



UNIVERSIDADE DE  
**VASSOURAS**



UNIVERSIDADE DE VASSOURAS

Curso de Graduação em Engenharia de Software

Aula 6  
25 MAR 2022

## Laboratório de Programação de Interfaces com o Usuário

**Prof. André Saraiva**

Mestre em Sistemas de Computação  
Especialista em Arquitetura e Projeto de Cloud Computing  
Analista Blue Team em Cibersegurança pela Kimoshiro  
Tutor EaD pela Universidade Federal Fluminense - UFF



## Atenção

### P1

- 01/04
- Entrega da Atividade de Scraping
- Questionário na Plataforma / Aula ? (não tem 2ª chamada)

## Tópicos

### Revisão de Python

- Dicionários
- Arquivos Texto
- Revisão

## Dicionários (dict)

Dicionários tem pares, compostos de **chave:valor**, iteráveis sobre a **chave**, podendo seus elementos serem acessados por uma estrutura **for**.

Além disso, um dicionário pode ser escrito diretamente no vídeo via comando **print**.

## Dicionários (dict)

- ✓ **dict()**: associa um dicionário vazio a uma variável.

`compras = dict()` ou `compras = {}`

- ✓ Para adicionar um novo par ao dicionário basta atribuir o valor ao nome do dicionário seguido pela chave entre colchetes. Caso já exista esta chave, o valor é atualizado

`compras = dict()`

`compras[2] = "ovo"`

`compras[2] = "arroz"`

## Dicionários (dict)

✓ **del:** A função **del** nomeDict[**chave**] retira o par **chave:valor** do dicionário. Caso a **chave** não esteja no dicionário um erro ocorre - **KeyError**.

```
compras = {51:"Boa Ideia", 1: "leite", 2:"arroz"}
```

```
del compras[1] ou compras.pop(1)
```

```
compras = {51:"Boa Ideia, 2:"arroz"}
```

✓ A função **len()** retorna o tamanho do dicionário

```
print(len(compras)) # 2
```

## Dicionários (dict)

- ✓ **d.items()**: retorna uma visualização de todos os pares (chave, valor) de um dicionário **d**.
- ✓ **d.keys()**: retorna uma visualização de todas as chaves de um dicionário **d**.
- ✓ **d.values()**: retorna uma visualização de todos os valores de um dicionário **d**.



## Atividade Prática

### ✓ Especificação do Problema:

Crie um dicionário que faz a leitura de cinco nomes e telefones e cria um dicionário com até cinco nomes distintos digitados pelo usuário da forma **nome:fone** . Ao término, seu programa deverá escrever na saída padrão todo dicionário, depois somente as chaves e depois somente os valores.

## Dicionários (dict)

✓ A função **get(chave)** retorna **None** se não existir aquela **chave** no dicionário, caso contrário retorna o respectivo **valor**. A função **get(chave,default)** retorna **default** se não existir aquela **chave** no dicionário, caso contrário retorna o respectivo **valor**.

```
compras = {51:"Boa Ideia", 1: "leite", 2:"arroz"}
```

```
print(compras.get(5))           # retorna None
```

```
print(compras.get(1))          # retorna leite
```

```
print(compras.get(3, "out"))    # retorna Out
```

## Arquivos Textos

Um **arquivo texto** é uma sequência de caracteres, organizada em linhas, que reside em uma área de armazenamento (e.g., disco rígido) sob um mesmo nome.

Arquivos texto podem ser criados, visualizados e alterados por editores ou processadores de texto.

## Arquivo Texto

- ✓ Em Python, antes de ser utilizado, um arquivo texto precisa ser associado a um nome no diretório de arquivos e ser aberto, via operação **open**

*variavel = open(caminho do arquivo)*

ou

*variavel = open(caminho do arquivo, modo)*

- ✓ Os modos de operação de um arquivo são:
  - “r” = somente leitura
  - “w” = somente escrita
  - “a” = escrita no final do arquivo

## Arquivos Textos

`dados = open("teste.txt", "r")` → Caso o arquivo exista: abre para leitura o arquivo "teste.txt", e coloca a cabeça de leitura sobre o primeiro caractere da primeira linha. Caso ele não exista: causa erro **FileNotFoundError**.

`dados = open("teste.txt", "w")` → Caso o arquivo exista: apaga seu conteúdo antigo e coloca a cabeça de escrita no início do arquivo. Caso ele não exista: cria o arquivo no diretório e coloca a cabeça de escrita no início do arquivo

## Arquivos Textos

`dados = open("teste.txt", "a")` → Caso o arquivo exista: abre o arquivo para escrita e coloca a cabeça de escrita no fim do arquivo. Isto é: pronto a anexar novas informações no seu final. Caso ele não exista: cria o arquivo no diretório e coloca a cabeça de escrita no fim do arquivo, que neste caso é igual ao início.

## Arquivo Texto

✓ A operação **close()** permite que um arquivo texto seja fechado. Sempre que não for mais ser utilizado, um arquivo deve ser fechado

```
variavel = open
```

```
...
```

```
variavel.close()
```

## Arquivo Texto

✓ A operação **readline()**, aplicada sobre um arquivo texto aberto, retorna uma linha completa do arquivo, incluindo o fim de linha: “\n”. A cabeça de leitura avança para a próxima linha. Uma string vazia é retornada quando o fim de arquivo é encontrado.

```
variavel = open(“teste.txt”, “r”)
```

```
linha = variavel.readline()
```

```
print(linha, end=“ “)
```

```
variavel.close()
```



## Arquivo Texto

✓ A operação **readline()**, aplicada sobre um arquivo texto aberto, retorna uma linha completa do arquivo, incluindo o fim de linha: “\n”. A cabeça de leitura avança para a próxima linha. Uma string vazia é retornada quando o fim de arquivo é encontrado.

```
variavel = open(“teste.txt”, “r”)  
linha = variavel.readline()  
print(linha, end=“ “)  
variavel.close()
```

O arquivo teste.txt pode ser lido. O primeiro caractere a ser lido será o primeiro caractere do arquivo

## Arquivo Texto

✓ A operação **readline()**, aplicada sobre um arquivo texto aberto, retorna uma linha completa do arquivo, incluindo o fim de linha: “\n”. A cabeça de leitura avança para a próxima linha. Uma string vazia é retornada quando o fim de arquivo é encontrado.

```
variavel = open(“teste.txt”, “r”)
```

```
linha = variavel.readline()
```

```
print(linha, end=“ “)
```

```
variavel.close()
```

Lê a primeira linha  
ou o fim do arquivo

## Arquivo Texto

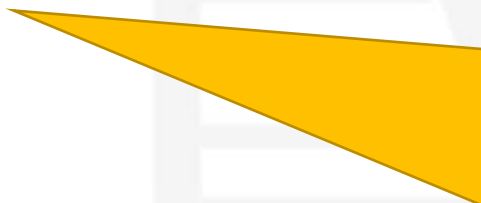
✓ A operação **readline()**, aplicada sobre um arquivo texto aberto, retorna uma linha completa do arquivo, incluindo o fim de linha: “\n”. A cabeça de leitura avança para a próxima linha. Uma string vazia é retornada quando o fim de arquivo é encontrado.

```
variavel = open(“teste.txt”, “r”)
```

```
linha = variavel.readline()
```

```
print(linha, end=“ “)
```

```
variavel.close()
```



Mostra a linha na  
tela

## Arquivo Texto

✓ A operação **readline()**, aplicada sobre um arquivo texto aberto, retorna uma linha completa do arquivo, incluindo o fim de linha: “\n”. A cabeça de leitura avança para a próxima linha. Uma string vazia é retornada quando o fim de arquivo é encontrado.

```
variavel = open(“teste.txt”, “r”)
```

```
linha = variavel.readline()
```

```
print(linha, end=“ “)
```

```
variavel.close()
```

Ao final, sempre  
fechar o arquivo.

