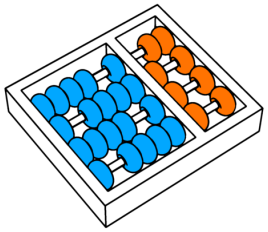

MC102 – Aula 12
Arquivos

Algoritmos e Programação de Computadores

Zanoni Dias

2019

Instituto de Computação

Roteiro

Arquivos

Arquivo Texto

Exercícios

Arquivos

Tipos de Arquivos

- Arquivos podem ter o mais variado conteúdo, mas do ponto de vista dos programas existem apenas dois tipos de arquivos:
  - Arquivo texto: Armazena caracteres que podem ser mostrados diretamente na tela ou modificados por um editor de textos simples. Exemplos: código fonte Python, documento texto simples, páginas HTML (*HyperText Markup Language*), arquivos CSV (*Comma-Separated Values*).
  - Arquivo binário: Sequência de bits sujeita às convenções do programa que o gerou, não legíveis diretamente por um humano. Exemplos: arquivos executáveis, arquivos compactados, documentos do Word.

Arquivo Texto

Arquivo Texto

- Para trabalharmos com arquivos devemos abri-lo e associá-lo com uma variável utilizando a função **open**.
- A função **open** recebe como parâmetros o nome do arquivo (incluindo o caminho até ele) e o modo desejado para abrir o arquivo.
  - `r` Leitura: nesse modo podemos somente ler os dados do arquivo.
  - `w` Escrita: nesse modo podemos escrever/modificar os dados do arquivo.
  - `r+` Leitura/escrita: nesse modo podemos ler e também escrever/modificar os dados do arquivo.
  - `a` Anexação: nesse modo podemos somente adicionar novos dados no final do arquivo.

Arquivo Texto

- Ao tentar abrir um arquivo inexistente para leitura (`r`), a função `open` gerará um erro.
- Ao abrir um arquivo para escrita (`w`), seu conteúdo é primeiramente apagado. Se o arquivo não existir, um novo arquivo será criado.
- Ao tentar abrir um arquivo inexistente para leitura leitura/escrita (`r+`), a função `open` gerará um erro. Se o arquivo existir, seu conteúdo não será primeiramente apagado.
- Ao tentar abrir um arquivo inexistente para anexação (`a`), um novo arquivo será criado.

Arquivo Texto

- Exemplo:

```
arq = open("teste1.txt", "r")
# abrindo o arquivo teste1.txt com modo leitura
arq = open("teste2.txt", "w")
# abrindo o arquivo teste2.txt com modo escrita
arq = open("teste3.txt", "r+")
# abrindo o arquivo teste3.txt com modo leitura/escrita
arq = open("teste4.txt", "a")
# abrindo o arquivo teste4.txt com modo anexação
```

Arquivo Texto

- Exemplo:

```
arq = open("MC102/teste.txt", "r")
# abrindo o arquivo teste.txt no diretório MC102
# usando modo de leitura
arq = open("arqs/arquivo.log", "r+")
# abrindo o arquivo arquivo.log no diretório arqs
# usando modo de leitura/escrita
```

Arquivo Texto

- A função **open** retorna um objeto do tipo **_io.TextIOWrapper** que possui métodos para ler e escrever em um arquivo.

```
1 arq = open("teste.txt", "r")
2 print(arq)
3 # <_io.TextIOWrapper name='teste.txt' mode='r'
4 #  encoding='UTF-8'>
5 print(type(arq))
6 # <class '_io.TextIOWrapper'>
```

Arquivo Texto - Leitura

- O método `read` é utilizado para ler os dados de um arquivo.
- O método `read` recebe como parâmetro o número de caracteres que devem ser lidos.
- O método `read` retorna uma string compatível com a quantidade de caracteres especificados.
- Caso a quantidade de caracteres não seja especificada, o método `read` irá retornar o conteúdo completo do arquivo.
- Para utilizar o método `read`, o arquivo deve ser aberto no modo de leitura (`r`) ou leitura/escrita (`r+`).
- Considere o arquivo `teste.txt` com o seguinte conteúdo:

```
MC102
Unicamp - Python
```

Arquivo Texto - Leitura

- Lendo o arquivo `teste.txt`:

```
arq = open("teste.txt", "r")
texto = arq.read()
print(texto, end = "")
# MC102
# Unicamp - Python
```

- Lendo os 5 primeiros caracteres do arquivo `teste.txt`:

```
arq = open("teste.txt", "r")
texto = arq.read(5)
print(texto)
# MC102
```

Arquivo Texto - Leitura

- Quando um arquivo é aberto, um indicador de posição no arquivo é criado, e este recebe a posição do início do arquivo.
- Para cada dado lido ou escrito no arquivo, este indicador de posição é automaticamente incrementado para a próxima posição do arquivo.
- O método `read` retorna uma string vazia caso o indicador de posição esteja no fim do arquivo.

