Aula 9

Professores:

Dante Corbucci Filho
Leandro A. F. Fernandes

Conteúdo:

- Persistência de Dados
  - Arquivo Texto

Arquivos

- Os programas apresentados neste curso até o momento são chamados **interativos**.
  - Programas interativos leem os dados de entrada do teclado e apresentam os dados de saída na tela.
  - Este tipo de programação é utilizada quando poucos dados são processados ou quando necessitam de interação humana.

Arquivos

- Os programas apresentados neste curso até o momento são chamados **interativos**.
  - Programas interativos leem os dados de entrada do teclado e apresentam os dados de saída na tela.
  - Este tipo de programação é utilizada quando poucos dados são processados ou quando necessitam de interação humana.

- Quando grandes quantidades de dados são processadas, **arquivos** são utilizados para armazenar os dados de entrada e os de saída.
  - Estes são chamados, respectivamente, arquivos de entrada e arquivos de saída.

Arquivo Texto

- Há basicamente dois tipos de arquivo: **texto** e **binário**.
  - Arquivos binários serão vistos nas próximas aulas.

Arquivo Texto

- Há basicamente dois tipos de arquivo: **texto** e **binário**.
  - Arquivos binários serão vistos nas próximas aulas.

- Um **arquivo texto** é uma sequência de caracteres, organizada em linhas, que reside em uma área de armazenamento (e.g., disco rígido, pen drive, CD/DVD) sob um mesmo nome.

Arquivo Texto

- Há basicamente dois tipos de arquivo: **texto** e **binário**.
  - Arquivos binários serão vistos nas próximas aulas.

- Um **arquivo texto** é uma sequência de caracteres, organizada em linhas, que reside em uma área de armazenamento (e.g., disco rígido, pen drive, CD/DVD) sob um mesmo nome.

- Arquivos texto podem ser criados, visualizados e alterados por editores ou processadores de texto.
  - O arquivo de entrada de um programa pode ser criado em um editor de texto e o arquivo de saída pode ser consultado utilizando-se também um editor.

Arquivo Texto

- Um arquivo texto é armazenado em disco como uma sequência de caracteres.

1	0	0		C	a	b	o		F	r	i	o	\n	2	6	1		L	a	g	u	n	a	\n	\0

Arquivo Texto

- Um arquivo texto é armazenado em disco como uma sequência de caracteres.

1	0	0		C	a	b	o		F	r	i	o	\n	2	6	1		L	a	g	u	n	a	\n	\0

52 bytes para UNICODE

- O arquivo acima contém 26 caracteres, entre eles, dígitos, brancos, letras e caracteres especiais: “\n” e “\0”.

Arquivo Texto

- Um arquivo texto é armazenado em disco como uma sequência de caracteres.

1	0	0		C	a	b	o		F	r	i	o	\n	2	6	1		L	a	g	u	n	a	\n	\0

52 bytes para UNICODE

- O arquivo acima contém 26 caracteres, entre eles, dígitos, brancos, letras e caracteres especiais: “\n” e “\0”.

- O caractere “\n” indica fim de linha e o caractere “\0” indica fim do arquivo.

1 decimal 49
A decimal 65
\n decimal 10
\0 decimal 0

Abrindo um Arquivo de Texto

- Em Python, antes de ser utilizado, um arquivo texto precisa ser associado a um nome no diretório de arquivos e ser aberto, via operação **open**:

variável = **open**(*caminho do arquivo*)

ou

variável = **open**(*caminho do arquivo*, *modo*)

Abrindo um Arquivo de Texto

- Em Python, antes de ser utilizado, um arquivo texto precisa ser associado a um nome no diretório de arquivos e ser aberto, via operação **open**:

variável = **open**(*caminho do arquivo*)

ou

variável = **open**(*caminho do arquivo*, *modo*)

- Os modos de operação de um arquivo são:

"r" : apenas leitura (se omitido = "r");

Abrindo um Arquivo de Texto

- Em Python, antes de ser utilizado, um arquivo texto precisa ser associado a um nome no diretório de arquivos e ser aberto, via operação **open**:

variável = **open**(*caminho do arquivo*)

ou

variável = **open**(*caminho do arquivo*, *modo*)

- Os modos de operação de um arquivo são:
  - "r" : apenas leitura (se omitido = "r");
  - "w" : apenas escrita;

Abrindo um Arquivo de Texto

- Em Python, antes de ser utilizado, um arquivo texto precisa ser associado a um nome no diretório de arquivos e ser aberto, via operação **open**:

variável = **open**(*caminho do arquivo*)

ou

variável = **open**(*caminho do arquivo*, *modo*)

- Os modos de operação de um arquivo são:
  - "r" : apenas leitura (se omitido = "r");
  - "w" : apenas escrita;
  - "a" : escrita no final do arquivo;

Abrindo um Arquivo de Texto

- Em Python, antes de ser utilizado, um arquivo texto precisa ser associado a um nome no diretório de arquivos e ser aberto, via operação **open**:

variável = **open**(*caminho do arquivo*)

ou

variável = **open**(*caminho do arquivo*, *modo*)

- Os modos de operação de um arquivo são:
  - “r” : apenas leitura (se omitido = “r”);
  - “w” : apenas escrita;
  - “a” : escrita no final do arquivo;
  - “r+” : leitura e escrita (não visto aqui).

Abrindo um Arquivo de Texto

dados = **open**(“teste.txt”, “r”)

Caso o arquivo exista: abre para leitura o arquivo “teste.txt”, e coloca a cabeça de leitura sobre o primeiro caractere da primeira linha.

Caso ele não exista: causa erro **FileNotFoundError**.

