Algoritmos Computacionais

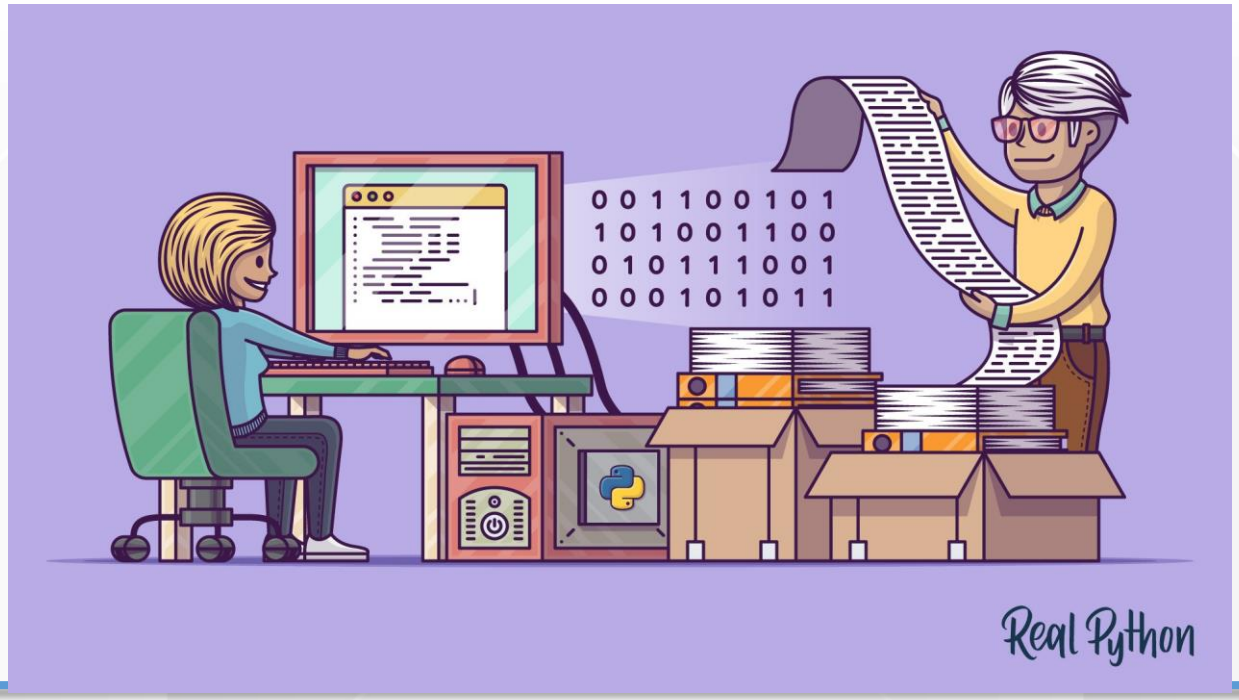
Conteúdo: Arquivos

Prof. Dsc. Giomar Sequeiros
giomar@eng.uerj.br

Arquivos

Arquivos

- Um arquivo é uma área em **disco** onde podemos ler e **gravar informações**.
  - Essa área é **gerenciada** pelo **sistema operacional** do computador, ou seja, não precisamos nos preocupar em como esse espaço é organizado em disco.
- Do ponto de vista de programas, um arquivo é acessado por nome e é onde podemos **ler** e **escrever** linhas de texto ou dados em geral.



Arquivos: função open

- Para **acessar** um arquivo, precisamos **abri-lo**.
  - Durante a abertura, informamos o **nome** do **arquivo**, com o nome do **diretório** onde ele se encontra (se necessário) e que **operações** queremos realizar: **leitura** e/ou **escrita**.
- A função **open** utiliza os parâmetros nome e modo. O nome é o nome do arquivo em si, por exemplo, leiame.txt. O modo indica as operações que vamos realizar

Modo	Operações
r	leitura
w	escrita, apaga o conteúdo se já existir
a	escrita, mas preserva o conteúdo se já existir
b	modo binário
+	atualização (leitura e escrita)

Modos de abertura de arquivos

Arquivos: função open

- Os modos podem ser **combinados** (“r+”,“w+”, “a+”, “r+b”, “w+b”, “a+b”).
- A função open retorna um objeto do tipo file (arquivo).
  - É esse objeto que vamos utilizar para ler e escrever os dados no arquivo.
- Utilizamos o método **write** para escrever ou gravar dados no arquivo, **read** para ler e close para fechá-lo.
- Ao trabalharmos com arquivos, devemos sempre realizar o seguinte **ciclo**:
  - abertura,
  - leitura e/ou escrita,
  - fechamento.
- O **fechamento** do arquivo é muito importante, pois cada arquivo aberto consome recursos do computador. Só o fechamento do arquivo garante a liberação desses recursos e preserva a integridade dos dados do arquivo.

