

## Leitura de imagens locais-transformar base 64

<https://platform.openai.com/docs/guides/vision>

Devido ao GPT não aceitar o uso de imagens no formato padrão, precisamos utilizar o encaminhamento destas imagens em formato 64.

O formato 64 nada mais é do que codificar imagens em linhas de 64 caracteres, isso possibilita a transferência via rede.

Para processo em questão iniciei fazendo uma chamada das bibliotecas que irei utilizar.

PYTHON

```
import base64
import requests
from dotenv import load_dotenv
import os
```

A biblioteca base64 vai ser usada para uma função de codificação de imagens  
A request para fazer a requisição para página do chatgpt  
dotenv e os para "importar" a variável de ambiente da key do gpt.

PYTHON

```
load_dotenv()
api_key = os.getenv("OPENAI_API_KEY")

def encode_image(image_path):
    with open(image_path, "rb") as image_file:
        return
base64.b64encode(image_file.read()).decode('utf-8')

image_path = "/home/oem/Imagens/thumb-1920-522193.jpg"

base64_image = encode_image(image_path)
```

load\_dotenv() é a importação do arquivo .env no diretório base.

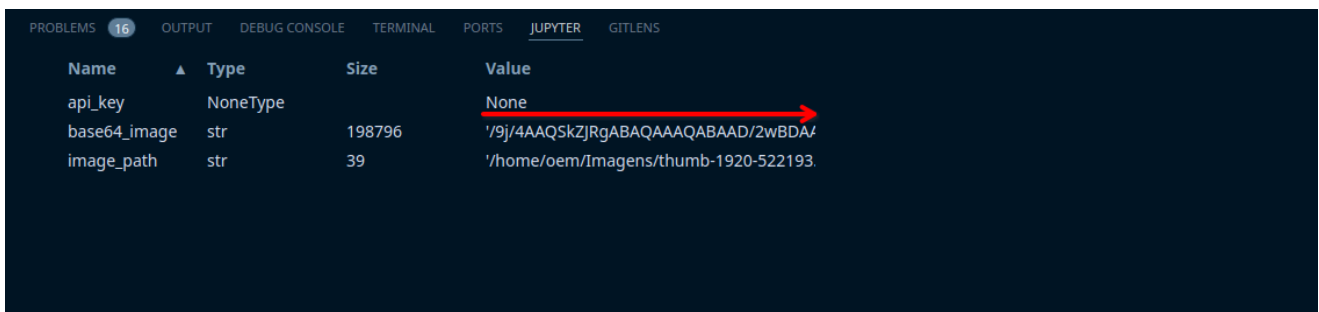
Após isto crio uma variável chamada api\_key que através da biblioteca os utiliza o atributo getenv para variável chamada "OPENAI\_API\_KEY" que é o nome da variável de ambiente definida no .env com a chave do gpt.

Crio então uma função de codificação que recebe um caminho, a partir disso utilizo uma função de abertura de arquivo, que recebe o caminho da imagem e o atributo de r=leitura b=binaria, a partir disso defino isto como a variável image\_file.

O retorno dela será feito a partir de um método da biblioteca base64 que codifica a imagem.

Definido a função, faço a inicialização de uma variável que recebe o caminho da imagem no meu ambiente.

Inicializo outra variável que recebe o valor a partir da função acima.



Name	Type	Size	Value
api_key	NoneType		None
base64_image	str	198796	'/9j/4AAQSkZJRgABAQAAQABAAD/2wBDA/
image_path	str	39	'/home/oem/Imagens/thumb-1920-522193

Se formos pegar o código completo o retorno é uma série gigantesca de caracteres. A partir disso inicializo um dicionário:

PYTHON

```
headers = {
    "Content-Type": "application/json",
    "Authorization": f"Bearer {api_key}"
}
```

Esta requisição inicia indicando que o tipo de conteúdo será enviado no formato json. Abaixo indica o tipo de autorização que definimos a api\_key.

