Curso básico de Programação em Python

Encontro 8 - Requisição de arquivos

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

Instituto Federal de Santa Catarina
Campus São José

Índice

Sumário

Introdução às APIs

API significa Application Programming Interface, e basicamente é uma ferramenta para trocar informações entre computadores.

De forma mais completa, APIs são serviços disponibilizados online para acessar recursos de alguma aplicação web.

A grande vantagem das APIs é que usam como protocolo HTML, de forma que muitas plataformas, como twitter, github, spotify disponibilizam este serviço para desenvolvedores.

Toda API possui 4 partes:

- base URL
- endpoint
- recurso
- verbos HTTP

Introdução às APIs

API significa Application Programming Interface, e basicamente é uma ferramenta para trocar informações entre computadores.

De forma mais completa, APIs são serviços disponibilizados online para acessar recursos de alguma aplicação web.

A grande vantagem das APIs é que usam como protocolo HTML, de forma que muitas plataformas, como twitter, github, spotify disponibilizam este serviço para desenvolvedores.

Toda API possui 4 partes:

- base URL
- endpoint
- recurso
- verbos HTTP

Introdução às APIs

API significa Application Programming Interface, e basicamente é uma ferramenta para trocar informações entre computadores.

De forma mais completa, APIs são serviços disponibilizados online para acessar recursos de alguma aplicação web.

A grande vantagem das APIs é que usam como protocolo HTML, de forma que muitas plataformas, como twitter, github, spotify disponibilizam este serviço para desenvolvedores.

Toda API possui 4 partes:

- base URL
- endpoint
- recurso
- verbos HTTP

Introdução às APIs

API significa Application Programming Interface, e basicamente é uma ferramenta para trocar informações entre computadores.

De forma mais completa, APIs são serviços disponibilizados online para acessar recursos de alguma aplicação web.

A grande vantagem das APIs é que usam como protocolo HTML, de forma que muitas plataformas, como twitter, github, spotify disponibilizam este serviço para desenvolvedores.

Toda API possui 4 partes:

- base URL
- endpoint
- recurso
- verbos HTTP

Introdução às APIs

API significa Application Programming Interface, e basicamente é uma ferramenta para trocar informações entre computadores.

De forma mais completa, APIs são serviços disponibilizados online para acessar recursos de alguma aplicação web.

A grande vantagem das APIs é que usam como protocolo HTML, de forma que muitas plataformas, como twitter, github, spotify disponibilizam este serviço para desenvolvedores.

Toda API possui 4 partes:

- base URL
- endpoint
- recurso
- verbos HTTP

Introdução às APIs

API significa Application Programming Interface, e basicamente é uma ferramenta para trocar informações entre computadores.

De forma mais completa, APIs são serviços disponibilizados online para acessar recursos de alguma aplicação web.

A grande vantagem das APIs é que usam como protocolo HTML, de forma que muitas plataformas, como twitter, github, spotify disponibilizam este serviço para desenvolvedores.

Toda API possui 4 partes:

- base URL
- endpoint
- recurso
- verbos HTTP

Introdução às APIs

API significa Application Programming Interface, e basicamente é uma ferramenta para trocar informações entre computadores.

De forma mais completa, APIs são serviços disponibilizados online para acessar recursos de alguma aplicação web.

A grande vantagem das APIs é que usam como protocolo HTML, de forma que muitas plataformas, como twitter, github, spotify disponibilizam este serviço para desenvolvedores.

Toda API possui 4 partes:

- base URL
- endpoint
- recurso
- verbos HTTP

Sumário

APIs oficiais e não oficiais

APIs Oficiais

Há APIs oficiais e não oficiais, que servem para aprendizagem e treinamento, permitindo uma boa fonte para produzir projetos.

Uma boa lista de APIs se encontra em: Lista de APIs github

Lista de APIs públicas: Lista de APIs públicas

Uma boa forma de proceder é ir ao github e fazer uma busca por APIs, sempre há muitos resultados em python para trabalhar.

APIs não Oficiais

APIs não oficiais são as não fornecidas pela empresa que disponibiliza os dados, geralmente tem mais funcionalidades que as APIs oficiais. Um exemplo é dado em API instagram não oficial.

Digitando no Google, `instagram unofficial api` será possível receber uma longa lista de APIs feita por pessoas comuns para poder trocar informações com os servidores do Instagram, do Google, Facebook, etc.

Não há lei que proíba, mas se o Instagram descobrir pode banir você.

APIs oficiais e não oficiais

APIs Oficiais

Há APIs oficiais e não oficiais, que servem para aprendizagem e treinamento, permitindo uma boa fonte para produzir projetos.

Uma boa lista de APIs se encontra em: Lista de APIs github

Lista de APIs públicas: Lista de APIs públicas

Uma boa forma de proceder é ir ao github e fazer uma busca por APIs, sempre há muitos resultados em python para trabalhar.

APIs não Oficiais

APIs não oficiais são as não fornecidas pela empresa que disponibiliza os dados, geralmente tem mais funcionalidades que as APIs oficiais. Um exemplo é dado em API instagram não oficial.

Digitando no Google, `instagram unofficial api` será possível receber uma longa lista de APIs feita por pessoas comuns para poder trocar informações com os servidores do Instagram, do Google, Facebook, etc.

Não há lei que proíba, mas se o Instagram descobrir pode banir você.

APIs oficiais e não oficiais

APIs Oficiais

Há APIs oficiais e não oficiais, que servem para aprendizagem e treinamento, permitindo uma boa fonte para produzir projetos.

Uma boa lista de APIs se encontra em: Lista de APIs github

Lista de APIs públicas: Lista de APIs públicas

Uma boa forma de proceder é ir ao github e fazer uma busca por APIs, sempre há muitos resultados em python para trabalhar.

APIs não Oficiais

APIs não oficiais são as não fornecidas pela empresa que disponibiliza os dados, geralmente tem mais funcionalidades que as APIs oficiais. Um exemplo é dado em API instagram não oficial.

Digitando no Google, `instagram unofficial api` será possível receber uma longa lista de APIs feita por pessoas comuns para poder trocar informações com os servidores do Instagram, do Google, Facebook, etc.

Não há lei que proíba, mas se o Instagram descobrir pode banir você.

