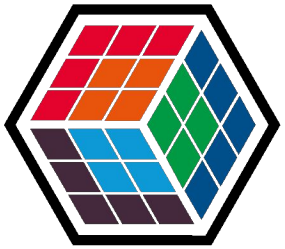


THE DEVELOPER'S CONFERENCE

# Criando uma Aplicação CLI com AsyncIO

**Élysson Mendes Rezende**
Desenvolvedor/Arquiteto de Software

# Agenda

THE DEVELOPER'S CONFERENCE

- Arquitetura Sincrona X AsyncIO
- Como fazer um CLI utilizando AsyncIO
- Upload de imagens para o Google Photos
- Conclusão

# Hello World

THE DEVELOPER'S CONFERENCE

## Élysson MR

Desenvolvedor Python/NodeJS/GO atuando com micro serviços na LuizaLabs, curioso por natureza e padawan em Arquitetura de Software.

github.com/elyssonmr

linkedin.com/in/elyssonmr

# Arquitetura Sincrona x AsyncIO

THE DEVELOPER'S CONFERENCE

## Tradicional

- O IO é bloqueante;
- Necessário muitos recursos para escalar;
- Simples de codificar;
- Fácil de encontrar bibliotecas;

## AsyncIO

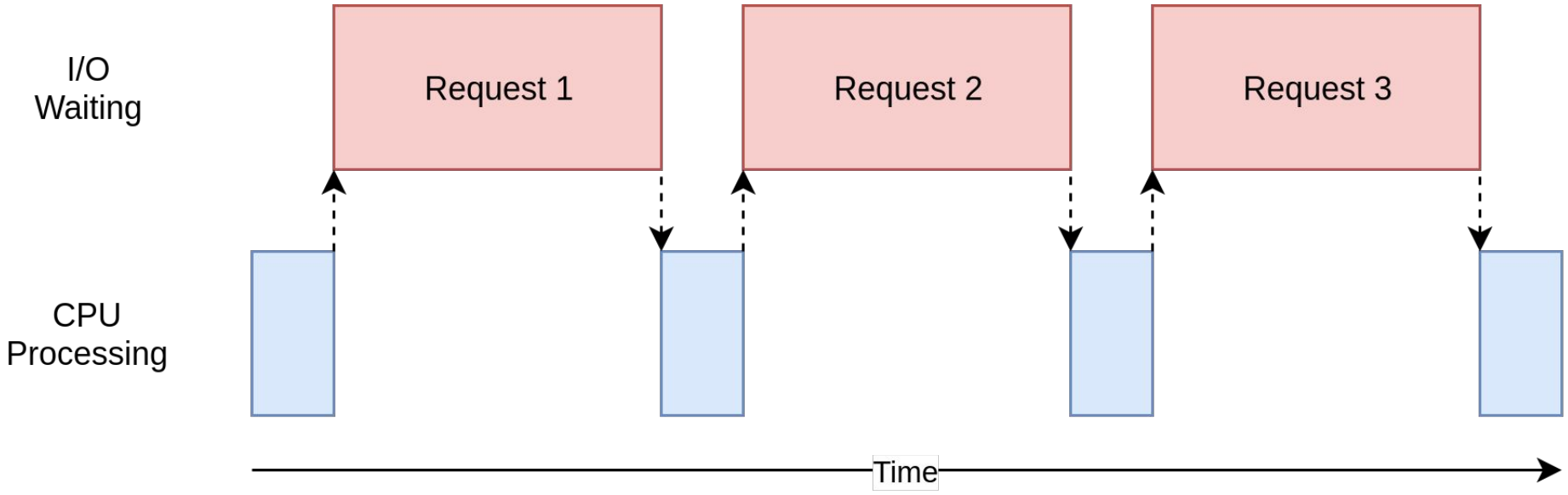
- O IO não é bloqueante;
- Não é necessário muitos recursos para escalar;
- Complicado de codificar;
- Não existem muitas bibliotecas Async (por enquanto);

# Arquitetura Sincrona x AsyncIO

THE DEVELOPER'S CONFERENCE

```python
import requests


def how_is_the_wheather(city="Florianopolis"):
    api_key = "API_KEY"
    api_url = ("http://api.openweathermap.org/data/2.5/"
               f"weather?q={city}&units=metric&APPID={api_key}")

    response = requests.get(api_url)

    wheather = response.json()

    return wheather.get("main", {}).get("temp", "0.0")


if __name__ == "__main__":
    wheather = how_is_the_wheather("São Paulo")
    print(f"Temperatura: {wheather} ºC")
```

