MAC2166 - Introdução à Computação

Prof. Dr. Helder Oliveira

Agenda

- Manipulação de Arquivos Texto
- Funções e métodos básicos

Arquivos

- Até agora, programas manipulavam dados digitados pelos usuários. Outra forma de entrada de dados usada com frequência em programas é a leitura de dados gravados em um arquivo.

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 nativos, arquivos compactados e arquivos de imagens.

Arquivo de Texto

- Para trabalharmos com um arquivo 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.
- **open(nome_arquivo, modo)**:
  - 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 de 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/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 de Texto

- Exemplo:

```
1 arq = open ("arquivo1.txt", "r")
2 # abrindo o arquivo arquivo1.txt com modo leitura
3 arq = open ("arquivo2.txt","w")
4 # abrindo o arquivo arquivo2.txt com modo escrita
5 arq = open ("arquivo3.txt","r+")
6 # abrindo o arquivo arquivo3.txt com modo leitura/escrita
7 arq = open ("arquivo4.txt","a")
8 # abrindo o arquivo arquivo4.txt com modo anexação
```

Arquivo de Texto

- Exemplo:

```
1 arq = open ("MAC2166/aula.txt","r")
2 # abrindo o arquivo teste.txt no diretório MAC2166
3 # usando modo de leitura
4 arq = open ("arqs/arquivo.log","r+")
5 # abrindo o arquivo arquivo.log no diretório arqs
6 # usando modo de leitura/escrita
```

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 aula.txt com o seguinte conteúdo:

```
MAC2166
USP - Python
```

Arquivo Texto - Leitura

- Lendo o arquivo aula.txt:

```
1 arq = open ("aula.txt","r")
2 texto = arq.read ()
3 print (texto , end = "")
4 #MAC2166
5 #USP - Python
```

- Lendo os 7 primeiros caracteres do arquivo aula.txt:

```
1 arq = open ("aula.txt","r")
2 texto = arq.read (7)
3 print (texto)
4 #MAC2166
```

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:

```
1  #Abre o arquivo chamado "aula.txt" no modo leitura ("r")
2  arq = open ("aula.txt","r")
3  #Lê o primeiro caractere do arquivo.
4  c = arq.read (1)
5  texto = c
6  while c:
7      c = arq.read (1)
8      texto = texto + c
9  print (texto , end = "")
10 #MAC2166
11 #USP - 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 no modo de leitura (r) ou leitura/escrita (r+).

Arquivo Texto - Leitura

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

```
1 arq = open ("aula.txt","r")
2 linha = arq.readline ()
3 while linha:
4     print (linha , end = "")
5     linha = arq.readline ()
6 #MAC2166
7 #USP - Python
```

Arquivo Texto - Leitura

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

```
1 arq = open ("aula.txt","r")
2 for linha in arq:
3     print (linha , end = "")
4 #MAC2166
5 #USP - 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 aula.txt duas vezes:

```
1  arq = open ("aula.txt","r")
2  linha = arq.readline ()
3  print (linha , end = "")
4  #MAC2166
5  arq.seek (0) # Voltando para o início do arquivo
6  linha = arq.readline ()
7  print (linha , end = "")
8  #MAC2166
9  linha = arq.readline ()
10 print (linha , end = "")
11 #USP - Python
```

Arquivo Texto - Leitura

- Avançando e retrocedendo num arquivo:

```
1  arq = open ("aula.txt","r")
2  linha = arq.readline ()
3  print (linha , end = "")
4  #MAC2166
5
6  #retorna a posição atual do cursor no arquivo
7  print ("Posição =", arq.tell ())
8
9  #recua o cursor 3 bytes no arquivo para ler a partir desse ponto.
10 arq.seek(arq.tell () - 3)
11 #Lê a linha a partir da nova posição do cursor
12 linha = arq.readline ()
13 print (linha , end = "")
14 #Imprime a posição atual do cursor após a segunda leitura.
15 print ("Posição =", arq.tell ())
```

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 - Close

- 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 aula2.txt:

```
1  arq = open ("aula2.txt", "w")
2  arq.write("Hello World!\n")
3  arq.write("Hello World!\n")
4  arq.close ()
5
6  arq = open ("aula2.txt", "r")
7  texto = arq.read ()
8  arq.close ()
9  print (texto , end = "")
10 # Hello World!
11 # Hello World!
```