Abrindo um Arquivo de Texto

dados = **open**(“teste.txt”, “r”)

Caso o arquivo exista: abre para leitura o arquivo “teste.txt”, e coloca a cabeça de leitura sobre o primeiro caractere da primeira linha.

Caso ele não exista: causa erro **FileNotFoundError**.

dados = **open**(“teste.txt”, “w”)

Caso o arquivo exista: apaga seu conteúdo antigo e coloca a cabeça de escrita no início do arquivo.

Caso ele não exista: cria o arquivo no diretório e coloca a cabeça de escrita no início do arquivo.

Abrindo um Arquivo de Texto

dados = **open**(“teste.txt”, “r”)

Caso o arquivo exista: abre para leitura o arquivo “teste.txt”, e coloca a cabeça de leitura sobre o primeiro caractere da primeira linha.

Caso ele não exista: causa erro **FileNotFoundError**.

dados = **open**(“teste.txt”, “w”)

Caso o arquivo exista: apaga seu conteúdo antigo e coloca a cabeça de escrita no início do arquivo.

Caso ele não exista: cria o arquivo no diretório e coloca a cabeça de escrita no início do arquivo.

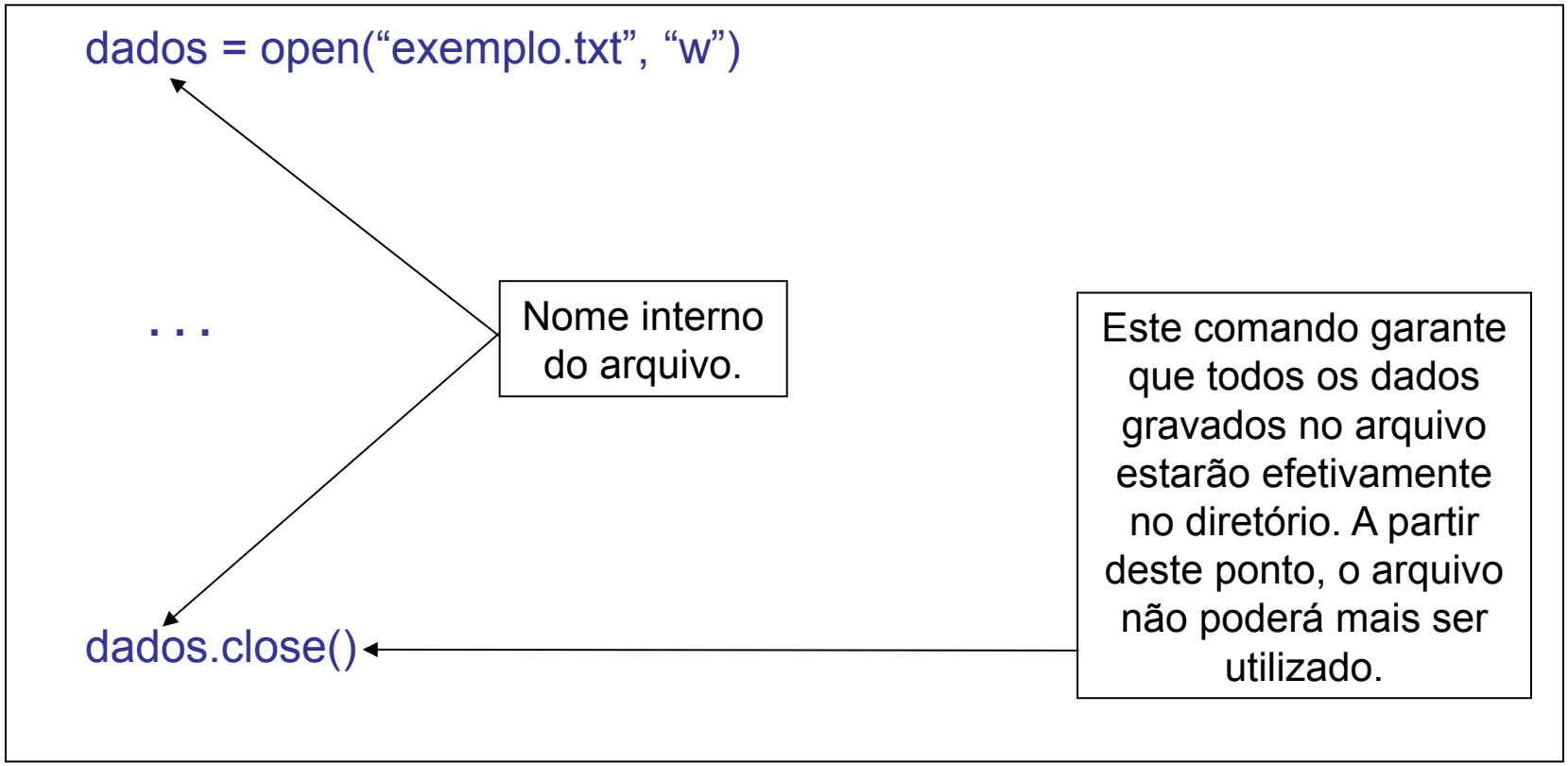
dados = **open**(“teste.txt”, “a”)

Caso o arquivo exista: abre o arquivo para escrita e coloca a cabeça de escrita no fim do arquivo. Isto é: pronto a anexar novas informações no seu final.

Caso ele não exista: cria o arquivo no diretório e coloca a cabeça de escrita no fim do arquivo, que neste caso é igual ao início.

Fechando um Arquivo de Texto

A operação **close()** permite que um arquivo texto seja fechado. Sempre que não for mais ser utilizado, um arquivo deve ser fechado.



