Desafio 1 - Consumo de APIs

Objetivos:

Se desenvolver através de atividade prática como consumir APIs - **Application Programming Interface** (Interface de Programação de Aplicação). Recurso extremamente utilizado na área de Engenheiro de Dados. Consiste em puxar as informações de alguma fonte de dados e trazê-la para dentro de sua infraestrutura.

Conhecimento exigido:

* Lógica de programação
  + Curso de Lógica: <https://www.youtube.com/watch?v=iF2MdbrTiBM>
* Linguagem de programação (**Python**, JavaScript, Java e etc)
  + Curso de Python: <https://www.youtube.com/watch?v=GQpQha2Mfpg>
  + <https://web.dio.me/lives/como-aprender-a-programar-python-do-zero-getting-started-26>
  + <https://www.udemy.com/course/programacao-python-do-basico-ao-avancado/>
  + <https://www.youtube.com/watch?v=3J1xBL7zJXg&ab_channel=ProgramadorPython>

* APIs
  + <https://www.youtube.com/watch?v=ghTrp1x_1As>

Softwares Necessários:

* IDE Visual Studio: <https://code.visualstudio.com/>
* Interpretador e Plugins do VS:
  + [Python - Visual Studio Marketplace](https://marketplace.visualstudio.com/items?itemName=ms-python.python)
  + [Code Runner - Visual Studio Marketplace](https://marketplace.visualstudio.com/items?itemName=formulahendry.code-runner)
  + [Flake8 - Visual Studio Marketplace](https://marketplace.visualstudio.com/items?itemName=ms-python.flake8)
* Plugin Chrome
  + [https://chrome.google.com/webstore/detail/json-viewer/gbmdgpbipfallnflgajpaliibnhdgobh](https://chrome.google.com/webstore/detail/json-viewer/gbmdgpbipfallnflgajpaliibnhdgobh?utm_source=ext_sidebar&hl=fr)

Bibliotecas Utilizadas:

* Requests: <https://pypi.org/project/requests/>
* Grequests: <https://pypi.org/project/grequests/>

Desafio:

**Descrição**

Eu como desenvolvedor tenho a necessidade de consumir algumas informações de uma API voltada para dados de Pokémons e assim realizar algumas extrações dessas informações.

**Recursos do Desafio**

Endpoint da API Principal:  <https://pokeapi.co/docs/v2>

Endpoint da API Species: <https://pokeapi.co/api/v2/pokemon-species/>

Volume de dados: Puxar a primeira geração

Formato das Informações:

{

"id": 1,

"name": "bulbasaur",

"height": 7,

"weight": 69,

"types":"grass/poison", types": [ { "slot": 1, "type": { "name": "grass",

"moves","azor-wind/cut/C", (PODE ESTOURAR OS CARACTERES) LIMITAR A 3 / NÃO ACEITAR NULOS

"power": "80/NULL/30", LIMITAR A 3 / NÃO ACEITAR NULOS

"hp" : 45,

"attack" :49,

"defense" : 49

"is\_legendary":false

}

Gerações:

|  |  |  |  |
| --- | --- | --- | --- |
| id | name | start | end |
| 1 | Kanto | 1 | 151 |
| 2 | Johto | 152 | 251 |
| 3 | Hoenn | 252 | 386 |
| 4 | Sinnoh | 387 | 493 |
| 5 | Unova | 494 | 649 |
| 6 | Kalos | 650 | 721 |
| 7 | Alola | 722 | 809 |
| 8 | Galar/Hisui | 810 | 905 |
| 9 | Paldea | 906 | 1015 |

Passos:

1 - Desenvolver uma função que faça a requisição na API do Pokemon, onde terá como assinatura do método a seguinte linha:

def grequests\_pokemon\_api(start:int, end:int):

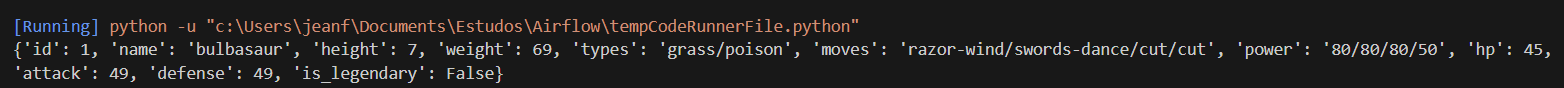
    #start - primeiro pokemon da lista

    #end - ultimo pokemon da lista

    #exemplo def grequests\_pokemon\_api(1, 151):

2 - Prepare os dados da consulta na API e transforme eles em: DataFrame, List, Class ou Dictionary. Qualquer uma das opções de armazenamento de estruturas.

3 - Imprima o resultado dessa consulta com os Pokemons da primeira geração.



Dica: No início reduza o escopo da consulta para **1** Pokémon para não perder tempo puxando muitos dados.

4 - Saída esperada:

{

   "id":1,

   "name":"bulbasaur",

   "height":7,

   "weight":69,

   "types":"grass/poison",

   "moves":"razor-wind/swords-dance/cut/cut",

   "power":"80/80/80/50",

   "hp":45,

   "attack":49,

   "defense":49,

   "is\_legendary":false

}

Extras:

* Otimizar a consulta para o menor tempo possível:
  + Grequests
  + Pesquisar estrutura menos recurso de memória.
* Colocar logs de acompanhamento na execução.
  + import logging
* Realizar testes unitários.
  + PyTest
  + Unittest

Desafio 2 - Consumo de APIs

Objetivos:

Se desenvolver através de atividade prática como consumir APIs - **Application Programming Interface** (Interface de Programação de Aplicação). Recurso extremamente utilizado na área de Engenheiro de Dados. Consiste em puxar as informações de alguma fonte de dados e trazê-la para dentro de sua infraestrutura.

