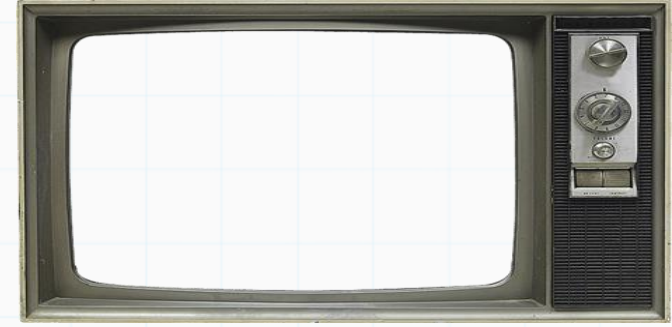


Programação De Computadores

Professor : Yuri Frota

www.ic.uff.br/~yuri/prog.html

yuri@ic.uff.br



Strings

Um pouco mais sobre strings (variáveis de texto):
uma string se comporta muitas vezes como uma lista

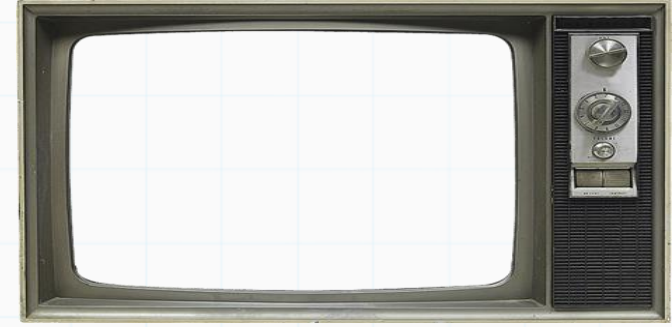
```
1 nome = 'Zelda'
2 print(nome)
3 print(nome[2])
4
5 for i in range(len(nome)):
6     print(nome[i])
7
```

Shell ×

```
Zelda
l
Z
e
l
d
a
```

- podemos acessar por índices
- podemos usar "len"

0	1	2	3	4
Z	e	l	d	a



Strings

Um pouco mais sobre strings (variáveis de texto):

Mas são imutáveis (assim como tuplas)

```
1 nome = 'Zelda'
2 nome[0] = 'L'
3 print(nome)
```

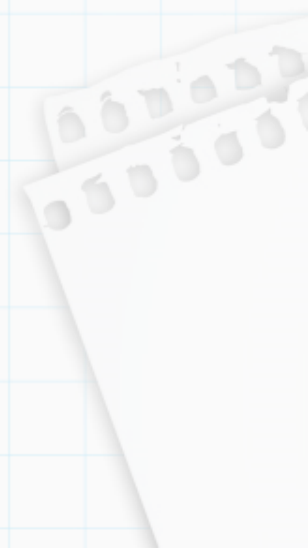
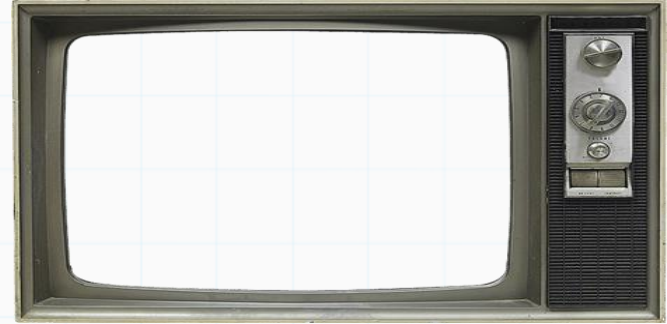
Shell ×

Traceback (most recent call last):

File "C:\Users\Yuri\Desktop\teste.py", line 2, in <module>

nome[0] = 'L'

TypeError: 'str' object does not support item assignment



Strings

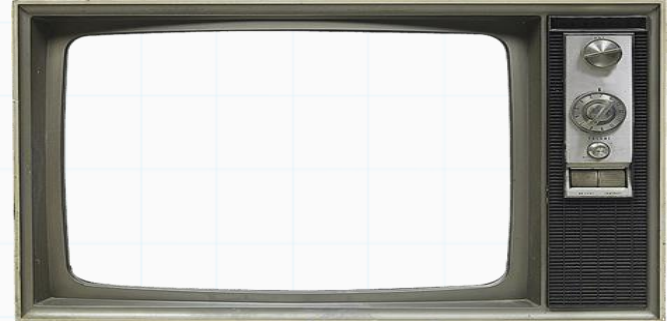
Um pouco mais sobre strings (variáveis de texto):

podemos percorrer a string pelo índice ou por elemento (assim como as listas)

```
1 nome = 'Zelda'
2
3 for i in range(len(nome)):
4     print(nome[i])
5
6 for letra in nome:
7     print(letra)
```

Shell ×

Z
e
l
d
a
Z
e
l
d
a



Strings

Um pouco mais sobre strings (variáveis de texto):

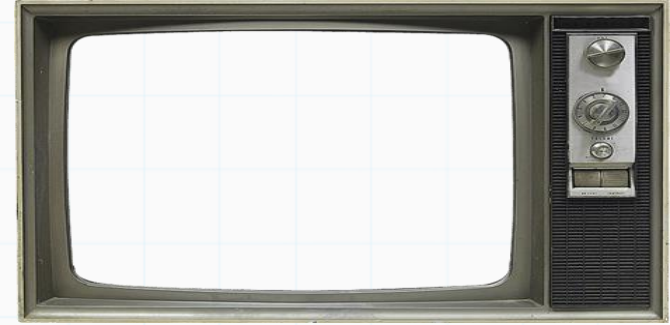
podemos concatenar com “+” (já usamos isso)

```
1 nome = 'Zelda'
2 nome2 = ' não é meu nome'
3 nome3 = nome + nome2
4 print(nome3)
```

Shell ×

```
>>> %Run teste.py
```

```
Zelda não é meu nome
```



Strings

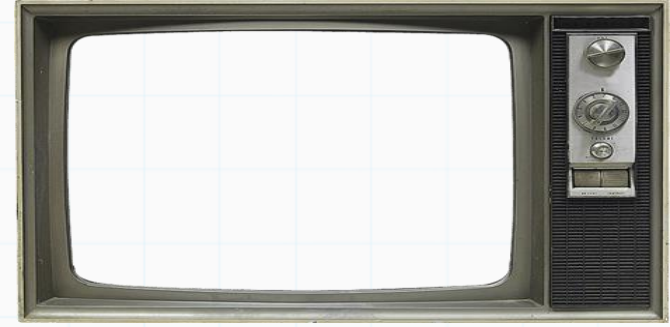
Um pouco mais sobre strings (variáveis de texto):

caracteres especiais: quebra de linha “\n”

```
1 nome = 'meu nome\n não é\n Zelda'
2 print(nome)
3
4 nome = 'meu nome\nn não é\nZelda'
5 print(nome)
```

Shell ×

```
meu nome
 não é
Zelda
meu nome
 não é
Zelda
```



Strings

- comando strip: retira caracteres do começo e do fim da string (limpa a string)

formato: `<string>.strip(<c>)`

```
1 n = 'xx Ryu Ken Chun-li xx'
2 print(n)
3
4 n2 = n.strip('x')
5 print(n2)
```

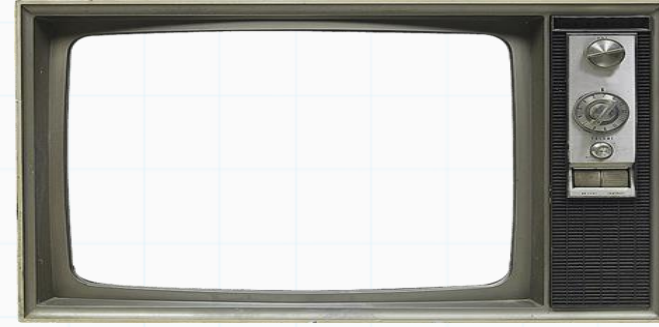
Shell x

```
xx Ryu Ken Chun-li xx
Ryu Ken Chun-li
```

```
1 n = 'xx Ryu xx Ken xx Chun-li xx'
2 print(n)
3
4 n2 = n.strip('x')
5 print(n2)
```

Shell x

```
xx Ryu xx Ken xx Chun-li xx
Ryu xx Ken xx Chun-li
```



Strings

- comando strip: retira caracteres do começo e do fim da string (limpa a string)

formato: `<string>.strip(<c>)`

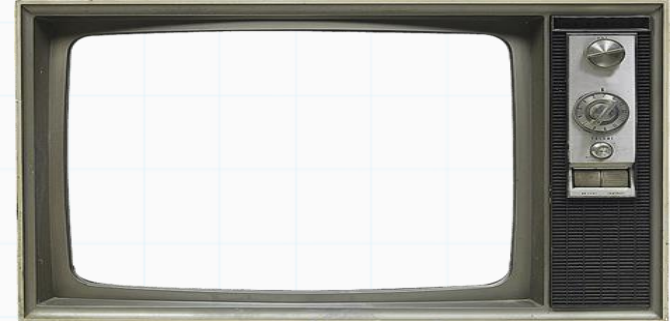
```
1 n = '\n Ryu Ken Chun-li \n'
2 print(n)
3
4 n2 = n.strip()
5 print(n2)
```

Shell ×

Ryu Ken Chun-li

Ryu Ken Chun-li

Se não tiver parâmetros, retira
espaços e quebras de linhas
das extremidades



Strings

- comando split : Quebra a string em uma lista de strings separadas por um separador. Se o separador não for especificado, é usado espaços em branco, tabs e quebras de linha como separador.

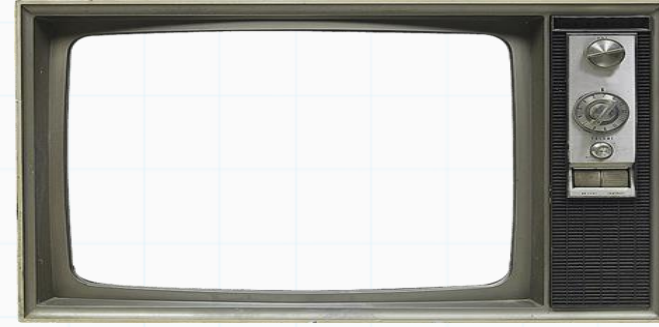
formato: <string>.split(<sep>)

```
1 n = 'Liu Kang,Johnny Cage,Sonya Blade'
2 print(n)
3
4 n2 = n.split(',')
5 print(n2)
```

Shell x

```
/// teste.py

Liu Kang,Johnny Cage,Sonya Blade
['Liu Kang', 'Johnny Cage', 'Sonya Blade']
```



Strings

- comando split : Quebra a string em uma lista de strings separadas por um separador. **Se o separador não for especificado, é usado espaços em branco, tabs e quebras de linha como separador.**

formato: <string>.split(<sep>)

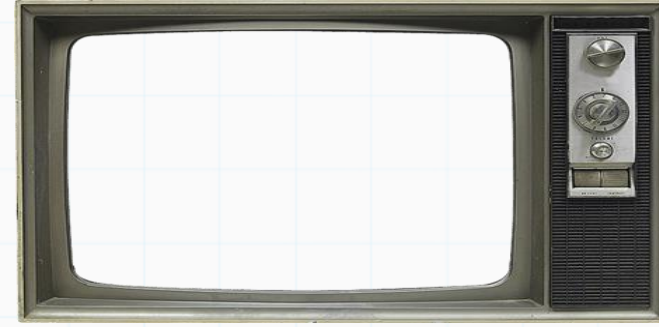
```
1 n = 'Liu Kang derrota\n Johnny Cage'
2 print(n)
3
4 n2 = n.split()
5 print(n2)
```

Shell x

```
>>> %Run teste.py
```

```
Liu Kang derrota
Johnny Cage
```

```
['Liu', 'Kang', 'derrota', 'Johnny', 'Cage']
```



Strings

- comando `split` : Quebra a string em uma lista de strings separadas por um separador. **Se o separador não for especificado, é usado espaços em branco, tabs e quebras de linha como separador.**

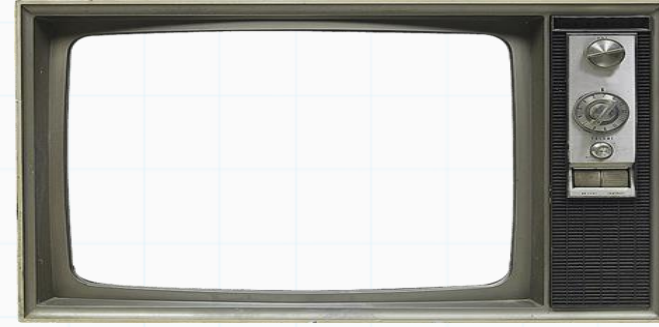
formato: `<string>.split(<sep>)`

```
1 n = ' Liu Kang derrota\n Johnny Cage '
2 print(n)
3
4 for palavra in n.split():
5     print(palavra)
```

Shell ×

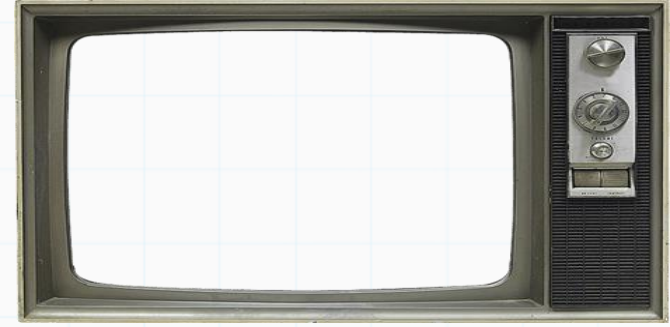
```
Liu Kang derrota
Johnny Cage
Liu
Kang
derrota
Johnny
Cage
```

Percorrendo as palavras de uma frase



Arquivos

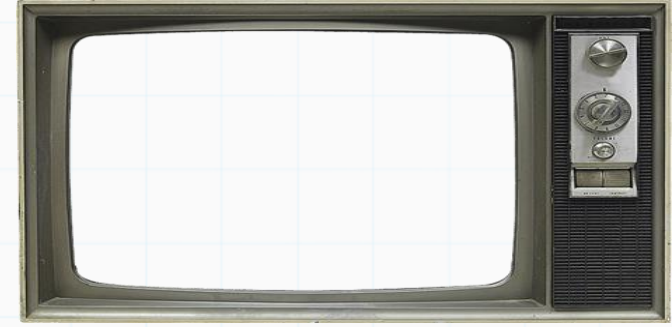
- Até o momento, foram tratados exemplos de programas que apenas obtiveram os dados de **IO** (entrada/saída) de usuários via teclado (função `input()`) ou mostrados na tela/display (função `print()`)
- Uma outra forma de ingressar ou gerar dados como entradas/saídas de programa é através de **arquivos texto** (ou binários, ex. arquivos de áudio e/ou imagens)