Arquivo Texto - Leitura

- Exemplo de como ler os dados de um arquivo caractere por caractere:

```
arq = open("teste.txt", "r")
texto = ""
c = arq.read(1)

while c:
  texto = texto + c
  c = arq.read(1)

print(texto, end = "")
# MC102
# Unicamp - Python
```

Arquivo Texto - Leitura

- O método `readline` retorna uma string referente a uma linha do arquivo.
- Similar ao método `read`, o método `readline` retorna uma string vazia caso o indicador de posição esteja no fim do arquivo.
- Para utilizar o método `readline`, o arquivo deve ser aberto bo modo de leitura (`r`) ou leitura/escrita (`r+`).

Arquivo Texto - Leitura

- Exemplo de como ler os dados de um arquivo linha por linha:

```
arq = open("teste.txt", "r")
linha = arq.readline()

while linha:
  print(linha, end = "")
  linha = arq.readline()

# MC102
# Unicamp - Python
```

Arquivo Texto - Leitura

- Outra forma de ler os dados de um arquivo linha por linha:

```
1 arq = open("teste.txt", "r")
2
3 for linha in arq:
4   print(linha, end = "")
5
6 # MC102
7 # Unicamp - Python
```

Arquivo Texto - Leitura

- O método `tell` retorna a posição atual no arquivo.
- Podemos alterar o indicador de posição de um arquivo utilizando o método `seek`.
- O método `seek` recebe a nova posição, em relação ao início do arquivo.
- Podemos usar os métodos `seek` e `tell` combinados para alterar a posição do arquivo com base na posição atual.

Arquivo Texto - Leitura

- Lendo a primeira linha do arquivo `teste.txt` duas vezes:

```
1  arq = open("teste.txt", "r")
2
3  linha = arq.readline()
4  print(linha, end = "")
5  # MC102
6
7  arq.seek(0) # Voltando para o início do arquivo
8
9  linha = arq.readline()
10 print(linha, end = "")
11 # MC102
12
13 linha = arq.readline()
14 print(linha, end = "")
15 # Unicamp - Python
```

Arquivo Texto - Leitura

- Avançando e retrocedendo num arquivo:

```
1  arq = open("teste.txt", "r")
2
3  linha = arq.readline()
4  print(linha, end = "")
5  # MC102
6  print("Posição =", arq.tell())
7  # Posição = 6
8
9  arq.seek(arq.tell() - 3)
10
11 linha = arq.readline()
12 print(linha, end = "")
13 # 02
14 print("Posição =", arq.tell())
15 # Posição = 6
```

Arquivo Texto - Leitura

- Avançando e retrocedendo num arquivo:

```
1  arq = open("teste.txt", "r")
2
3  linha = arq.readline()
4  print(linha, end = "")
5  # MC102
6  print("Posição =", arq.tell())
7  # Posição = 6
8
9  arq.seek(arq.tell() + 3)
10
11 linha = arq.readline()
12 print(linha, end = "")
13 # camp - Python
14 print("Posição =", arq.tell())
15 # Posição = 23
```

Arquivo Texto - Escrita

- Para escrevermos em um arquivo utilizamos o método `write`.
- O método `write` recebe como parâmetro a string que será escrita no arquivo.
- Para utilizar o método `write`, o arquivo deve ser aberto com o modo de escrita (`w`), leitura/escrita (`r+`) ou anexação (`a`).

Arquivo Texto - Escrita

- O método `close` deve sempre ser usado para fechar um arquivo que foi aberto.
- Quando escrevemos dados em um arquivo, este comando garante que os dados serão efetivamente escritos no arquivo.
- Ele também libera recursos que são alocados para manter a associação da variável com o arquivo.

