Algoritmos e Programação de Computadores

Arquivos Binários

Ref.: material original (1o S., T. KLMN). por **Profa. Sandra Avila**, Instituto de Computação (IC/ Unicamp)

Agenda

- Arquivos Binários
  - Abrindo um arquivo binário
  - Lendo e escrevendo no arquivo binário
  - Exemplo: Cadastro de alunos

Arquivos

- Arquivos podem ter os mais variados conteúdos (imagens, videos, audios, documentos, etc), mas do ponto de vista dos programas existem apenas dois tipos de arquivo:

Arquivos

- Arquivos podem ter o mais variado conteúdo (imagens, videos, audios, documentos, etc), mas do ponto de vista dos programas existem apenas dois tipos de arquivo:
  - **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.

Arquivos

- Arquivos podem ter o mais variado conteúdo (imagens, videos, audios, documentos, etc), mas do ponto de vista dos programas existem apenas dois tipos de arquivo:
  - **Arquivo binário**: Sequência de bits sujeita às convenções dos programas que o gerou, não legíveis diretamente.
  - Exemplos: arquivos executáveis, arquivos compactados, documentos do Office.

Arquivos Binários

- A motivação principal é que objetos (como inteiros, listas, dicionários) na sua representação em binário, **ocupam menos espaço na memória**, comparado com sua representação em formato texto:
  - Para representar o número na forma texto, dever ser convertido para *strings*, sendo necessários 12 bytes **(*)** para representar este número (96 bits)… **(*)** depende do coding/*hardware* do sistema…(UTF-8)
  - Sua representação binária, no entanto ocupa sempre 64 bits (ou 8 bytes). Ex. IEEE 754 std. 64-bit double precision.

Arquivos Binários

- Armazenar dados em arquivos de forma análoga à utilizada em memória permite:
  - Reduzir o tamanho do arquivo.
  - Guardar estruturas de dados mais complexas, facilitando o acesso (escrita e/ou leitura).

Abrindo um Arquivo Binário

- Assim como em arquivos texto, devemos abrir o arquivo com a função `open` e atribuir o objeto arquivo resultante para uma variável.
- Desta forma a variável estará associada ao arquivo, com métodos para leitura e escrita sobre este.

```
arquivo = open("nome do arquivo", "modo")
```

Abrindo um Arquivo Binário

```
arquivo = open("nome do arquivo", "modo")
```

modo	operador
`rb`	leitura
`wb`	escrita
`r+b`	leitura e escrita

Abrindo um Arquivo Binário

- Se um arquivo for aberto para leitura (`rb`) e não existir, a função gera uma exceção.
- Se um arquivo for aberto para escrita (`wb`) e não existir, um novo arquivo é criado. Se ele existir, é sobrescrito.
- Se um arquivo for aberto para leitura/gravação (`r+b`) e existir, ele NÃO é sobrescrito. Se o arquivo não existir, a função gera uma exceção.

Lendo e Escrevendo no Arquivo Binário

- Utilizaremos o módulo `pickle` para ler e escrever objetos para um arquivo.
- Primeiramente deve-se importar este módulo:

```
import pickle
```

- Após isso podemos escrever um objeto em arquivo com o método `pickle.dump` e podemos ler um objeto em arquivo com o método `pickle.load`.

Lendo e Escrevendo no Arquivo Binário

- Para **escrever um objeto em um arquivo binário** usamos o método `pickle.dump`.

```
pickle.dump(objeto, arquivo)
```

- `objeto`: este é o objeto a ser salvo em arquivo.
- `arquivo`: esta é a variável associada a um arquivo previamente aberto em modo binário.

Lendo e Escrevendo no Arquivo Binário

- Exemplo: O programa abaixo salva uma lista em arquivo.

```
import pickle
try:
    arquivo = open("teste.bin", "wb")
    lista = [1, 2, 3]
    pickle.dump(lista, arquivo)
    arquivo.close()
except:
        print("Problemas com o arquivo.")
```

Lendo e Escrevendo no Arquivo Binário

- Para **ler um objeto de um arquivo binário** usamos o método `pickle.load`.

```
objeto = pickle.load(arquivo)
```

- `arquivo`: esta é a variável associada a um arquivo previamente aberto em modo binário.
- O método automaticamente reconhece o tipo de objeto salvo em arquivo, carrega este para a memória e atribui para a variável `objeto`.

Lendo e Escrevendo no Arquivo Binário

- Exemplo: O programa abaixo lê a lista previamente salva em arquivo.

```
import pickle
try:
    arquivo = open("teste.bin", "rb")
    l = pickle.load(arquivo)
    print(l)
    arquivo.close()
except:
        print("Problemas com o arquivo.")
```

Exemplo: Cadastro de Alunos

- Vamos criar um programa que simula um cadastro de alunos de uma turma de MC102.
- Para representar o aluno vamos criar um dicionário com os campos nome e notas que irão armazenar, respectivamente, o nome do aluno e as notas dos labs de MC102.
- Para representar o cadastro criaremos um objeto do tipo lista que contém a lista de alunos.

Exemplo: Cadastro de Alunos

- Além disso, o programa deverá fornecer um menu de operações com as funções de inserção e remoção do aluno no cadastro, a inserção de notas e a visualização da lista de alunos.
- Os dados deverão ser salvos em arquivo antes do programa encerrar, para serem utilizados posteriormente.

Exemplo: Cadastro de Alunos

- Para representar o aluno vamos criar um dicionário com os campos nome e notas que irão armazenar, respectivamente, o nome do aluno e as notas dos labs de MC102.

```
def cadastrarAluno(cadastro, nome):
    aluno = {}
    aluno["nome"] = nome
    aluno["notas"] = []
    cadastro.append(aluno)
```

Exemplo: Cadastro de Alunos

- Além disso, o programa deverá fornecer um menu de operações com as funções de inserção e **remoção** do aluno no cadastro, a inserção de notas e a visualização da lista de alunos.

```
def excluirAluno(cadastro, nome):
    for aluno in cadastro:
        if aluno["nome"] == nome:
            cadastro.remove(aluno)
            return
    print(nome, " não encontrado!")
```

Exemplo: Cadastro de Alunos

- Além disso, o programa deverá fornecer um menu de operações com as funções de inserção e remoção do aluno no cadastro, a **inserção de notas** e a visualização da lista de alunos.

```
def inserirNotas(cadastro, nome, notas):
    for aluno in cadastro:
        if aluno["nome"] == nome:
            aluno["notas"] += notas.split()
            return
    print(nome, " não encontrado!")
```

Exemplo: Cadastro de Alunos

- Além disso, o programa deverá fornecer um menu de operações com as funções de inserção e remoção do aluno no cadastro, a inserção de notas e a **visualização da lista de alunos**.

```
def listarAlunos(cadastro):
    for aluno in cadastro:
        for chave in aluno:
            print(chave, ": ", aluno[chave])
```

```
def menuPrincipal(cadastro):
    while True:
        print("\nEscolha uma opção:\n 1- Incluir Aluno\n 2- Excluir Aluno")
        print(" 3- Incluir Notas\n 4- Listar Turma\n 5- Sair\n")
        opcao = int(input())
        if opcao == 1:
            nome = input("Digite o nome do aluno: ")
            cadastrarAluno(cadastro, nome)
        elif opcao == 2:
            nome = input("Digite o nome do aluno: ")
            excluirAluno(cadastro, nome)
        elif opcao == 3:
            nome = input("Digite o nome do aluno: ")
            notas = input("Digite as notas do aluno separados por espaço: ")
            inserirNotas(cadastro, nome,notas)
        elif opcao == 4:
            listarAlunos(cadastro)
        elif opcao == 5:
            return
        else:
            print("Opção inválida!")
```

```
import pickle

try:
    cadastro = pickle.load(open("cadastro.bin", "rb"))
    menuPrincipal(cadastro)
    print("\nSalvando cadastro...")
    pickle.dump(cadastro,open("cadastro.bin", "wb"))
except FileNotFoundError:
    print("Criando cadastro...")
    cadastro = []
    menuPrincipal(cadastro)
    pickle.dump(cadastro,open("cadastro.bin", "wb"))
except IOError:
    print("Problemas no arquivo de cadastro.")
```

Exemplo: Cadastro de Alunos

- Quando o programa é executado pela primeira vez, o cadastro em arquivo `cadastro.bin` não existe.
- Ao tentar abrir o arquivo será gerado a exceção `FileNotFoundError`, que é tratada no bloco `except FileNotFoundError`.
- Neste bloco o arquivo é criado, e após a execução do `menuPrincipal(cadastro)` o cadastro é escrito no arquivo .bin.

Exemplo: Cadastro de Alunos

- Nas demais execuções do programa o arquivo `cadastro.bin` já vai existir, então será executado o bloco `try`.
- O arquivo é aberto para escrita, `wb`, e então o arquivo .bin é carregado na memória, associando-o com a variável `cadastro`.
- Executa-se o programa normalmente com `menuPrincipal(cadastro)`.
- Após a finalização do usuário, o cadastro é atualizado (gravado).

Referências

- Os slides dessa aula foram baseados no material de MC102 dos profs. Sandra Avila, Eduardo Xavier e Marcio Pereira (IC/Unicamp).
- Livro:
  - “Beginning Python , Using Python 2.6 and Python 3.1”, por James Pyne, chapter 8: Files and Directories; Wiley Pub. Inc,, 2010.