Arquivo Texto - Escrita

- Adicionando mais dados no arquivo aula2.txt:

```
1  arq = open ("aula2.txt", "a")
2  arq.write("MAC2166\n")
3  arq.write("USP - Python\n")
4  arq.close ()
5
6  arq = open ("aula2.txt", "r")
7  texto = arq.read ()
8  arq.close ()
9
10 print (texto , end = "")
11 # Hello World!
12 # Hello World!
13
14 #MAC2166
15 #USP - 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 ("aula2.txt","w")
2  print ("Utilizando a função print", file = arq)
3  arq.close ()
4
5  arq = open ("aula2.txt", "r")
6  texto = arq.read ()
7  arq.close ()
8  print (texto)
9  # Utilizando a função print
```

try-except-else

- Essa estrutura é usada para **tratar exceções** — ou seja, lidar com erros que podem acontecer durante a execução do programa, evitando que ele pare abruptamente.
  - **try**: Aqui você coloca o código que pode gerar uma exceção (erro). O Python vai tentar executar esse bloco.
  - **except**: Se alguma exceção ocorrer dentro do try, o código dentro do except será executado. Ou seja, é o tratamento do erro.
  - **else**: Se não acontecer nenhum erro dentro do try, o código dentro do else será executado.

try-except-else

- **try:** — tenta abrir o arquivo que o usuário digitou.
- **except:** — se o arquivo não existir, ou não puder ser aberto, imprime uma mensagem dizendo que não conseguiu abrir.
- **else:** — só executa o cálculo das médias se o arquivo foi aberto com sucesso (ou seja, não houve erro).

```
1  def ler_arquivo(nome_arquivo):
2      try:
3          arquivo = open(nome_arquivo, 'r')   # tenta abrir o arquivo
4      except FileNotFoundError:
5          print("Arquivo não encontrado:", nome_arquivo)  # erro específico para arquivo não existir
6      else:
7          conteudo = arquivo.read()  # lê o conteúdo do arquivo
8          print("Conteúdo do arquivo:")
9          print(conteudo)
10         arquivo.close()  # fecha o arquivo
```

Exercício

- Crie um arquivo chamado nomes.txt e escreva os nomes: Ana, Bruno e Carlos em linhas separadas.

Exercício

- Crie um arquivo chamado nomes.txt e escreva os nomes: Ana, Bruno e Carlos em linhas separadas.

```
1  # abre para escrita (cria se não existir)
2  arquivo = open('nomes.txt', 'w')
3  arquivo.write('Ana\n')
4  arquivo.write('Bruno\n')
5  arquivo.write('Carlos\n')
6  arquivo.close()  # sempre fecha o arquivo
```

Exercício

- Leia o arquivo nomes.txt criado no exercício anterior e imprima cada nome na tela.

Exercício

- Leia o arquivo nomes.txt criado no exercício anterior e imprima cada nome na tela.

```
try:
    arquivo = open('nomes.txt', 'r')  # abre para leitura
except:
    print("Não consegui abrir o arquivo nomes.txt")
else:
    for linha in arquivo:
        print(linha.strip())  # remove \n e imprime
    arquivo.close()
```

Exercício

- Adicione os nomes Daniel e Eduarda ao final do arquivo nomes.txt sem apagar o conteúdo anterior.

Exercício

- Adicione os nomes Daniel e Eduarda ao final do arquivo nomes.txt sem apagar o conteúdo anterior.

```
1  # abre para acrescentar no final
2  arquivo = open('nomes.txt', 'a')
3  arquivo.write('Daniel\n')
4  arquivo.write('Eduarda\n')
5  arquivo.close()
```

Exercício

- Leia o arquivo nomes.txt e mostre quantos nomes (linhas) existem no arquivo.

Exercício

- Leia o arquivo nomes.txt e mostre quantos nomes (linhas) existem no arquivo.

```
1 try:
2     arquivo = open('nomes.txt', 'r')
3 except:
4     print("Não consegui abrir o arquivo nomes.txt")
5 else:
6     linhas = arquivo.readlines()
7     print(f"O arquivo possui {len(linhas)} nomes.")
8     arquivo.close()
```

Exercício

- Crie um arquivo chamado numeros.txt com os números: 10, 20, 30, 40, 50. Depois, leia este arquivo, some todos os números e imprima o resultado.

