

Atividades do Módulo 4 - QA NA PRÁTICA



Esse [repositório](#) é dedicado às atividades realizadas durante o Módulo 4 - M4 - LÓGICA DE PROGRAMAÇÃO COM PYTHON do curso de Quality Assurance oferecido pelo [Instituto Jogajunto](#).

💡 Descrição da 28ª Atividade: 🌟

💡 EM SQUAD Faça um ambiente virtual, instale o request e faça o código com os seguintes requisitos:

- Tenha uma estrutura de dicionário com nome e cep de cada integrante. Essa estrutura deverá ser salva em uma variável apenas;
- Faça uma requisição e imprima o nome e a cidade de cada integrante do squad;
- Gere um arquivo chamado requirements.txt que contenha todas as dependências do seu projeto.

Ao final, suba a atividade em seu github.

```
import requests
import os
import subprocess

integrantes = {
    "Leanderson": "06412-140",
    "Beatrix Souza": "01302-000",
    "Bruno Soares": "70002-900",
    "Rebeca Borges": "04571-060",
    "Sara Cruz": "22031-000"
}

def obter_dados_do_cep(cep):
    url = f"http://viacep.com.br/ws/{cep}/json/"
    response = requests.get(url)
    if response.status_code == 200:
        data = response.json()
        return data.get("localidade")
    else:
        return "CEP não encontrado"

for nome, cep in integrantes.items():
    cidade = obter_dados_do_cep(cep)
    print(f"Nome: {nome}, Cidade: {cidade}")

atividades_path = os.path.join(os.path.dirname(__file__),
"atividade28_requirements.txt")

with open(atividades_path, "w") as file:
    result = subprocess.run(["pip", "freeze"], stdout=subprocess.PIPE, text=True)
    file.write(result.stdout)
```

Obs: Nenhum desses CEPs são os CEPs reais onde os integrantes da squad moram, como é um dado pessoal, optei por listar ceps aleatórios.

Com o propósito de exercitar o conteúdo de "Teste de Mesa" que aprendi recentemente vou explicar esse código por etapas, começando pelas bibliotecas:

```
import requests  
import os  
import subprocess
```

- **import requests:**

Serve para que eu possa usar o método *GET* e fazer uma requisição na API [Via Cep](#).

- **import os:**

Serve para que eu possa salvar o arquivo "atividade28_requirements.txt" na pasta Atividades dentro desse mesmo repositório.

- **import subprocess:**

Serve para que eu possa usar o comando `pip freeze` e obter as dependências desse projeto e salvá-las no arquivo "atividade28_requirements.txt".

```
def obter_dados_do_cep(cep):  
    url = f"http://viacep.com.br/ws/{cep}/json/"  
    response = requests.get(url)  
    if response.status_code == 200:  
        data = response.json()  
        return data.get("localidade")  
    else:  
        return "CEP não encontrado"
```

- **def obter_dados_do_cep(cep):**

Define uma função chamada `obter_dados_do_cep` que recebe um CEP como argumento.

- **url = f"http://viacep.com.br/ws/{cep}/json/":**

Monta a URL da API do ViaCEP com base no CEP fornecido. Essa URL será usada para fazer a solicitação HTTP para obter informações do CEP.

- **response = requests.get(url):**

Faz uma solicitação HTTP GET para a URL do ViaCEP usando a biblioteca requests e armazena a resposta na variável response.

- **if response.status_code == 200:**

Verifica se a resposta da solicitação HTTP tem o status code 200, que indica uma resposta bem-sucedida.

- **data = response.json():**

Se a resposta for bem-sucedida, converte o conteúdo da resposta em um formato JSON e armazena-o na variável data.

- **return data.get("localidade"):**

Retorna o nome da cidade (localidade) obtido a partir dos dados do CEP.

- **else:**

Se a resposta não for bem-sucedida (status code diferente de 200), entra no bloco else.

- **return "CEP não encontrado":**

Retorna a mensagem "CEP não encontrado" para indicar que os dados do CEP não puderam ser recuperados.

```
for nome, cep in integrantes.items():
    cidade = obter_dados_do_cep(cep)
    print(f"Nome: {nome}, Cidade: {cidade}")
```

- **for nome, cep in integrantes.items():**

Inicia um loop que percorre todos os itens (nome e CEP) do dicionário integrantes.

- **cidade = obter_dados_do_cep(cep):**

Chama a função obter_dados_do_cep para obter o nome da cidade com base no CEP atual do loop.

- **print(f"Nome: {nome}, Cidade: {cidade}":)**

Imprime o nome e a cidade obtidos para cada integrante da squad.

Até essa parte do código obtemos o seguinte resultado no terminal:

```
Nome: Leanderson, Cidade: Barueri
Nome: Beatriz Souza, Cidade: São Paulo
Nome: Bruno Soares, Cidade: Brasília
Nome: Rebeca Borges, Cidade: São Paulo
Nome: Sara Cruz, Cidade: Rio de Janeiro
(pvenv)
```

```
atividades_path = os.path.join(os.path.dirname(__file__),
"atividade28_requirements.txt")
```

- **atividades_path = os.path.join(os.path.dirname(file), "atividade28_requirements.txt"):**

Cria o caminho completo para o arquivo "atividade28_requirements.txt" usando os.path.join. Isso garante que o arquivo seja criado no mesmo diretório em que o script está sendo executado.

```
with open(atividades_path, "w") as file:
    result = subprocess.run(["pip", "freeze"], stdout=subprocess.PIPE, text=True)
    file.write(result.stdout)
```

- **with open(atividades_path, "w") as file:**

Abre o arquivo "atividade28_requirements.txt" em modo de escrita usando um bloco with, que garante que o arquivo seja fechado corretamente após o uso.

- **result = subprocess.run(["pip", "freeze"], stdout=subprocess.PIPE, text=True):**

Executa o comando pip freeze usando subprocess.run e redireciona a saída (a lista de dependências) para uma variável chamada result. A opção stdout=subprocess.PIPE permite capturar a saída padrão do comando.

- **file.write(result.stdout):**

Escreve a saída do comando pip freeze (a lista de dependências) no arquivo "atividade28_requirements.txt". Isso cria um arquivo de requisitos que lista todas as dependências do projeto e suas versões.

Como resultado dessa última parte do código, no arquivo Atividades\atividade28_requirements.txt temos as seguintes informações:

```
certifi==2023.7.22
charset-normalizer==3.2.0
idna==3.4
```

```
requests==2.31.0  
urllib3==2.0.4
```

E por fim, para desativar o ambiente virtual, bastou digitar o comando deactivate no terminal.

```
deactivate
```

O arquivo dessa atividade está nesse repositório dentro da pasta Atividades: Atividades\Atividade28.py.

Integrantes da Squad:

| Beatriz Souza | [Bruno Soares](#) | [Leanderson Lima](#) | [Rebeca Borges](#) | Sara Cruz |

Atividades do Módulo 4 - QA NA PRÁTICA



Esse [repositório](#) é dedicado às atividades realizadas durante o Módulo 4 - M4 - LÓGICA DE PROGRAMAÇÃO COM PYTHON do curso de Quality Assurance oferecido pelo [Instituto Jogajunto](#).

💡 Descrição da 29ª Atividade: 🌟

💡 DESAFIO DO CAIQUE Vamos explorar o poder da biblioteca OS! Prepare-se para mergulhar no mundo da interação entre o Python e o seu sistema operacional. Vamos aprender a usar a biblioteca OS em conjunto com funções nativas do Python para criar algo. O desafio é o seguinte: você vai criar uma lista de dados e, usando a biblioteca OS, interagir com o seu sistema operacional. Além disso, também criará uma nova pasta para salvar o arquivo de texto txt.

Para essa atividade, tomei como base o arquivo Atividades\atividade29_dados.csv que está nesse repositório. Aqui está o início do conteúdo desse arquivo.

```
,nome,endereco,email,idade,renda
0,Rafaela Rezende,"Avenida de Ribeiro
Santa Helena
14342136 Caldeira das Pedras / SP",mouraheitor@example.org,46,15594.52
1,Srta. Beatriz das Neves,"Loteamento de Fogaça, 61
Vila Novo São Lucas
11233-022 Monteiro / SC",luiz-otaviocorreia@example.org,24,16478.26
2,Ana Clara das Neves,"Favela Pinto, 97
São José
60325-119 Rodrigues / MS",isisalmeida@example.org,28,13729.48
3,Olivia Silva,"Vereda de Duarte, 12
Vila Satélite
91893-339 Souza / RN",lpeixoto@example.org,23,18478.46
4,Alexia Martins,"Conjunto João Pedro Caldeira, 888
Sion
81988-738 Azevedo da Mata / MA",carvalhoana-beatriz@example.com,54,9988.47
5,Lorena Almeida,"Morro Luiz Gustavo Castro, 61
Liberdade
02189859 Pereira / PI",mdas-neves@example.com,66,2224.67
6
.
.
.
```