Arquivos: Exemplo escrita

- Vejamos um exemplo onde vamos escrever o arquivo números.txt com 20 linhas

```
arquivo = open('numeros.txt','w')
for linha in range(1,21):
    arquivo.write('%d\n' % linha)
arquivo.close()
```

- O programa cria um novo arquivo (numeros.txt) no diretório atual, ou seja, no mesmo diretório do programa.
- Utilizamos a função **open** para **abrir** o arquivo números.txt.
  - Observe que o nome do arquivo é uma string e deve ser escrito entre aspas.
- O modo escolhido foi "w", indicando **escrita** ou gravação.
  - O modo "w" cria o arquivo se ele não existir. Caso já exista, seu conteúdo é apagado

Arquivos: Exemplo escrita (outro método)

- Este código cria um arquivo chamado "exemplo.txt" e escreve a string "Olá, mundo!" dentro dele.

```
# Abre um arquivo no modo "w" para escrever nele
with open("exemplo.txt", "w") as arquivo:
    arquivo.write("Olá, mundo!")
```

- O uso do **with** garante que o **arquivo** será **fechado** automaticamente ao final do bloco de código.

Arquivos: Exemplo leitura

Vejamos agora um programa para **ler** o arquivo e **imprimir** suas linhas na tela

```
arquivo = open('numeros.txt','r')
for linha in arquivo.readlines():
    print(linha)
arquivo.close()
```

- Utilizamos a função **open** para **abrir** o arquivo.
  - O nome é o mesmo utilizado durante a gravação, mas o modo agora é **"r"**, ou seja, leitura.
- O método **readlines**, que gera uma **lista** em que cada **elemento** é uma **linha** do **arquivo**.
  - Imprimimos a linha na tela.
- Finalmente fechamos o arquivo

Arquivos: Exemplo leitura (outro método)

O código abre o arquivo "exemplo.txt" e lê seu conteúdo para a variável conteudo. Em seguida, imprime o conteúdo na tela.

```
# Abre um arquivo no modo "r" para ler seu conteúdo
with open("exemplo.txt", "r") as arquivo:
    conteudo = arquivo.read()
    print(conteudo)
```

- O uso do **with** garante que o arquivo será **fechado automaticamente** ao final do bloco de código.

Exemplo: Geração de arquivos

Vejamos o programa abaixo, que gera dois arquivos com 50 linhas cada.

```
impares = open('impares.txt','w')
pares = open('pares.txt','w')

for n in range(0,100):
    if n % 2 == 0:
        pares.write('%d\n' % n)
    else:
        impares.write('%d\n' % n)

impares.close()
pares.close()
```

- O programa distribui os números ímpares e pares em arquivos diferentes.
- Observe que para gravar em arquivos diferentes, utilizamos um **if** e dois arquivos abertos para escrita.
  - Utilizando o método write dos objetos pares e mpares, fizemos a seleção de onde gravar o número.
  - Observe que incluímos o \n para indicar fim de linha

Exemplo: Leitura e escrita

Podemos realizar diversas operações com arquivos; entre elas ler, processar e gerar novos arquivos. Utilizando o arquivo pares.txt criado no exemplo anterior, vejamos como filtrá-lo de forma a gerar um novo arquivo, apenas com números múltiplos de 4

```
multiplos4 = open('multiplos4.txt','w')
pares = open('pares.txt')

for linha in pares.readlines():
    if int(linha) % 4 == 0:
        multiplos4.write(linha)

pares.close()
multiplos4.close()
```

Veja que convertemos a linha lida de string para inteiro antes de fazer os cálculos

Exemplo: Processamento de um arquivo

- Podemos também processar as linhas de um arquivo de entrada com se fossem comandos.
- Crie um arquivo com as seguintes linhas e salve-o como entrada.txt

```
;Esta linha não deve ser impressa.
>Esta linha deve ser impressa alinhada a direita
*Esta linha deve ser centralizada
Uma linha normal
Outra linha normal
```

- Criaremos um programa na qual as linhas cujo primeiro caractere é igual a ; serão ignoradas; as com > serão impressas alinhadas à direita; as com <, alinhadas à esquerda; e as com * serão centralizadas.

