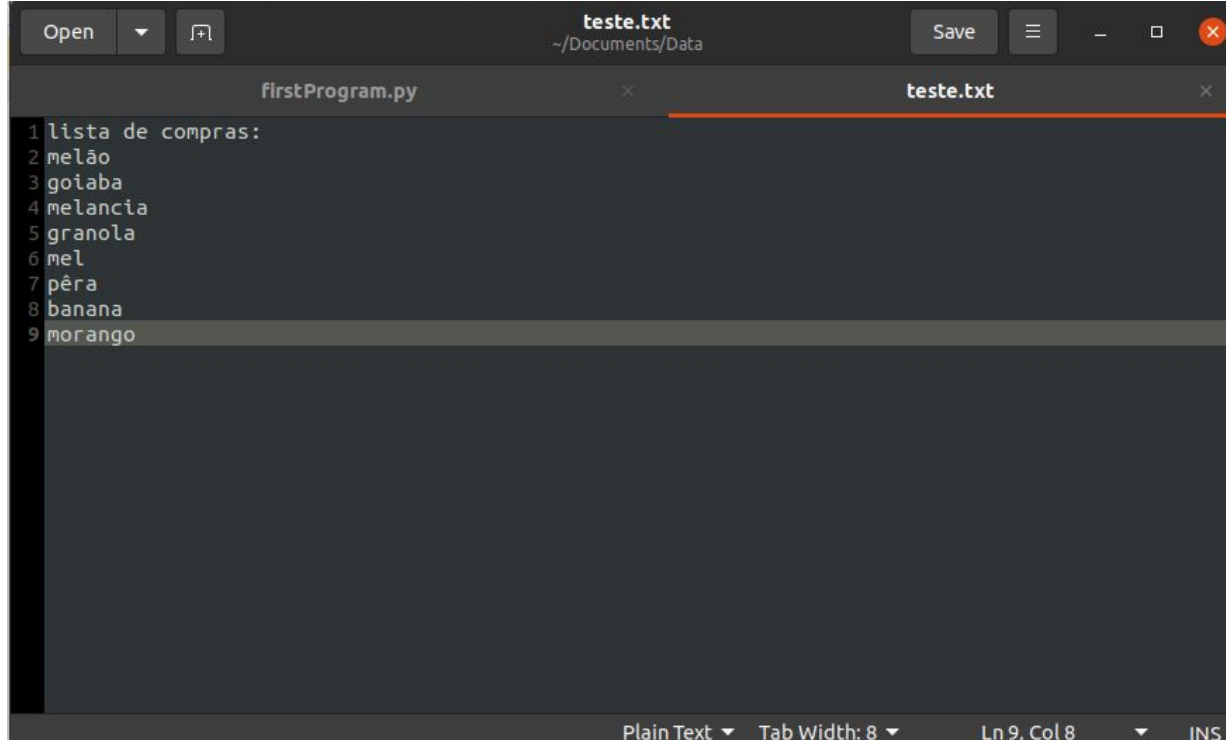
Escrevendo no arquivo

- Mas e se quiséssemos apenas adicionar informação ?
- Para isso usamos o modo 'a'
 - Escreve no fim do arquivo
 - Mantendo a informação original
- Vamos editar escrever novamente nossa lista manualmente
- E mudar o modo de abertura no programa



Escrevendo no arquivo

- Escrevendo novamente a lista :



The screenshot shows a text editor window with a dark theme. The title bar indicates the file is 'teste.txt' located at '~/Documents/Data'. The window has tabs for 'firstProgram.py' and 'teste.txt', with 'teste.txt' being the active tab. The editor contains a shopping list with 9 items, each on a new line and numbered. The text is as follows:

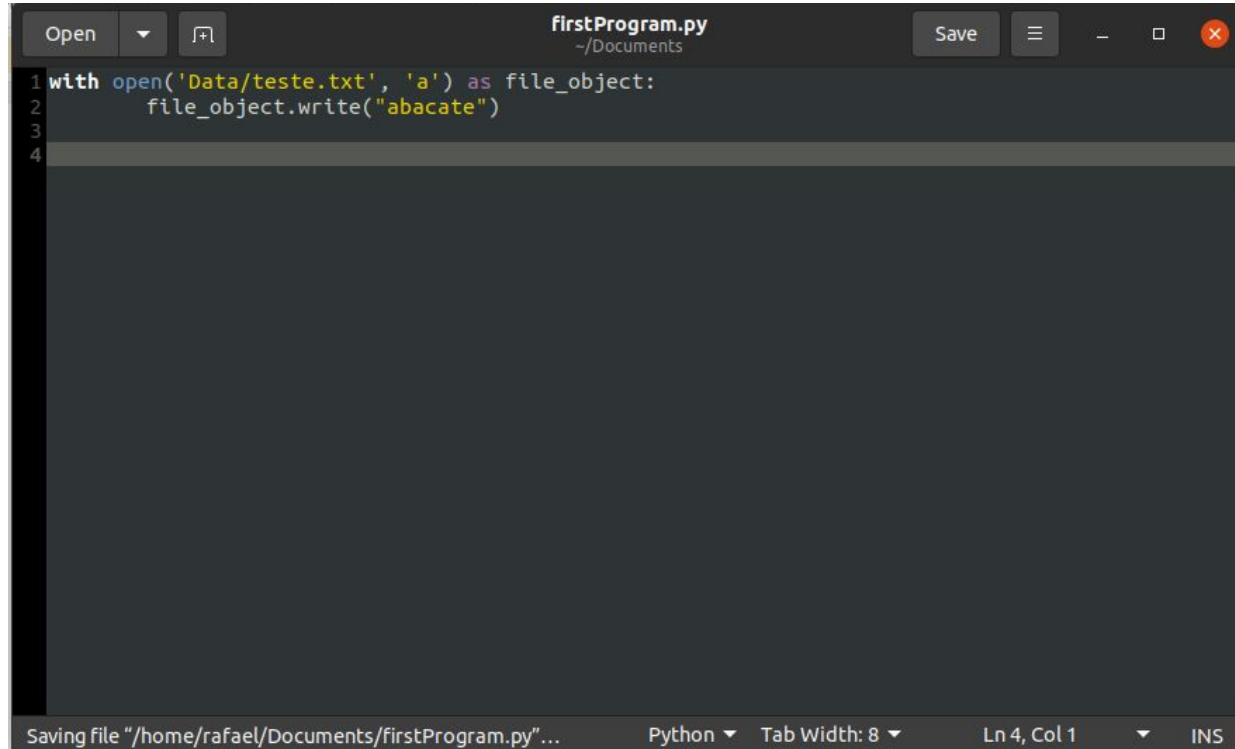
```
1 lista de compras:  
2 melão  
3 goiaba  
4 melancia  
5 granola  
6 mel  
7 pêra  
8 banana  
9 morango
```