Para executar o que essa tarefa pedia, executei o seguinte código:

```
import csv
import os

dados = []

caminho_csv = 'Atividades/atividade29_dados.csv'

if os.path.exists(caminho_csv):
    with open(caminho_csv, newline='') as csvfile:
        leitor = csv.DictReader(csvfile)
        for linha in leitor:
            dados.append(linha)
else:
    print(f'O arquivo CSV '{caminho_csv}' não foi encontrado.')

nome_pasta = 'atividade29'
caminho_pasta = 'Atividades'
if not os.path.exists(caminho_pasta):
    os.mkdir(caminho_pasta)

for linha in dados:
    print(f'Nome: {linha["nome"]}, Endereço: {linha["endereco"]}, Email: {linha["email"]}, Idade: {linha["idade"]}, Renda: {linha["renda"]}')

caminho_arquivo_txt = os.path.join(caminho_pasta, 'atividade29_dados.txt')

with open(caminho_arquivo_txt, 'w') as arquivo_txt:
    for linha in dados:
        arquivo_txt.write(f'Nome: {linha["nome"]}, Idade: {linha["idade"]}\n')
```

Este código, primeiro, importa algumas ferramentas úteis para lidar com informações em um formato chamado CSV.

Em seguida, pega as informações do arquivo que está guardado em 'Atividades/atividade29_dados.csv' e guarda essas informações em uma lista chamada 'dados'.

Antes de fazer isso, o código verifica se esse arquivo existe para não ter problemas.

Depois, o código verifica se há uma pasta chamada 'Atividades', caso ela não exista ela será criada. Como ela já existe nesse repositório, não será executada nenhuma ação.

Com um tipo de repetição, mostramos no terminal os nomes e idades das informações do arquivo CSV. Como nesse exemplo:

```
$ python -u "c:\Users\...\GitHub\Squad02_M4\Atividades\Atividade29.py"
Nome: Rafaela Rezende, Endereço: Avenida de Ribeiro
Santa Helena
14342136 Caldeira das Pedras / SP, Email: mouraheitor@example.org, Idade: 46,
```

Renda: 15594.52
Nome: Srta. Beatriz das Neves, Endereco: Loteamento de Fogaça, 61
Vila Novo São Lucas
11233-022 Monteiro / SC, Email: luiz-otaviocorreia@example.org, Idade: 24, Renda: 16478.26
Nome: Ana Clara das Neves, Endereco: Favela Pinto, 97
São José
60325-119 Rodrigues / MS, Email: isisalmeida@example.org, Idade: 28, Renda: 13729.48
Nome: Olivia Silva, Endereco: Vereda de Duarte, 12
Vila Satélite
91893-339 Souza / RN, Email: lpeixoto@example.org, Idade: 23, Renda: 18478.46
Nome: Alexia Martins, Endereco: Conjunto João Pedro Caldeira, 888
Sion
81988-738 Azevedo da Mata / MA, Email: carvalhoana-beatriz@example.com, Idade: 54, Renda: 9988.47
Nome: Lorena Almeida, Endereco: Morro Luiz Gustavo Castro, 61
Liberdade
02189859 Pereira / PI, Email: mdas-neves@example.com, Idade: 66, Renda: 2224.67
.
.
.

Para terminar, o código cria um novo arquivo de texto ('Atividades/atividade29_dados.txt') e coloca essas mesmas informações nele.

Nome: Rafaela Rezende, Endereco: Avenida de Ribeiro
Santa Helena
14342136 Caldeira das Pedras / SP, Email: mouraheitor@example.org, Idade: 46, Renda: 15594.52
Nome: Srta. Beatriz das Neves, Endereco: Loteamento de Fogaça, 61
Vila Novo São Lucas
11233-022 Monteiro / SC, Email: luiz-otaviocorreia@example.org, Idade: 24, Renda: 16478.26
Nome: Ana Clara das Neves, Endereco: Favela Pinto, 97
São José
60325-119 Rodrigues / MS, Email: isisalmeida@example.org, Idade: 28, Renda: 13729.48
Nome: Olivia Silva, Endereco: Vereda de Duarte, 12
Vila Satélite
91893-339 Souza / RN, Email: lpeixoto@example.org, Idade: 23, Renda: 18478.46
Nome: Alexia Martins, Endereco: Conjunto João Pedro Caldeira, 888
Sion
81988-738 Azevedo da Mata / MA, Email: carvalhoana-beatriz@example.com, Idade: 54, Renda: 9988.47
.
.
.

O arquivo dessa atividade está nesse repositório dentro da pasta Atividades: Atividades\Atividade29.py.

Integrantes da Squad:

| Beatriz Souza | Bruno Soares | Leanderson Lima | Rebeca Borges | Sara Cruz |