## Arquivo Texto

```
nomeArquivo = input("Digite o nome do arquivo que deseja visualizar: ")
```

```
dados = open(nomeArquivo, "r")
```

```
linha = dados.readline()
```

Lê primeira linha

```
while linha != "":
```

Enquanto não é o fim

```
    print(linha, end="")
```

Mostra na tela

```
    linha = dados.readline()
```

Lê próxima linha

```
dados.close()
```

Ao final, sempre  
fechar o arquivo.

A partir deste ponto,  
o arquivo  
**nomeArquivo** pode  
ser lido.

O primeiro caracter  
a ser lido será o  
primeiro caracter do  
arquivo (onde  
estará inicialmente  
a "cabeça de  
leitura").

## Arquivo Texto

```
nomeArquivo = input("Digite o nome do arquivo que deseja visualizar: ")
```

```
dados = open(nomeArquivo, "r")
```

```
for linha in dados:
```

```
    print(linha, end="")
```

```
dados.close()
```

Iterando linha a linha  
sobre o conteúdo de  
um arquivo texto.

Ao final, sempre  
fechar o arquivo.

A partir deste ponto, o  
arquivo **nomeArquivo**  
pode ser lido.

O primeiro caracter a ser  
lido será o primeiro  
caracter do arquivo (onde  
estará inicialmente a  
"cabeça de leitura").

## Arquivo Texto

✓ Para escrever uma sequência de caracteres em um arquivo texto, no modo “w” ou “a”, podemos utilizar o método **write(desejada)**. Que escreverá a String **desejada** a partir do ponto em que a cabeça de escrita do arquivo estiver posicionada. Ao final, a cabeça de escrita ficará posicionada após o último caractere da String **desejada**.



Pula linha

```
variavel = open(“teste.txt”, “w”)
```

```
variavel.write(“qualquer texto pode ser escrito.\n”)
```

```
variavel.close()
```

## Atividade Prática

### ✓ Especificação do Problema:

Crie um programa que faça cópia de um arquivo texto (backup). Seu programa deverá solicitar o nome do arquivo de origem e o nome do novo arquivo de destino. Após isso, uma função copiar() deverá abrir o arquivo de origem, ler o conteúdo e copiar em novo arquivo de destino.



## Atividade Prática para P1

### ✓ Especificação do Problema:

Escreva um programa para um supermercado onde cada produto possui:

- (a) Código não repetido (a chave) – representado por String;
- (b) Descrição – representada por String;
- (c) Quantidade – representada por inteiro;
- (d) Preço – representado por número de ponto flutuante; e
- (e) Data de Validade – representada por Tupla (dia, mês, ano).

## Atividade Prática para P1

### ✓ Especificação do Problema:

As funções necessárias são:

`mostrar(...)`

`preencher(...)`

`vender(...)`

`main()`

## Atividade Prática para P1

### ✓ Especificação do Problema:

No mai():

```
produtos = { }  
preencher(produtos, 5)  
mostrar(produtos)  
vender(produtos, {"abc":3, "xyz":2})  
mostrar(produtos)
```

Prazo: 01/04 às 20h

Valor: até 1,5

## Atividade Prática para 09/04

### ✓ Especificação do Problema:

Crie um programa que copie um arquivo texto. Seu programa deverá abrir um arquivo texto, ler linha a linha e escrever em outro arquivo texto.

Atenção: Arquivo texto disponibilizado no AVA

## Próximos Tópicos

### P1

- Avaliação P1

**Contato**



**Professor:**

André Saraiva, MSc

**E-mail:**

[andre.saraiva@universidadedevassouras.edu.br](mailto:andre.saraiva@universidadedevassouras.edu.br)