Arquivo Texto - Escrita

- Criando um arquivo `teste.txt`:

```
1  arq = open("teste.txt", "w")
2  arq.write("Hello World!\n")
3  arq.write("Hello World!\n")
4  arq.close()
5
6  arq = open("teste.txt", "r")
7  texto = arq.read()
8  arq.close()
9
10 print(texto, end = "")
11 # Hello World!
12 # Hello World!
```

Arquivo Texto - Escrita

- Adicionando mais dados no arquivo `teste.txt`:

```
1  arq = open("teste.txt", "a")
2  arq.write("MC102\n")
3  arq.write("Unicamp - Python\n")
4  arq.close()
5
6  arq = open("teste.txt", "r")
7  texto = arq.read()
8  arq.close()
9
10 print(texto, end = "")
11 # Hello World!
12 # Hello World!
13 # MC102
14 # Unicamp - Python
```

Arquivo Texto - Escrita

- A função **`print`** também pode ser utilizada para escrever dados em um arquivo.
- Para isso, basta utilizar o parâmetro **`file`**, indicando em qual arquivo, adequadamente aberto, a mensagem deve ser escrita.
- Exemplo:

```
1  arq = open("teste.txt", "w")
2  print("Utilizando a função print", file = arq)
3  arq.close()
4
5  arq = open("teste.txt", "r")
6  texto = arq.read()
7  arq.close()
8
9  print(texto)
10 # Utilizando a função print
```

Exercícios

Exercícios

Nos dois exercícios abaixo considere a existência de um arquivo `financeiro.log` com os registros financeiros de uma empresa, com o seguinte conteúdo inicial:

```
1000 capital inicial
-500 compra de matéria-prima
-200 mão de obra
400 venda do primeiro lote
300 venda do segundo lote
-300 aluguel da fábrica
```

1. Escreva um programa que leia o arquivo `financeiro.log` e imprima o saldo financeiro da empresa.
2. Escreva um programa que leia um valor e uma descrição, e inclua uma nova linha no arquivo `financeiro.log`, conforme o formato ilustrado acima.

Exercícios

3. Escreva um programa que leia o nome de dois arquivos e duas strings. Seu programa deve ler o conteúdo do primeiro arquivo e escrevê-lo no segundo arquivo, substituindo todas as ocorrências da primeira pela segunda string.
4. Escreva um programa que, dado o nome de um arquivo no formato CSV (*Comma-Separated Values*) e uma string representando o separador, leia e armazene o conteúdo do arquivo numa lista bidimensional.

Exercício 1 - Saldo

```
arq = open("financeiro.txt", "r")
saldo = 0

for linha in arq:
  saldo = saldo + int(linha.split()[0])

print("Saldo =", saldo)

arq.close()
```

Exercício 2 - Movimentação Financeira

```
1 arq = open("financeiro.txt", "a")
2
3 valor = input()
4 descrição = input()
5
6 print(valor, descrição, file = arq)
7
8 arq.close()
```

Exercício 3 - Substituição de Strings

```
arq1 = input()
arq2 = input()
str1 = input()
str2 = input()

entrada = open(arq1, "r")
saída = open(arq2, "w")

for linha in entrada:
  nova = linha.replace(str1, str2)
  print(nova, end = "", file = saída)

entrada.close()
saída.close()
```

Exercício 4 - Exemplo de Arquivo CSV

```
Bulbasaur,0.7m,6.9kg,Seed,Overgrow
Charmander,0.6m,8.5kg,Lizard,Blaze
Squirtle,0.5m,9.0kg,Tyny Turtle,Torrent
Pikachu,0.4m,6.0kg,Mouse,Static
Jigglypuff,0.5m,5.5kg,Ballon,Cute Charm
Snorlax,2.1m,460.0kg,Sleeping,Immunity
```

Exercício 4 - Lendo um Arquivo CSV

```
1  csv = input()
2  sep = input()
3
4  arq = open(csv, "r")
5
6  m = []
7  for linha in arq:
8    linha = linha.replace("\n", "")
9    m.append(linha.split(sep))
10
11 arq.close()
12
13 print(m)
```

Exercício 4 - Exemplo de Lista Bidimensional Gerada

```
[['Bulbasaur', '0.7m', '6.9kg', 'Seed', 'Overgrow'],
 ['Charmander', '0.6m', '8.5kg', 'Lizard', 'Blaze'],
 ['Squirtle', '0.5m', '9.0kg', 'Tyny Turtle', 'Torrent'],
 ['Pikachu', '0.4m', '6.0kg', 'Mouse', 'Static'],
 ['Jigglypuff', '0.5m', '5.5kg', 'Ballon', 'Cute Charm'],
 ['Snorlax', '2.1m', '460.0kg', 'Sleeping', 'Immunity']]
```