Curso básico de Programação em Python

Encontro 11 - Manipulação de arquivos

Prof. Louis Augusto
louis.augusto@ifsc.edu.br

Instituto Federal de Santa Catarina
Campus São José

Índice

Sumário

Armazenamento de arquivos

Os arquivos são guardados em pastas no disco do computador, e nestas pastas podem estar os arquivos em que trabalhamos ou outras pastas.

A pasta raiz é a pasta (tecnicamente um diretório) de mais baixo acesso no computador, no linux é a pasta `/home` para um usário não administrador, e no windows é uma letra que representa uma partição no disco rígido, geralmente `C:\` ou `D:\`. A partir daí depositamos arquivos ou criamos outras pastas nesta pasta raiz.

No Linux, ao abrir o terminal e digitar `~$ pwd`, que significa *print working directory* obtemos como resposta o caminho até o diretório em que se está trabalhando.

Ainda no terminal ao digitarmos `~$ ls` obtemos a lista de todos os arquivos e pastas contidos no diretório corrente. Para alterar o diretório de tradigitamos `~$ cd nomedodiretorio` se houver algum diretório acima, `~$ cd ..` para ir a um diretório abaixo e `~$ cd` para ir ao diretório raiz.

Os comandos `~$ mkdir NomeDoDiretorio` criam um diretório na pasta e `~$ rm NomeArquivo` apaga um arquivo e `~$ rm -r NomeDiretorio` apaga o diretório com tudo o que tiver dentro.

Armazenamento de arquivos

Os arquivos são guardados em pastas no disco do computador, e nestas pastas podem estar os arquivos em que trabalhamos ou outras pastas.

A pasta raiz é a pasta (tecnicamente um diretório) de mais baixo acesso no computador, no linux é a pasta `/home` para um usário não administrador, e no windows é uma letra que representa uma partição no disco rígido, geralmente `C:\` ou `D:\`. A partir daí depositamos arquivos ou criamos outras pastas nesta pasta raiz.

No Linux, ao abrir o terminal e digitar `~$ pwd`, que significa *print working directory* obtemos como resposta o caminho até o diretório em que se está trabalhando.

Ainda no terminal ao digitarmos `~$ ls` obtemos a lista de todos os arquivos e pastas contidos no diretório corrente. Para alterar o diretório de tradigitamos `~$ cd nomedodiretorio` se houver algum diretório acima, `~$ cd ..` para ir a um diretório abaixo e `~$ cd` para ir ao diretório raiz.

Os comandos `~$ mkdir NomeDoDiretorio` criam um diretório na pasta e `~$ rm NomeArquivo` apaga um arquivo e `~$ rm -r NomeDiretorio` apaga o diretório com tudo o que tiver dentro.

Armazenamento de arquivos

Os arquivos são guardados em pastas no disco do computador, e nestas pastas podem estar os arquivos em que trabalhamos ou outras pastas.

A pasta raiz é a pasta (tecnicamente um diretório) de mais baixo acesso no computador, no linux é a pasta `/home` para um usário não administrador, e no windows é uma letra que representa uma partição no disco rígido, geralmente `C:\` ou `D:\`. A partir daí depositamos arquivos ou criamos outras pastas nesta pasta raiz.

No Linux, ao abrir o terminal e digitar `~$ pwd`, que significa *print working directory* obtemos como resposta o caminho até o diretório em que se está trabalhando.

Ainda no terminal ao digitarmos `~$ ls` obtemos a lista de todos os arquivos e pastas contidos no diretório corrente. Para alterar o diretório de tradigitamos `~$ cd nomedodiretorio` se houver algum diretório acima, `~$ cd ..` para ir a um diretório abaixo e `~$ cd` para ir ao diretório raiz.

Os comandos `~$ mkdir NomeDoDiretorio` criam um diretório na pasta e `~$ rm NomeArquivo` apaga um arquivo e `~$ rm -r NomeDiretorio` apaga o diretório com tudo o que tiver dentro.

Armazenamento de arquivos

Os arquivos são guardados em pastas no disco do computador, e nestas pastas podem estar os arquivos em que trabalhamos ou outras pastas.

A pasta raiz é a pasta (tecnicamente um diretório) de mais baixo acesso no computador, no linux é a pasta `/home` para um usário não administrador, e no windows é uma letra que representa uma partição no disco rígido, geralmente `C:\` ou `D:\`. A partir daí depositamos arquivos ou criamos outras pastas nesta pasta raiz.

No Linux, ao abrir o terminal e digitar `~$ pwd`, que significa *print working directory* obtemos como resposta o caminho até o diretório em que se está trabalhando.

Ainda no terminal ao digitarmos `~$ ls` obtemos a lista de todos os arquivos e pastas contidos no diretório corrente. Para alterar o diretório de tradigitamos `~$ cd nomedodiretorio` se houver algum diretório acima, `~$ cd ..` para ir a um diretório abaixo e `~$ cd` para ir ao diretório raiz.

Os comandos `~$ mkdir NomeDoDiretorio` criam um diretório na pasta e `~$ rm NomeArquivo` apaga um arquivo e `~$ rm -r NomeDiretorio` apaga o diretório com tudo o que tiver dentro.

Armazenamento de arquivos

Os arquivos são guardados em pastas no disco do computador, e nestas pastas podem estar os arquivos em que trabalhamos ou outras pastas.

A pasta raiz é a pasta (tecnicamente um diretório) de mais baixo acesso no computador, no linux é a pasta `/home` para um usário não administrador, e no windows é uma letra que representa uma partição no disco rígido, geralmente `C:\` ou `D:\`. A partir daí depositamos arquivos ou criamos outras pastas nesta pasta raiz.

No Linux, ao abrir o terminal e digitar `~$ pwd`, que significa *print working directory* obtemos como resposta o caminho até o diretório em que se está trabalhando.

Ainda no terminal ao digitarmos `~$ ls` obtemos a lista de todos os arquivos e pastas contidos no diretório corrente. Para alterar o diretório de tradigitamos `~$ cd nomedodiretorio` se houver algum diretório acima, `~$ cd ..` para ir a um diretório abaixo e `~$ cd` para ir ao diretório raiz.

Os comandos `~$ mkdir NomeDoDiretorio` criam um diretório na pasta e `~$ rm NomeArquivo` apaga um arquivo e `~$ rm -r NomeDiretorio` apaga o diretório com tudo o que tiver dentro.

Manipulação de arquivos no Python

Todo arquivo, independentemente do sistema operacional tem caminho (path) e um nome.

- Caminho é a sequência de diretórios até chegar ao arquivo.
- Nome é a forma de referenciar o arquivo, geralmente precedido por uma extensão, `.jpg` por exemplo, que é uma dica para o sistema operacional escolher um programa que abra este arquivo. No Linux não é obrigatório.

Se estivermos com o terminal aberto, `~$ pwd` mostra o caminho da pasta e adicionando o nome do arquivo teremos o caminho do arquivo.