Exemplo: Processamento de um arquivo

- A seguir temos o programa que faz o processamento do arquivo anterior

```
LARGURA = 79
entrada = open('entrada.txt')

for linha in entrada.readlines():
    if linha[0]==';':
        continue
    elif linha[0]=='>':
        print(linha[1:].rjust(LARGURA))
    elif linha[0]=='*':
        print(linha[1:].center(LARGURA))
    else:
        print(linha)

entrada.close()
```

Outro exemplo de processamento de um arquivo

- Neste exemplo, vamos ler um arquivo de texto contendo informações sobre **funcionários** e **imprimir** na tela as informações de cada funcionário. Em seguida, vamos criar um novo arquivo de texto adicionando um salário fictício ao funcionário (idade*1000).
- Considere que o arquivo de entrada **funcionarios.txt** possui a seguinte estrutura:

```
João Silva,30,Masculino
Maria Santos,25,Feminino
Pedro Oliveira,45,Masculino
```

Outro exemplo de processamento de um arquivo(2)

```
# Abre o arquivo de entrada
with open("funcionarios.txt", "r") as arquivo_entrada:
    # Lê todas as linhas do arquivo
    linhas = arquivo_entrada.readlines()

    # Itera sobre as linhas
    for linha in linhas:
        # Separa as informações de cada funcionário
        funcionario = linha.strip().split(",")

        # Imprime as informações na tela
        print(f"Nome: {funcionario[0]}")
        print(f"Idade: {funcionario[1]}")
        print(f"Gênero: {funcionario[2]}")
        print()
```

Outro exemplo de processamento de um arquivo(3)

```
    # Cria um novo arquivo de saída
with open("funcionarios_com_salario.txt", "w") as arquivo_saida:
    # Itera sobre as linhas do arquivo de entrada novamente
    for linha in linhas:
        # Separa as informações de cada funcionário
        funcionario = linha.strip().split(",")

        # Calcula um salário fictício para o funcionário
        salario = int(funcionario[1]) * 1000

        # Escreve as informações do funcionário e o salário no novo arquivo
        arquivo_saida.write(f"{funcionario[0]}, {funcionario[1]}, {funcionario[2]}, {salario}\n")
```

Outro exemplo de processamento de um arquivo(4)

- Ao executar o programa, a saída:

```
Nome: João Silva
Idade: 30
Gênero: Masculino

Nome: Maria Santos
Idade: 25
Gênero: Feminino

Nome: Pedro Oliveira
Idade: 45
Gênero: Masculino
```

- O conteúdo do arquivo **funcionarios_com_salario.txt** seria:

```
João Silva, 30, Masculino, 30000
Maria Santos, 25, Feminino, 25000
Pedro Oliveira, 45, Masculino, 45000
```

Geração de HTML

- Toda página web é escrita em uma linguagem de marcação chamada **HTML** (Hypertext Mark-up Language – Linguagem de Marcação de Hipertexto).
- Como o formato **HTML** é definido apenas com **texto simples**, ou seja, sem caracteres especiais de controle, ele utiliza **marcações**, que são sequências especiais de texto delimitado pelos caracteres de menor (**<**) e maior (**>**).
- Essas sequências são chamadas de **tags** e podem iniciar ou finalizar um elemento.

Geração de HTML

- O elemento de mais alto nível de um documento **HTML** é chamado de <**html**>.
- Escreveremos nossas páginas web entre as tags **<html>** e **</html>,** onde a primeira marca o início do documento; e a segunda, seu fim.
- Exemplo, considere o seguinte arquivo de texto com extensão .html:

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
<meta charset="utf-8">
<title>Título da pagina</title>
</head>
<body>
Olá!
</body>
</html>
```

Pode escrever essa página usando um editor de textos. Grave o arquivo com o nome de ola.html. Abra em seu browser (navegador) de internet preferido: FireFox, Chrome, Opera etc.

Geração de HTML: Exemplo

- Execute o seguinte programa para gerar um arquivo html

```
pagina = open('pagina.html','w', encoding='utf-8')
pagina.write('<!DOCTYPE html>\n')
pagina.write('<html lang="pt-BR">\n')
pagina.write('<head>\n')
pagina.write('<meta charset="utf-8">\n')
pagina.write('<title>Título da pagina</title>\n')
pagina.write('</head>\n')
pagina.write('<body>\n')
pagina.write('Olá!')

for l in range(10):
    pagina.write('<p>%d</p>\n' % l)

pagina.write('</body>\n')
pagina.write('</html>\n')
pagina.close()
```

Geração de HTML: Exemplo – versão aprimorada

- Uso de aspas triplas para escrever as strings

```
pagina = open('pagina2.html','w', encoding='utf-8')
pagina.write('''
<!DOCTYPE html>
<html lang="pt-BR">
<head>
<meta charset="utf-8">
<title>Título da Página</title>
</head>
<body>
Olá!
''')
for l in range(10):
    pagina.write("<p>%d</p>\n" % l)
pagina.write('''
</body>
</html>
''')
pagina.close()
```