O Método *readline()*

A operação **readline()**, aplicada sobre um arquivo texto aberto, retorna uma linha completa do arquivo, incluindo o fim de linha: “\n”. A cabeça de leitura avança para a próxima linha. Uma string vazia é retornada quando o fim de arquivo é encontrado.

```
dados = open(“exemplo.txt", “r”)

linha = dados.readline()


print(linha, end=“”)


dados.close()
```

O Método *readline()*

A operação **readline()**, aplicada sobre um arquivo texto aberto, retorna uma linha completa do arquivo, incluindo o fim de linha: "\n". A cabeça de leitura avança para a próxima linha. Uma string vazia é retornada quando o fim de arquivo é encontrado.

```
dados = open("exemplo.txt", "r")

linha = dados.readline()


print(linha, end="")


dados.close()
```

A partir deste ponto, o arquivo **"exemplo.txt"** pode ser lido. O primeiro caracter a ser lido será o primeiro caracter do arquivo (onde estará inicialmente a "cabeça de leitura").

O Método *readline()*

A operação **readline()**, aplicada sobre um arquivo texto aberto, retorna uma linha completa do arquivo, incluindo o fim de linha: "\n". A cabeça de leitura avança para a próxima linha. Uma string vazia é retornada quando o fim de arquivo é encontrado.

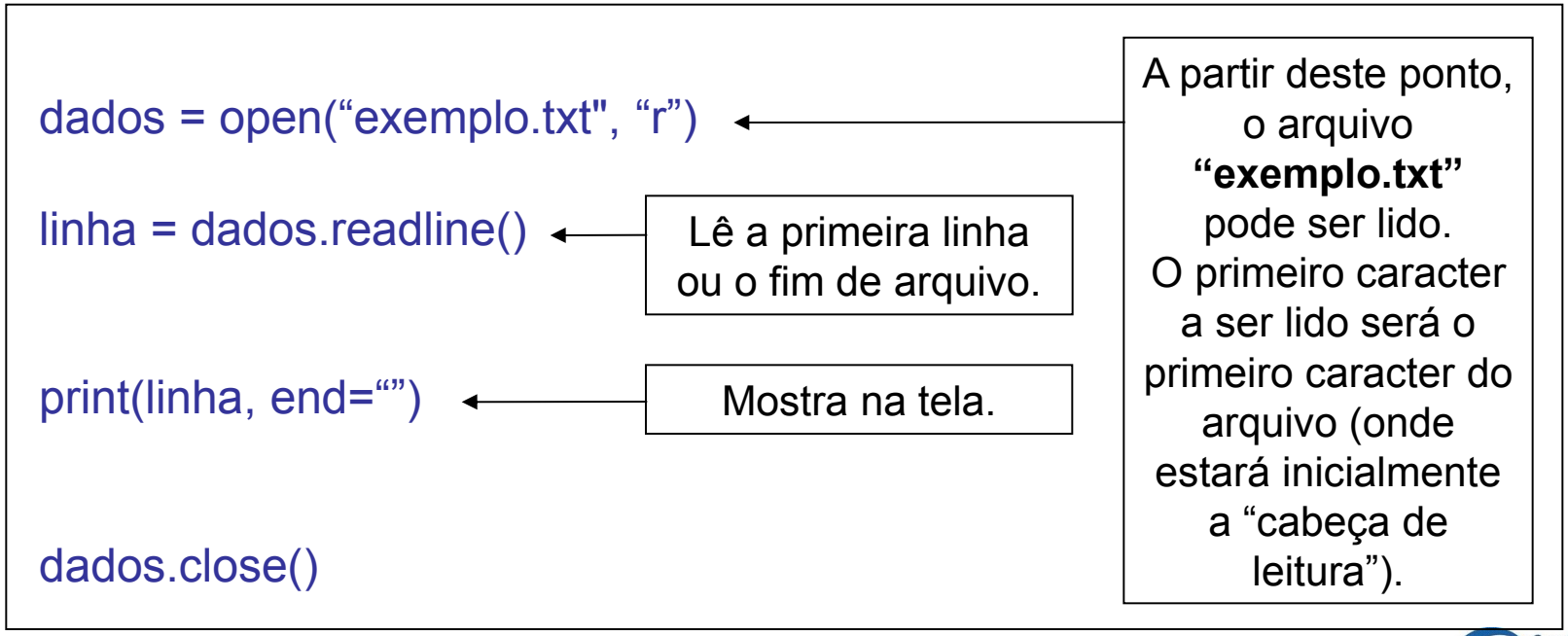
```
dados = open("exemplo.txt", "r")

linha = dados.readline()

print(linha, end="")


dados.close()
```

A partir deste ponto, o arquivo **"exemplo.txt"** pode ser lido. O primeiro caracter a ser lido será o primeiro caracter do arquivo (onde estará inicialmente a "cabeça de leitura").

Lê a primeira linha ou o fim de arquivo.