Feito isto encontramos a parte mais "chata" do código:

```

payload = {
    "model": "gpt-4o-mini",
    "messages": [
        {
            "role": "user",
            "content": [
                {
                    "type": "text",
                    "text": "O que há nesta imagem, qual
personagem da mesma?"
                },
                {
                    "type": "image_url",
                    "image_url": {
                        "url": f"data:image/jpeg;base64,
{base64_image}"
                    }
                }
            ]
        },
        {
            "max_tokens": 300
        }
    ]
}

```

Me refiro a chato em um sentido de complexidade na escrita. Inicializo o dicionário indicando o modelo da IA que será utilizado, após a message que será levada. Nela defino primeiro a mensagem que será levada pelo usuário, que leva um conteúdo do tipo texto e o texto em questão, que é a pergunta " que há nesta imagem, qual personagem da mesma?", feito isto encaminho uma URL da imagem, que basicamente leva o código base64 que definimos anteriormente.

Ao concluir isto, defino o max\_tokens que irá retornar no máximo 300 caracteres.

A ultima parte do código é a conclusão, utilizando a biblioteca requests:

```

response = requests.post(
    "https://api.openai.com/v1/chat/completions",
    headers=headers, json=payload)

if response.status_code == 200:
    response_data = response.json()
    print(response_data['choices'][0]['message']['content'])
else:
    print(f"Erro: {response.status_code}")
    print(response.json())

```

Aqui basicamente o que o código faz é uma request como post para o link do openai, usa o cabeçalho que definimos no início do código e o formato json a partir do payload.

Se o retorno desta requisição for positivo (Status 200) ele irá me retornar printando o retorno positivo.

Caso não ele irá me retornar com o código da falha e a resposta da request.


Com isto tive algumas respostas:



```

3 from dotenv import load_dotenv
4 import os
5
6 load_dotenv()
7
8 api_key = os.getenv("OPENAI_API_KEY")
9
10
11 def encode_image(image_path):
12     with open(image_path, "rb") as image_file:
13         return base64.b64encode(image_file.read()).decode('utf-8')
14
15
16 image_path = "/home/oem/Imagens/super_mario_wallpaper_by_inklingmain_df2dg7z-fullview.jpg - 1280x720 - 50% — Gw..."
17
18 base64_image = encode_image(image_path)
19
20 headers = {
21     "Content-Type": "application/json",
22     "Authorization": f"Bearer {api_key}"
23 }
24
25 payload = {
26     "model": "gpt-4o-mini",
27     "messages": [
28         {
29             "role": "user"

```



```

30         }
31     ]
32 }

```

(tf-gpu) oem@Leonardo:~/pythonGPU/PythonMaximiza/MaximizaEstagio/Task2\$ python3 base64GPT.py


A imagem apresenta vários personagens do universo "Super Mario". Entre os personagens, podemos identificar Mario, Luigi, Princesa Peach, Donkey Kong, Yoshi, Toad, Wario e Waluigi. Eles são figuras icônicas de jogos da Nintendo.

(tf-gpu) oem@Leonardo:~/pythonGPU/PythonMaximiza/MaximizaEstagio/Task2\$

```

3 from dotenv import load_dotenv
4 import os
5
6 load_dotenv()
7
8 api_key = os.getenv("OPENAI_API_KEY")
9
10
11 def encode_image(image_path):
12     with open(image_path, "rb") as image_file:
13         return base64.b64encode(image_file.read()).decode('utf-8')
14
15
16 image_path = "/home/oem/Imagens/D1B9C77C-8B13-4FD0-86EB-72B1A44AA5C1.png"
17
18 base64_image = encode_image(image_path)
19
20 headers = {
21     "Content-Type": "application/json",
22     "Authorization": f"Bearer {api_key}"
23 }
24
25 payload = {
26     "model": "gpt-4o-mini",
27     "messages": [
28         {
29             "role": "user"

```



```

30         }
31     ]
32 }

```

(tf-gpu) oem@Leonardo:~/pythonGPU/PythonMaximiza/MaximizaEstagio/Task2\$ python3 base64GPT.py

A imagem é do jogo "Castlevania: Symphony of the Night". O personagem representado é Alucard, que é o protagonista do jogo e filho de Dracula. Se precisar de mais informações sobre o jogo ou o personagem, é só avisar!

(tf-gpu) oem@Leonardo:~/pythonGPU/PythonMaximiza/MaximizaEstagio/Task2\$

Num geral tive boas respostas, tirando o fato do erro da primeira imagem ele acertou o jogo, só errando o personagem. Utilizei nos testes 2 imagens num formato .jpg e uma no formato .png, ambas tiveram um tempo rápido de request.