

Arquivos

Vanessa Braganholo
vanessa@ic.uff.br

Aula de hoje

- ▶ Até então só consideramos teclado e monitor como mecanismos de entrada e saída



- ▶ Veremos como ler e escrever em arquivos



Motivação

- ▶ Em algumas situações é desejado ler dados de arquivos e escrever dados em arquivos
 - ▶ Não é necessário digitar via teclado os dados a cada execução do programa
 - ▶ Os resultados do programa podem ser impressos ou enviados para outras pessoas com mais facilidade
 - ▶ O estado do programa (jogo, por exemplo) pode ser salvo e recarregado em outro momento



Operações Básicas

- ▶ **Abertura do arquivo**
 - ▶ Liga uma variável do programa com o arquivo físico
 - ▶ Essa variável deve ser usada no programa para manipular o arquivo (ler e escrever no arquivo)
- ▶ **Fechamento do arquivo**
 - ▶ Encerramento da conexão da variável com o arquivo físico
- ▶ **Leitura do conteúdo do arquivo**
- ▶ **Escrita no arquivo**

Escrita de arquivos

- ▶ É muito parecido com escrita no monitor, só que precisa conectar com o arquivo antes (abrir o arquivo)
- ▶ Vamos ver um exemplo...



Exemplo: escrevendo números aleatórios **no monitor**

```
import random

def escreverNumerosAleatorios(qtdNumeros):
    for i in range(qtdNumeros):
        print(random.randint(0,100))

escreverNumerosAleatorios(100)
```



Exemplo: escrevendo números aleatórios em um arquivo

```
import random

def escreverNumerosAleatorios(qtdNumeros, nomeArquivo):
    arquivoNumeros = open(nomeArquivo, 'w')
    for i in range(qtdNumeros):
        arquivoNumeros.write(str(random.randint(0,100)))
    arquivoNumeros.close()

escreverNumerosAleatorios(100, 'aleatorios.txt')
```



Exemplo: escrevendo números aleatórios em um arquivo

O arquivo aparecerá na raiz do projeto do PyCharm

```
import random

def escreverNumerosAleatorios(qtdNumeros, nomeArquivo):
    arquivoNumeros = open(nomeArquivo, 'w')
    for i in range(qtdNumeros):
        arquivoNumeros.write(str(random.randint(0,100)))
    arquivoNumeros.close()

escreverNumerosAleatorios(100, 'aleatorios.txt')
```

Abertura do arquivo para escrita e posterior fechamento



Exemplo: escrevendo números aleatórios em um arquivo

```
import random

def escreverNumerosAleatorios(qtdNumeros, nomeArquivo):
    arquivoNumeros = open(nomeArquivo, 'w')
    for i in range(qtdNumeros):
        arquivoNumeros.write(str(random.randint(0,100))
    arquivoNumeros.close()

escreverNumerosAleatorios(100, 'aleatorios.txt')
```

Escrita no arquivo



Leitura de arquivos

- ▶ Novamente, é muito parecido com leitura do teclado, só que precisa conectar com o arquivo antes (abrir o arquivo)
- ▶ Vamos ver um exemplo...



Exemplo: lendo números **do teclado** e escrevendo a média

```
def escreverMedia(qtdNumeros):  
    soma = 0  
    for i in range(qtdNumeros):  
        num = eval(input("Digite o número:"))  
        soma += num  
    return soma/qtdNumeros  
  
escreverMedia(100)
```



Exemplo: lendo números **de um arquivo** e escrevendo a média

O arquivo será procurado na raiz do projeto do PyCharm

```
def escreverMedia(qtdNumeros, nomeArquivo):  
    arquivoNumeros = open(nomeArquivo)  
    soma = 0  
  
    for i in range(qtdNumeros):  
        num = float(arquivoNumeros.readline().strip())  
        soma += num  
    arquivoNumeros.close()  
    return soma/qtdNumeros  
  
escreverMedia(100, 'media.txt')  

```

Abertura do arquivo para leitura e posterior fechamento

Exemplo: lendo números **de um arquivo** e escrevendo a média

```
def escreverMedia(qtdNumeros, nomeArquivo):  
    arquivoNumeros = open(nomeArquivo)  
    soma = 0  
  
    for i in range(qtdNumeros):  
        num = float(arquivoNumeros.readline().strip())  
        soma += num  
    arquivoNumeros.close()  
    return soma/qtdNumeros  
  
escreverMedia(100, 'media.txt')
```