Manipulação de arquivos no Python

Todo arquivo, independentemente do sistema operacional tem caminho (path) e um nome.

- Caminho é a sequência de diretórios até chegar ao arquivo.
- Nome é a forma de referenciar o arquivo, geralmente precedido por uma extensão, `.jpg` por exemplo, que é uma dica para o sistema operacional escolher um programa que abra este arquivo. No Linux não é obrigatório.

Se estivermos com o terminal aberto, `~$ pwd` mostra o caminho da pasta e adicionando o nome do arquivo teremos o caminho do arquivo.

Manipulação de arquivos no Python

Todo arquivo, independentemente do sistema operacional tem caminho (path) e um nome.

- Caminho é a sequência de diretórios até chegar ao arquivo.
- Nome é a forma de referenciar o arquivo, geralmente precedido por uma extensão, `.jpg` por exemplo, que é uma dica para o sistema operacional escolher um programa que abra este arquivo. No Linux não é obrigatório.

Se estivermos com o terminal aberto, `~$ pwd` mostra o caminho da pasta e adicionando o nome do arquivo teremos o caminho do arquivo.

Manipulação de arquivos no Python

Todo arquivo, independentemente do sistema operacional tem caminho (path) e um nome.

- Caminho é a sequência de diretórios até chegar ao arquivo.
- Nome é a forma de referenciar o arquivo, geralmente precedido por uma extensão, `.jpg` por exemplo, que é uma dica para o sistema operacional escolher um programa que abra este arquivo. No Linux não é obrigatório.

Se estivermos com o terminal aberto, `~$ pwd` mostra o caminho da pasta e adicionando o nome do arquivo teremos o caminho do arquivo.

Manipulação de arquivos no Python

Todo arquivo, independentemente do sistema operacional tem caminho (path) e um nome.

- Caminho é a sequência de diretórios até chegar ao arquivo.
- Nome é a forma de referenciar o arquivo, geralmente precedido por uma extensão, `.jpg` por exemplo, que é uma dica para o sistema operacional escolher um programa que abra este arquivo. No Linux não é obrigatório.

Se estivermos com o terminal aberto, `~$ pwd` mostra o caminho da pasta e adicionando o nome do arquivo teremos o caminho do arquivo.

Manipulação de arquivos no Python

Vimos como se cria, apaga pastas e arquivos via terminal, e no Python temos comandos equivalentes, que permitem durante a execução de um programa manipular pastas e arquivos.

Para utilizar o Python para manipular pastas precisamos do módulo `os` e do módulo `shutil`. Então precisamos usar uma linha `import os, shutil` para usar estas funcionalidades. Comandos equivalentes no python:

os.getcwd()	`~$ pwd`. No python *get current working directory*.
os.listdir()	`~$ ls`
os.remove("NomeArquivo")	`rm NomeArquivo`
shutil.rmtree("NomeDiretorio")	`~$ rm -r NomeDiretorio`
os.mkdir("NomeDiretorio")	`~$ mkdir NomeDiretorio`
os.chdir("NomeDiretorio")	`~$ cd NomeDiretorio`

Outras ações importantes:

- Copiar arquivo de um diretorio para outro: `shutil.copy(arq1, arq2)`, onde arq1 é o caminho do arquivo a copiar e arq2 é o caminho onde a cópia será feita.
- `shutil.copytree(dir1, dir2)` faz o serviço de `shutil.copy(...)` para diretórios completos.

Manipulação de arquivos no Python

Vimos como se cria, apaga pastas e arquivos via terminal, e no Python temos comandos equivalentes, que permitem durante a execução de um programa manipular pastas e arquivos.

Para utilizar o Python para manipular pastas precisamos do módulo `os` e do módulo `shutil`. Então precisamos usar uma linha `import os, shutil` para usar estas funcionalidades. Comandos equivalentes no python:

- `os.getcwd()` `~$ pwd`. No python *get current working directory*.
- `os.listdir()` `~$ ls`
- `os.remove("NomeArquivo")` `rm NomeArquivo`
- `shutil.rmtree("NomeDiretorio")` `~$ rm -r NomeDiretorio`
- `os.mkdir("NomeDiretorio")` `~$ mkdir NomeDiretorio`
- `os.chdir("NomeDiretorio")` `~$ cd NomeDiretorio`

Outras ações importantes:

- Copiar arquivo de um diretorio para outro: `shutil.copy(arq1, arq2)`, onde arq1 é o caminho do arquivo a copiar e arq2 é o caminho onde a cópia será feita.
- `shutil.copytree(dir1, dir2)` faz o serviço de `shutil.copy(.,.)` para diretórios completos.

Manipulação de arquivos no Python

Vimos como se cria, apaga pastas e arquivos via terminal, e no Python temos comandos equivalentes, que permitem durante a execução de um programa manipular pastas e arquivos.

Para utilizar o Python para manipular pastas precisamos do módulo `os` e do módulo `shutil`. Então precisamos usar uma linha `import os, shutil` para usar estas funcionalidades. Comandos equivalentes no python:

`os.getcwd()` `~$ pwd`. No python *get current working directory*.

`os.listdir()` `~$ ls`

`os.remove("NomeArquivo")` `rm NomeArquivo`

`shutil.rmtree("NomeDiretorio")` `~$ rm -r NomeDiretorio`

`os.mkdir("NomeDiretorio")` `~$ mkdir NomeDiretorio`

`os.chdir("NomeDiretorio")` `~$ cd NomeDiretorio`

Outras ações importantes:

- Copiar arquivo de um diretorio para outro: `shutil.copy(arq1, arq2)`, onde arq1 é o caminho do arquivo a copiar e arq2 é o caminho onde a cópia será feita.
- `shutil.copytree(dir1, dir2)` faz o serviço de `shutil.copy(.,.)` para diretórios completos.

Manipulação de arquivos no Python

Vimos como se cria, apaga pastas e arquivos via terminal, e no Python temos comandos equivalentes, que permitem durante a execução de um programa manipular pastas e arquivos.

Para utilizar o Python para manipular pastas precisamos do módulo `os` e do módulo `shutil`. Então precisamos usar uma linha `import os, shutil` para usar estas funcionalidades. Comandos equivalentes no python:

os.getcwd()	`~$ pwd`. No python *get current working directory*.
os.listdir()	`~$ ls`
os.remove("NomeArquivo")	`rm NomeArquivo`
shutil.rmtree("NomeDiretorio")	`~$ rm -r NomeDiretorio`
os.mkdir("NomeDiretorio")	`~$ mkdir NomeDiretorio`
os.chdir("NomeDiretorio")	`~$ cd NomeDiretorio`

Outras ações importantes:

- Copiar arquivo de um diretorio para outro: `shutil.copy(arq1, arq2)`, onde arq1 é o caminho do arquivo a copiar e arq2 é o caminho onde a cópia será feita.
- `shutil.copytree(dir1, dir2)` faz o serviço de `shutil.copy(.,.)` para diretórios completos.

