Fundamentos de Programação

Leitura e Escrita em ARQUIVOS

Aula 16

Post It da Aula Passada

Funções (Teoria)

- “Sub algoritmos” dentro de algoritmos
- Finalidade
  - Melhorar a organização do código
  - Evitar repetição de código

Funções (Prática)

- **Estrutura**: *def*, nome, argumentos e operações!
  - **Nome**: mesmas regras da nomeação de variáveis
  - **Argumentos**: entre parentêses
  - **Operações**: bloco indentado

Arquivos

Até esse momento estávamos **utilizando apenas a memória principal do computador** para armazenar dados

O “problema” (característica) em relação a isso é que não persistimos os nossos dados... ou seja, toda a vez que executarmos um programa as variáveis serão inicializadas com os valores padrão, não havendo referência aos dados da última execução

Para mantermos os dados de uma execução do programa para a outra **podemos utilizar a memória secundária do computador**

Arquivos

Existem várias maneiras de persistir nossos dados!

As alternativas mais comerciais consistem em bancos de dados, relacionais ou não, que fornecem sistemas de gerenciamento complexos e completos

Entretanto, podemos optar por uma alternativa simplificada para salvar nossos dados no computador:

ARQUIVOS

Arquivos

Mas... o que são arquivos?

“Repositório ou coleção de qualquer espécie de documentos ou outros materiais, como manuscritos, fotos, correspondência, etc., importantes para instituições civis ou governamentais, ou que têm valor cultural, estratégico, histórico, informativo, etc.” — **Michaelis**

Em computação, a ideia é a mesma! Porém, mapeada para um contexto digital

Arquivos

“Seção de dados de um computador, como lista de endereços, textos, contas de clientes, etc., em forma de registros individuais que podem conter dados, caracteres, dígitos ou gráficos” — **Michaelis**

Em um arquivo físico, **podemos escrever qualquer tipo de dado**...
O mesmo ocorre em arquivo digital!

.txt; .bmp; .zip; .py

Arquivos

A linguagem de programação Python fornece uma forma nativa e simplificada de ler e escrever arquivos

O primeiro **conceito que precisamos entender para manipularmos arquivos em Python (e em qualquer outra linguagem) é o de caminho do arquivo** (vinculado ao sistema de arquivos do computador)

Todo o computador apresenta uma pasta raiz, geralmente associada ao caminho:

/

Arquivos

A partir da pasta raiz do programa, um sistema de arquivos se organiza como uma hierarquia de diretórios. Navegamos através desses diretórios utilizando o símbolo “/”, que indica cada passo do caminho a ser percorrido:

/user/
/user/desktop/
/user/documents/

Se estamos navegando entre diretórios no terminal do computador, geralmente utilizamos o comando **cd**

Arquivos

Diretórios, no que lhes concernem, podem armazenar diferentes arquivos. Geralmente esses arquivos são lidos (da maneira correta) por programas específicos

Alguns arquivos são “especiais”. Ou seja, além de armazenar dados, esses arquivos podem ser, por exemplo, executados

A extensão de um arquivo facilita a previsão dos dados que existem no mesmo. Entretanto, apenas a extensão não garante o conteúdo de um arquivo

Arquivos

/user/config.cfg

Um arquivo de **configuração de sistema**, geralmente não é feito para ser lido por humanos, mas provavelmente pode ser aberto por um editor de texto

/user/documents/informacoes.txt

Um **arquivo de texto** que provavelmente pode ser aberto por qualquer editor de texto disponível no computador

/user/images/foto.bmp

Um **arquivo que contém matrizes de cores que representam uma imagem**. Poderá ser aberto por um leitor de imagens

Abrindo Arquivos

Em Python, as **funções de manipulação básica de arquivos são providas nativamente**

A função de arquivos que permite a utilização das demais é denominada open():

```
open(caminho, modo)
```

Caminho (obrigatório): caminho da pasta atual (ou da origem) até o arquivo que deve ser aberto (com extensão, se houver)
Modo (opcional, padrão: r): modo de leitura do arquivo (w, r, a)

Abrindo Arquivos

Vamos assumir que existe um arquivo de texto em uma área de trabalho que apresenta o seguinte caminho:

/user/desktop/teste.txt

Podemos abrir esse arquivo em três modos básicos diferentes:

- w: modo de **escrita**, arquivo é criado ou aberto e limpo (se já existe)
- r: modo de **leitura**, arquivo é aberto e deve estar disponível
- a: modo de **concatenação**, arquivo é criado ou aberto para adição (se já existe)

Abrindo Arquivos

Além disso, vale ressaltar que **a abertura de um arquivo retorna como resultado um *IOWrapper***, que serve para a manipulação do mesmo (dependendo do modo de abertura, leitura ou escrita)

arquivo = open(**“/user/desktop/teste.txt”**)	#1 - Leitura
arquivo = open(**“/user/desktop/teste.txt”, “w”**)	#2 - Escrita
arquivo = open(**“/user/desktop/teste.txt”, “r”**)	#3 - Leitura
arquivo = open(**“/user/desktop/teste.txt”, “a”**)	#4 - Conc.

Leitura de Arquivos

Se um arquivo foi aberto em modo de leitura (“r”), ele pode ser naturalmente lido através da linguagem Python. Existem algumas **funções acessadas por notação ponto no *IOWrapper*** para fazer isso:

```
arquivo.read()
```

A função ***read* lê todo o conteúdo de um arquivo**, retornando o seu conteúdo como uma única *string*. A *string* retornada pode ser atribuída em uma variável e ser manipulada por qualquer operação nativa deste tipo de dado

Leitura de Arquivos

`arquivo.readline()`

A função ***readline*** **lê o conteúdo de um arquivo linha a linha**, retornando cada uma delas como uma *string* independente. Cada linha pode ser atribuída e manipulada normalmente

`arquivo.readlines()`

A função ***readlines*** **lê todo o conteúdo de um arquivo, porém linha por linha**. Assim, uma lista contendo todas as linhas do arquivo é retornada. Isso quer dizer que os elementos da lista são *strings* de cada linha do arquivo

Leitura de Arquivos

arquivo = **open(“/user/desktop/teste.txt”, “r”)**

```
dados = arquivo.read()
print(type(dados))
print(dados)
```

```
dados = arquivo.readline()
print(type(dados))
print(dados)
```

```
dados = arquivo.readlines()
print(type(dados))
print(dados)
```

Escrita em Arquivos

Arquivos abertos em **modo de escrita (w) ou concatenação (a)** não podem ser lidos. Entretanto, podemos adicionar dados neles através da operação de escrita

Para isso utilizamos a função *write* em notação ponto no *IOWrapper* obtido após a abertura do arquivo:

arquivo = **open(“/user/desktop/teste.txt”, “w”)**
arquivo**.write(“Uma string qualquer”)**

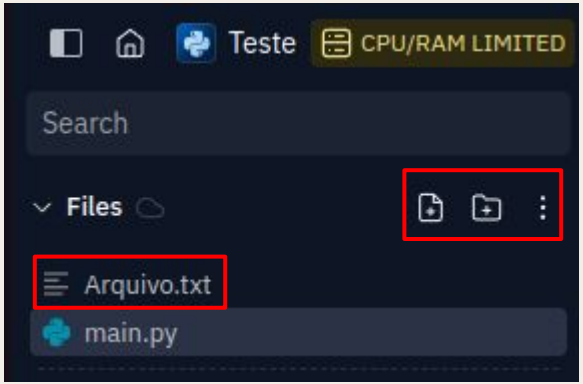
A função *write* recebe uma *string* como argumento

Arquivos no ReplIt

E no ReplIt, onde ficam armazenados os meus arquivos?

Todos os programas no ReplIt apresentam um pequeno sistema de arquivos disponível na nuvem para a utilização

Podemos gerenciar nossos arquivos **no canto esquerdo da tela**



Formatos de Arquivo

Existem formatos textuais de arquivos que facilitam a organização dos dados

Alguns são bastante simples: **.csv**
Outros são mais complexos e permitem definir estruturas de dados hierarquizadas: **.json, .yaml**

A linguagem Python apresenta **suporte nativo para csv e json**

Arquivos na Prática

Faça um programa que **implemente um jogo de par ou ímpar**, requisitando ao usuário a sua jogada a cada rodada pela entrada padrão. O computador deve gerar um número aleatório como a sua jogada

A cada rodada jogada, armazene em memória principal o ganhador da rodada

Quando o jogador parar de jogar, **gere um relatório contendo o número de vitórias do computador, o número de vitórias do usuário e o campeão geral** (aquele com mais vitórias). Esse arquivo deve ser criado e armazenado na pasta local do programa

Exercício #16

Considere o programa anterior (jogo de par ou ímpar) e faça as seguintes modificações:

- Adicione uma **opção de carregar um placar no menu do jogo**;
  - Essa opção deve ler o arquivo do último relatório do jogo na pasta do mesmo e carregar o número de vitórias do computador e do usuário nos contadores
- **Faça um *hacking mode***, você pode escolher o código para ativar ele. O *hacking mode* deve exibir na tela a próxima jogada do computador

Fundamentos de Programação

Aula 16

Obrigado e ATÉ A PRÓXIMA!

©Prof. Vinicius Fulber-Garcia - 2025