APIs oficiais e não oficiais

APIs Oficiais

Há APIs oficiais e não oficiais, que servem para aprendizagem e treinamento, permitindo uma boa fonte para produzir projetos.

Uma boa lista de APIs se encontra em: ▸ Lista de APIs github

Lista de APIs públicas: ▸ Lista de APIs públicas

Uma boa forma de proceder é ir ao github e fazer uma busca por APIs, sempre há muitos resultados em python para trabalhar.

APIs não Oficiais

APIs não oficiais são as não fornecidas pela empresa que disponibiliza os dados, geralmente tem mais funcionalidades que as APIs oficiais. Um exemplo é dado em ▸ API instagram não oficial.

Digitando no Google, `instagram unofficial api` será possível receber uma longa lista de APIs feita por pessoas comuns para poder trocar informações com os servidores do Instagram, do Google, Facebook, etc.

Não há lei que proíba, mas se o Instagram descobrir pode banir você.

APIs oficiais e não oficiais

APIs Oficiais

Há APIs oficiais e não oficiais, que servem para aprendizagem e treinamento, permitindo uma boa fonte para produzir projetos.

Uma boa lista de APIs se encontra em: ▸ Lista de APIs github

Lista de APIs públicas: ▸ Lista de APIs públicas

Uma boa forma de proceder é ir ao github e fazer uma busca por APIs, sempre há muitos resultados em python para trabalhar.

APIs não Oficiais

APIs não oficiais são as não fornecidas pela empresa que disponibiliza os dados, geralmente tem mais funcionalidades que as APIs oficiais. Um exemplo é dado em ▸ API instagram não oficial.

Digitando no Google, `instagram unofficial api` será possível receber uma longa lista de APIs feita por pessoas comuns para poder trocar informações com os servidores do Instagram, do Google, Facebook, etc.

Não há lei que proíba, mas se o Instagram descobrir pode banir você.

APIs oficiais e não oficiais

APIs Oficiais

Há APIs oficiais e não oficiais, que servem para aprendizagem e treinamento, permitindo uma boa fonte para produzir projetos.

Uma boa lista de APIs se encontra em: ▸ Lista de APIs github

Lista de APIs públicas: ▸ Lista de APIs públicas

Uma boa forma de proceder é ir ao github e fazer uma busca por APIs, sempre há muitos resultados em python para trabalhar.

APIs não Oficiais

APIs não oficiais são as não fornecidas pela empresa que disponibiliza os dados, geralmente tem mais funcionalidades que as APIs oficiais. Um exemplo é dado em ▸ API instagram não oficial.

Digitando no Google, `instagram unofficial api` será possível receber uma longa lista de APIs feita por pessoas comuns para poder trocar informações com os servidores do Instagram, do Google, Facebook, etc.

Não há lei que proíba, mas se o Instagram descobrir pode banir você.

APIs oficiais e não oficiais

APIs Oficiais

Há APIs oficiais e não oficiais, que servem para aprendizagem e treinamento, permitindo uma boa fonte para produzir projetos.

Uma boa lista de APIs se encontra em: ▸ Lista de APIs github

Lista de APIs públicas: ▸ Lista de APIs públicas

Uma boa forma de proceder é ir ao github e fazer uma busca por APIs, sempre há muitos resultados em python para trabalhar.

APIs não Oficiais

APIs não oficiais são as não fornecidas pela empresa que disponibiliza os dados, geralmente tem mais funcionalidades que as APIs oficiais. Um exemplo é dado em ▸ API instagram não oficial.

Digitando no Google, `instagram unofficial api` será possível receber uma longa lista de APIs feita por pessoas comuns para poder trocar informações com os servidores do Instagram, do Google, Facebook, etc.

Não há lei que proíba, mas se o Instagram descobrir pode banir você.

APIs oficiais e não oficiais

APIs Oficiais

Há APIs oficiais e não oficiais, que servem para aprendizagem e treinamento, permitindo uma boa fonte para produzir projetos.

Uma boa lista de APIs se encontra em: ▸ Lista de APIs github

Lista de APIs públicas: ▸ Lista de APIs públicas

Uma boa forma de proceder é ir ao github e fazer uma busca por APIs, sempre há muitos resultados em python para trabalhar.

APIs não Oficiais

APIs não oficiais são as não fornecidas pela empresa que disponibiliza os dados, geralmente tem mais funcionalidades que as APIs oficiais. Um exemplo é dado em ▸ API instagram não oficial.

Digitando no Google, `instagram unofficial api` será possível receber uma longa lista de APIs feita por pessoas comuns para poder trocar informações com os servidores do Instagram, do Google, Facebook, etc.

Não há lei que proíba, mas se o Instagram descobrir pode banir você.

APIs oficiais e não oficiais

APIs Oficiais

Há APIs oficiais e não oficiais, que servem para aprendizagem e treinamento, permitindo uma boa fonte para produzir projetos.

Uma boa lista de APIs se encontra em: ▸ Lista de APIs github

Lista de APIs públicas: ▸ Lista de APIs públicas

Uma boa forma de proceder é ir ao github e fazer uma busca por APIs, sempre há muitos resultados em python para trabalhar.

APIs não Oficiais

APIs não oficiais são as não fornecidas pela empresa que disponibiliza os dados, geralmente tem mais funcionalidades que as APIs oficiais. Um exemplo é dado em ▸ API instagram não oficial.

Digitando no Google, `instagram unofficial api` será possível receber uma longa lista de APIs feita por pessoas comuns para poder trocar informações com os servidores do Instagram, do Google, Facebook, etc.

Não há lei que proíba, mas se o Instagram descobrir pode banir você.

APIs oficiais e não oficiais

APIs Oficiais

Há APIs oficiais e não oficiais, que servem para aprendizagem e treinamento, permitindo uma boa fonte para produzir projetos.

Uma boa lista de APIs se encontra em: ▸ Lista de APIs github

Lista de APIs públicas: ▸ Lista de APIs públicas

Uma boa forma de proceder é ir ao github e fazer uma busca por APIs, sempre há muitos resultados em python para trabalhar.

APIs não Oficiais

APIs não oficiais são as não fornecidas pela empresa que disponibiliza os dados, geralmente tem mais funcionalidades que as APIs oficiais. Um exemplo é dado em ▸ API instagram não oficial.
Digitando no Google, `instagram unofficial api` será possível receber uma longa lista de APIs feita por pessoas comuns para poder trocar informações com os servidores do Instagram, do Google, Facebook, etc.

