

***INTELIGÊNCIA ARTIFICIAL***

**UA02 / LABORATÓRIO # 2**

**O código:**

import pandas as pd

# Dados sobre inflação do BACEN (Banco Central do Brasil)

import requests

url = 'https://api.bcb.gov.br/dados/serie/bcdata.sgs.433/dados?formato=json'

response = requests.get(url).json()

response

**Explicando o código:**

1.) Importação de Bibliotecas

import pandas as pd

- Aqui estamos importando a biblioteca Pandas, que é amplamente usada para análise e manipulação de dados.

- O “as pd” é um alias (apelido) que permite usarmos “pd” no lugar de “pandas” no código.

- Exemplo: Em vez de escrever “pandas.read\_csv()”, escrevemos “pd.read\_csv()”

2.) Fonte de Dados via API

import requests

- A biblioteca “requests” é importada para fazer solicitações HTTP, o que nos permite acessar APIs web.

- Requests é como um navegador dentro do Python.

- Serve para acessar páginas web e APIs (serviços de dados online).

3.) Configuração da URL da API

url = 'https://api.bcb.gov.br/dados/serie/bcdata.sgs.433/dados?formato=json'

- `url` é o endereço onde os dados estão guardados.

- Esta URL específica contém dados do IPCA (inflação oficial do Brasil).

- `bcdata.sgs.433` é o código do IPCA no Banco Central.

- O parâmetro `formato=json` pede os dados em formato JSON (um formato fácil para programas entenderem).

4.) Requisição e Conversão dos Dados

response = requests.get(url).json()

Vamos dividir em partes:

1. `requests.get(url)` - Faz o mesmo que digitar um endereço no navegador

2. `.json()` - Converte a resposta para um formato que o Python entende.

5.) Saída (Resposta)

response

- Isso simplesmente exibe o conteúdo da variável `response`.

- Mostra o conteúdo que veio da API.

- Normalmente é uma lista de informações mensais.

A variável `response` vai conter algo assim:

[

{'data': '01/01/2010', 'valor': '0.75'},

{'data': '01/02/2010', 'valor': '0.78'},

{'data': '01/03/2010', 'valor': '0.52'},

# ... e assim por diante com todos os meses

]

Cada item é um mês com:

- `data`: quando foi medido (dia/mês/ano).

- `valor`: percentual da inflação naquele mês.

O que o código faz:

O código coleta dados históricos de inflação (IPCA) diretamente da API do Banco Central do Brasil e os converte para um formato que pode ser manipulado em Python.

Saída esperada:

A variável `response` conterá uma lista de dicionários, onde cada dicionário representa uma observação mensal do IPCA. Cada entrada tem normalmente este formato:

{

"data": "01/01/2023",

"valor": "0.53"

}

- "data": data da observação (no formato dd/mm/aaaa).

- "valor": valor do IPCA no período (em percentual).

Exemplo de como a saída pode parecer (apenas primeiros itens):

[

{'data': '01/01/1979', 'valor': '4.48'},

{'data': '01/02/1979', 'valor': '3.67'},

{'data': '01/03/1979', 'valor': '2.24'},

...

]

**Enunciado para Prova de IA**

**Título:** Análise de Dados Econômicos usando Python

Você foi contratada como Cientista de Dados para analisar indicadores econômicos do Brasil. Seu primeiro desafio é coletar e interpretar dados oficiais de inflação (IPCA) disponibilizados pelo Banco Central.

**Tarefa:**

1.) Escreva um código em Python que:

- Acesse a API do Banco Central para obter a série histórica do IPCA (código SGS 433).

- Converta os dados recebidos em um formato adequado para análise.

- Exiba os dados brutos retornados pela API.

2.) Explique:

- Por que cada biblioteca foi utilizada.

- Qual a função de cada parte do código.

- Como esse código poderia ser expandido para uma análise mais completa.

**Resposta Esperada (Código + Explicação)**

**Código:**

# Para manipulação de dados em formato de tabela  
import pandas as pd

# Para fazer requisições HTTP e acessar APIs  
import requests

# Definindo a URL da API do Banco Central (IPCA - Série 433)

url = 'https://api.bcb.gov.br/dados/serie/bcdata.sgs.433/dados?formato=json'

# Fazendo a requisição GET e convertendo a resposta para JSON

response = requests.get(url).json()

# Exibindo os dados brutos retornados

response

**Explicação:**

1.) Bibliotecas Utilizadas

- `pandas` (`pd`):

- Biblioteca essencial para análise de dados em Python.

- Permite trabalhar com estruturas tabulares (DataFrames), facilitando filtros, cálculos e visualizações.

- No código atual, ainda não estamos usando diretamente, mas seria útil para transformar `response` em uma tabela.

**- `requests`:**

- Usada para fazer chamadas a APIs web (como a do Banco Central).

- O método `requests.get()` envia uma solicitação HTTP GET para a URL especificada.