Arquivos

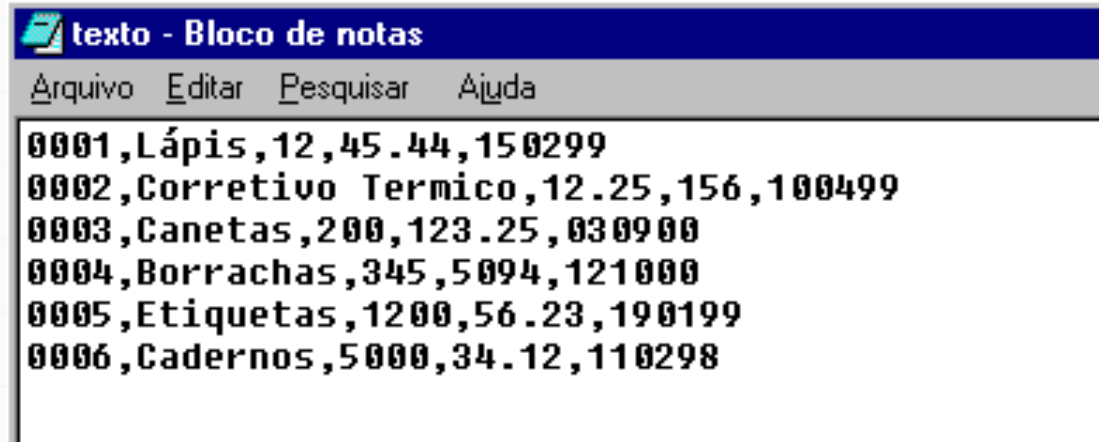
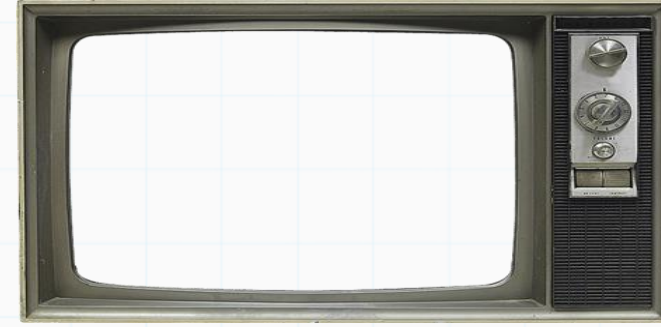
- Vantagens:

- Volume de dados : O conjunto de dados/informação pode ser muito maior.
- O Desempenho/Eficiência: Os dados podem ser inseridos e processados pelos programas muito mais rapidamente.



Arquivos

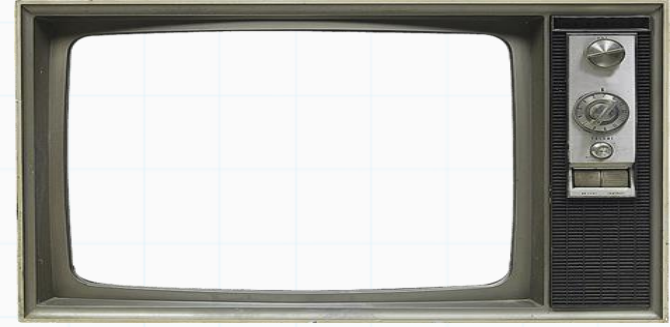
- Operações Básicas: Para poder usar arquivos temos que saber
 - Abrir arquivo
 - Fechar arquivo
 - Ler arquivo
 - Escrever arquivo



Arquivos

- Abrindo arquivo: em Python, temos que associar uma variável ao arquivo com o comando open

formato: `<var> = open(<nome>, <modo>)`



Arquivos

- Abrindo arquivo: em Python, temos que associar uma variável ao arquivo com o comando open

formato: `<var> = open(<nome>, <modo>)`

variável
que
representa
arquivo

nome do arquivo
(pode incluir
caminho). Se não
tiver, irá procurar o
arquivo no
diretório do
programa

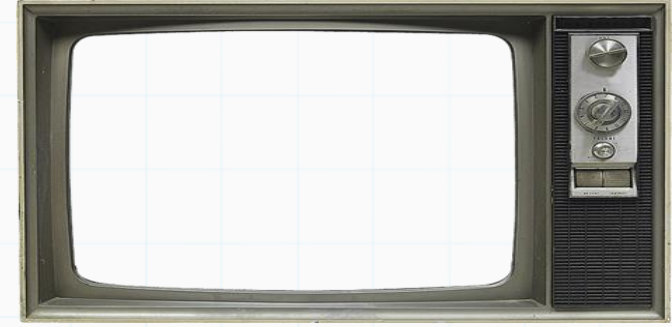
Modo de abertura:

“r” - leitura

“w” - escrita (se o arquivo existir, será apagado)

“a” - escrita a partir do final do arquivo

“+r” - leitura e escrita



Arquivos

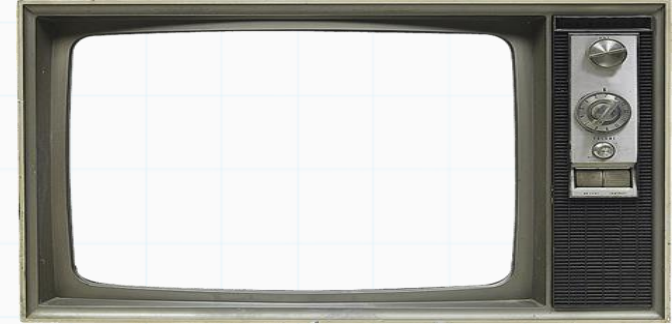
- Abrindo arquivo: em Python, temos que associar uma variável ao arquivo com o comando open

formato: `<var> = open(<nome>, <modo>)`

```
1  arq = open('teste.txt', 'r')
2
```

```
Shell x
Python 3.7.7 (bundled)
>>> %Run teste.py
Traceback (most recent call last):
  File "C:\Users\Yuri\Desktop\teste.py", line 1, in <module>
    arq = open('teste.txt', 'r')
FileNotFoundError: [Errno 2] No such file or directory: 'teste.txt'
>>>
```

se o arquivo "teste.txt" não
estiver no diretório do
programa



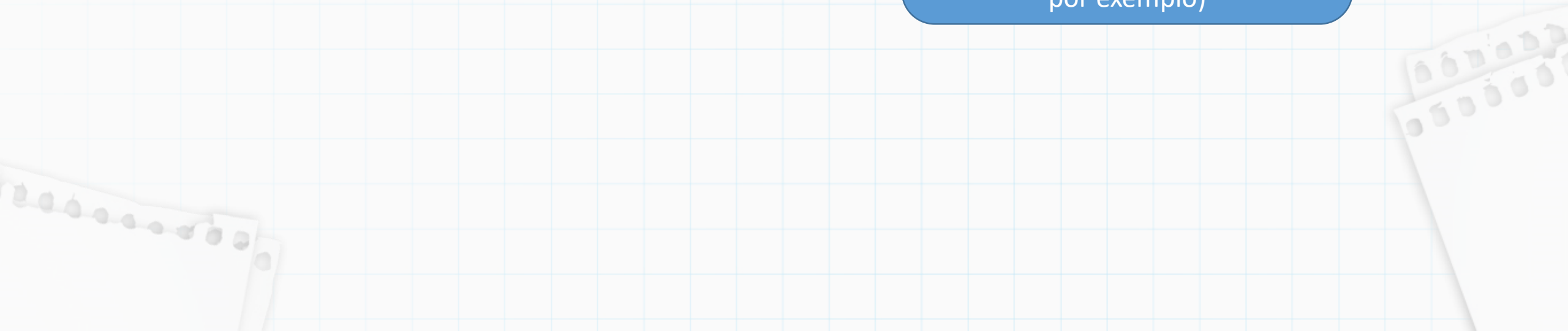
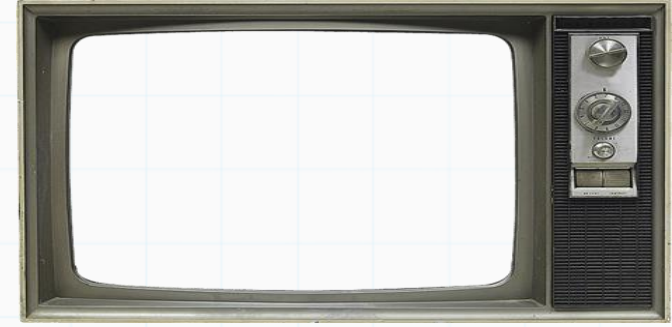
Arquivos