Não há lei que proíba, mas se o Instagram descobrir pode banir você.

APIs oficiais e não oficiais

APIs Oficiais

Há APIs oficiais e não oficiais, que servem para aprendizagem e treinamento, permitindo uma boa fonte para produzir projetos.

Uma boa lista de APIs se encontra em: ▸ Lista de APIs github

Lista de APIs públicas: ▸ Lista de APIs públicas

Uma boa forma de proceder é ir ao github e fazer uma busca por APIs, sempre há muitos resultados em python para trabalhar.

APIs não Oficiais

APIs não oficiais são as não fornecidas pela empresa que disponibiliza os dados, geralmente tem mais funcionalidades que as APIs oficiais. Um exemplo é dado em ▸ API instagram não oficial.
Digitando no Google, `instagram unofficial api` será possível receber uma longa lista de APIs feita por pessoas comuns para poder trocar informações com os servidores do Instagram, do Google, Facebook, etc.

Não há lei que proíba, mas se o Instragram descobrir pode banir você.

Sumário

Base URL

Por exemplo, para obter cotações de criptomoedas a corretora MercadoBitcoin oferece o serviço de API usando como base:

```
https://www.mercadobitcoin.net/api
```

Se o endereço acima for uma base URL todos as páginas para requisições de arquivos terão esta base e alterações na URL retornarão informações diferentes.

Cada API terá uma documentação e um meio de operação. Para o MercadoBitcoin, se quisermos saber os preços das negociações de bitcoin entre dois momentos Unix, fazemos:

```
https://www.mercadobitcoin.net/api/BTC/trades/1700000000/1706973186/
```

e teremos o retorno de um arquivo com os valores de Bitcoin entre os momentos Unix 1700000000 e 1706973186. Um momento Unix é a quantidade em segundos desde o dia 01/01/1970. ▸ Momento Unix

Base URL

Por exemplo, para obter cotações de criptomoedas a corretora MercadoBitcoin oferece o serviço de API usando como base:

```
https://www.mercadobitcoin.net/api
```

Se o endereço acima for uma base URL todos as páginas para requisições de arquivos terão esta base e alterações na URL retornarão informações diferentes.

Cada API terá uma documentação e um meio de operação. Para o MercadoBitcoin, se quisermos saber os preços das negociações de bitcoin entre dois momentos Unix, fazemos:

```
https://www.mercadobitcoin.net/api/BTC/trades/1700000000/1706973186/
```

e teremos o retorno de um arquivo com os valores de Bitcoin entre os momentos Unix 1700000000 e 1706973186. Um momento Unix é a quantidade em segundos desde o dia 01/01/1970. ▸ Momento Unix

Base URL

Por exemplo, para obter cotações de criptomoedas a corretora MercadoBitcoin oferece o serviço de API usando como base:

```
https://www.mercadobitcoin.net/api
```

Se o endereço acima for uma base URL todos as páginas para requisições de arquivos terão esta base e alterações na URL retornarão informações diferentes.

Cada API terá uma documentação e um meio de operação. Para o MercadoBitcoin, se quisermos saber os preços das negociações de bitcoin entre dois momentos Unix, fazemos:

```
https://www.mercadobitcoin.net/api/BTC/trades/1700000000/1706973186/
```

e teremos o retorno de um arquivo com os valores de Bitcoin entre os momentos Unix 1700000000 e 1706973186. Um momento Unix é a quantidade em segundos desde o dia 01/01/1970. ▸ Momento Unix

Base URL

Por exemplo, para obter cotações de criptomoedas a corretora MercadoBitcoin oferece o serviço de API usando como base:

```
https://www.mercadobitcoin.net/api
```

Se o endereço acima for uma base URL todos as páginas para requisições de arquivos terão esta base e alterações na URL retornarão informações diferentes.

Cada API terá uma documentação e um meio de operação. Para o MercadoBitcoin, se quisermos saber os preços das negociações de bitcoin entre dois momentos Unix, fazemos:

```
https://www.mercadobitcoin.net/api/BTC/trades/1700000000/1706973186/
```

e teremos o retorno de um arquivo com os valores de Bitcoin entre os momentos Unix 1700000000 e 1706973186. Um momento Unix é a quantidade em segundos desde o dia 01/01/1970. ▸ Momento Unix

Base URL

Por exemplo, para obter cotações de criptomoedas a corretora MercadoBitcoin oferece o serviço de API usando como base:

`https://www.mercadobitcoin.net/api`

Se o endereço acima for uma base URL todos as páginas para requisições de arquivos terão esta base e alterações na URL retornarão informações diferentes.

Cada API terá uma documentação e um meio de operação. Para o MercadoBitcoin, se quisermos saber os preços das negociações de bitcoin entre dois momentos Unix, fazemos:

`https://www.mercadobitcoin.net/api/BTC/trades/1700000000/1706973186/`

e teremos o retorno de um arquivo com os valores de Bitcoin entre os momentos Unix 1700000000 e 1706973186. Um momento Unix é a quantidade em segundos desde o dia 01/01/1970. ▸ Momento Unix

Endpoint e recurso

Endpoint é a parte da URL que vem depois da base URL. Em

```
https://www.mercadobitcoin.net/api/BTC/trades/1700000000/1706973186/
```

temos como endpoint `BTC/trades/1700000000/1706973186/`
Em suma um API é composto de uma base URL e um endpoint.

Chamamos de recurso (resource) ao conjunto de informações enviadas e recebidas dentro das APIs.

São chamadas verbos HTTP às ações que os usuários podem remeter à API. Temos 4 verbos:

- get - Obter dados existentes
- post - Criar novo recurso (enviar dados)
- patch (às vezes put) - Atualizar dados
- delete - Excluir dados

Endpoint e recurso

Endpoint é a parte da URL que vem depois da base URL. Em

```
https://www.mercadobitcoin.net/api/BTC/trades/1700000000/1706973186/
```

temos como endpoint `BTC/trades/1700000000/1706973186/`
Em suma um API é composto de uma base URL e um endpoint.

Chamamos de recurso (resource) ao conjunto de informações enviadas e recebidas dentro das APIs.