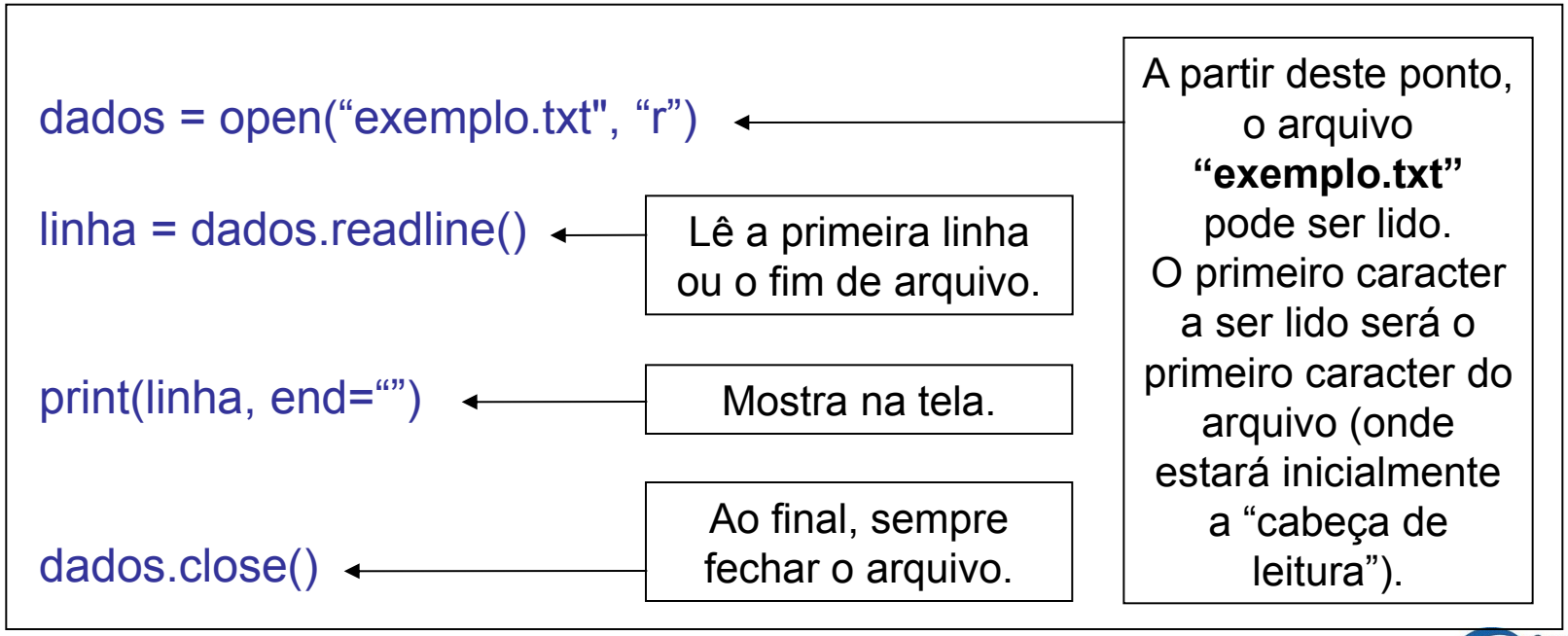
O Método *readline()*

A operação **readline()**, aplicada sobre um arquivo texto aberto, retorna uma linha completa do arquivo, incluindo o fim de linha: "\n". A cabeça de leitura avança para a próxima linha. Uma string vazia é retornada quando o fim de arquivo é encontrado.



O Método *readline()*

A operação **readline()**, aplicada sobre um arquivo texto aberto, retorna uma linha completa do arquivo, incluindo o fim de linha: "\n". A cabeça de leitura avança para a próxima linha. Uma string vazia é retornada quando o fim de arquivo é encontrado.



Lendo o Arquivo Texto com o Método *readline()*

O programa abaixo pede ao usuário que escolha um nome de arquivo, existente em seu diretório, e exibe seu conteúdo na tela.

```
nomeArquivo = input("Digite o nome do arquivo que deseja visualizar: ")

dados = open(nomeArquivo, "r")

linha = dados.readline()

while linha != "":

    print(linha, end="")

    linha = dados.readline()

dados.close()
```

Lendo o Arquivo Texto com o Método *readline()*

O programa abaixo pede ao usuário que escolha um nome de arquivo, existente em seu diretório, e exibe seu conteúdo na tela.

```
nomeArquivo = input("Digite o nome do arquivo que deseja visualizar: ")

dados = open(nomeArquivo, "r")

linha = dados.readline()

while linha != "":

    print(linha, end="")

    linha = dados.readline()

dados.close()
```

A partir deste ponto, o arquivo **nomeArquivo** pode ser lido. O primeiro caracter a ser lido será o primeiro caracter do arquivo (onde estará inicialmente a "cabeça de leitura").

Lendo o Arquivo Texto com o Método *readline()*

O programa abaixo pede ao usuário que escolha um nome de arquivo, existente em seu diretório, e exibe seu conteúdo na tela.

```
nomeArquivo = input("Digite o nome do arquivo que deseja visualizar: ")

dados = open(nomeArquivo, "r")

linha = dados.readline()

while linha != "":

    print(linha, end="")

    linha = dados.readline()

dados.close()
```

Lê primeira linha

A partir deste ponto, o arquivo **nomeArquivo** pode ser lido. O primeiro caracter a ser lido será o primeiro caracter do arquivo (onde estará inicialmente a "cabeça de leitura").

Lendo o Arquivo Texto com o Método *readline()*

O programa abaixo pede ao usuário que escolha um nome de arquivo, existente em seu diretório, e exibe seu conteúdo na tela.

```
nomeArquivo = input("Digite o nome do arquivo que deseja visualizar: ")

dados = open(nomeArquivo, "r")

linha = dados.readline()

while linha != "":

    print(linha, end="")

    linha = dados.readline()

dados.close()
```

Lê primeira linha

Enquanto não é o fim

A partir deste ponto, o arquivo **nomeArquivo** pode ser lido. O primeiro caracter a ser lido será o primeiro caracter do arquivo (onde estará inicialmente a "cabeça de leitura").

Lendo o Arquivo Texto com o Método *readline()*

O programa abaixo pede ao usuário que escolha um nome de arquivo, existente em seu diretório, e exibe seu conteúdo na tela.

```
nomeArquivo = input("Digite o nome do arquivo que deseja visualizar: ")

dados = open(nomeArquivo, "r")

linha = dados.readline()

while linha != "":

    print(linha, end="")

    linha = dados.readline()

dados.close()
```

A partir deste ponto, o arquivo **nomeArquivo** pode ser lido. O primeiro caracter a ser lido será o primeiro caracter do arquivo (onde estará inicialmente a "cabeça de leitura").

Lê primeira linha

Enquanto não é o fim

Mostra na tela

Lendo o Arquivo Texto com o Método *readline()*

O programa abaixo pede ao usuário que escolha um nome de arquivo, existente em seu diretório, e exibe seu conteúdo na tela.

```
nomeArquivo = input("Digite o nome do arquivo que deseja visualizar: ")

dados = open(nomeArquivo, "r")

linha = dados.readline()

while linha != "":

    print(linha, end="")

    linha = dados.readline()

dados.close()
```

Lê primeira linha

Enquanto não é o fim

Mostra na tela

Lê próxima linha

A partir deste ponto, o arquivo **nomeArquivo** pode ser lido. O primeiro caracter a ser lido será o primeiro caracter do arquivo (onde estará inicialmente a "cabeça de leitura").

Lendo o Arquivo Texto com o Método *readline()*

O programa abaixo pede ao usuário que escolha um nome de arquivo, existente em seu diretório, e exibe seu conteúdo na tela.

```
nomeArquivo = input("Digite o nome do arquivo que deseja visualizar: ")

dados = open(nomeArquivo, "r")

linha = dados.readline()

while linha != "":

    print(linha, end="")

    linha = dados.readline()

dados.close()
```

A partir deste ponto, o arquivo **nomeArquivo** pode ser lido. O primeiro caracter a ser lido será o primeiro caracter do arquivo (onde estará inicialmente a "cabeça de leitura").

Lê primeira linha

Enquanto não é o fim

Mostra na tela

Lê próxima linha

Ao final, sempre fechar o arquivo.

Lendo com o Iterador sobre o Arquivo

O programa abaixo pede ao usuário que escolha um nome de arquivo, existente em seu diretório, e exibe seu conteúdo na tela.

```
nomeArquivo = input("Digite o nome do arquivo que deseja visualizar: ")

dados = open(nomeArquivo, "r")

for linha in dados:

    print(linha, end="")

dados.close()
```

Lendo com o Iterador sobre o Arquivo

O programa abaixo pede ao usuário que escolha um nome de arquivo, existente em seu diretório, e exibe seu conteúdo na tela.

```
nomeArquivo = input("Digite o nome do arquivo que deseja visualizar: ")

dados = open(nomeArquivo, "r")

for linha in dados:

    print(linha, end="")

dados.close()
```

A partir deste ponto, o arquivo **nomeArquivo** pode ser lido.
O primeiro caracter a ser lido será o primeiro caracter do arquivo (onde estará inicialmente a "cabeça de leitura").

Lendo com o Iterador sobre o Arquivo

O programa abaixo pede ao usuário que escolha um nome de arquivo, existente em seu diretório, e exibe seu conteúdo na tela.

```
nomeArquivo = input("Digite o nome do arquivo que deseja visualizar: ")

dados = open(nomeArquivo, "r")

for linha in dados:

    print(linha, end="")

dados.close()
```

A partir deste ponto, o arquivo **nomeArquivo** pode ser lido. O primeiro caracter a ser lido será o primeiro caracter do arquivo (onde estará inicialmente a "cabeça de leitura").

Iterando linha a linha sobre o conteúdo de um arquivo texto.

Lendo com o Iterador sobre o Arquivo

O programa abaixo pede ao usuário que escolha um nome de arquivo, existente em seu diretório, e exibe seu conteúdo na tela.

```
nomeArquivo = input("Digite o nome do arquivo que deseja visualizar: ")

dados = open(nomeArquivo, "r")

for linha in dados:

    print(linha, end="")

dados.close()
```

A partir deste ponto, o arquivo **nomeArquivo** pode ser lido. O primeiro caracter a ser lido será o primeiro caracter do arquivo (onde estará inicialmente a "cabeça de leitura").

Iterando linha a linha sobre o conteúdo de um arquivo texto.

Ao final, sempre fechar o arquivo.

Lendo Todas as Linhas para uma Lista

O programa abaixo, que funciona apenas para pequenos arquivos, pede ao usuário que escolha um nome de arquivo, existente em seu diretório, e exibe seu conteúdo na tela.

```
nomeArquivo = input("Digite o nome do arquivo que deseja visualizar: ")

dados = open(nomeArquivo, "r")

linhas = dados.readlines()

for linha in linhas:

    print(linha, end="")

dados.close()
```

Lendo Todas as Linhas para uma Lista

O programa abaixo, que funciona apenas para pequenos arquivos, pede ao usuário que escolha um nome de arquivo, existente em seu diretório, e exibe seu conteúdo na tela.

```
nomeArquivo = input("Digite o nome do arquivo que deseja visualizar: ")

dados = open(nomeArquivo, "r")

linhas = dados.readlines()

for linha in linhas:

    print(linha, end="")

dados.close()
```

Restrição (ERRO): Será que o arquivo cabe na memória principal?

Lendo Todas as Linhas para uma Lista

O programa abaixo, que funciona apenas para pequenos arquivos, pede ao usuário que escolha um nome de arquivo, existente em seu diretório, e exibe seu conteúdo na tela.