Exercício

- Crie um arquivo chamado numeros.txt com os números: 10, 20, 30, 40, 50. Depois, leia este arquivo, some todos os números e imprima o resultado.

```python
# Criar arquivo com números
arquivo = open('numeros.txt', 'w')
for numero in [10, 20, 30, 40, 50]:
    arquivo.write(str(numero) + '\n')
arquivo.close()

# Ler arquivo e somar números
try:
    arquivo = open('numeros.txt', 'r')
except:
    print("Não consegui abrir o arquivo numeros.txt")
else:
    soma = 0
    for linha in arquivo:
        soma += int(linha.strip())
    print('Soma dos números:', soma)
    arquivo.close()
```

Exercício

- Crie um arquivo alunos.csv com as colunas: nome, idade, nota. Adicione os seguintes dados:

nome	idade	nota
Ana	20	8.5
Bruno	22	7.0
Carlos	19	9.0

Exercício

- Crie um arquivo alunos.csv com as colunas: nome, idade, nota. Adicione os seguintes dados:

nome	idade	nota
Ana	20	8.5
Bruno	22	7.0
Carlos	19	9.0

```
1 arquivo = open('alunos.csv', 'w')
2 arquivo.write('nome,idade,nota\n')
3 arquivo.write('Ana,20,8.5\n')
4 arquivo.write('Bruno,22,7.0\n')
5 arquivo.write('Carlos,19,9.0\n')
6 arquivo.close()
```

Exercício

- Leia o arquivo alunos.csv e imprima os alunos com nota >= 7.5

Exercício

- Leia o arquivo alunos.csv e imprima os alunos com nota >= 7.5

```python
try:
    arquivo = open('alunos.csv', 'r')
except:
    print("Não consegui abrir o arquivo alunos.csv")
else:
    primeira_linha = True
    for linha in arquivo:
        if primeira_linha:
            primeira_linha = False  # pula cabeçalho
            continue
        dados = linha.strip().split(',')  # separa por vírgula
        nome = dados[0]
        nota = float(dados[2])
        if nota >= 7.5:
            print(f"Aluno: {nome}, Nota: {nota}")
    arquivo.close()
```

Exercício

- Atualize a nota do Bruno para 8.0

Exercício

- Atualize a nota do Bruno para 8.0

```python
try:
    arquivo = open('alunos.csv', 'r+')
except:
    print("Não consegui abrir o arquivo alunos.csv")
else:
    while True:
        pos = arquivo.tell()          # salva posição do início da linha
        linha = arquivo.readline()
        if not linha:
            break                     # chegou ao fim do arquivo
        dados = linha.strip().split(',')
        if len(dados) < 3:
            continue                  # linha inválida ou cabeçalho
        if dados[0] == 'Bruno':
            # Atualiza a nota para 8.0
            dados[2] = '8.0'
            # Monta a linha modificada, com mesmo tamanho (incluindo \n)
            nova_linha = ','.join(dados)
            # Preenche com espaços à direita se necessário para não alterar tamanho
            if len(nova_linha) < len(linha.strip()):
                nova_linha = nova_linha.ljust(len(linha.strip()))
            nova_linha += '\n'  # adiciona a quebra de linha
            arquivo.seek(pos)           # volta para início da linha
            arquivo.write(nova_linha)  # sobrescreve a linha
            break                     # terminou a alteração
    arquivo.close()
```

Exercício

- Escreva um programa que lê um arquivo e o imprime com as linhas numeradas.

Exercício

- Escreva um programa que lê um arquivo e o imprime com as linhas numeradas.

```
1  def numera_linha(nome_arq):
2      arquivo = open(nome_arq, 'r')          #open(nome_do_arquivo, 'r', encoding='utf8')
3      nlin = 1
4      for linha in arquivo:
5          print(nlin, ':', linha)
6          nlin += 1
7      arquivo.close()
8
9  def main():
10     nome_do_arquivo = input('Nome do arquivo: ')
11     numera_linha(nome_do_arquivo)
12
13 main()
```

Exercício

- Escreva um programa que lê um arquivo e o imprime com as linhas numeradas.
  - A solução não ficou muito boa.
  - As linhas ficam separadas por uma linha em branco e os números não ficaram alinhados.
  - O problema das linhas em branco acontece porque quando lemos cada linha do arquivo, ela já vem com uma quebra de linha no final.
  - E quando usamos o comando print exibir a linha, ele inclui uma outra quebra de linha.
  - Então, para resolver o problema, podemos usar o print passando para ele o parâmetro end=
  - A chamada linha.rstrip() devolve uma cópia da string linha sem os caracteres em branco que ela tiver em seu final.