São chamadas verbos HTTP às ações que os usuários podem remeter à API.
Temos 4 verbos:

- get - Obter dados existentes
- post - Criar novo recurso (enviar dados)
- patch (às vezes put) - Atualizar dados
- delete - Excluir dados

Endpoint e recurso

Endpoint é a parte da URL que vem depois da base URL. Em

```
https://www.mercadobitcoin.net/api/BTC/trades/1700000000/1706973186/
```

temos como endpoint `BTC/trades/1700000000/1706973186/`
Em suma um API é composto de uma base URL e um endpoint.

Chamamos de recurso (resource) ao conjunto de informações enviadas e recebidas dentro das APIs.

São chamadas verbos HTTP às ações que os usuários podem remeter à API. Temos 4 verbos:

- get - Obter dados existentes
- post - Criar novo recurso (enviar dados)
- patch (às vezes put) - Atualizar dados
- delete - Excluir dados

Endpoint e recurso

Endpoint é a parte da URL que vem depois da base URL. Em

```
https://www.mercadobitcoin.net/api/BTC/trades/1700000000/1706973186/
```

temos como endpoint `BTC/trades/1700000000/1706973186/`
Em suma um API é composto de uma base URL e um endpoint.

Chamamos de recurso (resource) ao conjunto de informações enviadas e recebidas dentro das APIs.

São chamadas verbos HTTP às ações que os usuários podem remeter à API. Temos 4 verbos:

- get - Obter dados existentes
- post - Criar novo recurso (enviar dados)
- patch (às vezes put) - Atualizar dados
- delete - Excluir dados

Endpoint e recurso

Endpoint é a parte da URL que vem depois da base URL. Em

```
https://www.mercadobitcoin.net/api/BTC/trades/1700000000/1706973186/
```

temos como endpoint `BTC/trades/1700000000/1706973186/`
Em suma um API é composto de uma base URL e um endpoint.

Chamamos de recurso (resource) ao conjunto de informações enviadas e recebidas dentro das APIs.

São chamadas verbos HTTP às ações que os usuários podem remeter à API. Temos 4 verbos:

- get - Obter dados existentes
- post - Criar novo recurso (enviar dados)
- patch (às vezes put) - Atualizar dados
- delete - Excluir dados

Endpoint e recurso

Endpoint é a parte da URL que vem depois da base URL. Em

```
https://www.mercadobitcoin.net/api/BTC/trades/1700000000/1706973186/
```

temos como endpoint `BTC/trades/1700000000/1706973186/`
Em suma um API é composto de uma base URL e um endpoint.

Chamamos de recurso (resource) ao conjunto de informações enviadas e recebidas dentro das APIs.

São chamadas verbos HTTP às ações que os usuários podem remeter à API. Temos 4 verbos:

- get - Obter dados existentes
- post - Criar novo recurso (enviar dados)
- patch (às vezes put) - Atualizar dados
- delete - Excluir dados

Status codes

Status codes são os códigos que as APIs retornam para informar o estado de uma requisição, temos a convenção:

- 1xx - Informação
- 2xx - Sucesso
- 3xx - Redirecionamento
- 4xx - Erro no cliente
- 5xx - Erro no servidor

Estes identificadores são atrelados à situação de uma dada requisição. Um erro 400 ou 404 indica erro no servidor, e 200, ou 204 sucesso na operação requisitada.

Se no browser Mozilla em qualquer site apertar `F12` e ir para a aba `network` o usuário tem a informação do que está acontecendo no momento. Se com a aba `network` aberta apertar `F5` terá acesso às informações sobre requisições feitas pelo cliente e os *status codes* referentes.

Status codes

Status codes são os códigos que as APIs retornam para informar o estado de uma requisição, temos a convenção:

- 1xx - Informação
- 2xx - Sucesso
- 3xx - Redirecionamento
- 4xx - Erro no cliente
- 5xx - Erro no servidor

Estes identificadores são atrelados à situação de uma dada requisição. Um erro 400 ou 404 indica erro no servidor, e 200, ou 204 sucesso na operação requisitada.

Se no browser Mozilla em qualquer site apertar `F12` e ir para a aba `network` o usuário tem a informação do que está acontecendo no momento. Se com a aba `network` aberta apertar `F5` terá acesso às informações sobre requisições feitas pelo cliente e os *status codes* referentes.

Status codes

Status codes são os códigos que as APIs retornam para informar o estado de uma requisição, temos a convenção:

- 1xx - Informação
- 2xx - Sucesso
- 3xx - Redirecionamento
- 4xx - Erro no cliente
- 5xx - Erro no servidor

Estes identificadores são atrelados à situação de uma dada requisição. Um erro 400 ou 404 indica erro no servidor, e 200, ou 204 sucesso na operação requisitada.

Se no browser Mozilla em qualquer site apertar `F12` e ir para a aba `network` o usuário tem a informação do que está acontecendo no momento. Se com a aba `network` aberta apertar `F5` terá acesso às informações sobre requisições feitas pelo cliente e os *status codes* referentes.

Status codes

Status codes são os códigos que as APIs retornam para informar o estado de uma requisição, temos a convenção:

- 1xx - Informação
- 2xx - Sucesso
- 3xx - Redirecionamento
- 4xx - Erro no cliente
- 5xx - Erro no servidor

Estes identificadores são atrelados à situação de uma dada requisição. Um erro 400 ou 404 indica erro no servidor, e 200, ou 204 sucesso na operação requisitada.

Se no browser Mozilla em qualquer site apertar `F12` e ir para a aba `network` o usuário tem a informação do que está acontecendo no momento. Se com a aba `network` aberta apertar `F5` terá acesso às informações sobre requisições feitas pelo cliente e os *status codes* referentes.

Status codes

Status codes são os códigos que as APIs retornam para informar o estado de uma requisição, temos a convenção:

- 1xx - Informação
- 2xx - Sucesso
- 3xx - Redirecionamento
- 4xx - Erro no cliente
- 5xx - Erro no servidor

Estes identificadores são atrelados à situação de uma dada requisição. Um erro 400 ou 404 indica erro no servidor, e 200, ou 204 sucesso na operação requisitada.

Se no browser Mozilla em qualquer site apertar `F12` e ir para a aba `network` o usuário tem a informação do que está acontecendo no momento. Se com a aba `network` aberta apertar `F5` terá acesso às informações sobre requisições feitas pelo cliente e os *status codes* referentes.