The status bar at the bottom shows 'Plain Text', 'Tab Width: 8', 'Ln 9, Col 8', and 'INS'.



Escrevendo no arquivo

- Agora mudando o modo de abertura :



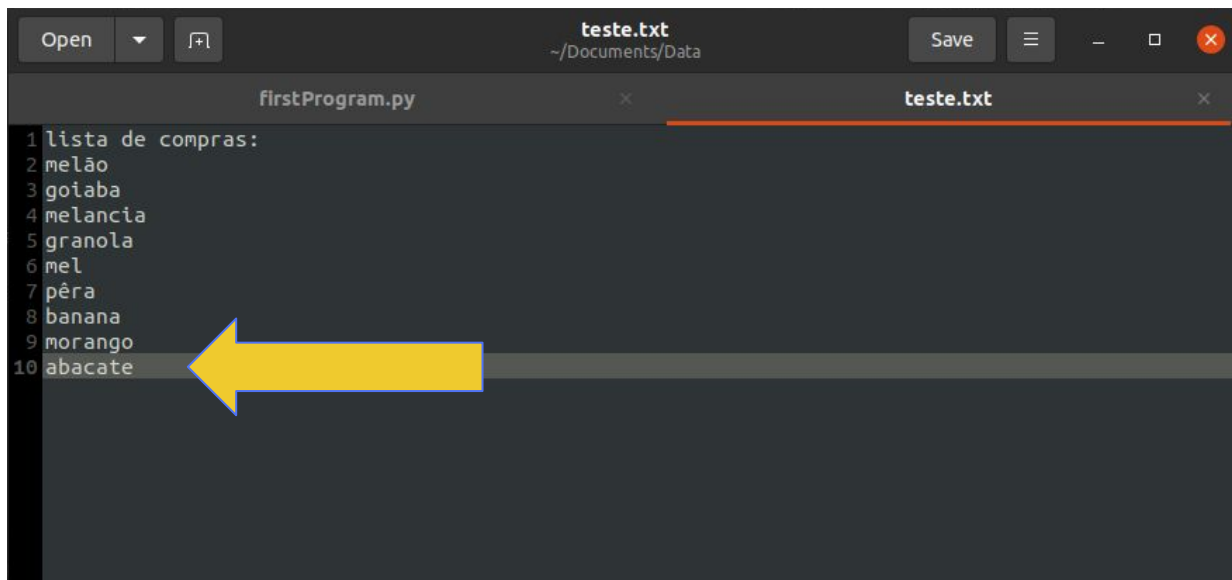
```
1 with open('Data/teste.txt', 'a') as file_object:
2     file_object.write("abacate")
3
4
```

The screenshot shows a code editor window titled 'firstProgram.py' with a path of '~/Documents'. The editor contains a Python script that opens a file named 'Data/teste.txt' in append mode ('a') and writes the string 'abacate' to it. The status bar at the bottom indicates the file is being saved, the language is Python, the tab width is 8, and the cursor is at line 4, column 1.



Escrevendo no arquivo

- Ao rodar o arquivo, podemos ver que adicionamos a informação no fim
- Mas mantemos as informações que já estavam no arquivo



The screenshot shows a code editor window with two tabs: 'firstProgram.py' and 'teste.txt'. The 'teste.txt' tab is active and shows a list of items. A yellow arrow points to the last line of the list, '10 abacate'.

```
1 lista de compras:
2 melão
3 goiaba
4 melancia
5 granola
6 mel
7 pêra
8 banana
9 morango
10 abacate
```



- Escreva um programa para ler as n primeiras linhas de um arquivo.
- Escreva um programa que vai ler um arquivo, mostrar o conteúdo na tela e e depois copiar n linhas para outro arquivo.
- Escreva um programa para combinar cada linha do primeiro arquivo com a linha correspondente no segundo arquivo e no fim montar um string nova com as linhas criadas.

A decorative red geometric pattern consisting of various triangles and polygons, some outlined in white, located on the left side of the slide.

**VAMOS PRATICAR E APRENDER
MAIS COISAS !!!**

