***ANOTAÇÕES SOBRE DISCIPLINA***

***Jogos Digitais – Universidade Estácio de Sá (UNESA)***

***Disciplina: Desenvolvimento Rápido de Aplicações em Python***

***Discente: Breno Freitas Aguirres***

**Manipulação de dados em Arquivos**

1. *Funções de Manipulação de Arquivos.*

Operações Básicas – Abrir, Fechar, Ler, Escrever.

* arquivo = open(caminho, modo\_leitura)

Exemplo:

Absoluto:

open(“C:\Downloads\arquivo.txt”) – utilizado em ambientes MS Windows.  
open(“/home/usuario/arquivo.txt”) – utilizado em ambientes Linux.

Relativo:

open(“arquivo.txt”), para os casos em que o arquivo está no mesmo diretório do **script**.  
open(“../arquivo.txt”), para os casos em que o arquivo está no diretório acima do **script**.

* import os
* print(os.path.relpath(arquivo.name))
* print(os.path.abspath(arquivo.name))
* print(arquivo)

Desmembrando essa saída, temos:

O tipo do objeto, TextIOWrapper, que trata de arquivos de texto.

O nome do arquivo, name='dados.txt'.

O modo de acesso ao arquivo, mode='r'.

A codificação do arquivo, encoding='cp1252'.].

Modos de acesso ao arquivo:

- Modo padrão é leitura (‘r’)

- Adicionar + (leitura e escrita) -> r+

- Abrir arquivo binário (imagem p.e.) -> rb

* ‘r’ – leitura
* ‘w’ – escrita, trunca o arquivo primeiro
* ‘x’ – cria arquivo para escrita e falha caso o arquivo exista
* ‘a’ – abre para escrita, acrescentando conteúdo ao final do arquivo, caso exista.
* ‘b’ – modo binário
* ‘t’ – modo texto
* ‘+’ – abre para atualização (leitura e escrita).

Atributos do obj do tipo arquivo:

* print(“Nome do arquivo: ”, arquivo.name)
* print(“Modo do arquivo: ”, arquivo.mode)
* print(“Arquivo está fechado? ”, arquivo.closed)

Fechando arquivos:

Precisamos fechar o arquivo para liberar a memória alocada pelo interpretador e liberar seu uso para outros programas.

* arquivo.close()

Lendo o conteúdo de um arquivo:

* arquivo.read() -> retorna todo o conteúdo em 1 string.
* arquivo.readline() -> retorna uma linha do arquivo, incluindo \n e avança o cursor para a próxima linha.
* arquivo.readlines() -> retorna uma lista em que cada item da lista é uma linha do arquivo.

Iterando sobre um arquivo:

* for linha in arquivo:
  + print(repr(linha))

Quando precisamos abrir um arquivo muito grande, é inviável utilizar os métodos read e readlines, pois eles retornam todo o conteúdo do arquivo de uma só vez, seja na forma de string, seja na forma de lista. Isso pode consumir todos os recursos do computador, travando seu programa.

Após utilizar qualquer um dos métodos para leitura do arquivo apresentado, não podemos utilizá-los novamente. Isso acontece porque o cursor estará posicionado ao final do arquivo, e as chamadas aos métodos read, readline ou readlines retornarão vazias.

Retornando ao inicio:

* arquivo.seek(n) -> n é o número da linha.

Escrevendo em um arquivo:

* arquivo = open(“dados.txt”, “w”)
* arquivo.write(texto)
* arquivo.writelines(iteravel)
* Exemplo:
  + arquivo.write(“Olá Mundo!”)
  + lines = [“Conteúdo da primeira linha”, “\nConteúdo da segunda linha”]
  + arquivo.writelines(lines)

Python não insere quebra de linha automaticamente, é preciso fazer isso.

O modo w trunca o arquivo, então o conteúdo que existia antes será apagado.

