Introdução à linguagem Python

Thiago Martins

A Biblioteca padrão Python (cont.)

- Diz-se que python tem “pilhas inclusas” (*batteries included*). A biblioteca padrão.
- I/O de arquivos, Expressões Regulares, Matemática, redes, rpc via xml, json, servidor web, sql...
- **https://docs.python.org/3/library/**

IO de arquivos

- Um arquivo pode ser aberto com a função open.
- Arquivos devem ser fechados com o método `close`.

```
file = open("arquivo.txt", "r")
file.close()
```

- O primeiro parâmetro é o caminho até o arquivo, o segundo é o modo.
- Modos:

`'r'`	leitura
`'w'`	escrita (truncando)
`'x'`	criação *exclusiva*
`'a'`	escrita (concatenando)
`'b'`	modo binário
`'t'`	modo texto
`'+'`	escrita e leitura

IO de arquivos

- Sistemas operacionais requerem que arquivos sejam *fechados* após seu uso!
- A cada chamada de open deve *sempre* corresponder uma chadada a `close`!
- Como tratar exceções?

```
file = open(“arquivo.txt”, “r”)
try:
   …
finally:
   file.close()
```

Código pouco legível...

Gerenciadores de contexto

- Alguns objetos possuem *gerenciadores de contexto* que garantem que os recursos gerenciados são liberados uma vez que o contexto se encerra.
- São usados dentro de blocos `with`:

```
with open(“arquivo.txt”, “r”) as file:
    …
```

- Recursos associados a `file` são liberados quando o bloco é encerrado (arquivo é fechado!)
- É possível gerenciar múltiplos objetos dentro do mesmo bloco:

```
with open(“a1.txt”, “r”) as a1, open(“a2.txt”, “r”) as a2:
```

Gerenciadores de contexto

- Arquivos são o exemplo clássico. Há outros no entanto: Conexões de rede, clientes de bancos de dados, etc...
- Objetos com gerenciamento de contexto implementam os métodos `__enter__(self)` e `__exit__(self, type, value, traceback)` `__exit__` recebe informações de exceção, caso alguma ocorra dentro do bloco. Retorna `True` se a exceção for tratada.
- Uma maneira simples de se criar um gerenciador de contexto é com geradores e o decorador `contextlib.contextmanager`:

```
import contextlib
@contextmangaer
def cria_objeto():
    print(“criando objeto…”)
    yield 2
    print(“liberando objeto…”)
```

I/O de arquivos em Python3

- Métodos do objeto arquivo:
- read(tam): Lê tam caracteres/bytes do arquivo. Se invocada sem o tamanho, lê o arquivo inteiro. Retorna uma string para arquivos texto ou uma sequência de bytes para arquivos binários
- readline(): Lê uma linha do arquivo texto. Retorna uma string.
- readlines(): Lê *todas* as linhas do arquivo texto.
- write(obj): Escreve uma string (arq. texto) ou uma sequência de bytes em um arquivo.
- tell(): Retorna a *posição* no arquivo atual.
- seek(pos, ref): Altera a posição atual do arquivo.

Acessando recursos da internet

- Pacote urllib
- Módulo `request`

```
import urllib
resposta = urllib.request.urlopen('https://api6.ipify.org/')
codificacao='UTF-8'
for content_spec in resposta.headers['Content-Type'].split(';'):
    if content_spec.startswith('charset'):
      encoding = content_spec.split('=')[1].lstrip()
print(resposta.read().decode(codificacao))
```

- A Função `urlopen` busca um recurso na web e devolve-o em um objeto do tipo `HTTPResponse`.
- O objeto `HTTPResponse` é um objeto *complexo*, com cabeçalhos, contexto e outros aspectos do protocolo http.
- O método `read` retorna uma cadeia de *bytes* com o conteúdo, não uma string!
- O exemplo acima usa os cabeçalhos para identificar a *codificação* da resposta e convertê-la em string.

Acessando recursos da internet

- Métodos http:
  - GET, POST, PUT, DELETE...
- Parâmetros opcionais data, method
  - Caso omitidos, método padrão é GET
  - Caso só se tenha data, método padrão é POST

```
import urllib.request
import json
with urllib.request.urlopen('https://httpbin.org/post', data = "valor=1234".encode("UTF-8")) as request:
  print(json.load(request)['form'])
```

O site `https://httpbin.org/` permite testar diversas requisições http.

Recursos em JSON

- JSON: JavaScript Object Notation
  - Formato de intercâmbio de dados em formato de texto.
  - Baseado na sintaxe JavaScript (objetos são codificados como as equivalentes literais constantes em javascript).
  - O padrão encontra um uso crescente na internet, suplantando SOAP e afins.

- Python possui o módulo `json`.
  - Funções principais:
    - `dumps(obj)`: Codifica o objeto obj como uma string em formato JSON.
    - `loads(str)`: Decodifica o JSON na string str e retorna um objeto Python equivalente.
    - dump(obj, arquivo): Escreve a representação em JSON de obj no arquivo aberto para escrita em formato texto.
    - load(arquivo): Lê uma string do arquivo (aberto para leitura) e retorna o objeto correspondente.