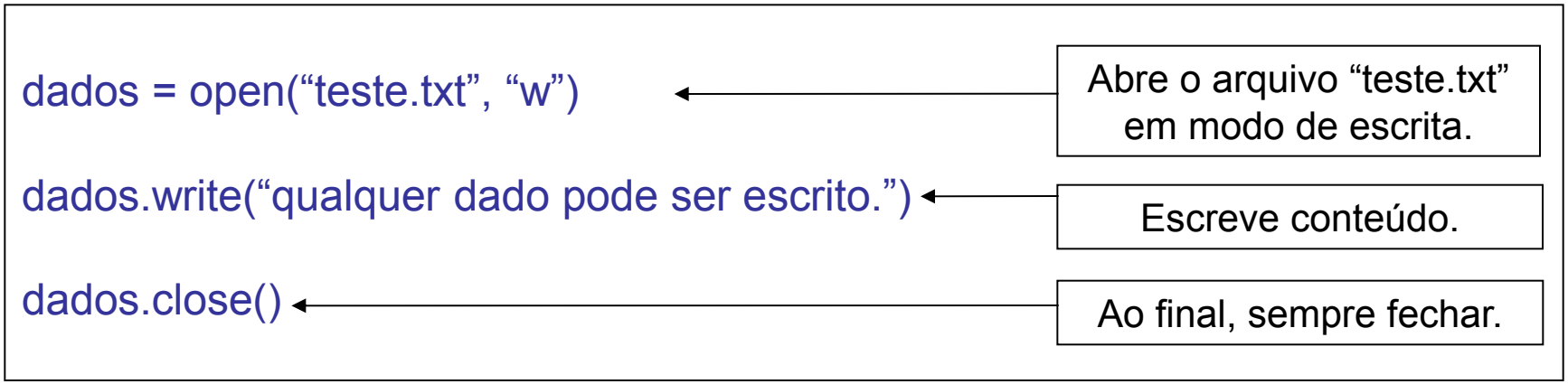
```
nomeArquivo = input("Digite o nome do arquivo que deseja visualizar: ")

dados = open(nomeArquivo, "r")

linhas = dados.readlines()

for linha in linhas:

    print(linha, end="")

dados.close()
```

Restrição (ERRO): Será que o arquivo cabe na memória principal?

Iterando linha a linha sobre a lista de linhas.

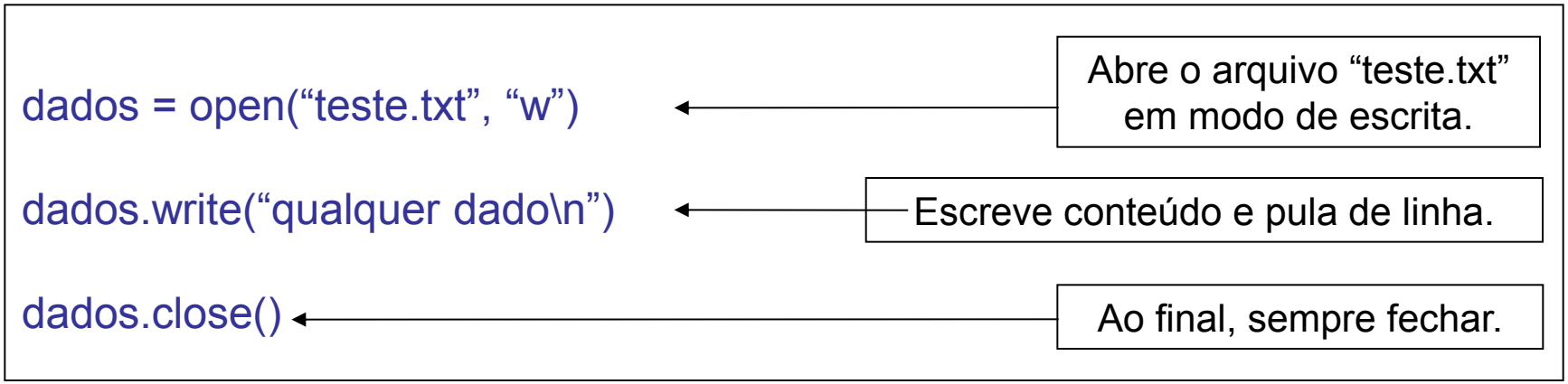
Produzindo Arquivo Texto: o Método *write()*

Para escrever uma sequência de caracteres em um arquivo texto, no modo “w” ou “a”, podemos utilizar o método **write**(**desejada**). Que escreverá a String **desejada** a partir do ponto em que a cabeça de escrita do arquivo estiver posicionada. Ao final, a cabeça de escrita ficará posicionada após o último caractere da String **desejada**.



Produzindo Arquivo Texto: *write()* com "\n"

Para escrever uma linha de texto, necessitamos colocar o caractere que representa o fim de linha. Esse caractere é o "\n".



Criando um Arquivo Texto

O programa abaixo pede ao usuário que escolha um nome de arquivo e quantidade de linhas que deseja escrever, em seguida os seus conteúdos são lidos do teclado e escritos no arquivo.

```
nomeArquivo = input("Digite o nome do arquivo que deseja criar: ")

quantasLinhas = int(input("Quantas linhas: "))

dados = open(nomeArquivo, "w")

for i in range(quantasLinhas):

    nova = input("Linha " + str(i+1) + ": ")

    dados.write(nova + "\n")

dados.close()
```

Criando um Arquivo Texto

O programa abaixo pede ao usuário que escolha um nome de arquivo e quantidade de linhas que deseja escrever, em seguida os seus conteúdos são lidos do teclado e escritos no arquivo.

```
nomeArquivo = input("Digite o nome do arquivo que deseja criar: ")

quantasLinhas = int(input("Quantas linhas: "))

dados = open(nomeArquivo, "w")

for i in range(quantasLinhas):

    nova = input("Linha " + str(i+1) + ": ")

    dados.write(nova + "\n")

dados.close()
```

A partir deste ponto, o arquivo **nomeArquivo** pode ser escrito. O primeiro caracter a ser escrito será o primeiro caracter do arquivo (onde estará inicialmente a "cabeça de escrita").