* arquivo = open(“dados.txt”, “a”)
* arquivo.write(“Conteúdo adicional\n”)

Usando o modo append o arquivo não será truncado.

Uma boa prática é utilizar a keyword “with” para abrir arquivos, assim não precisamos nos lembrar de fechar o arquivo explicitamente após o uso.

* with open(caminho, modo) as nome\_variavel:

//seu código ficará aqui.

//após sair da identação, o arquivo será fechado

1. *Funções de Manipulação de Strings.*

Métodos de manipulação de Strings.

Método Strip

* linha.strip() -> remove caracteres em branco ou outros especificados do início e final de uma string.
* linha.lstrip() -> apenas do início
* linha.rstrip() -> apenas do final
* Exemplo:
  + contador = 0
  + for linha in arquivo:
    - if linha.strip():
      * contador += 1
  + print(contador)
  + # esse script vai contar a qtd de linhas não vazias em um arquivo

Métodos Count e Split

* minha\_string.count(palavra) -> retorna a qtd de vezes em que essa palavra aparece na string.

OBS: esse método também conta as palavras que contêm parte da string passada como argumento.

Para contornar esse problema podemos utilizar o método split.

* separador = ‘ ‘ -> caractere de espaço (poderia ser um caractere ou uma string, caso não coloque separador vai separar por espaço).
* lista\_palavras = minha\_string.split(separador)
* for p in lista\_palavras:
  + if p == palavra\_para\_contar:
    - contador += 1

A string utilizada como separador **não** aparecerá em nenhum elemento da lista retornada.

Método Join

* conector = “, ” -> ou qualquer outra string
* string\_final = conector.join(iteravel)
* string\_final = “\n”.join(iterável)

Cria uma string com base em um iterável de strings com um determinado conector no meio.

Formatação de strings.

F-Strings

* minha\_string = f”Olá Mundo e {expr}” # expr pode se tratar de variáveis, literais (2 > 3 ou [1, 2] p.e.), e métodos também.
* from datetime import datetime
* frutas = [‘Jabuticaba’, ‘Laranja’, ‘Uva’, ‘Banana’]
* for fruta in frutas:
  + minha\_fruta = f”Nome: {fruta:12} – Letras: {len(fruta):3}”
* pi = 3.1415
* meu\_num = f”O número pi é: {pi:.1f}”
* meu\_num\_deslocado = f”Pi deslocado: {pi:6.1f}”
* meu\_num\_preciso = f”Pi preciso: {pi:.4f}”
* data = datetime.now()
* minha\_data = f”A data de hoje é {data}”
* minha\_data\_formatada = f”A data de hoje formatada é {data:%d/%m/%y}”

1. *Tratamento de exceções e outras operações.*

Try/Except:

* try:
  + # código que pode gerar exceção
* except Erro1 as erro:
  + # o que fazer quando encontrar Erro1
* except Erro2 as erro:
  + # o que fazer quando encontrar Erro2

Tipos comuns de exceções:

* PermissionError, FileExistsError, FileNotFoundError – essas exceções herdam de OSError que herda de Exception.

Operações adicionais em arquivos:

* import os
* os.remove(path)
* os.rmdir(path)
* Exception IsADirectoryError (remove só apaga arquivo)
* os.rename(pathorigin, finalpath)
* os.replace(pathorigin, finalpath) -> rename mas sobrescreve se houver um nome igual ao final path.

Manipulação de Diretórios.

* os.mkdir(path)
* os.rmdir(path)
* except OSError as erro:
  + if erro.errno == errno.ENDTEMPTY: # confere qual o tipo do erro genérico OSError.
    - print(“Diretório não está vazio”)

Listando conteúdo do diretório:

* os.scandir(path) # retorna um iterator do tipo os.DirEntry que pode ser pasta ou arquivo.
* Atributos e métodos do os.DirEntry:
  + Name, Path, is\_dir(), is\_file(), stat() # esse ultimo tem atributos como tamanho