Status codes

Status codes são os códigos que as APIs retornam para informar o estado de uma requisição, temos a convenção:

- 1xx - Informação
- 2xx - Sucesso
- 3xx - Redirecionamento
- 4xx - Erro no cliente
- 5xx - Erro no servidor

Estes identificadores são atrelados à situação de uma dada requisição. Um erro 400 ou 404 indica erro no servidor, e 200, ou 204 sucesso na operação requisitada.

Se no browser Mozilla em qualquer site apertar `F12` e ir para a aba `network` o usuário tem a informação do que está acontecendo no momento. Se com a aba `network` aberta apertar `F5` terá acesso às informações sobre requisições feitas pelo cliente e os *status codes* referentes.

Status codes

Status codes são os códigos que as APIs retornam para informar o estado de uma requisição, temos a convenção:

- 1xx - Informação
- 2xx - Sucesso
- 3xx - Redirecionamento
- 4xx - Erro no cliente
- 5xx - Erro no servidor

Estes identificadores são atrelados à situação de uma dada requisição. Um erro 400 ou 404 indica erro no servidor, e 200, ou 204 sucesso na operação requisitada.

Se no browser Mozilla em qualquer site apertar `F12` e ir para a aba `network` o usuário tem a informação do que está acontecendo no momento. Se com a aba `network` aberta apertar `F5` terá acesso às informações sobre requisições feitas pelo cliente e os *status codes* referentes.

Status codes

Status codes são os códigos que as APIs retornam para informar o estado de uma requisição, temos a convenção:

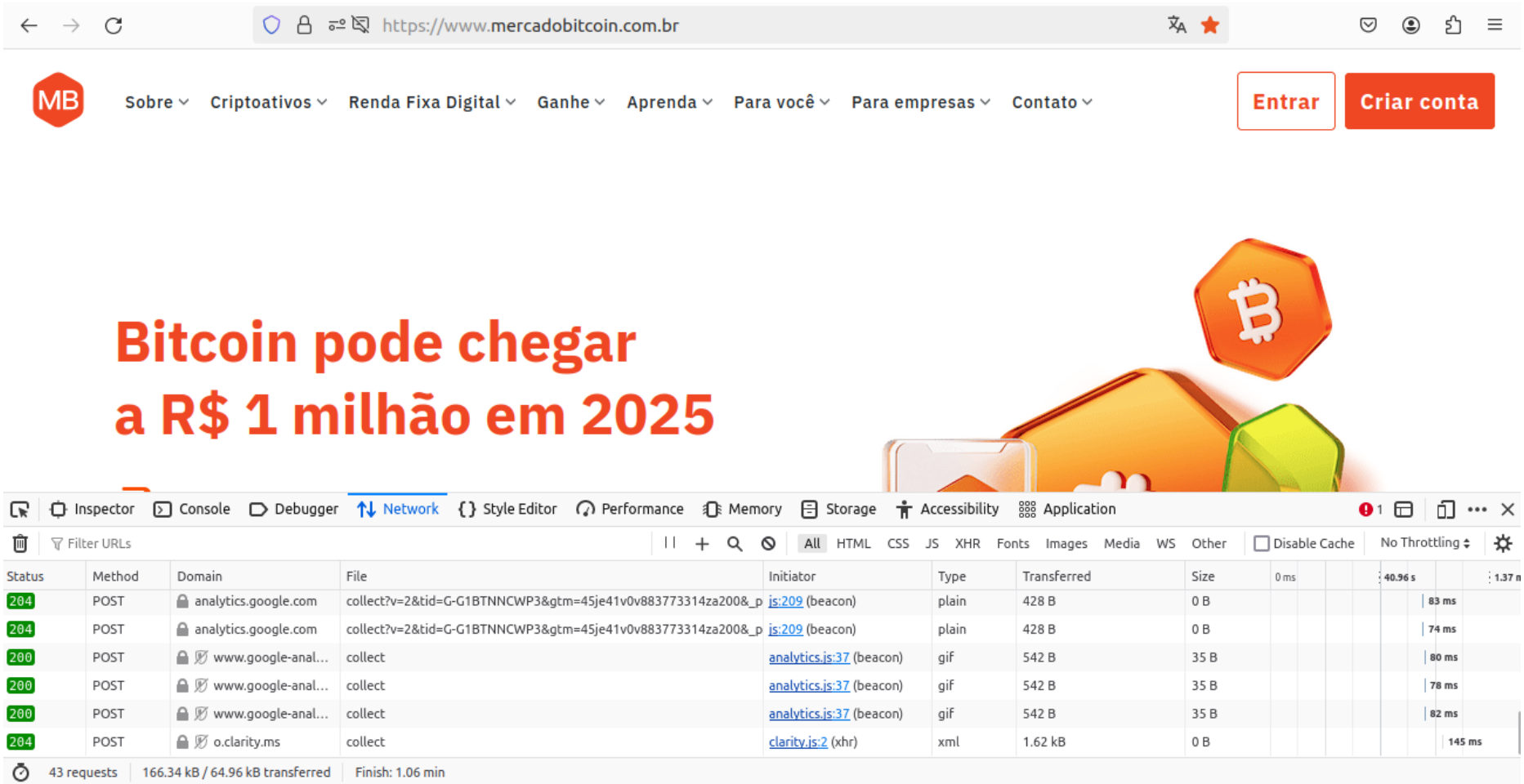
- 1xx - Informação
- 2xx - Sucesso
- 3xx - Redirecionamento
- 4xx - Erro no cliente
- 5xx - Erro no servidor

Estes identificadores são atrelados à situação de uma dada requisição. Um erro 400 ou 404 indica erro no servidor, e 200, ou 204 sucesso na operação requisitada.

Se no browser Mozilla em qualquer site apertar `F12` e ir para a aba `network` o usuário tem a informação do que está acontecendo no momento. Se com a aba `network` aberta apertar `F5` terá acesso às informações sobre requisições feitas pelo cliente e os *status codes* referentes.