Leitura do arquivo



Exemplo: lendo números **de um arquivo** e escrevendo a média

```
def escreverMedia(qtdNumeros, nomeArquivo):  
    arquivoNumeros = open(nomeArquivo)  
    soma = 0  
  
    for i in range(qtdNumeros):  
        num = float(arquivoNumeros.readline().strip())  
        soma += num  
    arquivoNumeros.close()  
    return soma/qtdNumeros  
  
escreverMedia(100, 'media.txt')
```

**Interação no arquivo quando a
quantidade de valores no arquivo é
conhecida**



Fazendo de outra forma...

```
def escreverMedia(nomeArquivo):  
    arquivoNumeros = open(nomeArquivo)  
    soma = 0  
    qtdNumeros = 0  
    for num in arquivoNumeros:  
        num = float(num.strip())  
        soma += num  
        qtdNumeros += 1  
    arquivoNumeros.close()  
    return soma/qtdNumeros  
  
escreverMedia('media.txt')
```

Lendo todos os valores do
arquivo com o comando **for**



Agora usando while...

Lendo todos os valores do
arquivo com o comando
while

```
def escreverMedia(nomeArquivo):  
    arquivoNumeros = open(nomeArquivo)  
    soma = 0  
    qtdNumeros = 0  
    num = arquivoNumeros.readline()  
  
    while num != "":  
        num = float(num.strip())  
        soma += num  
        qtdNumeros += 1  
        num = arquivoNumeros.readline()  
    arquivoNumeros.close()  
    return soma/qtdNumeros  
  
escreverMedia('media.txt')
```



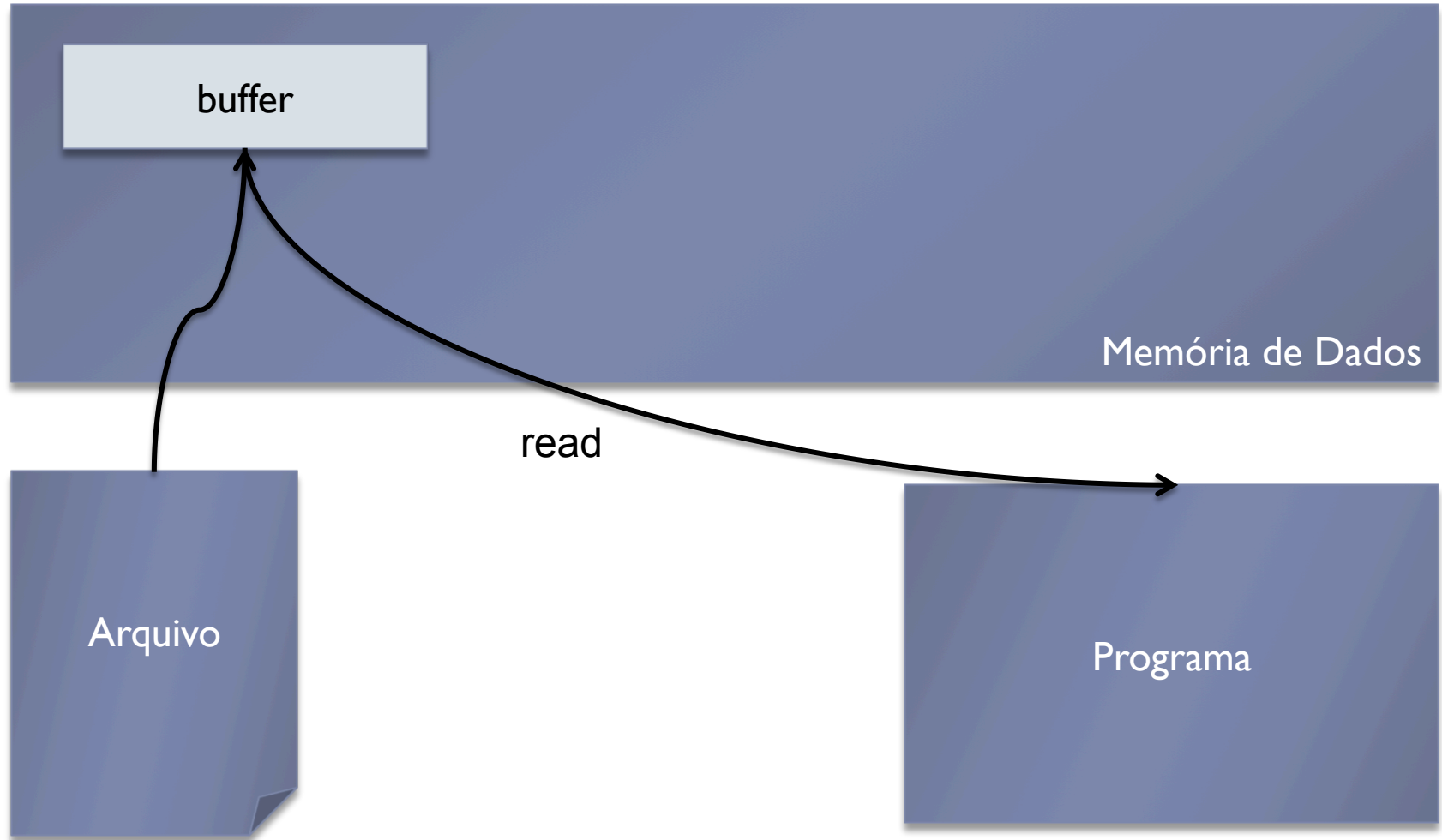
Detalhes do comando de abertura do arquivo