Manipulação de arquivos no Python

Vimos como se cria, apaga pastas e arquivos via terminal, e no Python temos comandos equivalentes, que permitem durante a execução de um programa manipular pastas e arquivos.

Para utilizar o Python para manipular pastas precisamos do módulo `os` e do módulo `shutil`. Então precisamos usar uma linha `import os, shutil` para usar estas funcionalidades. Comandos equivalentes no python:

os.getcwd() `~$ pwd`. No python *get current working directory*.

os.listdir() `~$ ls`

os.remove("NomeArquivo") `rm NomeArquivo`

shutil.rmtree("NomeDiretorio") `~$ rm -r NomeDiretorio`

os.mkdir("NomeDiretorio") `~$ mkdir NomeDiretorio`

os.chdir("NomeDiretorio") `~$ cd NomeDiretorio`

Outras ações importantes:

- Copiar arquivo de um diretorio para outro: `shutil.copy(arq1, arq2)`, onde arq1 é o caminho do arquivo a copiar e arq2 é o caminho onde a cópia será feita.
- `shutil.copytree(dir1, dir2)` faz o serviço de `shutil.copy(.,.)` para diretórios completos.

Manipulação de arquivos no Python

Vimos como se cria, apaga pastas e arquivos via terminal, e no Python temos comandos equivalentes, que permitem durante a execução de um programa manipular pastas e arquivos.

Para utilizar o Python para manipular pastas precisamos do módulo `os` e do módulo `shutil`. Então precisamos usar uma linha `import os, shutil` para usar estas funcionalidades. Comandos equivalentes no python:

os.getcwd() `~$ pwd`. No python *get current working directory*.

os.listdir() `~$ ls`

os.remove("NomeArquivo") `rm NomeArquivo`

shutil.rmtree("NomeDiretorio") `~$ rm -r NomeDiretorio`

os.mkdir("NomeDiretorio") `~$ mkdir NomeDiretorio`

os.chdir("NomeDiretorio") `~$ cd NomeDiretorio`

Outras ações importantes:

- Copiar arquivo de um diretorio para outro: `shutil.copy(arq1, arq2)`, onde arq1 é o caminho do arquivo a copiar e arq2 é o caminho onde a cópia será feita.
- `shutil.copytree(dir1, dir2)` faz o serviço de `shutil.copy(.,.)` para diretórios completos.

Manipulação de arquivos no Python

Vimos como se cria, apaga pastas e arquivos via terminal, e no Python temos comandos equivalentes, que permitem durante a execução de um programa manipular pastas e arquivos.

Para utilizar o Python para manipular pastas precisamos do módulo `os` e do módulo `shutil`. Então precisamos usar uma linha `import os, shutil` para usar estas funcionalidades. Comandos equivalentes no python:

os.getcwd() `~$ pwd`. No python *get current working directory*.

os.listdir() `~$ ls`

os.remove("NomeArquivo") `rm NomeArquivo`

shutil.rmtree("NomeDiretorio") `~$ rm -r NomeDiretorio`

os.mkdir("NomeDiretorio") `~$ mkdir NomeDiretorio`

os.chdir("NomeDiretorio") `~$ cd NomeDiretorio`

Outras ações importantes:

- Copiar arquivo de um diretorio para outro: `shutil.copy(arq1, arq2)`, onde arq1 é o caminho do arquivo a copiar e arq2 é o caminho onde a cópia será feita.
- `shutil.copytree(dir1, dir2)` faz o serviço de `shutil.copy(.,.)` para diretórios completos.

Manipulação de arquivos no Python

Vimos como se cria, apaga pastas e arquivos via terminal, e no Python temos comandos equivalentes, que permitem durante a execução de um programa manipular pastas e arquivos.

Para utilizar o Python para manipular pastas precisamos do módulo `os` e do módulo `shutil`. Então precisamos usar uma linha `import os, shutil` para usar estas funcionalidades. Comandos equivalentes no python:

os.getcwd() `~$ pwd`. No python *get current working directory*.

os.listdir() `~$ ls`

os.remove("NomeArquivo") `rm NomeArquivo`

shutil.rmtree("NomeDiretorio") `~$ rm -r NomeDiretorio`

os.mkdir("NomeDiretorio") `~$ mkdir NomeDiretorio`

os.chdir("NomeDiretorio") `~$ cd NomeDiretorio`

Outras ações importantes:

- Copiar arquivo de um diretorio para outro: `shutil.copy(arq1, arq2)`, onde arq1 é o caminho do arquivo a copiar e arq2 é o caminho onde a cópia será feita.
- `shutil.copytree(dir1, dir2)` faz o serviço de `shutil.copy(.,.)` para diretórios completos.

Manipulação de arquivos no Python

Vimos como se cria, apaga pastas e arquivos via terminal, e no Python temos comandos equivalentes, que permitem durante a execução de um programa manipular pastas e arquivos.

Para utilizar o Python para manipular pastas precisamos do módulo `os` e do módulo `shutil`. Então precisamos usar uma linha `import os, shutil` para usar estas funcionalidades. Comandos equivalentes no python:

os.getcwd() `~$ pwd`. No python *get current working directory*.

os.listdir() `~$ ls`

os.remove("NomeArquivo") `rm NomeArquivo`

shutil.rmtree("NomeDiretorio") `~$ rm -r NomeDiretorio`

os.mkdir("NomeDiretorio") `~$ mkdir NomeDiretorio`

os.chdir("NomeDiretorio") `~$ cd NomeDiretorio`

Outras ações importantes:

- Copiar arquivo de um diretorio para outro: `shutil.copy(arq1, arq2)`, onde arq1 é o caminho do arquivo a copiar e arq2 é o caminho onde a cópia será feita.
- `shutil.copytree(dir1, dir2)` faz o serviço de `shutil.copy(...)` para diretórios completos.

Manipulação de arquivos no Python

Vimos como se cria, apaga pastas e arquivos via terminal, e no Python temos comandos equivalentes, que permitem durante a execução de um programa manipular pastas e arquivos.

Para utilizar o Python para manipular pastas precisamos do módulo `os` e do módulo `shutil`. Então precisamos usar uma linha `import os, shutil` para usar estas funcionalidades. Comandos equivalentes no python:

os.getcwd() `~$ pwd`. No python *get current working directory*.

os.listdir() `~$ ls`

os.remove("NomeArquivo") `rm NomeArquivo`

shutil.rmtree("NomeDiretorio") `~$ rm -r NomeDiretorio`

os.mkdir("NomeDiretorio") `~$ mkdir NomeDiretorio`

os.chdir("NomeDiretorio") `~$ cd NomeDiretorio`

Outras ações importantes:

- Copiar arquivo de um diretorio para outro: `shutil.copy(arq1, arq2)`, onde arq1 é o caminho do arquivo a copiar e arq2 é o caminho onde a cópia será feita.
- `shutil.copytree(dir1, dir2)` faz o serviço de `shutil.copy(...)` para diretórios completos.