- Abrindo arquivo: em Python, temos que associar uma variável ao arquivo com o comando open

formato: `<var> = open(<nome>, <modo>)`

```
1  arq = open("C:\\Users\\Yuri\\Desktop\\teste.txt", 'r')  
2
```

Podemos passar o caminho todo também. Mas coloque “\\” invés de “\” para evitar erro de interpretação (conflito com “\n” por exemplo)



Arquivos

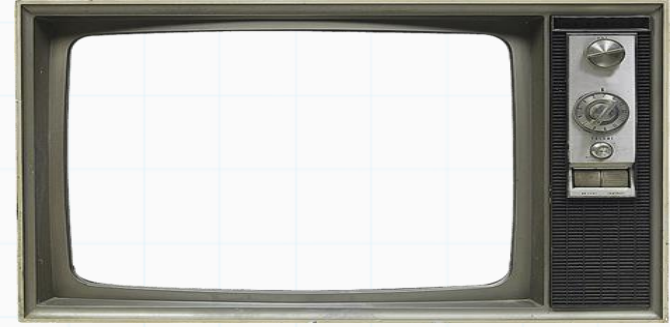
- Abrindo arquivo: em Python, temos que associar uma variável ao arquivo com o comando open

formato: `<var> = open(<nome>, <modo>)`

- Fechando arquivo: comando close (quando escrevemos dados em um arquivo, este comando garante que os dados serão efetivamente escritos no arquivo)

formato: `<var>.close()`

```
1  arq = open("C:\\Users\\Yuri\\Desktop\\teste.txt", 'r')
2  #
3  arq.close()
4
```



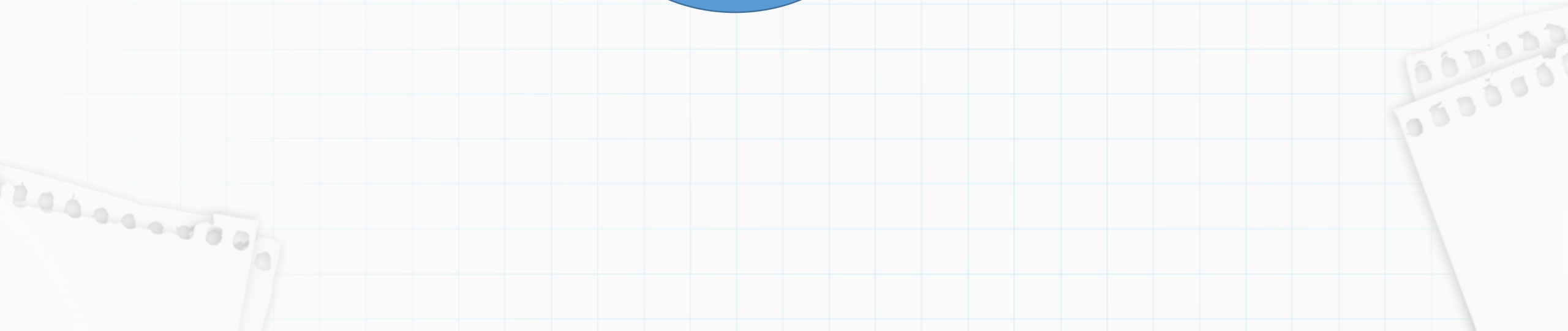
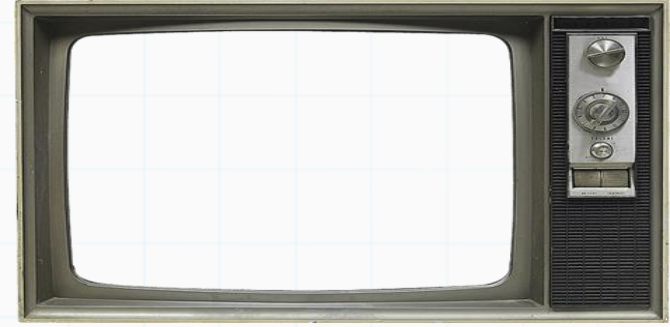
Arquivos

- Escrevendo arquivo: comando write (Necessário que o arquivo não tenha sido aberto em modo “r”)

formato: `<var>.write(<string>)`

variável
que
representa
arquivo

string a ser
escrita no fim
do arquivo.
String única

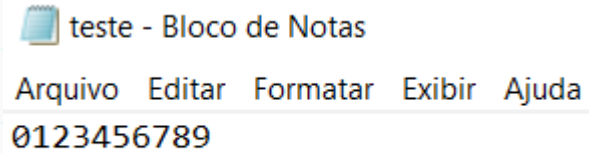


Arquivos

- Escrevendo arquivo: comando write (Necessário que o arquivo não tenha sido aberto em modo "r")

formato: <var>.write(<string>)

```
1  arq = open("teste.txt", 'w')
2
3  for i in range(10):
4      arq.write(str(i))
5
6  arq.close()
```

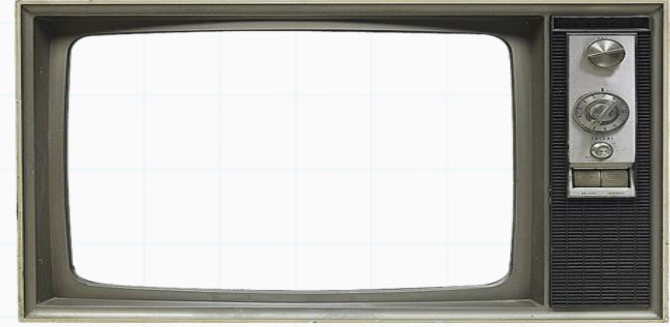


teste - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

0123456789

converte
para string
antes de
gravar

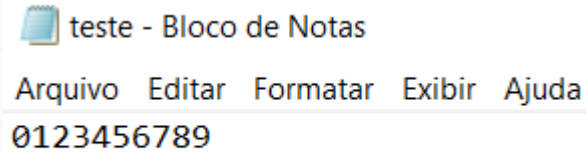


Arquivos

- Escrevendo arquivo: comando write (Necessário que o arquivo não tenha sido aberto em modo "r")

formato: <var>.write(<string>)

```
1  arq = open("teste.txt", 'w')
2
3  for i in range(10):
4      arq.write(str(i))
5
6  arq.close()
```

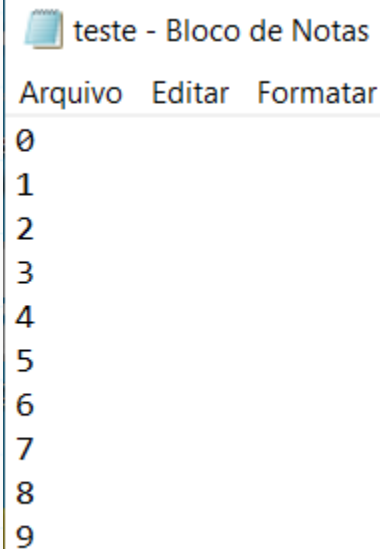


teste - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

0123456789

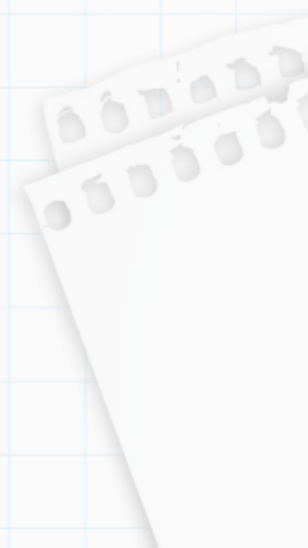
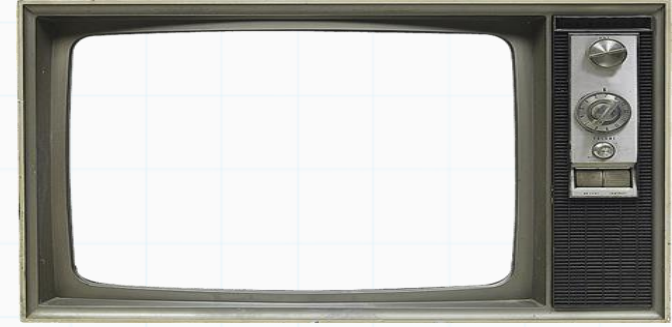
```
1  arq = open("teste.txt", 'w')
2
3  for i in range(10):
4      arq.write(str(i)+"\n")
5
6  arq.close()
```