Criando um Arquivo Texto

O programa abaixo pede ao usuário que escolha um nome de arquivo e quantidade de linhas que deseja escrever, em seguida os seus conteúdos são lidos do teclado e escritos no arquivo.

```
nomeArquivo = input("Digite o nome do arquivo que deseja criar: ")

quantasLinhas = int(input("Quantas linhas: "))

dados = open(nomeArquivo, "w")

for i in range(quantasLinhas):

    nova = input("Linha " + str(i+1) + ": ")

    dados.write(nova + "\n")

dados.close()
```

A partir deste ponto, o arquivo **nomeArquivo** pode ser escrito. O primeiro caracter a ser escrito será o primeiro caracter do arquivo (onde estará inicialmente a "cabeça de escrita").

Escreve conteúdo e pula para a próxima linha.

Criando um Arquivo Texto

O programa abaixo pede ao usuário que escolha um nome de arquivo e quantidade de linhas que deseja escrever, em seguida os seus conteúdos são lidos do teclado e escritos no arquivo.

```
nomeArquivo = input("Digite o nome do arquivo que deseja criar: ")

quantasLinhas = int(input("Quantas linhas: "))

dados = open(nomeArquivo, "w")

for i in range(quantasLinhas):

    nova = input("Linha " + str(i+1) + ": ")

    dados.write(nova + "\n")

dados.close()
```

A partir deste ponto, o arquivo **nomeArquivo** pode ser escrito. O primeiro caracter a ser escrito será o primeiro caracter do arquivo (onde estará inicialmente a "cabeça de escrita").

Escreve conteúdo e pula para a próxima linha.

Ao final, sempre fechar.

Criando um Arquivo Texto de Pontos 2D

O programa abaixo cria um arquivo, chamado “pontos.txt”, com 30 pontos bidimensionais (2D), com coordenadas aleatórias (x,y) no intervalo 0 a 400. Ao final, mostra na tela o conteúdo do arquivo gerado.

```
 # Subprogramas
def criaArqPts(nome, qtd, min, max):
      from random import randint
      arq = open(nome, “w”)
      for pos in range(qtd):
          arq.write(str(randint(min,max))+“ ”+str(randint(min, max))+“\n”)
      arq.close()
      return None
 def mostra(nome):
      arq = open(nome, “r”)
      for pt in arq:
         print(pt, end=“”)
      arq.close()
      return None
# Programa Principal – Cria e Mostra Arquivo de Pontos 2D
 criaArqPts(“pontos.txt”, 30, 0, 400)
 mostra(“pontos.txt”)
```

Criando um Arquivo Texto de Pontos 2D

O programa abaixo cria um arquivo, chamado “pontos.txt”, com 30 pontos bidimensionais (2D), com coordenadas aleatórias (x,y) no intervalo 0 a 400. Ao final, mostra na tela o conteúdo do arquivo gerado.

```
 # Subprogramas
def criaArqPts(nome, qtd, min, max):
      from random import randint
      arq = open(nome, “w”)
      for pos in range(qtd):
           arq.write(str(randint(min,max))+“ ”+str(randint(min, max))+“\n”)
      arq.close()
      return None
 def mostra(nome):
      arq = open(nome, “r”)
      for pt in arq:
          print(pt, end=“”)
      arq.close()
      return None
# Programa Principal – Cria e Mostra Arquivo de Pontos 2D
 criaArqPts(“pontos.txt”, 30, 0, 400)
 mostra(“pontos.txt”)
```

Criando um Arquivo Texto de Pontos 2D

O programa abaixo cria um arquivo, chamado “pontos.txt”, com 30 pontos bidimensionais (2D), com coordenadas aleatórias (x,y) no intervalo 0 a 400. Ao final, mostra na tela o conteúdo do arquivo gerado.

```
 # Subprogramas
def criaArqPts(nome, qtd, min, max):
      from random import randint
      arq = open(nome, “w”)
      for pos in range(qtd):
          arq.write(str(randint(min,max))+“ ”+str(randint(min, max))+“\n”)
      arq.close()
      return None
 def mostra(nome):
      arq = open(nome, “r”)
      for pt in arq:
         print(pt, end=“”)
      arq.close()
      return None
# Programa Principal – Cria e Mostra Arquivo de Pontos 2D
 criaArqPts(“pontos.txt”, 30, 0, 400)
 mostra(“pontos.txt”)
```

Criando um Arquivo Texto de Pontos 2D

O programa abaixo cria um arquivo, chamado “pontos.txt”, com 30 pontos bidimensionais (2D), com coordenadas aleatórias (x,y) no intervalo 0 a 400. Ao final, mostra na tela o conteúdo do arquivo gerado.

```
 # Subprogramas
def criaArqPts(nome, qtd, min, max):
      from random import randint
      arq = open(nome, “w”)
      for pos in range(qtd):
          arq.write(str(randint(min,max))+“ ”+str(randint(min, max))+“\n”)
      arq.close()
      return None
 def mostra(nome):
      arq = open(nome, “r”)
      for pt in arq:
         print(pt, end=“”)
      arq.close()
      return None
# Programa Principal – Cria e Mostra Arquivo de Pontos 2D
 criaArqPts(“pontos.txt”, 30, 0, 400)
 mostra(“pontos.txt”)
```