Sumário

Arquivos de texto e arquivos binários

Há três tipos de arquivos que podem ser lidos para obtenção de dados, arquivos de texto e binários.

Arquivos de texto: são arquivos que podem ser lidos por pessoas, como este texto, e são editáveis por editores de texto, como o Pluma ou Writer (no Linux) ou Bloco de Notas ou Word (no Windows). O preço a ser pago é que a leitura pelo computador é lenta. Normalmente tem extensões .txt, .dat, .doc etc.

Arquivos binários: são arquivos que não podem ser lido por pessoas, precisam ser lidos, e possivelmente editados, por um programa executável, sendo que este deve conhecer muito bem o formato em detalhes. São rápidos para salvar e carregar em comparação com arquivos de texto. Como exemplos temos arquivos de imagem .jpg, de vídeo .mp4, de áudio .mp3 etc.

Arquivos de texto e arquivos binários

Há três tipos de arquivos que podem ser lidos para obtenção de dados, arquivos de texto e binários.

Arquivos de texto: são arquivos que podem ser lidos por pessoas, como este texto, e são editáveis por editores de texto, como o Pluma ou Writer (no Linux) ou Bloco de Notas ou Word (no Windows). O preço a ser pago é que a leitura pelo computador é lenta. Normalmente tem extensões .txt, .dat, .doc etc.

Arquivos binários: são arquivos que não podem ser lido por pessoas, precisam ser lidos, e possivelmente editados, por um programa executável, sendo que este deve conhecer muito bem o formato em detalhes. São rápidos para salvar e carregar em comparação com arquivos de texto. Como exemplos temos arquivos de imagem .jpg, de vídeo .mp4, de áudio .mp3 etc.

Arquivos de texto e arquivos binários

Há três tipos de arquivos que podem ser lidos para obtenção de dados, arquivos de texto e binários.

Arquivos de texto: são arquivos que podem ser lidos por pessoas, como este texto, e são editáveis por editores de texto, como o Pluma ou Writer (no Linux) ou Bloco de Notas ou Word (no Windows). O preço a ser pago é que a leitura pelo computador é lenta. Normalmente tem extensões .txt, .dat, .doc etc.

Arquivos binários: são arquivos que não podem ser lido por pessoas, precisam ser lidos, e possivelmente editados, por um programa executável, sendo que este deve conhecer muito bem o formato em detalhes. São rápidos para salvar e carregar em comparação com arquivos de texto. Como exemplos temos arquivos de imagem .jpg, de vídeo .mp4, de áudio .mp3 etc.

Sumário

Abertura de arquivos

Até o momento gravamos ou lemos informações usando a forma padrão de entrada e saída do Python (input() e print()) e redirecionamento.

A partir de agora será conveniente guardar informações de forma mais especializada. A linguagem Python oferece funções que servem para manipular arquivos sem muitas dificuldades.

Operação (A função open())

A sintaxe básica da função built-in open() é:

```
Objeto_arquivo = open(nome_do_arquivo, modo, buffer )
```

Um exemplo básico de abertura é:

```
-*- coding: utf-8 -*-
fo = open("teste.txt", "wb")
```

```
try:
    fo = open("teste.txt", "rb")
except IOError:
    print ("Erro na abertura do arquivo")
```

O erro de `IO` significa que o arquivo não existe no caminho indicado, ou outro erro de input/output.

Abertura de arquivos

Até o momento gravamos ou lemos informações usando a forma padrão de entrada e saída do Python (input() e print()) e redirecionamento.

A partir de agora será conveniente guardar informações de forma mais especializada. A linguagem Python oferece funções que servem para manipular arquivos sem muitas dificuldades.

Operação (A função open())

A sintaxe básica da função built-in open() é:

```
Objeto_arquivo = open(nome_do_arquivo, modo, buffer )
```

Um exemplo básico de abertura é:

```
-*- coding: utf-8 -*-
fo = open("teste.txt", "wb")
```

```
try:
    fo = open("teste.txt", "rb")
except IOError:
    print ("Erro na abertura do arquivo")
```

O erro de `IO` significa que o arquivo não existe no caminho indicado, ou outro erro de input/output.

Abertura de arquivos

Até o momento gravamos ou lemos informações usando a forma padrão de entrada e saída do Python (input() e print()) e redirecionamento.

A partir de agora será conveniente guardar informações de forma mais especializada. A linguagem Python oferece funções que servem para manipular arquivos sem muitas dificuldades.

Operação (A função open())

A sintaxe básica da função built-in open() é:

```
Objeto_arquivo = open(nome_do_arquivo, modo, buffer )
```

Um exemplo básico de abertura é:

```
-*- coding: utf-8 -*-
fo = open("teste.txt", "wb")
```

```
try:
    fo = open("teste.txt", "rb")
except IOError:
    print ("Erro na abertura do arquivo")
```

O erro de `IO` significa que o arquivo não existe no caminho indicado, ou outro erro de input/output.

Abertura de arquivos

Até o momento gravamos ou lemos informações usando a forma padrão de entrada e saída do Python (input() e print()) e redirecionamento.

A partir de agora será conveniente guardar informações de forma mais especializada. A linguagem Python oferece funções que servem para manipular arquivos sem muitas dificuldades.

Operação (A função open())

A sintaxe básica da função built-in open() é:

```
Objeto_arquivo = open(nome_do_arquivo, modo, buffer )
```

Um exemplo básico de abertura é:

```
-*- coding: utf-8 -*-
fo = open("teste.txt", "wb")
```

```
try:
    fo = open("teste.txt", "rb")
except IOError:
    print ("Erro na abertura do arquivo")
```

O erro de `IO` significa que o arquivo não existe no caminho indicado, ou outro erro de input/output.

Abertura de arquivos

Até o momento gravamos ou lemos informações usando a forma padrão de entrada e saída do Python (input() e print()) e redirecionamento.

A partir de agora será conveniente guardar informações de forma mais especializada. A linguagem Python oferece funções que servem para manipular arquivos sem muitas dificuldades.

Operação (A função open())

A sintaxe básica da função built-in open() é:

```
Objeto_arquivo = open(nome_do_arquivo, modo, buffer )
```

Um exemplo básico de abertura é:

```
-*- coding: utf-8 -*-
fo = open("teste.txt", "wb")
```

```
try:
    fo = open("teste.txt", "rb")
except IOError:
    print ("Erro na abertura do arquivo")
```

O erro de `IO` significa que o arquivo não existe no caminho indicado, ou outro erro de input/output.

Abertura de arquivos

Até o momento gravamos ou lemos informações usando a forma padrão de entrada e saída do Python (input() e print()) e redirecionamento.

A partir de agora será conveniente guardar informações de forma mais especializada. A linguagem Python oferece funções que servem para manipular arquivos sem muitas dificuldades.