teste - Bloco de Notas

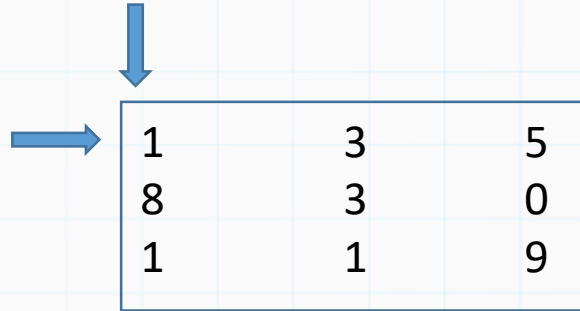
Arquivo Editar Formatar

0
1
2
3
4
5
6
7
8
9



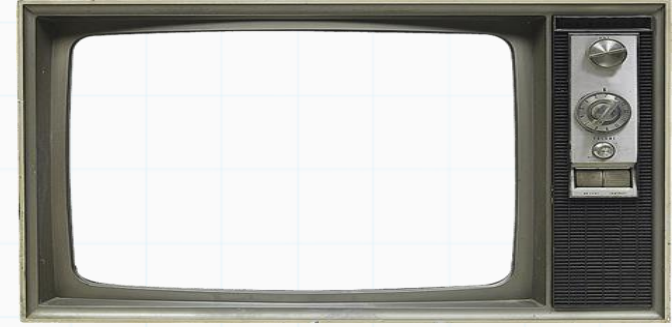
Arquivos

- Lendo arquivo:
 - Quando um arquivo é aberto, um **indicador de posição** no arquivo é criado, recebendo a posição do início do arquivo.



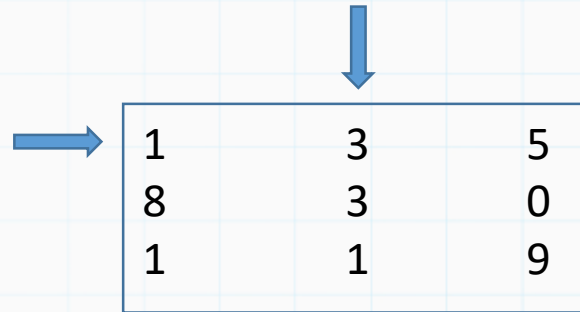
A diagram illustrating a file structure. A blue arrow points to the first row of a table, indicating the current position (cursor) in the file. The table contains three columns of numbers.

1	3	5
8	3	0
1	1	9



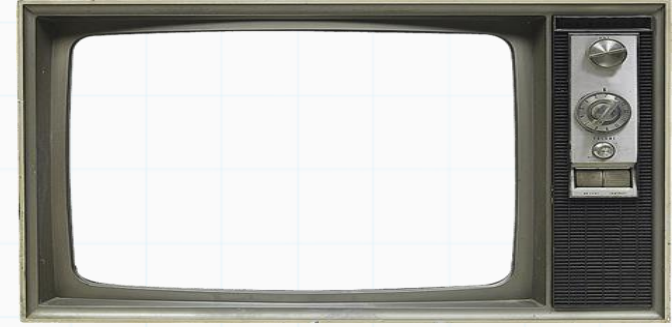
Arquivos

- Lendo arquivo:
 - Quando um arquivo é aberto, um **indicador de posição** no arquivo é criado, recebendo a posição do início do arquivo.
 - Para cada dado lido do arquivo, este indicador de posição é automaticamente incrementado para o próximo dado não lido



A diagram illustrating file reading. A blue arrow points from the left towards a table. A second blue arrow points from above down to the first row of the table. The table contains three columns of numbers.

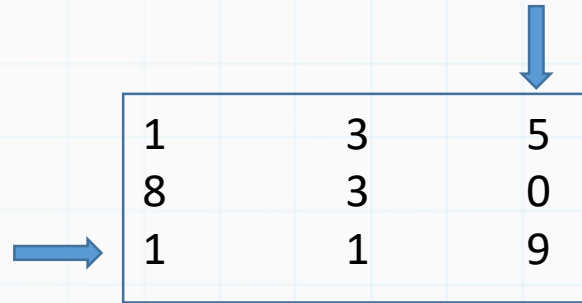
1	3	5
8	3	0
1	1	9



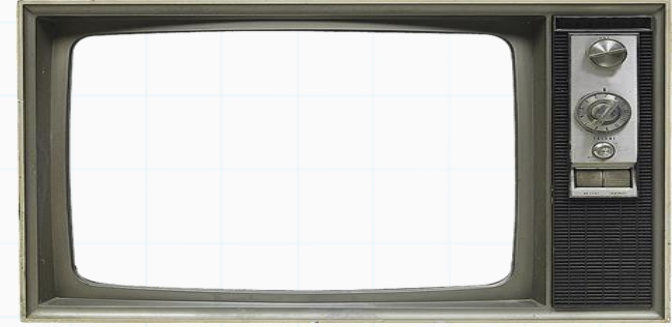
Arquivos

- Lendo arquivo:

- Quando um arquivo é aberto, um **indicador de posição** no arquivo é criado, recebendo a posição do início do arquivo.
- Para cada dado lido do arquivo, este indicador de posição é automaticamente incrementado para o próximo dado não lido
- Eventualmente o indicador de posição chega ao fim do arquivo



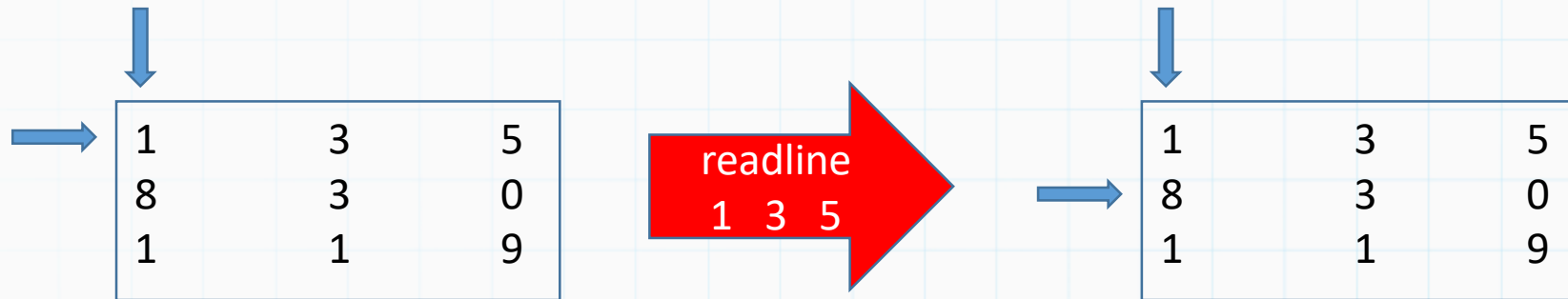
1	3	5
8	3	0
1	1	9



Arquivos

- Lendo arquivo: comando `readline` lê uma linha do arquivo e retorna como string (Necessário que o arquivo não tenha sido aberto em modo “w”)

formato: `<var>.readline()`



Arquivos

- Lendo arquivo: comando `readline` lê uma linha do arquivo e retorna como string (Necessário que o arquivo não tenha sido aberto em modo "w")

formato: `<var>.readline()`

```
1  arq = open("teste.txt", 'r')
2
3  soma = 0
4  num = arq.readline()
5  while num != '':
6      soma = soma + int(num)
7      num = arq.readline()
8  print(soma)
9
10 arq.close()
```