Processando um Arquivo Texto de Pontos 2D

O programa abaixo faz a leitura de um arquivo, chamado “pontos.txt”, com pontos bidimensionais (2D), com coordenadas (x,y). Calcula e escreve o centroide de todos os pontos lidos.

```
# Subprogramas
def centroide(nome):
    arquivo = open(nome, “r”)
    qtdPts = 0
    somaX = 0
    somaY = 0
    for coordenada in arquivo:
        partes = coordenada.split()
        somaX += float(partes[0])
        somaY += float(partes[1])
        qtdPts+=1
    arquivo.close()
    if qtdPts == 0:
        print(arquivo.name, “- vazio!!!”)
    else:
        print(“Ponto calculado: (”, somaX/qtdPts, “,”, somaY/qtdPts, “).”)
    return None
# Programa Principal – Calcula e escreve o centroide de pontos.
centroide(“pontos.txt”)
```

Copiando um Arquivo Texto

```
# Subprogramas
def mostra(nome):
    infos = open(nome, "r")
    for linha in infos:
        print(linha.strip())
    infos.close()
    return None
def copiar(nomeOrigem, nomeDestino):
    orig = open(nomeOrigem,"r")
    dest = open(nomeDestino, "w")
    for linha in orig:
        dest.write(linha)
    orig.close()
    dest.close()
    return None
# Programa Principal
nomes = input("Escreva os nomes dos arquivos, original e destino: ").split()
mostra(nomes[0])
copiar(nomes[0], nomes[1])
mostra(nomes[1])
```

Copiando um Arquivo Texto

```
# Subprogramas
def mostra(nome):
    infos = open(nome, "r")
    for linha in infos:
        print(linha.strip())
    infos.close()
    return None
def copiar(nomeOrigem, nomeDestino):
    orig = open(nomeOrigem,"r")
    dest = open(nomeDestino, "w")
    for linha in orig:
        dest.write(linha)
    orig.close()
    dest.close()
    return None
# Programa Principal
nomes = input("Escreva os nomes dos arquivos, original e destino: ").split()
mostra(nomes[0])
copiar(nomes[0], nomes[1])
mostra(nomes[1])
```

Copiando um Arquivo Texto

```
# Subprogramas
def mostra(nome):
    infos = open(nome, "r")
    for linha in infos:
        print(linha.strip())
    infos.close()
    return None
def copiar(nomeOrigem, nomeDestino):
    orig = open(nomeOrigem,"r")
    dest = open(nomeDestino, "w")
    for linha in orig:
        dest.write(linha)
    orig.close()
    dest.close()
    return None
# Programa Principal
nomes = input("Escreva os nomes dos arquivos, original e destino: ").split()
mostra(nomes[0])
copiar(nomes[0], nomes[1])
mostra(nomes[1])
```

Copiando um Arquivo Texto

```
# Subprogramas
def mostra(nome):
    infos = open(nome, "r")
    for linha in infos:
        print(linha.strip())
    infos.close()
    return None
def copiar(nomeOrigem, nomeDestino):
    orig = open(nomeOrigem,"r")
    dest = open(nomeDestino, "w")
    for linha in orig:
        dest.write(linha)
    orig.close()
    dest.close()
    return None
# Programa Principal
nomes = input("Escreva os nomes dos arquivos, original e destino: ").split()
mostra(nomes[0])
copiar(nomes[0], nomes[1])
mostra(nomes[1])
```

Copiando um Arquivo Texto

```
# Subprogramas
def mostra(nome):
    infos = open(nome, “r”)
    for linha in infos:
        print(linha.strip())
    infos.close()
    return None
def copiar(nomeOrigem, nomeDestino):
    orig = open(nomeOrigem,”r”)
    dest = open(nomeDestino, “w”)
    for linha in orig:
        dest.write(linha)
    orig.close()
    dest.close()
    return None
# Programa Principal
nomes = input(“Escreva os nomes dos arquivos, original e destino: ”).split()
mostra(nomes[0])
copiar(nomes[0], nomes[1])
mostra(nomes[1])
```

Copiando um Arquivo Texto

```
# Subprogramas
def mostra(nome):
    infos = open(nome, “r”)
    for linha in infos:
        print(linha.strip())
    infos.close()
    return None
def copiar(nomeOrigem, nomeDestino):
    orig = open(nomeOrigem,”r”)
    dest = open(nomeDestino, “w”)
    for linha in orig:
        dest.write(linha)
    orig.close()
    dest.close()
    return None
# Programa Principal
nomes = input(“Escreva os nomes dos arquivos, original e destino: ”).split()
mostra(nomes[0])
copiar(nomes[0], nomes[1])
mostra(nomes[1])
```

Anexando uma Nova Linha ao Final de um Arquivo

```
nome = input("Diga o nome do arquivo que deseja anexar linha ao final: ")

arquivo = open(nome, "a")

novaLinha = input("Diga a nova linha: ")

arquivo.write(novaLinha + "\n")

arquivo.close()
```

Erros de Entrada e Saída para Arquivos Texto

- Nos programas vistos até agora, um erro de entrada e saída que ocorra fará seu programa terminar em estado de erro (abortará).

Erros de Entrada e Saída para Arquivos Texto

- Nos programas vistos até agora, um erro de entrada e saída que ocorra fará seu programa terminar em estado de erro (abortará).

- É possível se evitar este término abrupto do programa pelo uso de tratamento de exceções, que será visto em aulas futuras.
  - Lá veremos que os erros de entrada e saída que ocorrerem dentro de uma região do código onde as exceções são tratadas não abortarão o programa.

Faça os Exercícios Relacionados a essa Aula

Clique no botão para visualizar os enunciados:

Aula 9

Professores:

Dante Corbucci Filho
Leandro A. F. Fernandes

Conteúdo:

- Persistência de Dados
  - Arquivo Texto