- ▶ `varArquivo = open(nomeArquivoFísico, modo, buffering)`
- ▶ **modo**
 - ▶ `r`: leitura (default) – o arquivo deve existir
 - ▶ `w`: escrita – conteúdo existente no arquivo será apagado
 - ▶ `b`: binário
 - ▶ `a`: escrita a partir do final do arquivo (*append*)
 - ▶ `+`: (usado com `r`) indica leitura e escrita

Detalhes do comando de abertura do arquivo

- ▶ `varArquivo = open(nomeArquivoFísico, modo, buffering)`
- ▶ **buffering (opcional)**
 - ▶ Indica se memória (buffer) é usada para acelerar operações de entrada e saída
 - ▶ 0: buffers não são usados
 - ▶ 1 (ou qq número negativo): um buffer de tamanho padrão é usado (default)
 - ▶ 2 ou maior: tamanho do buffer em bytes

Como funciona um buffer?



Detalhes do comando de escrita

- ▶ Necessário que o arquivo **não** tenha sido aberto em modo **r**
- ▶ `varArquivo.write(string)`
 - ▶ Escreve a string no arquivo
 - ▶ Devido ao uso de buffers, a escrita pode não ser feita imediatamente
 - ▶ Use o comando `flush()` ou `close()` para assegurar a escrita física no arquivo

Detalhes do comando de escrita

- ▶ Necessário que o arquivo **não** tenha sido aberto em modo **r**
- ▶ `varArquivo.writelines(sequencia)`
 - ▶ Escreve a lista (ou qualquer sequencia) de strings no arquivo, uma por uma
 - ▶ Caracteres de final de linha não são acrescentados no arquivo
 - ▶ Fica tudo numa única linha

Detalhes do comando de leitura

- ▶ Necessário que o arquivo tenha sido aberto em modo leitura ou leitura/escrita
- ▶ `varString = varArquivo.readline()`
 - ▶ Lê uma linha do arquivo e a retorna como string
- ▶ `varListaString = varArquivo.readlines()`
 - ▶ Lê o arquivo do ponto atual até o final, e retorna o conteúdo em uma lista de strings
 - ▶ Cada linha do arquivo é guardada em uma posição da lista

Detalhes do comando de leitura

- ▶ Necessário que o arquivo tenha sido aberto em modo leitura ou leitura/escrita
- ▶ `varString = varArquivo.read(numBytes)`
 - ▶ Lê **numBytes** do arquivo e os retorna em uma string
 - ▶ Se **numBytes** não for especificado, todos os bytes, desde o ponto atual do arquivo até o final do arquivo são retornados

Exemplo: Copiando dois arquivos

```
def copiaArquivo(velhoArquivo, novoArquivo):  
    f1 = open(velhoArquivo, "r")  
    f2 = open(novoArquivo, "w")  
    while True:  
        # lê a linha inteira  
        texto = f1.readline()  
        # chegou no fim do arquivo?  
        if texto == "":  
            break  
        f2.write(texto)  
    f1.close()  
    f2.close()
```

```
copiaArquivo("velho.txt", "novo.txt")
```


Sempre String?

- ▶ Para inserir valores em um arquivo, primeiro é necessário convertê-los para strings

```
>>>arq.write(str(12.3))
```

```
>>>arq.write(str([1, 2, 3]))
```

- ▶ O problema é que quando você lê esses valores de volta, você obtém uma string. O tipo original do dado foi perdido...

```
>>> arq.readline()
```

```
'12.3[1, 2, 3]'
```

Como tratar diferentes tipos?