Conhecimento exigido:

* Lógica de programação
  + Curso de Lógica: <https://www.youtube.com/watch?v=iF2MdbrTiBM>
* Linguagem de programação (**Python**, JavaScript, Java e etc)
  + Curso de Python: <https://www.youtube.com/watch?v=GQpQha2Mfpg>
  + <https://web.dio.me/lives/como-aprender-a-programar-python-do-zero-getting-started-26>
  + <https://www.udemy.com/course/programacao-python-do-basico-ao-avancado/>
  + <https://www.youtube.com/watch?v=3J1xBL7zJXg&ab_channel=ProgramadorPython>
* APIs
  + <https://www.youtube.com/watch?v=ghTrp1x_1As>

Softwares Necessários:

* IDE Visual Studio: <https://code.visualstudio.com/>
* Interpretador e Plugins do VS:
  + [Python - Visual Studio Marketplace](https://marketplace.visualstudio.com/items?itemName=ms-python.python)
  + [Code Runner - Visual Studio Marketplace](https://marketplace.visualstudio.com/items?itemName=formulahendry.code-runner)
  + [Flake8 - Visual Studio Marketplace](https://marketplace.visualstudio.com/items?itemName=ms-python.flake8)
* Anki: <https://apps.ankiweb.net/>
* Ankidroid: <https://play.google.com/store/apps/details?id=com.ichi2.anki&hl=pt>

Bibliotecas Utilizadas:

* Googletrans: <https://pypi.org/project/googletrans/>
* gTTS: <https://pypi.org/project/gTTS/>
* Elevenlabs: <https://github.com/elevenlabs/elevenlabs-python>
* texttospeech: <https://cloud.google.com/python/docs/reference/texttospeech/latest>

Links Úteis:

* <https://docs.ankiweb.net/files.html>
* <https://forums.ankiweb.net/t/change-text-color-to-all-white-for-every-card-in-the-deck-using-find-and-replace/31863/5>

Desafio:

**Descrição**

Eu como desenvolvedor tenho a necessidade de construir um script que tenha como resultado um arquivo .csv que seja capaz de ser interpretado pelo aplicativo Anki. Essa solução terá como entrada um documento de texto .txt com frases em um idioma escolhido como Inglês com pelo menos 50 frases mais utilizadas no dia a dia (Usar ChatGPT) para gerar, depois de lido esse .txt realizar a tradução em tempo de execução para português e também criar o speech voice para cada frase com utilização de IA para vozes neurais. Esses dados vão ser lidos no Anki e lá serão gerados flashcards para o estudo do idioma escolhido.

**Recursos do Desafio**

Usar alguma das libs do python (gTTS, Elevenlabs, texttospeech) para transformar texto em vóz, utilizar a API do Google Googletrans para tradução em tempo de execução do texto.

Exemplo .TXT

Je suis ingénieur de données.

Tu es compétent en analyse de données.

Il est un expert en traitement de données.

Schema do .CSV

|  |  |  |  |
| --- | --- | --- | --- |
| french | portuguese | sound | tag |
| Je suis ingénieur de données. | Sou engenheiro de dados | [sound:Je suis ingnieur de donnes.mp3] | (Être) (Ser/estar) (Verbo) |

Passos:

1. Crie o arquivo de texto contendo as frases no idioma escolhido, use o chatGPT ou o Bard para criar, peça 50 frases mais comumente utilizadas.
2. Faça a leitura linha a linha desse arquivo se preocupando com a codificação(encode/decode) dos caracteres para não ter problemas no speech voice.
3. Faça a tradução da frase usando o Googletrans também se preocupando com os caracteres.
4. Gere o áudio a partir da frase usando uma das libs(gTTS, Elevenlabs, texttospeech)
5. Gere a nomenclatura do áudio para ser interpretada no Anki [sound:<nome\_arquivo>]
6. Salve essas strings no .csv respeitando o formato [french, portuguese, sound, tag]

Dica: No início reduza o escopo para uma frase para não perder tempo puxando muitos dados.

4 - Saída esperada:

Je suis ingénieur de données.;Sou engenheiro de dados.;[sound:Je suis ingnieur de donnes.mp3]

Extras:

* Otimizar a qualidade da voz com IA
* Criar processo com 3 velocidades de áudio easy, normal, hard
* Integrar com chatGPT pra receber as frases pela API.
* Colocar logs de acompanhamento na execução.
  + import logging
* Realizar testes unitários.
  + PyTest
  + Unittest
* Colocar estilos no Anki

Exemplo:

FRONT

<h1>{{word}}</h1>

STYLE

.card {

 font-family: -apple-system, BlinkMacSystemFont, Helvetica, Arial, sans-serif;

 font-size: 2em;

 background-color: #fafafa;

 max-width: 40rem;

 padding: 2rem;

 margin: auto;

 color: #222;

 }

BACK

{{FrontSide}}

<hr />

{{definition}}

Resultados:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente