MC-102 – Aula 25
Arquivos Binários

Prof. Luiz F. Bittencourt

Turmas QR

Instituto de Computação – Unicamp

2019

Conteúdo adaptado de slides fornecidos pelo Prof. Eduardo Xavier.

Roteiro

Motivação

- Vimos que existem dois tipos de arquivos: texto e binário.
- Objetos (como **int**, **float**, **list**, etc.), na sua representação em binário, ocupam pouco espaço na memória quando comparado com sua representação em formato texto.
  - Se representássemos 123456789.00 na forma textual, gastaríamos 12 bytes para representar este número.
  - Mas sua representação binária ocupa 64 bits (ou 8 bytes) [1].
- Armazenar dados em arquivos de forma análoga a utilizada em memória permite:
  - Reduzir o tamanho do arquivo.
  - Guardar estruturas complicadas tendo acesso simples.

[1] Depende da implementação do python utilizada

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.

```
arq = open(nome_do_arquivo, modo)
```

Abrindo um Arquivo Binário

```
arq = open(nome_do_arquivo, modo)
```

Modos de abertura de arquivo binário

modo	operações
rb	leitura
wb	escrita
r+b	leitura e escrita

- 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 sobre-escrito.
- Se um arquivo for aberto para leitura/gravação (**r+b**) e existir, este NÃO é sobreescrito. Se o arquivo não existir a função gera uma exceção.

Lendo e Escrevendo no Arquivo

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**.

Warning: The pickle module is not intended to be secure against erroneous or maliciously constructed data. Never unpickle data received from an untrusted or unauthenticated source.

Lendo e Escrevendo no Arquivo

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

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

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

O programa abaixo salva uma lista em arquivo:

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

Lendo e Escrevendo no Arquivo

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

```
var_objeto = pickle.load(var_arquivo)
```

- **var_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 **var_objeto**.

O programa abaixo lê a lista previamente salva em arquivo:

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

Cadastro de Alunos

- Vamos fazer uma aplicação que salva dados de alunos de uma certa turma.
- Teremos operações para inclusão de aluno, exclusão, etc.
- Os dados deverão ser salvos em arquivo antes do programa encerrar, para serem utilizados posteriormente.

Cadastro de Alunos

Vamos usar a classe Aluno já vista em uma aula anterior:

```
class Aluno:
    def __init__(self):
        self.nome=""
        self.notas=[]
    def printAluno(self):
        print("Nome: ", self.nome)
        print("Notas: ", self.notas)
    def incluiNota(self, nota):
        self.notas.append(nota)
```

Cadastro de Alunos

Vamos usar também a classe Cadastro, vista em uma aula anterior:

```
class Cadastro:
    def __init__(self):
        self.cadastro=[]
    def incluiAluno(self,a):
        self.cadastro.append(a)
    def excluiAluno(self, nome):
        for i in self.cadastro:
            if i.nome==nome:
                self.cadastro.remove(i)
    def printCad(self):
        print("Imprimindo Cadastro")
        for a in self.cadastro:
            a.printAluno()
```

O programa deverá ler de um arquivo um objeto do tipo Cadastro, fazer operações sobre este e, antes de encerrar, deverá salvar o cadastro atualizado no arquivo novamente.

Cadastro de Alunos

- Abaixo temos a função principal do programa que recebe um cadastro e o atualiza conforme opções do usuário.
- Todas alterações ficam em memória principal por enquanto.

```
def menuPrincipal(cadastro):
    while True:
        print("\n\nEscolha uma opção\n1-Incluir Aluno\n2-Excluir Aluno\n3-Incluir")
        print("Nota\n4-Listar Turma\n5-Sair")
        op = input()
        if(op=="1"):
            a = Aluno()
            a.nome = input("Digite nome do aluno:")
            cadastro.incluiAluno(a)
        elif(op=="2"):
            nome = input("Digite nome do aluno:")
            cadastro.excluiAluno(nome)
        elif(op=="3"):
            nome = input("Digite nome do aluno:")
            nota = float(input("Digite a nota:"))
            for a in cadastro.cadastro:
                if(a.nome == nome):
                    a.incluiNota(nota)
                    break
        elif(op=="4"):
            cadastro.printCad()
        elif(op=="5"):
            return
        else:
            print("Opção inválida")
```

Cadastro de Alunos

O programa começa com o código abaixo:

```
try:
    arq = open("cadastro.bin", "r+b")
    cadastro = pickle.load(arq)
    menuPrincipal(cadastro)
    print("salvando cadastro")
    arq.seek(0,0)
    pickle.dump(cadastro, arq)
    arq.close()
except FileNotFoundError:
    arq = open("cadastro.bin", "wb") #cria arquivo se não existir
    cadastro = Cadastro()
    menuPrincipal(cadastro)
    pickle.dump(cadastro, arq)
    arq.close()
except:
    print("Problemas no arquivo cadastro.bin")
```

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)** é feito a escrita do cadastro em arquivo.

```
try:
    arq = open("cadastro.bin", "r+b")
    ...
except FileNotFoundError:
    arq = open("cadastro.bin", "wb") #cria arquivo se não existir
    cadastro = Cadastro()
    menuPrincipal(cadastro)
    pickle.dump(cadastro, arq)
    arq.close()
except:
    print("Problemas no arquivo cadastro.bin")
```

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 leitura e escrita, **r+b**, e então é carregado o arquivo para a memória associando-o com a variável **cadastro**.
- Executa-se o programa normalmente com **menuPrincipal(cadastro)**.
- Após a finalização do usuário, voltamos ao início do arquivo com **arq.seek(0,0)** e então salvamos o **cadastro** atualizado.

```
try:
    arq = open("cadastro.bin", "r+b")
    cadastro = pickle.load(arq)
    menuPrincipal(cadastro)
    print("salvando cadastro")
    arq.seek(0,0)
    pickle.dump(cadastro, arq)
    arq.close()
except FileNotFoundError:
    ....
except:
    print("Problemas no arquivo cadastro.bin")
```

Exercício

- Como presidente da CBR (Confederação Brasileira de Rinhas), você é responsável por criar um banco de dados (arquivo binário) com os resultados da Rinha de Institutos. Você deve criar uma classe "Instituto", que contém informações sobre cada instituto (nome, quantidade de alunos, área que ocupa no campus, e qualquer outra informação que julgue pertinente como presidente da CBR) e uma classe "Rinha", que guardará informações sobre a disputa a cada ano. Para isso, a classe "Rinha"deverá conter as informações relevantes para uma disputa de um ano do torneio (e.g., ano do torneio, lista de embates e seus resultados, memes relacionados a cada embate, colocação de cada instituto, etc). Você pode criar métodos úteis para a gerência do campeonato, como cadastro de institutos, cadastro de embates e resultados, inclusão de memes, cômputo do das votações acumuladas para cada instituto, ordenação por pontos totais, etc). Como presidente da CBR, você tem total liberdade para criar mais classes e métodos de acordo com seu melhor julgamento.