Aba network do navegador

Aba network do browser Mozilla



Sumário

Requisições de APIs

A biblioteca `requests` permite requisição de recursos de APIs, por exemplo o código:

```
import requests
url = "https://www.mercadobitcoin.net/api/BTC/trades/1686731660/1686733260/"
requisicao = requests.get(url)
print(requisicao)
```

possui como resposta:

```
<Response [200]>
```

que mostra que a requisição `get` foi aceita.

Para guardar este resultado numa variável que pode ser utilizada, transformamos este arquivo num json usando a biblioteca `json`:

```
import requests, json
url = "https://www.mercadobitcoin.net/api/BTC/trades/1686731660/1686733260/"
requisicao = requests.get(url)
lista1 = requisicao.json()
print(lista1[0])
print("Quantidade de dados:", len(lista1))
```

E a resposta do terminal:

```
{'amount': 7.73e-05, 'date': 1686731766, 'price': 126787.01, 'tid': 15241674,
'type': 'buy'}
Quantidade de dados: 3
```

Requisições de APIs

A biblioteca `requests` permite requisição de recursos de APIs, por exemplo o código:

```
import requests
url = "https://www.mercadobitcoin.net/api/BTC/trades/1686731660/1686733260/"
requisicao = requests.get(url)
print(requisicao)
```

possui como resposta:

```
<Response [200]>
```

que mostra que a requisição `get` foi aceita.

Para guardar este resultado numa variável que pode ser utilizada, transformamos este arquivo num json usando a biblioteca `json`:

```
import requests, json
url = "https://www.mercadobitcoin.net/api/BTC/trades/1686731660/1686733260/"
requisicao = requests.get(url)
lista1 = requisicao.json()
print(lista1[0])
print("Quantidade de dados:", len(lista1))
```

E a resposta do terminal:

```
{'amount': 7.73e-05, 'date': 1686731766, 'price': 126787.01, 'tid': 15241674,
'type': 'buy'}
Quantidade de dados: 3
```

Requisições de APIs

A biblioteca `requests` permite requisição de recursos de APIs, por exemplo o código:

```
import requests
url = "https://www.mercadobitcoin.net/api/BTC/trades/1686731660/1686733260/"
requisicao = requests.get(url)
print(requisicao)
```

possui como resposta:

```
<Response [200]>
```

que mostra que a requisição `get` foi aceita.

Para guardar este resultado numa variável que pode ser utilizada, transformamos este arquivo num json usando a biblioteca `json`:

```
import requests, json
url = "https://www.mercadobitcoin.net/api/BTC/trades/1686731660/1686733260/"
requisicao = requests.get(url)
lista1 = requisicao.json()
print(lista1[0])
print("Quantidade de dados:", len(lista1))
```

E a resposta do terminal:

```
{'amount': 7.73e-05, 'date': 1686731766, 'price': 126787.01, 'tid': 15241674,
'type': 'buy'}
Quantidade de dados: 3
```

Requisições de APIs

A biblioteca `requests` permite requisição de recursos de APIs, por exemplo o código:

```
import requests
url = "https://www.mercadobitcoin.net/api/BTC/trades/1686731660/1686733260/"
requisicao = requests.get(url)
print(requisicao)
```

possui como resposta:

```
<Response [200]>
```

que mostra que a requisição `get` foi aceita.

Para guardar este resultado numa variável que pode ser utilizada, transformamos este arquivo num json usando a biblioteca `json`:

```
import requests, json
url = "https://www.mercadobitcoin.net/api/BTC/trades/1686731660/1686733260/"
requisicao = requests.get(url)
lista1 = requisicao.json()
print(lista1[0])
print("Quantidade de dados:", len(lista1))
```

E a resposta do terminal:

```
{'amount': 7.73e-05, 'date': 1686731766, 'price': 126787.01, 'tid': 15241674,
'type': 'buy'}
Quantidade de dados: 3
```

Requisições de APIs

A biblioteca `requests` permite requisição de recursos de APIs, por exemplo o código:

```
import requests
url = "https://www.mercadobitcoin.net/api/BTC/trades/1686731660/1686733260/"
requisicao = requests.get(url)
print(requisicao)
```

possui como resposta:

```
<Response [200]>
```

que mostra que a requisição `get` foi aceita.

Para guardar este resultado numa variável que pode ser utilizada, transformamos este arquivo num json usando a biblioteca `json`:

```
import requests, json
url = "https://www.mercadobitcoin.net/api/BTC/trades/1686731660/1686733260/"
requisicao = requests.get(url)
lista1 = requisicao.json()
print(lista1[0])
print("Quantidade de dados:", len(lista1))
```

E a resposta do terminal:

```
{'amount': 7.73e-05, 'date': 1686731766, 'price': 126787.01, 'tid': 15241674,
'type': 'buy'}
Quantidade de dados: 3
```

Sumário

Requisições de APIs

Vamos entrar numa api disponível para treinamento, chamada Json Placeholder. ▸ Link
Este site disponibiliza algumas apis para teste. Na seção `resources` uma boa api para trabalhar é a todos (*to do's* - coisas pendentes), que tem o endpoint:
`https://jsonplaceholder.typicode.com/todos`
Vamos fazer requisições com todos os verbos disponibilizados na api.

requests.get com try except

```
import requests, json
from pprint import*

url_base = 'https://jsonplaceholder.typicode.com/'
url_endpoint = 'todos'
url = url_base+url_endpoint

try:
    resultado_get = requests.get(url)
    resultado_get.raise_for_status()
except requests.exceptions.HTTPError as err:
    print("Abortando programa")
    raise SystemExit(err)

resultado_get= resultado_get.json()
for i in range(3):
    print(resultado_get[i])
```

E obtemos como resposta:

```
{'userId': 1, 'id': 1, 'title': 'delectus aut autem', 'completed': False}
{'userId': 1, 'id': 2, 'title': 'quis ut nam facilis et officia qui', 'completed': False}
{'userId': 1, 'id': 3, 'title': 'fugiat veniam minus', 'completed': False}
```

Sumário

Arquivos json

Arquivos json[1] são bastante comuns para guardar dados, principalmente quando se trabalha com dicionários. Os arquivos json são muito parecidos com vetores de dicionários.

```
{
    "nome":"Carol",
    "telefone":"47-992248852",
    "permissoes":"basico",
    "admin":true
}
```

▸ Link

Observe que a variável booleana tem valor `true`, e não `True`. Salienta-se `JSON` não é uma estrutura do Python, e sim do javascript, e segue a sintaxe do javascript. Mas há vantagens:

- JSON é facilmente intelegível para leitura de pessoas.
- JSON é fácil de ler e gravar em disco.
- JSON é portável entre várias linguagens, incluindo Python.