Shell ×

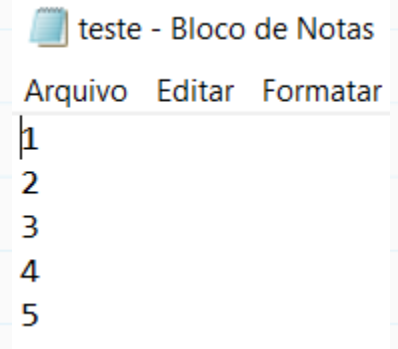
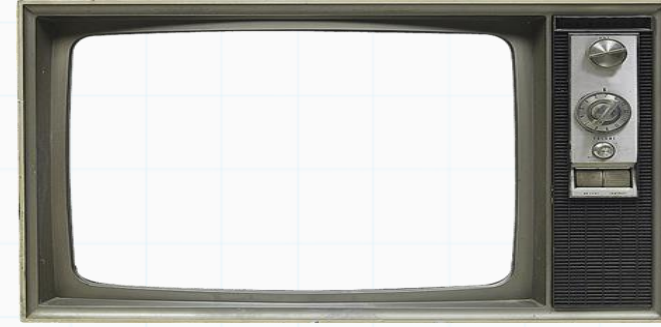
python teste.py

15

readline retorna vazio
no final do arquivo

tem que converter a
string

[código](#)



Arquivos

- Lendo arquivo: comando `readline` lê uma linha do arquivo e retorna como string (Necessário que o arquivo não tenha sido aberto em modo “w”)

formato: `<var>.readline()`

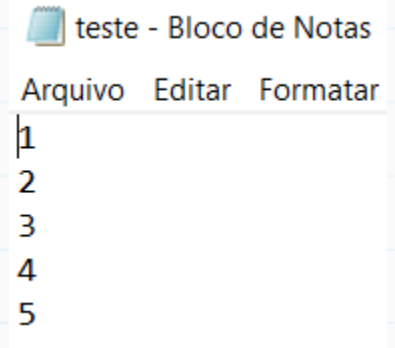
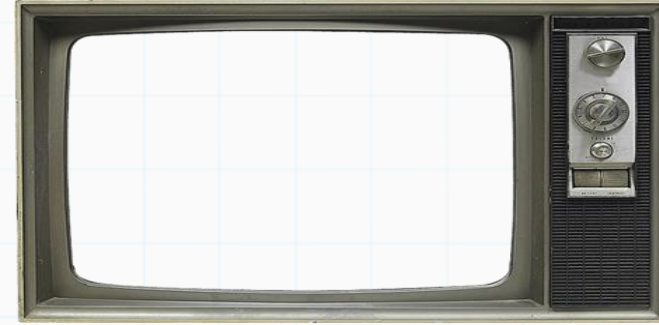
```
1 arq = open("teste.txt", 'r')
2
3 soma = 0
4 for num in arq:
5     soma = soma + int(num)
6 print(soma)
7
8 arq.close()
9
```

Shell ×

15

[código](#)

podemos usar o “for”
também para
caminhar pelas linhas
do arquivo (readline
implícito)



Arquivos

- Lendo arquivo: comando `readline` lê uma linha do arquivo e retorna como string (Necessário que o arquivo não tenha sido aberto em modo "w")

formato: `<var>.readline()`

Mas e se o arquivo tiver mais de um elemento por linha, como podemos ler ?

```
1  arq = open("teste.txt", 'r')
2
3  soma = 0
4  for num in arq:
5      soma = soma + int(num)
6  print(soma)
7
8  arq.close()
9
```

Shell ×

Python 3.7.7 (bundled)

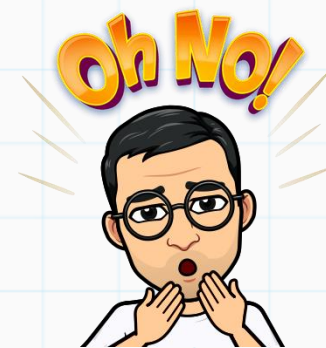
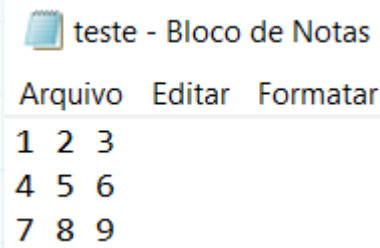
```
>>> %Run '24 - ex3.py'
```

Traceback (most recent call last):

File "C:\Users\Yuri\Desktop\24 - ex3.py", line 5, in <module>

`soma = soma + int(num)`

ValueError: invalid literal for int() with base 10: '1 2 3\n'



Arquivos


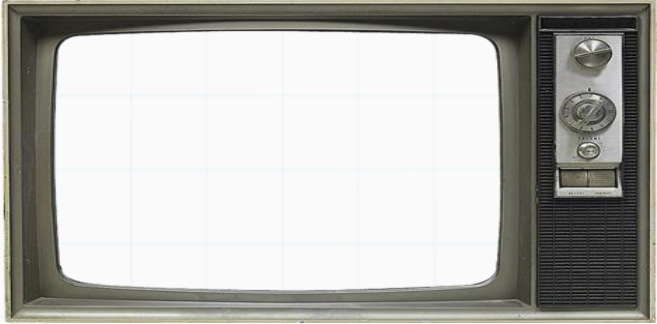
- Lendo arquivo: comando `readline` lê uma linha do arquivo e retorna como string (Necessário que o arquivo não tenha sido aberto em modo "w")

formato: `<var>.readline()`

Mas e se o arquivo tiver mais de um elemento por linha, como podemos ler ?

```
1  arq = open("teste.txt", 'r')
2
3  soma = 0
4  for linha in arq:
5      n1,n2,n3 = linha.split()
6      soma = soma + int(n1) + int(n2) + int(n3)
7  print(soma)
8
9  arq.close()
```

quebramos a linha em
3 partes (strings)



teste - Bloco de Notas			
Arquivo	Editar	Formatar	
1	2	3	
4	5	6	
7	8	9	

Shell x

%%Run 24 - ex1.py

45

[código](#)

Arquivos

- Lendo arquivo: comando `readline` lê uma linha do arquivo e retorna como string (Necessário que o arquivo não tenha sido aberto em modo "w")

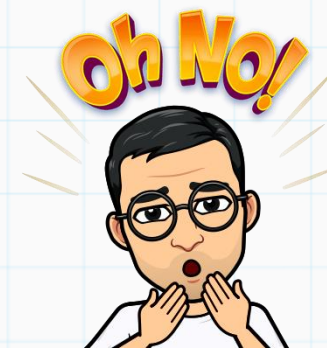
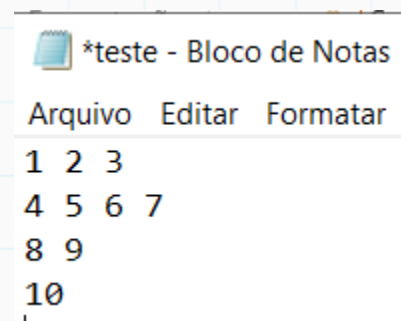
formato: `<var>.readline()`

Mas e se o arquivo tiver mais linhas de tamanhos diferentes ?

```
1  arq = open("teste.txt", 'r')
2
3  soma = 0
4  for linha in arq:
5      n1,n2,n3 = linha.split()
6      soma     = soma + int(n1) + int(n2) + int(n3)
7  print(soma)
8
9  arq.close()
```