- ▶ Usar o módulo *pickle*
 - ▶ Permite que o tipo do dado seja preservado na leitura e na escrita
- ▶ Importante 1: respeitar a estrutura do arquivo
 - ▶ O arquivo deve ser lido sempre na mesma ordem em que foi gravado
- ▶ Importante 2: abrir o arquivo no modo binário

```
>>> import pickle
```

```
>>> arq = open('teste.dat', 'wb')
```



Escrevendo com o tipo preservado

- ▶ Para escrever um valor no arquivo, preservando seu tipo, use **dump** ao invés de write/writeline
- ▶ `pickle.dump(valor_ou_var, varArquivo)`

```
import pickle
arq = open('test.dat', 'wb')
pickle.dump(12.3, arq)
pickle.dump([1, 2, 3], arq)
arq.close()
```

Lendo com o tipo preservado

- ▶ Para ler um dado do arquivo com seu tipo, use **load**

```
>>> import pickle
>>> arq = open('test.dat', 'rb')
>>> var1 = pickle.load(arq)
>>> print(var1)
12.3
>>> type(var1)
<type 'float'>
```

Continuação do exemplo anterior

```
>>> var2 = pickle.load(arq)
>>> print(var2)
[1, 2, 3]
>>> type(var2)
<type 'list'>
>>> arq.close()
```

Exemplo: Cadastro de pessoas

Concatena com os dados
preexistentes no arquivo

```
import pickle

arq = open("cadastro.dat", "ab")
nome = input("Digite o nome da pessoa: ")
pickle.dump(nome, arq)
idade = eval(input("Digite a idade: "))
pickle.dump(idade, arq)
telefone = input("Digite o telefone: ")
pickle.dump(telefone, arq)
arq.close()
```



Exemplo: Listagem das pessoas cadastradas

```
import pickle

arq = open("cadastro.dat", "rb")
while True:
    try:
        nome = pickle.load(arq)
        print(nome)
        idade = pickle.load(arq)
        print(idade)
        telefone = pickle.load(arq)
        print(telefone)
    except (EOFError, pickle.UnpicklingError):
        break
arq.close()
```



Interação com o OS

- ▶ Operações de entrada e saída são na verdade realizadas pelo sistema operacional
- ▶ O módulo **os** possui diversas variáveis e funções que ajudam um programa Python a se adequar ao sistema operacional

Funções do módulo **os**

- ▶ **os.getcwd()**
 - ▶ Retorna o diretório corrente
- ▶ **os.chdir(dir)**
 - ▶ Muda o diretório corrente para **dir**
- ▶ **os.sep**
 - ▶ É uma string que contém o caracter que separa os componentes de um caminho (‘/’ para Unix, “\\” para Windows)
- ▶ **os.path.exists(path)**
 - ▶ Retorna True se o arquivo **path** existe

Curiosidade

- ▶ As operações de entrada e saída do teclado também usam arquivos “por baixo dos panos”!
 - ▶ Sempre que um comando **print** é executado, o resultado vai para um arquivo chamado **sys.stdout**
 - ▶ Sempre que lemos um dado através do comando **input**, na verdade estamos lendo de um arquivo chamado **sys.stdin**
 - ▶ Mensagens de erro ou de rastreamento de exceções são enviadas para um arquivo chamado **sys.stderr**

Exercícios

1. Faça um programa que leia um número N e gere um arquivo com N nomes e idades aleatórios

- ▶ Faça uso de duas listas criadas na mão: uma que contenha 20 nomes e outra que contenha 20 sobrenomes
- ▶ Cada linha do arquivo resultante deve conter um nome completo e a sua idade

2. Estenda o exemplo do cadastro para considerar também a altura da pessoa

- ▶ Armazene a altura como float



Exercícios

3. Escreva uma função que recebe dois nomes de arquivos e copia o conteúdo do primeiro arquivo para o segundo arquivo. Considere que o conteúdo do arquivo de origem é um texto. Sua função não deve copiar linhas comentadas (que começam com //)
4. Faça um programa contendo uma função que recebe como argumentos os nomes de dois arquivos. O primeiro arquivo contém nomes de alunos e o segundo arquivo contém as notas dos alunos. No primeiro arquivo, cada linha corresponde ao nome de um aluno e no segundo arquivo, cada linha corresponde às notas dos alunos (uma ou mais). Leia os dois arquivos e gere um terceiro arquivo que contém o nome do aluno seguido da média de suas notas.



Exercícios

5. Faça um programa para alterar uma das notas de um aluno (usando os arquivos do exercício anterior). O programa deve ter uma função que recebe o nome do aluno, a nota velha e a nova nota. A função deve fazer a alteração no arquivo.

6. Faça uma função que leia um arquivo texto contendo uma lista de endereços IP e gere dois outros arquivos, um contendo os endereços IP válidos e outro contendo os endereços inválidos. O formato de um endereço IP é **num1.num.num.num**, onde **num1** vai de 1 a 255 e **num** vai de 0 a 255.

Referências

- ▶ Slides de Leonardo Murta e Aline Paes



Arquivos

Vanessa Braganholo
vanessa@ic.uff.br