[1] JSON é um acrônimo para javascript object notation.

Arquivos json

Arquivos json[1] são bastante comuns para guardar dados, principalmente quando se trabalha com dicionários. Os arquivos json são muito parecidos com vetores de dicionários.

```
{
    "nome":"Carol",
    "telefone":"47-992248852",
    "permissoes":"basico",
    "admin":true
}
```

▸ Link

Observe que a variável booleana tem valor `true`, e não `True`. Salienta-se `JSON` não é uma estrutura do Python, e sim do javascript, e segue a sintaxe do javascript. Mas há vantagens:

- JSON é facilmente intelegível para leitura de pessoas.
- JSON é fácil de ler e gravar em disco.
- JSON é portável entre várias linguagens, incluindo Python.

[1] JSON é um acrônimo para javascript object notation.

Arquivos json

Arquivos json[1] são bastante comuns para guardar dados, principalmente quando se trabalha com dicionários. Os arquivos json são muito parecidos com vetores de dicionários.

```
{
    "nome":"Carol",
    "telefone":"47-992248852",
    "permissoes":"basico",
    "admin":true
}
```

▸ Link

Observe que a variável booleana tem valor `true`, e não `True`. Salienta-se `JSON` não é uma estrutura do Python, e sim do javascript, e segue a sintaxe do javascript. Mas há vantagens:

- JSON é facilmente intelegível para leitura de pessoas.
- JSON é fácil de ler e gravar em disco.
- JSON é portável entre várias linguagens, incluindo Python.

[1] JSON é um acrônimo para javascript object notation.

Arquivos json

Embora a sintaxe dos arquivos `json` siga a notação de `javascript`, é muito próxima de como se usa dicionários e listas aninhadas no python. As diferenças são:

- `JSON` só aceita aspas duplas para strings.
- `JSON` usa `null` onde python usa `None`.
- `JSON` usa `true` e `false` com letras minúsculas.
- `JSON` aceita apenas strings como chaves, enquanto no python as chaves dos dicionários pode ter qualquer tipo.

Podemos construir um arquivo `json` com somente uma entrada ou com entradas aninhadas.

```
{
    "nome":"Carol",
    "telefone":"47-992248852",
    "permissoes":["basico","intermediario", "administrador"],
    "admin":true
}
```

Podemos ter níveis mais profundos para incluirmos mais informações.

Arquivos json

Embora a sintaxe dos arquivos `json` siga a notação de `javascript`, é muito próxima de como se usa dicionários e listas aninhadas no python. As diferenças são:

- `JSON` só aceita aspas duplas para strings.
- `JSON` usa `null` onde python usa `None`.
- `JSON` usa `true` e `false` com letras minúsculas.
- `JSON` aceita apenas strings como chaves, enquanto no python as chaves dos dicionários pode ter qualquer tipo.

Podemos construir um arquivo `json` com somente uma entrada ou com entradas aninhadas.

```
{
    "nome":"Carol",
    "telefone":"47-992248852",
    "permissoes":["basico","intermediario", "administrador"],
    "admin":true
}
```

Podemos ter níveis mais profundos para incluirmos mais informações.

Arquivos json

Embora a sintaxe dos arquivos `json` siga a notação de `javascript`, é muito próxima de como se usa dicionários e listas aninhadas no python. As diferenças são:

- `JSON` só aceita aspas duplas para strings.
- `JSON` usa `null` onde python usa `None`.
- `JSON` usa `true` e `false` com letras minúsculas.
- `JSON` aceita apenas strings como chaves, enquanto no python as chaves dos dicionários pode ter qualquer tipo.

Podemos construir um arquivo `json` com somente uma entrada ou com entradas aninhadas.

```
{
    "nome":"Carol",
    "telefone":"47-992248852",
    "permissoes":["basico","intermediario", "administrador"],
    "admin":true
}
```

Podemos ter níveis mais profundos para incluirmos mais informações.

Arquivos json

Embora a sintaxe dos arquivos `json` siga a notação de `javascript`, é muito próxima de como se usa dicionários e listas aninhadas no python. As diferenças são:

- `JSON` só aceita aspas duplas para strings.
- `JSON` usa `null` onde python usa `None`.
- `JSON` usa `true` e `false` com letras minúsculas.
- `JSON` aceita apenas strings como chaves, enquanto no python as chaves dos dicionários pode ter qualquer tipo.

Podemos construir um arquivo `json` com somente uma entrada ou com entradas aninhadas.

```
{
    "nome":"Carol",
    "telefone":"47-992248852",
    "permissoes":["basico","intermediario", "administrador"],
    "admin":true
}
```

Podemos ter níveis mais profundos para incluirmos mais informações.

Arquivos json

Embora a sintaxe dos arquivos `json` siga a notação de `javascript`, é muito próxima de como se usa dicionários e listas aninhadas no python. As diferenças são:

- `JSON` só aceita aspas duplas para strings.
- `JSON` usa `null` onde python usa `None`.
- `JSON` usa `true` e `false` com letras minúsculas.
- `JSON` aceita apenas strings como chaves, enquanto no python as chaves dos dicionários pode ter qualquer tipo.

Podemos construir um arquivo `json` com somente uma entrada ou com entradas aninhadas.

```
{
    "nome":"Carol",
    "telefone":"47-992248852",
    "permissoes":["basico","intermediario", "administrador"],
    "admin":true
}
```

Podemos ter níveis mais profundos para incluirmos mais informações.

Arquivos json

Embora a sintaxe dos arquivos `json` siga a notação de `javascript`, é muito próxima de como se usa dicionários e listas aninhadas no python. As diferenças são:

- `JSON` só aceita aspas duplas para strings.
- `JSON` usa `null` onde python usa `None`.
- `JSON` usa `true` e `false` com letras minúsculas.
- `JSON` aceita apenas strings como chaves, enquanto no python as chaves dos dicionários pode ter qualquer tipo.

Podemos construir um arquivo `json` com somente uma entrada ou com entradas aninhadas.

```
{
    "nome":"Carol",
    "telefone":"47-992248852",
    "permissoes":["basico","intermediario", "administrador"],
    "admin":true
}
```

Podemos ter níveis mais profundos para incluirmos mais informações.