Shell ×

```
Traceback (most recent call last):
  File "C:\Users\Yuri\Desktop\24 - ex1.py", line 5, in <module>
    n1,n2,n3 = linha.split()
ValueError: too many values to unpack (expected 3)
```



Arquivos

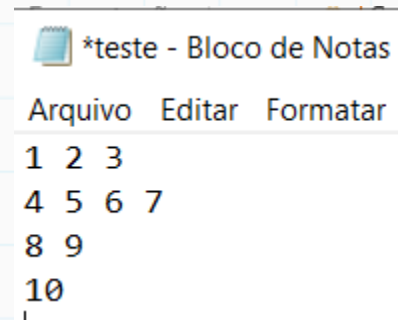
- Lendo arquivo: comando `readline` lê uma linha do arquivo e retorna como string (Necessário que o arquivo não tenha sido aberto em modo “w”)

formato: `<var>.readline()`

Mas e se o arquivo tiver mais linhas de tamanhos diferentes ?

```
1  arq = open("teste.txt", 'r')
2
3  soma = 0
4  for linha in arq:
5      for num in linha.split():
6          soma = soma + int(num)
7  print(soma)
8
9  arq.close()
```

usamos laço duplo,
para cada linha e para
cada palavra na linha



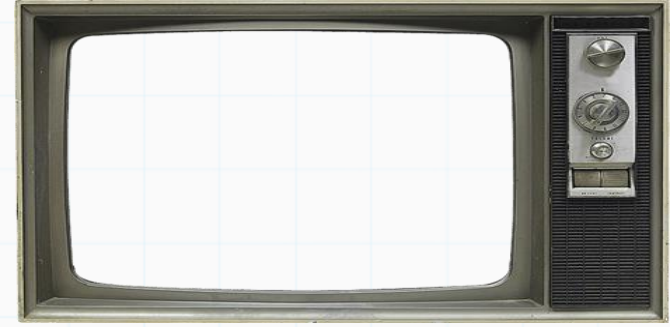
Shell x

55

[código](#)

Arquivos

- Existem dezenas de outros comandos e operadores de manipulação de arquivos (read, readlines, writeline, seek, flush, etc) que podem ser facilmente usados depois que os comandos básicos sejam dominados.



Arquivo

Exercício 1): O arquivo notas.txt contém uma linha para cada aluno de uma turma de estudantes. O nome de cada aluno está no início da linha e é seguido por suas 3 notas (separados por vírgula). Faça um programa que leia o arquivo e imprima os nomes dos alunos com média maior que 7 (imprima a média também)

notas - Bloco de Notas

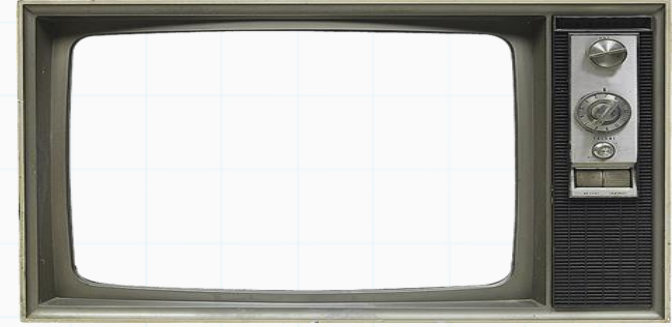
Arquivo Editar Formatar

```
Steve, 3, 4, 6  
Tony, 9, 9, 10  
Bruce, 8, 9, 8  
Natasha, 8, 7, 5  
Clint, 3, 4, 2  
|
```

saída:

Tony 9.333333333333334

Bruce 8.333333333333334

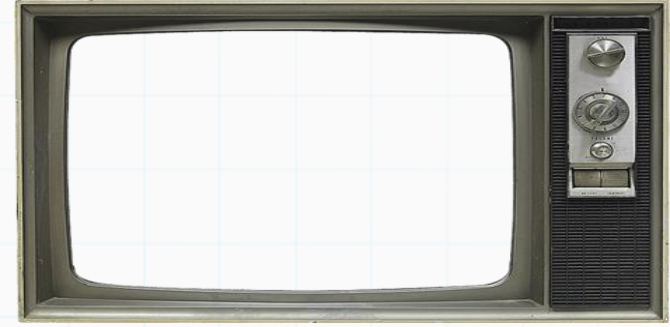


Arquivo

Exercício 1): O arquivo notas.txt contém uma linha para cada aluno de uma turma de estudantes. O nome de cada aluno está no início da linha e é seguido por suas 3 notas (separados por vírgula). Faça um programa que leia o arquivo e imprima os nomes dos alunos com média maior que 7 (imprima a média também)

```
1  arq = open("notas.txt", 'r')
2
3  for linha in arq:
4      nome,n1,n2,n3 = linha.split(',')
5      media = ((float(n1)+float(n2)+float(n3))/3)
6      if media >= 7:
7          print(nome,media)
8
9  arq.close()
```

[código](#)



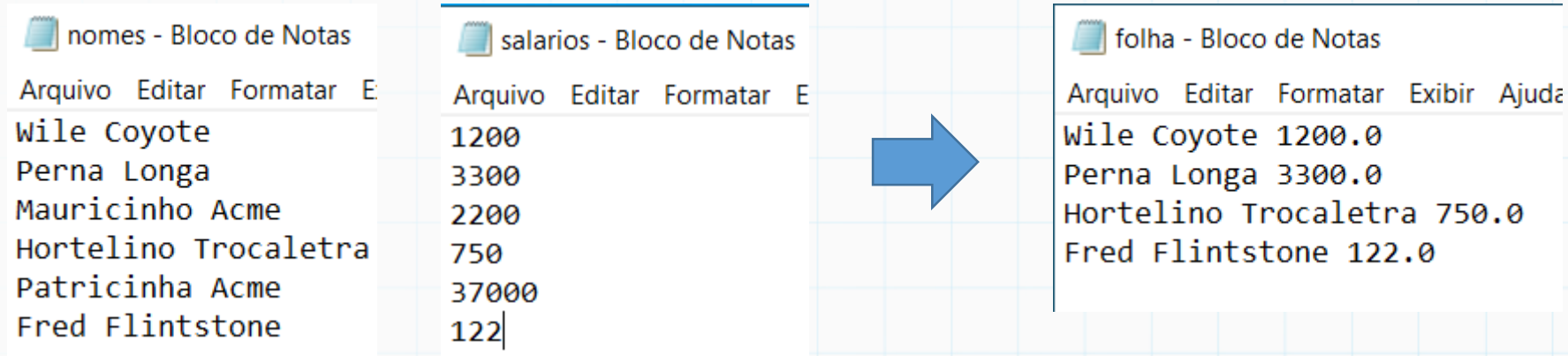
notas - Bloco de Notas

Arquivo	Editar	Formatar
Steve,	3,	4, 6
Tony,	9,	9, 10
Bruce,	8,	9, 8
Natasha,	8,	7, 5
Clint,	3,	4, 2

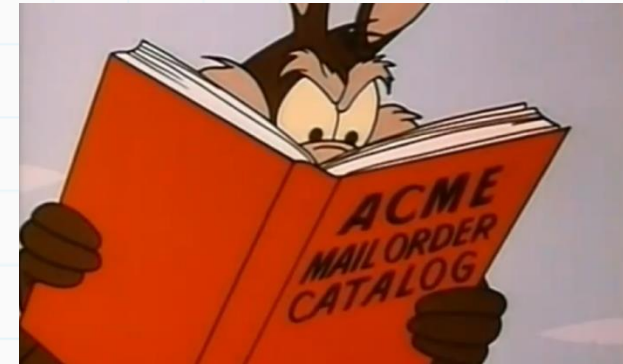
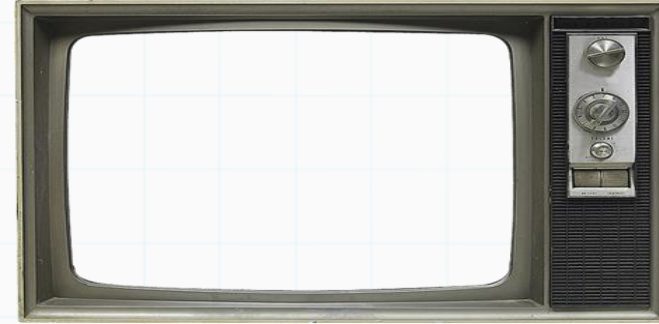