- `.json()` converte a resposta (que vem em formato JSON) para um objeto Python (lista/dicionário).

2.) Funcionamento do Código

**- `url = '...'`**

- Define o endpoint da API que contém os dados do IPCA (SGS 433).

- O parâmetro `formato=json` garante que os dados venham em JSON.

**- `response = requests.get(url).json()`**

- Faz a requisição à API e converte a resposta em uma estrutura Python.

- Se a API retornar sucesso (status 200), `response` será uma lista de dicionários, onde cada entrada representa um mês com `data` e `valor`.

**- `response`**

- Mostra os dados brutos no console, útil para verificar se a extração funcionou.

3.) Como Expandir para Análise Completa?

- Transformar em DataFrame:

df = pd.DataFrame(response)

# Converte valores para numérico  
 df['valor'] = df['valor'].astype(float)

- Calcular estatísticas:

media\_inflacao = df['valor'].mean()

max\_inflacao = df['valor'].max()

- Plotar gráficos (usando `matplotlib` ou `seaborn`):

import matplotlib.pyplot as plt

df.plot(x='data', y='valor', kind='line')

plt.title('Variação do IPCA ao Longo do Tempo')

plt.show()

**O código:**

# Dados do Site Oficial do National Art Gallery

import requests

from bs4 import BeautifulSoup

page = requests.get('https://web.archive.org/web/20121007172955/https://www.nga.gov/collection/anZ1.htm')

soup = BeautifulSoup(page.text, 'html.parser')

last\_links = soup.find(class\_='AlphaNav')

last\_links.decompose()

artist\_name\_list = soup.find(class\_='BodyText')

artist\_name\_list\_items = artist\_name\_list.find\_all('a')

# Usar .contents para pegar as tags <a> filhas

for artist\_name in artist\_name\_list\_items:

names = artist\_name.contents[0]

print(names)

1.) Importação das bibliotecas:

import requests

from bs4 import BeautifulSoup

- `requests`: Biblioteca para fazer requisições HTTP (usada para baixar o conteúdo da página web).

- `BeautifulSoup` (bs4): Biblioteca para extrair (parsear) dados de HTML/XML, permitindo navegar pela estrutura do documento.

2.) Baixando a página web:

page = requests.get('https://web.archive.org/web/20121007172955/https://www.nga.gov/collection/anZ1.htm')

- `requests.get()`: Faz uma requisição HTTP \*\*GET\*\* para a URL fornecida (nesse caso, um arquivo do Wayback Machine).

- `page`: Armazena a resposta da requisição (incluindo HTML, status, etc.).

3.) Parseando o HTML com BeautifulSoup:

soup = BeautifulSoup(page.text, 'html.parser')

- `page.text`: Extrai o conteúdo HTML da resposta.

- `BeautifulSoup(..., 'html.parser')`: Transforma o HTML em um objeto manipulável pelo BeautifulSoup, permitindo buscar elementos por tags, classes, IDs, etc.

4.) Removendo elementos indesejados:

last\_links = soup.find(class\_='AlphaNav')

last\_links.decompose()

- `soup.find(class\_='AlphaNav')`: Procura um elemento HTML com a classe `AlphaNav` (geralmente uma barra de navegação alfabética).

- `decompose()`: Remove esse elemento da árvore HTML, pois não queremos que ele apareça nos resultados.

5.) Buscando a lista de artistas:

artist\_name\_list = soup.find(class\_='BodyText')

artist\_name\_list\_items = artist\_name\_list.find\_all('a')

- `soup.find(class\_='BodyText')`: Encontra o elemento que contém o texto principal da página (onde estão os nomes dos artistas).

- `find\_all('a')`: Extrai todas as tags `<a>` (links) dentro desse elemento, pois cada nome de artista está dentro de um link.

6.) Extraindo e imprimindo os nomes dos artistas

for artist\_name in artist\_name\_list\_items:

names = artist\_name.contents[0]

print(names)

- Loop `for`: Itera sobre cada link (`<a>`) encontrado.

- `artist\_name.contents[0]`: Pega o primeiro conteúdo dentro da tag `<a>`, que é o nome do artista em formato de texto.

- `print(names)`: Exibe cada nome no console.

**Saída do código**

A lista impressa contém os nomes dos artistas que começam com "Z" (já que a URL acessada é `anZ1.htm`, referente à letra Z).

Zabaglia, Niccola

Zaccone, Fabian

Zadkine, Ossip

...

Zao Wou-Ki

**Resumo do que o código faz**

1.) Baixa o HTML de uma página de artistas da National Gallery.

2.) Remove elementos desnecessários (navegação alfabética).

3.) Extrai todos os links (`<a>`) que contêm nomes de artistas.

4.) Pega o texto dentro desses links e \*\*imprime\*\* os nomes.

5.) Para pegar outras letras, basta mudar a URL (ex: `anA1.htm` para artistas com "A").