Operação (A função open())

A sintaxe básica da função built-in open() é:

```
Objeto_arquivo = open(nome_do_arquivo, modo, buffer )
```

Um exemplo básico de abertura é:

```
-*- coding: utf-8 -*-
fo = open("teste.txt", "wb")
```

```
try:
    fo = open("teste.txt", "rb")
except IOError:
    print ("Erro na abertura do arquivo")
```

O erro de `IO` significa que o arquivo não existe no caminho indicado, ou outro erro de input/output.

Abertura de arquivos

Temos os elementos:

- `Objeto_arquivo`: Nome da variável que referencia o arquivo. A partir da leitura o arquivo é referenciável não por ser nome, mas pela variável.
- `Nome_do_arquivo`: é a string que recebe o nome do arquivo em disco.
- `modo`: é o tipo de arquivo que será manipulado, de texto ou binário, e se será um arquivo para leitura, escrita (podendo ser criado ou alterado). O próximo slide trabalhará este item.
- `buffer`: serve para indicar se a manipulação do arquivo será feita diretamente pelo sistema operacional ou não, podendo implicar numa leitura mais rápida. É um tópico avançado, que não será trabalhado aqui. Pode-se deixar o campo vazio ou usar -1 para o valor default.

Quando abrimos um arquivo recebemos um cursor, (ponteiro de posicionamento de arquivo), que indica a posição em que se está no arquivo, como um cursor num editor de texto.

Abertura de arquivos

Temos os elementos:

- `Objeto_arquivo`: Nome da variável que referencia o arquivo. A partir da leitura o arquivo é referenciável não por ser nome, mas pela variável.
- `Nome_do_arquivo`: é a string que recebe o nome do arquivo em disco.
- `modo`: é o tipo de arquivo que será manipulado, de texto ou binário, e se será um arquivo para leitura, escrita (podendo ser criado ou alterado). O próximo slide trabalhará este item.
- `buffer`: serve para indicar se a manipulação do arquivo será feita diretamente pelo sistema operacional ou não, podendo implicar numa leitura mais rápida. É um tópico avançado, que não será trabalhado aqui. Pode-se deixar o campo vazio ou usar -1 para o valor default.

Quando abrimos um arquivo recebemos um cursor, (ponteiro de posicionamento de arquivo), que indica a posição em que se está no arquivo, como um cursor num editor de texto.

Abertura de arquivos

Temos os elementos:

- `Objeto_arquivo`: Nome da variável que referencia o arquivo. A partir da leitura o arquivo é referenciável não por ser nome, mas pela variável.
- `Nome_do_arquivo`: é a string que recebe o nome do arquivo em disco.
- `modo`: é o tipo de arquivo que será manipulado, de texto ou binário, e se será um arquivo para leitura, escrita (podendo ser criado ou alterado). O próximo slide trabalhará este item.
- `buffer`: serve para indicar se a manipulação do arquivo será feita diretamente pelo sistema operacional ou não, podendo implicar numa leitura mais rápida. É um tópico avançado, que não será trabalhado aqui. Pode-se deixar o campo vazio ou usar -1 para o valor default.

Quando abrimos um arquivo recebemos um cursor, (ponteiro de posicionamento de arquivo), que indica a posição em que se está no arquivo, como um cursor num editor de texto.

Abertura de arquivos

Temos os elementos:

- `Objeto_arquivo`: Nome da variável que referencia o arquivo. A partir da leitura o arquivo é referenciável não por ser nome, mas pela variável.
- `Nome_do_arquivo`: é a string que recebe o nome do arquivo em disco.
- `modo`: é o tipo de arquivo que será manipulado, de texto ou binário, e se será um arquivo para leitura, escrita (podendo ser criado ou alterado). O próximo slide trabalhará este item.
- `buffer`: serve para indicar se a manipulação do arquivo será feita diretamente pelo sistema operacional ou não, podendo implicar numa leitura mais rápida. É um tópico avançado, que não será trabalhado aqui. Pode-se deixar o campo vazio ou usar -1 para o valor default.

Quando abrimos um arquivo recebemos um cursor, (ponteiro de posicionamento de arquivo), que indica a posição em que se está no arquivo, como um cursor num editor de texto.

Abertura de arquivos

Temos os elementos:

- `Objeto_arquivo`: Nome da variável que referencia o arquivo. A partir da leitura o arquivo é referenciável não por ser nome, mas pela variável.
- `Nome_do_arquivo`: é a string que recebe o nome do arquivo em disco.
- `modo`: é o tipo de arquivo que será manipulado, de texto ou binário, e se será um arquivo para leitura, escrita (podendo ser criado ou alterado). O próximo slide trabalhará este item.
- `buffer`: serve para indicar se a manipulação do arquivo será feita diretamente pelo sistema operacional ou não, podendo implicar numa leitura mais rápida. É um tópico avançado, que não será trabalhado aqui. Pode-se deixar o campo vazio ou usar -1 para o valor default.

Quando abrimos um arquivo recebemos um cursor, (ponteiro de posicionamento de arquivo), que indica a posição em que se está no arquivo, como um cursor num editor de texto.

Abertura de arquivos

Temos os elementos:

- `Objeto_arquivo`: Nome da variável que referencia o arquivo. A partir da leitura o arquivo é referenciável não por ser nome, mas pela variável.
- `Nome_do_arquivo`: é a string que recebe o nome do arquivo em disco.
- `modo`: é o tipo de arquivo que será manipulado, de texto ou binário, e se será um arquivo para leitura, escrita (podendo ser criado ou alterado). O próximo slide trabalhará este item.
- `buffer`: serve para indicar se a manipulação do arquivo será feita diretamente pelo sistema operacional ou não, podendo implicar numa leitura mais rápida. É um tópico avançado, que não será trabalhado aqui. Pode-se deixar o campo vazio ou usar -1 para o valor default.

Quando abrimos um arquivo recebemos um cursor, (ponteiro de posicionamento de arquivo), que indica a posição em que se está no arquivo, como um cursor num editor de texto.

Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *r* Abre um arquivo texto para somente leitura. O ponteiro de posição para o arquivo (como se fosse um cursor num editor de texto) se localiza no início do arquivo.
- *rb* Abre um arquivo binário para somente leitura. O ponteiro para o arquivo se localiza no início do arquivo.
- *rb+* Abre um arquivo binário para leitura e escrita. O ponteiro para o arquivo se localiza no início do arquivo.
- *w* Abre um arquivo texto para escrita. Se já houver um arquivo com o mesmo nome será sobrescrito.
- *wb* Abre um arquivo para somente escrita em format binário. Sobrescreve o arquivo caso já exista.
- *w+* Abre um arquivo em modo texto para leitura e escrita, sobrescreve o arquivo se já existir.
- *wb+* Abre um arquivo em modo binário para leitura e escrita em formato binário, sobrescreve o arquivo se já existir.

Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *r* Abre um arquivo texto para somente leitura. O ponteiro de posição para o arquivo (como se fosse um cursor num editor de texto) se localiza no início do arquivo.
- *rb* Abre um arquivo binário para somente leitura. O ponteiro para o arquivo se localiza no início do arquivo.
- *rb+* Abre um arquivo binário para leitura e escrita. O ponteiro para o arquivo se localiza no início do arquivo.
- *w* Abre um arquivo texto para escrita. Se já houver um arquivo com o mesmo nome será sobrescrito.
- *wb* Abre um arquivo para somente escrita em format binário. Sobrescreve o arquivo caso já exista.
- *w+* Abre um arquivo em modo texto para leitura e escrita, sobrescreve o arquivo se já existir.
- *wb+* Abre um arquivo em modo binário para leitura e escrita em formato binário, sobrescreve o arquivo se já existir.

Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *r* Abre um arquivo texto para somente leitura. O ponteiro de posição para o arquivo (como se fosse um cursor num editor de texto) se localiza no início do arquivo.
- *rb* Abre um arquivo binário para somente leitura. O ponteiro para o arquivo se localiza no início do arquivo.
- *rb+* Abre um arquivo binário para leitura e escrita. O ponteiro para o arquivo se localiza no início do arquivo.
- *w* Abre um arquivo texto para escrita. Se já houver um arquivo com o mesmo nome será sobrescrito.
- *wb* Abre um arquivo para somente escrita em format binário. Sobrescreve o arquivo caso já exista.
- *w+* Abre um arquivo em modo texto para leitura e escrita, sobrescreve o arquivo se já existir.
- *wb+* Abre um arquivo em modo binário para leitura e escrita em formato binário, sobrescreve o arquivo se já existir.

Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *r* Abre um arquivo texto para somente leitura. O ponteiro de posição para o arquivo (como se fosse um cursor num editor de texto) se localiza no início do arquivo.
- *rb* Abre um arquivo binário para somente leitura. O ponteiro para o arquivo se localiza no início do arquivo.
- *rb+* Abre um arquivo binário para leitura e escrita. O ponteiro para o arquivo se localiza no início do arquivo.
- *w* Abre um arquivo texto para escrita. Se já houver um arquivo com o mesmo nome será sobrescrito.
- *wb* Abre um arquivo para somente escrita em format binário. Sobrescreve o arquivo caso já exista.
- *w+* Abre um arquivo em modo texto para leitura e escrita, sobrescreve o arquivo se já existir.
- *wb+* Abre um arquivo em modo binário para leitura e escrita em formato binário, sobrescreve o arquivo se já existir.

Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *r* Abre um arquivo texto para somente leitura. O ponteiro de posição para o arquivo (como se fosse um cursor num editor de texto) se localiza no início do arquivo.
- *rb* Abre um arquivo binário para somente leitura. O ponteiro para o arquivo se localiza no início do arquivo.
- *rb+* Abre um arquivo binário para leitura e escrita. O ponteiro para o arquivo se localiza no início do arquivo.
- *w* Abre um arquivo texto para escrita. Se já houver um arquivo com o mesmo nome será sobrescrito.
- *wb* Abre um arquivo para somente escrita em format binário. Sobrescreve o arquivo caso já exista.
- *w+* Abre um arquivo em modo texto para leitura e escrita, sobrescreve o arquivo se já existir.
- *wb+* Abre um arquivo em modo binário para leitura e escrita em formato binário, sobrescreve o arquivo se já existir.

Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *r* Abre um arquivo texto para somente leitura. O ponteiro de posição para o arquivo (como se fosse um cursor num editor de texto) se localiza no início do arquivo.
- *rb* Abre um arquivo binário para somente leitura. O ponteiro para o arquivo se localiza no início do arquivo.
- *rb+* Abre um arquivo binário para leitura e escrita. O ponteiro para o arquivo se localiza no início do arquivo.
- *w* Abre um arquivo texto para escrita. Se já houver um arquivo com o mesmo nome será sobrescrito.
- *wb* Abre um arquivo para somente escrita em format binário. Sobrescreve o arquivo caso já exista.
- *w+* Abre um arquivo em modo texto para leitura e escrita, sobrescreve o arquivo se já existir.
- *wb+* Abre um arquivo em modo binário para leitura e escrita em formato binário, sobrescreve o arquivo se já existir.

Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *r* Abre um arquivo texto para somente leitura. O ponteiro de posição para o arquivo (como se fosse um cursor num editor de texto) se localiza no início do arquivo.
- *rb* Abre um arquivo binário para somente leitura. O ponteiro para o arquivo se localiza no início do arquivo.
- *rb+* Abre um arquivo binário para leitura e escrita. O ponteiro para o arquivo se localiza no início do arquivo.
- *w* Abre um arquivo texto para escrita. Se já houver um arquivo com o mesmo nome será sobrescrito.
- *wb* Abre um arquivo para somente escrita em format binário. Sobrescreve o arquivo caso já exista.
- *w+* Abre um arquivo em modo texto para leitura e escrita, sobrescreve o arquivo se já existir.
- *wb+* Abre um arquivo em modo binário para leitura e escrita em formato binário, sobrescreve o arquivo se já existir.

Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *a* Abre um arquivo para anexação de informação, ou seja, permite inclusão de informação no final do arquivo ($a = append$, que siginifica anexar). O cursor é posicionado no final do arquivo para anexação. Caso o arquivo não exista é criado um arquivo novo para escrita.
- *a+* Abre um arquivo para leitura e anexação. O ponteiro de posição se inicia no fim do arquivo, se existir, caso contrário cria um arquivo novo para escrita.
- *ab* Abre um arquivo para anexação em modo binário. O cursor é posicionado no final do arquivo, caso exista, caso contrário cria um arquivo para escrita.
- *ab+* Abre um arquivo para leitura e anexação em modo binário. O cursor é posicionado no final do arquivo, caso exista, do contrário cria um novo arquivo para leitura e escrita.

Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *a* Abre um arquivo para anexação de informação, ou seja, permite inclusão de informação no final do arquivo ($a = append$, que siginifica anexar). O cursor é posicionado no final do arquivo para anexação. Caso o arquivo não exista é criado um arquivo novo para escrita.
- *a+* Abre um arquivo para leitura e anexação. O ponteiro de posição se inicia no fim do arquivo, se existir, caso contrário cria um arquivo novo para escrita.
- *ab* Abre um arquivo para anexação em modo binário. O cursor é posicionado no final do arquivo, caso exista, caso contrário cria um arquivo para escrita.
- *ab+* Abre um arquivo para leitura e anexação em modo binário. O cursor é posicionado no final do arquivo, caso exista, do contrário cria um novo arquivo para leitura e escrita.

Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *a* Abre um arquivo para anexação de informação, ou seja, permite inclusão de informação no final do arquivo ($a = append$, que siginifica anexar). O cursor é posicionado no final do arquivo para anexação. Caso o arquivo não exista é criado um arquivo novo para escrita.
- *a+* Abre um arquivo para leitura e anexação. O ponteiro de posição se inicia no fim do arquivo, se existir, caso contrário cria um arquivo novo para escrita.
- *ab* Abre um arquivo para anexação em modo binário. O cursor é posicionado no final do arquivo, caso exista, caso contrário cria um arquivo para escrita.
- *ab+* Abre um arquivo para leitura e anexação em modo binário. O cursor é posicionado no final do arquivo, caso exista, do contrário cria um novo arquivo para leitura e escrita.

Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *a* Abre um arquivo para anexação de informação, ou seja, permite inclusão de informação no final do arquivo (*a* = *append*, que siginifica anexar). O cursor é posicionado no final do arquivo para anexação. Caso o arquivo não exista é criado um arquivo novo para escrita.
- *a+* Abre um arquivo para leitura e anexação. O ponteiro de posição se inicia no fim do arquivo, se existir, caso contrário cria um arquivo novo para escrita.
- *ab* Abre um arquivo para anexação em modo binário. O cursor é posicionado no final do arquivo, caso exista, caso contrário cria um arquivo para escrita.
- *ab+* Abre um arquivo para leitura e anexação em modo binário. O cursor é posicionado no final do arquivo, caso exista, do contrário cria um novo arquivo para leitura e escrita.

Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *a* Abre um arquivo para anexação de informação, ou seja, permite inclusão de informação no final do arquivo ($a = append$, que siginifica anexar). O cursor é posicionado no final do arquivo para anexação. Caso o arquivo não exista é criado um arquivo novo para escrita.
- *a+* Abre um arquivo para leitura e anexação. O ponteiro de posição se inicia no fim do arquivo, se existir, caso contrário cria um arquivo novo para escrita.
- *ab* Abre um arquivo para anexação em modo binário. O cursor é posicionado no final do arquivo, caso exista, caso contrário cria um arquivo para escrita.
- *ab+* Abre um arquivo para leitura e anexação em modo binário. O cursor é posicionado no final do arquivo, caso exista, do contrário cria um novo arquivo para leitura e escrita.

Exemplos

Vamos criar um arquivo texto bem simples.

```
#-*- coding: utf-8 -*-
t_str= "Eu gosto de Python. "
t2_str = "Vamos estudar python. "

#abertura e gravacao
ft=open("ArqTexto.dat",'w')
ft.write(t_str)
ft.write('\n')
ft.write(t_str)
ft.write('\n')
ft.close()
```

Abra o arquivo e veja como ficou.

Exemplos

Vamos criar um arquivo texto bem simples.

```
#-*- coding: utf-8 -*-
t_str= "Eu gosto de Python. "
t2_str = "Vamos estudar python. "

#abertura e gravacao
ft=open("ArqTexto.dat",'w')
ft.write(t_str)
ft.write('\n')
ft.write(t_str)
ft.write('\n')
ft.close()
```

Abra o arquivo e veja como ficou.

Exemplo em modo texto

Faça uma anexação ao arquivo:

```
ft2 = open("ArqTexto.dat",'a')
ft2.write(t2_str)
ft2.write('\n')
ft2.close()
```

Abra novamente o arquivo e veja como ficou.

Vamos agora abrir o arquivo em modo texto:

```
#Abertura e leitura
ft2 = open("ArqTexto.dat",'r')
linhas = ft2.readlines()
ft2.close()
#Impressão
for i in range(len(linhas)):
print(linhas[i],end = '')
print()
```

Exemplo em modo texto

Faça uma anexação ao arquivo:

```
ft2 = open("ArqTexto.dat",'a')
ft2.write(t2_str)
ft2.write('\n')
ft2.close()
```

Abra novamente o arquivo e veja como ficou.

Vamos agora abrir o arquivo em modo texto:

```
#Abertura e leitura
ft2 = open("ArqTexto.dat",'r')
linhas = ft2.readlines()
ft2.close()
#Impressão
for i in range(len(linhas)):
print(linhas[i],end = '')
print()
```

Exemplo em modo texto

Faça uma anexação ao arquivo:

```
ft2 = open("ArqTexto.dat",'a')
ft2.write(t2_str)
ft2.write('\n')
ft2.close()
```

Abra novamente o arquivo e veja como ficou.

Vamos agora abrir o arquivo em modo texto:

```
#Abertura e leitura
ft2 = open("ArqTexto.dat",'r')
linhas = ft2.readlines()
ft2.close()
#Impressão
for i in range(len(linhas)):
print(linhas[i],end = '')
print()
```

Exemplo em modo texto

Faça uma anexação ao arquivo:

```
ft2 = open("ArqTexto.dat",'a')
ft2.write(t2_str)
ft2.write('\n')
ft2.close()
```

Abra novamente o arquivo e veja como ficou.

Vamos agora abrir o arquivo em modo texto:

```
#Abertura e leitura
ft2 = open("ArqTexto.dat",'r')
linhas = ft2.readlines()
ft2.close()
#Impressão
for i in range(len(linhas)):
print(linhas[i],end = '')
print()
```

Exemplo em modo binário

```
#Modo binário:
t_strb= "Eu gosto de Python em modo binario. \n"
t2_strb = "Vamos estudar python em modo binario. "
primos = [2, 3, 5, 7]
#Conversao do array para binario
b_str= bytearray(t_strb, 'utf-8')
b2_str= bytearray(t2_strb,'utf-8')
#Funcao bytearray converte o arquivo para binario,
tem-se que indicar a codificacao.
```

Gravação

```
fb1 = open("ArqBin.b2n",'wb')
fb1.write(b_str)
fb1.close()
```

Anexação:

```
fb2 = open("ArqBin.b2n",'ab')
fb2.write(b2_str)
fb2.close()
```

Exemplo em modo binário

```
#Modo binário:
t_strb= "Eu gosto de Python em modo binario. \n"
t2_strb = "Vamos estudar python em modo binario. "
primos = [2, 3, 5, 7]
#Conversao do array para binario
b_str= bytearray(t_strb, 'utf-8')
b2_str= bytearray(t2_strb,'utf-8')
#Funcao bytearray converte o arquivo para binario,
tem-se que indicar a codificacao.
```

Gravação

```
fb1 = open("ArqBin.b2n",'wb')
fb1.write(b_str)
fb1.close()
```

Anexação:

```
fb2 = open("ArqBin.b2n",'ab')
fb2.write(b2_str)
fb2.close()
```

Exemplo em modo binário

```
#Modo binário:
t_strb= "Eu gosto de Python em modo binario. \n"
t2_strb = "Vamos estudar python em modo binario. "
primos = [2, 3, 5, 7]
#Conversao do array para binario
b_str= bytearray(t_strb, 'utf-8')
b2_str= bytearray(t2_strb,'utf-8')
#Funcao bytearray converte o arquivo para binario,
tem-se que indicar a codificacao.
```

Gravação

```
fb1 = open("ArqBin.b2n",'wb')
fb1.write(b_str)
fb1.close()
```

Anexação:

```
fb2 = open("ArqBin.b2n",'ab')
fb2.write(b2_str)
fb2.close()
```

Exemplo em modo binário

```
b3 = open("ArqBin.b2n",'rb')
lines = fb3.readlines() #Leitura
fb3.close()

for i in range(len(lines)):
print(lines[i],end = '')
print()

#Para converter em strings usamos a função decode()
for i in range(len(lines)):
print(lines[i].decode(),end = '')
print()
```

Atributos do objeto

Uma vez que o arquivo estiver aberto e a referência do arquivo estiver ativa podemos ter várias informações relativas ao arquivo.