Arquivo

Exercício 2): O arquivo nomes.txt contém os nomes dos funcionários da empresa Acme, (nome e sobrenome separado por espaço), enquanto que o arquivo salarios.txt contém os salários (a mesma linha nos 2 arquivos equivale ao mesmo funcionário). Faça um programa para gerar o arquivo chamado folha.txt que contenha o nome, sobrenome e o salário do funcionário se o sobrenome do funcionário não for Acme (para evitar os comentários maldosos). No fim, imprima (na tela) o total da folha salarial (sem os salários dos herdeiros)



folha= 5372.0

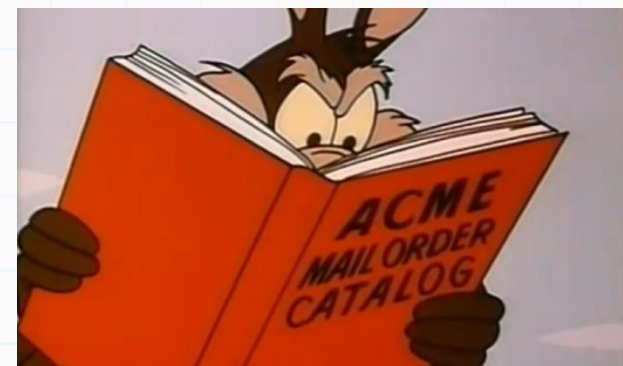
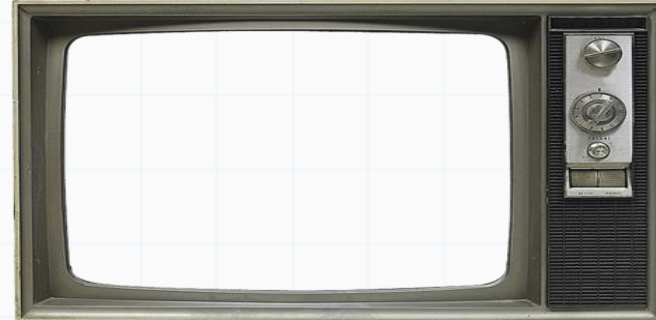



Dica: Para iterar (andar) em 2 arquivos ao mesmo tempo, não podemos usar o readline implícito com “for”, vamos usa-lo explicitamente com “while”. Não esquecendo de sempre chamar a próxima linha.

Arquivo

```
1 arq1 = open("nomes.txt", 'r')
2 arq2 = open("salarios.txt", 'r')
3 arq3 = open("folha.txt", 'w')
4
5 l1 = arq1.readline()
6 l2 = arq2.readline()
7
8 total = 0
9 while (l1 != ""):
10     nome, sobrenome = l1.split()
11     salario = float(l2)
12     if (sobrenome != "Acme"):
13         arq3.write(nome + " " + sobrenome + " " + str(salario) + "\n")
14         total = total + salario
15     l1 = arq1.readline()
16     l2 = arq2.readline()
17
18 print("folha=", total)
19 arq1.close()
20 arq2.close()
21 arq3.close()
```


[código](#)



 nomes - Bloco de Notas

Arquivo Editar Formatar E

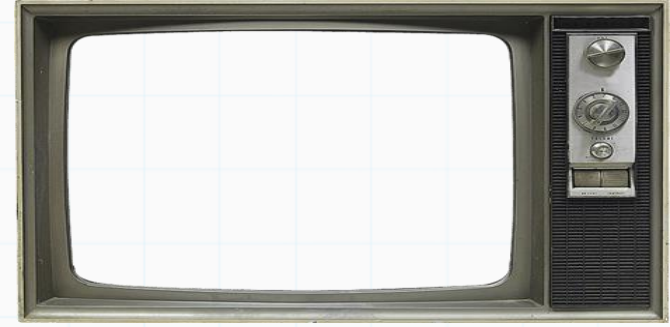
Wile Coyote
Perna Longa
Mauricinho Acme
Hortelino Trocaletra
Patricinha Acme
Fred Flintstone

 salarios - Bloco de Notas

Arquivo Editar Formatar E

1200
3300
2200
750
37000
122|

Arquivo



Exercício 3): Faça um programa que leia um arquivo texto contendo uma lista de endereços IP e gere dois outros arquivos, um contendo os endereços IP válidos e outro contendo os endereços inválidos. O formato de um endereço IP é:

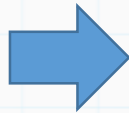
num1.num.num.num

onde **num1** vai de 1 a 255 e **num** vai de 0 a 255.

ips - Bloco de Notas

Arquivo Editar Formatar

```
127.55.122.29
255.0.0.256
400.1.2.3
100.200.111.222
0.0.0.0
255.255.255.255
|
```



v - Bloco de Notas

Arquivo Editar Formatar

```
127.55.122.29
100.200.111.222
255.255.255.255
|
```

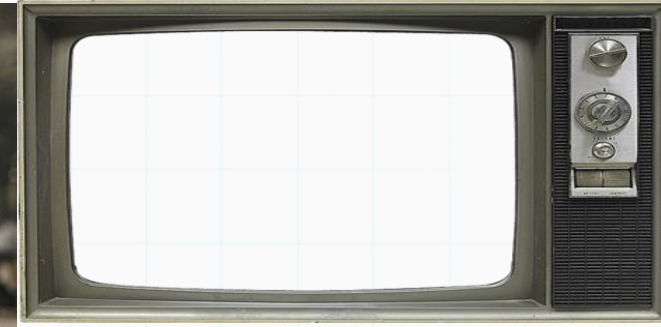
i - Bloco de Notas

Arquivo Editar Formatar

```
255.0.0.256
400.1.2.3
0.0.0.0
|
```



Arquivo



```
1 arq1=open('ips.txt','r')
2 arq2=open('v.txt','w')
3 arq3=open('i.txt','w')
4
5 for l1 in arq1:
6     n1,n2,n3,n4 = l1.split(".")
7     print(n1,n2,n3,n4)
8
9     if (int(n1)<=255 and int(n1)>=0) and (int(n2)<=255 and int(n2)>=1) and \
10        (int(n3)<=255 and int(n3)>=1) and (int(n4)<=255 and int(n4)>=1):
11         arq2.write(l1)
12     else:
13         arq3.write(l1)
14
15
16
17 arq1.close()
18 arq2.close()
19 arq3.close()
```

ips - Bloco de Notas

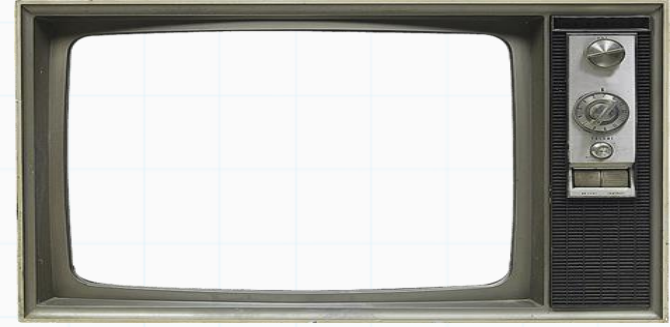
Arquivo Editar Formatar

```
127.55.122.29
255.0.0.256
400.1.2.3
100.200.111.222
0.0.0.0
255.255.255.255
|
```

quebra
de linha

[código](#)

Até a próxima



Slides baseados no curso de Vanessa Braganholo