Arquivos json

Embora a sintaxe dos arquivos `json` siga a notação de `javascript`, é muito próxima de como se usa dicionários e listas aninhadas no python. As diferenças são:

- `JSON` só aceita aspas duplas para strings.
- `JSON` usa `null` onde python usa `None`.
- `JSON` usa `true` e `false` com letras minúsculas.
- `JSON` aceita apenas strings como chaves, enquanto no python as chaves dos dicionários pode ter qualquer tipo.

Podemos construir um arquivo `json` com somente uma entrada ou com entradas aninhadas.

```
{
    "nome":"Carol",
    "telefone":"47-992248852",
    "permissoes":["basico","intermediario", "administrador"],
    "admin":true
}
```

Podemos ter níveis mais profundos para incluirmos mais informações.

Arquivos json

Embora a sintaxe dos arquivos `json` siga a notação de `javascript`, é muito próxima de como se usa dicionários e listas aninhadas no python. As diferenças são:

- `JSON` só aceita aspas duplas para strings.
- `JSON` usa `null` onde python usa `None`.
- `JSON` usa `true` e `false` com letras minúsculas.
- `JSON` aceita apenas strings como chaves, enquanto no python as chaves dos dicionários pode ter qualquer tipo.

Podemos construir um arquivo `json` com somente uma entrada ou com entradas aninhadas.

```
{
    "nome":"Carol",
    "telefone":"47-992248852",
    "permissoes":["basico","intermediario", "administrador"],
    "admin":true
}
```

Podemos ter níveis mais profundos para incluirmos mais informações.

Arquivos json

Dois arquivos `json` mais profundos:

```
{
    "usuários": [
        {
            "nome": "Carol",
            "telefone": "47-99224-8852",
            "permissões": [
                "basico",
                "intermediário",
                "administrador"
            ],
            "admin": true
        },
        {
            "nome": "Douglas",
            "telefone": "47-99224-8852",
            "permissões": [
                "basico",
                "intermediário"
            ],
            "admin": false
        }
    ]
}
```

```
{
    "usuários": [
        {
            "nome": "Carol",
            "telefone": "47-99224-8852",
            "permissões": [
                "basico",
                "intermediário",
                "administrador"
            ],
            "admin": true,
            "plano": {
                "nome": "basico",
                "preco": "R$20,00"
            }
        },
        {
            "nome": "Douglas",
            "telefone": "47-99224-8852",
            "permissões": [
                "basico",
                "intermediário"
            ],
            "admin": true,
            "plano": {
                "nome": "pro",
                "preco": "R$50,00"
            }
        }
    ]
}
```

Arquivos json

Dois arquivos `json` mais profundos:

```
{
    "usuários": [
        {
            "nome": "Carol",
            "telefone": "47-99224-8852",
            "permissões": [
                "basico",
                "intermediário",
                "administrador"
            ],
            "admin": true
        },
        {
            "nome": "Douglas",
            "telefone": "47-99224-8852",
            "permissões": [
                "basico",
                "intermediário"
            ],
            "admin": false
        }
    ]
}
```

```
{
    "usuários": [
        {
            "nome": "Carol",
            "telefone": "47-99224-8852",
            "permissões": [
                "basico",
                "intermediário",
                "administrador"
            ],
            "admin": true,
            "plano": {
                "nome": "basico",
                "preco": "R$20,00"
            }
        },
        {
            "nome": "Douglas",
            "telefone": "47-99224-8852",
            "permissões": [
                "basico",
                "intermediário"
            ],
            "admin": true,
            "plano": {
                "nome": "pro",
                "preco": "R$50,00"
            }
        }
    ]
}
```

Arquivos json

Dois arquivos `json` mais profundos:

```
{
    "usuários": [
        {
            "nome": "Carol",
            "telefone": "47-99224-8852",
            "permissões": [
                "basico",
                "intermediário",
                "administrador"
            ],
            "admin": true
        },
        {
            "nome": "Douglas",
            "telefone": "47-99224-8852",
            "permissões": [
                "basico",
                "intermediário"
            ],
            "admin": false
        }
    ]
}
```

```
{
    "usuários": [
        {
            "nome": "Carol",
            "telefone": "47-99224-8852",
            "permissões": [
                "basico",
                "intermediário",
                "administrador"
            ],
            "admin": true,
            "plano": {
                "nome": "basico",
                "preco": "R$20,00"
            }
        },
        {
            "nome": "Douglas",
            "telefone": "47-99224-8852",
            "permissões": [
                "basico",
                "intermediário"
            ],
            "admin": true,
            "plano": {
                "nome": "pro",
                "preco": "R$50,00"
            }
        }
    ]
}
```

Sumário

Criação e gravação de arquivos json

Para este estudo precisamos trabalhar com abertura e fechamento básicos de arquivos, que serão objeto de aula completa futura.

Observe a estrutura de dados abaixo:

```
#Dicionario nativo de python, ou outra estrutura
comp1 = {"marca":"Dell",
    "preço":7000
    }
comp2 = {"marca":"Pichau",
    "preço":8000
    }
#Juntando em uma lista os dois dicionarios
lista_comp = [comp1, comp2]
#Cria arquivo para gravação
nome_arquivo = "Comp.json"
Arq1 = open(nome_arquivo, "w", encoding = 'utf-8')
json.dump(lista_comp, Arq1)#Codifica no formato Json e grava no arquivo.
Arq1.close()
```

Assim é criado um arquivo chamado `Comp.json` que grava a estrutura de Python, que no caso é uma lista de dicionários.

Sumário

Leitura de arquivos json

A conversão de estruturas de dados obtidas de arquivos json pode ser feita usando a função `json.load(ponteiro_arquivo)`, e então o arquivo é convertido na estrutura de dados equivalente em Python.
Aproveitando o que foi feito no slide anterior:

```
nome_arquivo = "Comp.json"
Arq_L = open(nome_arquivo, "r", encoding= 'utf-8')
dados = json.load(Arq_L) #Converte Arq_L para tipo em Python
print(dados)
Arq_L.close()
```

E o arquivo foi convertido numa variável `dados`, que é uma lista de dicionários, conforme guardado em arquivo.