Exercício

- Escreva um programa que leia um arquivo onde cada linha contém o nome e as notas de um estudante e, para cada estudante, calcule e imprima a média das notas. Ao final, o programa deve imprimir também a média da turma.
- Exemplo: para o arquivo notas.txt com:

```
jose 10 15 20 30 40
pedro 23 16 19 22
suzana 8 22 17 14 32 17 24 21 2 9 11 17
gisela 12 28 21 45 26 10
joao 14 32 25 16 89
```

```
Digite o nome do arquivo: notas.txt
Aluno: jose      Média = 23.000000
Aluno: pedro      Média = 20.000000
Aluno: suzana      Média = 16.166667
Aluno: gisela      Média = 23.666667
Aluno: joao      Média = 35.200000
Média da turma = 23.606667
```

Exercício

```python
def main():
    nome = input("Digite o nome do arquivo: ")  # Nome do arquivo
    try:
        arquivo = open(nome, 'r')   # Tenta abrir o arquivo para leitura
    except:
        # Caso não consiga abrir o arquivo (por exemplo, arquivo não existe),
        print("Não consegui abrir o arquivo %s" %(nome))
    else:
        total = 0          # Variável para acumular a soma das médias
        cont_alunos = 0    # Contar quantos alunos foram processados
        # Para cada linha no arquivo, que representa um aluno e suas notas:
        for linha in arquivo:
            dados = linha.split()  # Divide a linha em palavras
            notas = dados[1:]      # Seleciona as notas (exceto o nome)
            soma = 0               # Inicializa a soma das notas daquele aluno
            for n in notas:
                soma += float(n)   # Converte a nota para número decimal e soma
            media = soma / len(notas)  # Calcula a média das notas do aluno
            # Exibe o nome do aluno e a média calculada
            print("Aluno: %s \t Média = %f"%(dados[0], media))

            total += media         # Soma a média daquele aluno ao total geral
            cont_alunos += 1       # Incrementa os alunos processados
        # Após processar todos os alunos, calcula e exibe a média geral
        print("Média da turma = %f"%(total / cont_alunos))
        arquivo.close()  # Fecha o arquivo aberto para liberar o recurso

# Chama a função principal para executar o programa
main()
```

Exercício – Parte 1

- Dado um número inteiro ímpar n > 0, faça uma função molduras_concentricas(n, v1, v2) que gera uma matriz quadrada nxn preenchida com um padrão de molduras concêntricas, alternando entre os dois valores fornecidos (v1 e v2) e iniciando com o valor v1 na posição central da matriz, tal como apresentado nos exemplos abaixo.

Para n=7, v1=0, v2=1:

```
[[1, 1, 1, 1, 1, 1, 1],
 [1, 0, 0, 0, 0, 0, 1],
 [1, 0, 1, 1, 1, 0, 1],
 [1, 0, 1, 0, 1, 0, 1],
 [1, 0, 1, 1, 1, 0, 1],
 [1, 0, 0, 0, 0, 0, 1],
 [1, 1, 1, 1, 1, 1, 1]]
```

Para n=5, v1=0, v2=1:

```
[[0, 0, 0, 0, 0],
 [0, 1, 1, 1, 0],
 [0, 1, 0, 1, 0],
 [0, 1, 1, 1, 0],
 [0, 0, 0, 0, 0]]
```

Exercício – Parte 1 - Solução

```python
#Cria matriz com m linhas e n colunas
def cria_matriz(m, n, valor):
    M = []
    for i in range(m):
        linha = []
        for j in range(n):
            linha.append(valor)
        M.append(linha)
    return M
```

```python
def molduras_concentricas(n, v1, v2):
    M = cria_matriz(n, n, 0)
    for i in range(n):
        for j in range(n):
            #dh = distancia horizontal ao centro
            dh = n//2 - j
            if dh < 0:
                dh = -dh
            #dv = distancia vertical ao centro
            dv = n//2 - i
            if dv < 0:
                dv = -dv
            if dh > dv:
                if dh % 2 == 0:
                    M[i][j] = v1
                else:
                    M[i][j] = v2
            else:
                if dv % 2 == 0:
                    M[i][j] = v1
                else:
                    M[i][j] = v2
    return M
```

Exercício – Parte 2

- Faça um programa que gera uma imagem em tons de cinza no formato PGM do padrão gerado no item anterior, usando zero (preto) para v1 e 255 (branco) para v2. Uma explicação sobre o formato de imagem PGM pode ser encontrada em https://www.ime.usp.br/~mac2166/2025/arquivos/pgm_ppm.pdf

Exercício – Parte 2 - solução

```
37 def grava_PGM(M):
38     arquivo = open("fig01.pgm", 'w')
39     arquivo.write("P2\n")
40     m = len(M)
41     n = len(M[0])
42     arquivo.write("%d %d\n"%(n,m))
43     arquivo.write("255\n")
44     for i in range(m):
45         for j in range(n):
46             arquivo.write(" %3d"%(M[i][j]))
47         arquivo.write("\n")
48     arquivo.close()
49
50
51 def main():
52     n = int(input("Digite n: "))
53     M = molduras_concentricas(n, 0, 255)
54     grava_PGM(M)
55
56 main()
```

Referencias

- Slides baseados em:
  - Zanoni Dias, MC102- Algoritmos e Programação de Computadores, 2020.

Dúvidas