- `ref_arquivo.closed`: Retorna `True` se o arquivo está fechado e `False` se estiver aberto.
- `ref_arquivo.mode`: Retorna o modo de acesso.
- `ref_arquivo.name` Retorna o nome do arquivo.

Como usar:

```
# Abrir um arquivo para gravacao binaria
ref_arquivo = open("foo.txt", "wb")
print ("Nome do arquivo: ", ref_arquivo.name)
print ("Arquivo fechado ou nao: ", ref_arquivo.closed)
print ("Modo de utilização : ", ref_arquivo.mode)
ref_arquivo.close()
```

Atributos do objeto

Uma vez que o arquivo estiver aberto e a referência do arquivo estiver ativa podemos ter várias informações relativas ao arquivo.

- `ref_arquivo.closed`: Retorna `True` se o arquivo está fechado e `False` se estiver aberto.
- `ref_arquivo.mode`: Retorna o modo de acesso.
- `ref_arquivo.name` Retorna o nome do arquivo.

Como usar:

```
# Abrir um arquivo para gravacao binaria
ref_arquivo = open("foo.txt", "wb")
print ("Nome do arquivo: ", ref_arquivo.name)
print ("Arquivo fechado ou nao: ", ref_arquivo.closed)
print ("Modo de utilização : ", ref_arquivo.mode)
ref_arquivo.close()
```

Atributos do objeto

Uma vez que o arquivo estiver aberto e a referência do arquivo estiver ativa podemos ter várias informações relativas ao arquivo.

- `ref_arquivo.closed`: Retorna `True` se o arquivo está fechado e `False` se estiver aberto.
- `ref_arquivo.mode`: Retorna o modo de acesso.
- `ref_arquivo.name` Retorna o nome do arquivo.

Como usar:

```
# Abrir um arquivo para gravacao binaria
ref_arquivo = open("foo.txt", "wb")
print ("Nome do arquivo: ", ref_arquivo.name)
print ("Arquivo fechado ou nao: ", ref_arquivo.closed)
print ("Modo de utilização : ", ref_arquivo.mode)
ref_arquivo.close()
```

Atributos do objeto

Uma vez que o arquivo estiver aberto e a referência do arquivo estiver ativa podemos ter várias informações relativas ao arquivo.

- `ref_arquivo.closed`: Retorna `True` se o arquivo está fechado e `False` se estiver aberto.
- `ref_arquivo.mode`: Retorna o modo de acesso.
- `ref_arquivo.name` Retorna o nome do arquivo.

Como usar:

```
# Abrir um arquivo para gravacao binaria
ref_arquivo = open("foo.txt", "wb")
print ("Nome do arquivo: ", ref_arquivo.name)
print ("Arquivo fechado ou nao: ", ref_arquivo.closed)
print ("Modo de utilização : ", ref_arquivo.mode)
ref_arquivo.close()
```

Atributos do objeto

Uma vez que o arquivo estiver aberto e a referência do arquivo estiver ativa podemos ter várias informações relativas ao arquivo.

- `ref_arquivo.closed`: Retorna `True` se o arquivo está fechado e `False` se estiver aberto.
- `ref_arquivo.mode`: Retorna o modo de acesso.
- `ref_arquivo.name` Retorna o nome do arquivo.

Como usar:

```
# Abrir um arquivo para gravacao binaria
ref_arquivo = open("foo.txt", "wb")
print ("Nome do arquivo: ", ref_arquivo.name)
print ("Arquivo fechado ou nao: ", ref_arquivo.closed)
print ("Modo de utilização : ", ref_arquivo.mode)
ref_arquivo.close()
```

Atributos do objeto

Uma vez que o arquivo estiver aberto e a referência do arquivo estiver ativa podemos ter várias informações relativas ao arquivo.

- `ref_arquivo.closed:` Retorna `True` se o arquivo está fechado e `False` se estiver aberto.
- `ref_arquivo.mode:` Retorna o modo de acesso.
- `ref_arquivo.name` Retorna o nome do arquivo.

Como usar:

```
# Abrir um arquivo para gravacao binaria
ref_arquivo = open("foo.txt", "wb")
print ("Nome do arquivo: ", ref_arquivo.name)
print ("Arquivo fechado ou nao: ", ref_arquivo.closed)
print ("Modo de utilização : ", ref_arquivo.mode)
ref_arquivo.close()
```

Sumário

Comando with para gravação

O comando `with` é uma forma de se evitar fazer um tratamento de exceção quando se trabalha com manipulação de arquivo. Uma forma segura de se trabalhar com abertura e fechamento seria:

```
Arq = open(nome_arquivo, 'w')
try:
    Arq.write('Lorem ipsum')
finally:
    Arq.close()
```

que pode ser trocado pelo comando `with`, com funcionalidade análoga:

```
with open(nome_arquivo, 'w') as Arq:
    Arq.write('Lorem ipsum')
```

O bloco de código que seria usado em `try` passa a ser usado no bloco do comando `with`. O bloco `with` faz internamente a verificação de erro e o fechamento do arquivo, de forma que não se precisa utilizar `Arq.close()`.

Comando with para leitura

Vamos ler o arquivo gravado anteriormente e exibir em tela cada uma das strings.

```
with open('celulares.txt', 'r') as Arq:
    for linha in Arq:
        print(linha, end = '')
```

Formas alternativas:

```
with open('celulares.txt', 'r') as Arq:
    Linhas = Arq.readlines()
    for linha in Linhas:
        print(linha, end = '')
```

Leitura completa do arquivo.

```
with open('celulares.txt', 'r') as Arq:
    while True:
        Linhas = Arq.readline()
        if not Linhas:
            break
        print(Linhas, end = '')
```

Procede-se à leitura de linha por linha.

Se fizermos, apenas:

```
with open('celulares.txt', 'r') as Arq:
    print(Arq)
```

Teremos informações sobre o arquivo, incluindo sua codificação.
A operação `r+` permite que se adicione informação ao final do arquivo, mas não consegue alterar o corpo do arquivo

Sumário

Comando with

O código abaixo gera um arquivo com cada string referenciada e um salto de linha.

```
with open('celulares.txt', 'w') as Arq:
    Arq.write('Celular 1\n')

with open('celulares.txt', 'a') as Arq:
    Arq.write('Celular 2\n')

nomes = ['Lucas', 'Carol', 'Hosana']
numeros = list(range(1,5))
with open('celulares.txt', 'a') as Arq:
    for nome in nomes:
        Arq.write(nome+'\n')
    for num in numeros:
        Arq.write(str(num)+'\n')
```