# Arquitetura Sincrona x AsyncIO



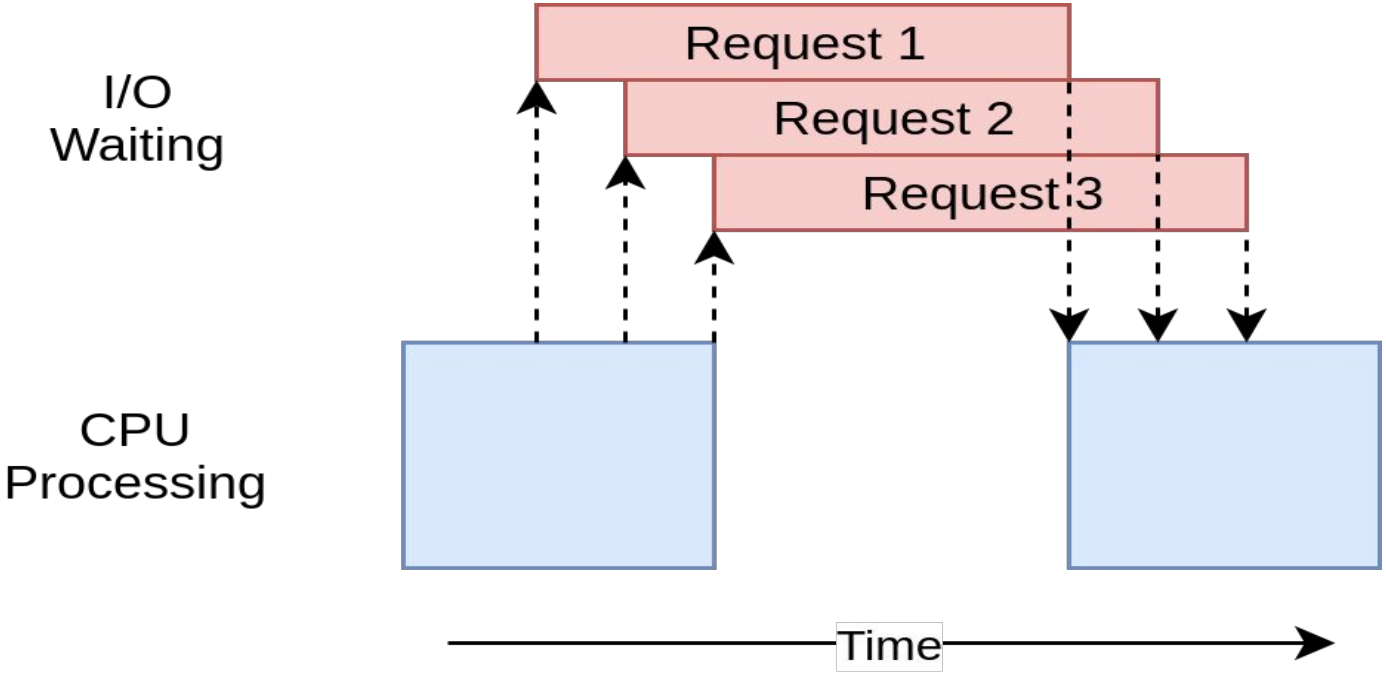

**Fonte:** https://realpython.com/python-concurrency/

# Arquitetura Sincrona x AsyncIO

THE DEVELOPER'S CONFERENCE

```python
import asyncio

from aiohttp import ClientSession


async def how_is_the_wheather(client, city="Florianopolis"):
    api_key = "API_KEY"
    api_url = ("http://api.openweathermap.org/data/2.5/"
               f"weather?q={city}&units=metric&APPID={api_key}")
    async with client.get(api_url) as r:
        wheather = await r.json()
        return wheather.get("main", {}).get("temp", "0.0")


async def main(loop):
    cities = ["Florianópolis", "São Paulo", "New York"]
    requests = []

    async with ClientSession() as session:
        for city in cities:
            requests.append(how_is_the_wheather(session, city))
        temps = await asyncio.gather(*requests)

        for city, temp in zip(cities, temps):
            print(f"Temperatura em {city} é de {temp} ºC")

if __name__ == "__main__":
    loop = asyncio.get_event_loop()

    loop.run_until_complete(main(loop))
```

# Arquitetura Sincrona x AsyncIO




**Fonte:** https://realpython.com/python-concurrency/

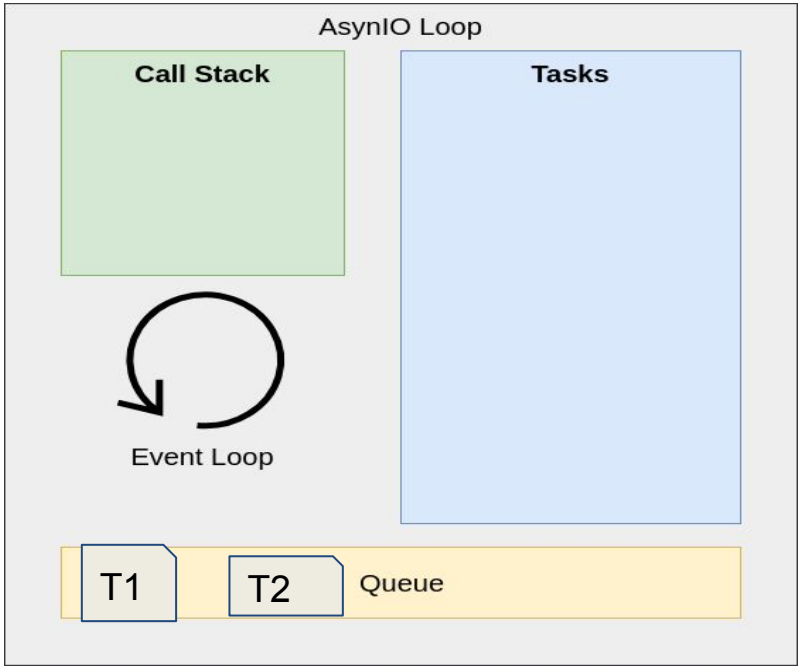
# Arquitetura Sincrona x AsyncIO

THE DEVELOPER'S CONFERENCE

- Keyword async e await;

```
import asyncio

async def do_something_slow():
    await asyncio.sleep(3)


if __name__ == "__main__":
    loop = asyncio.get_event_loop()
    loop.run_until_complete(do_something_slow())
    print("Done")
```

# Arquitetura Sincrona x AsyncIO

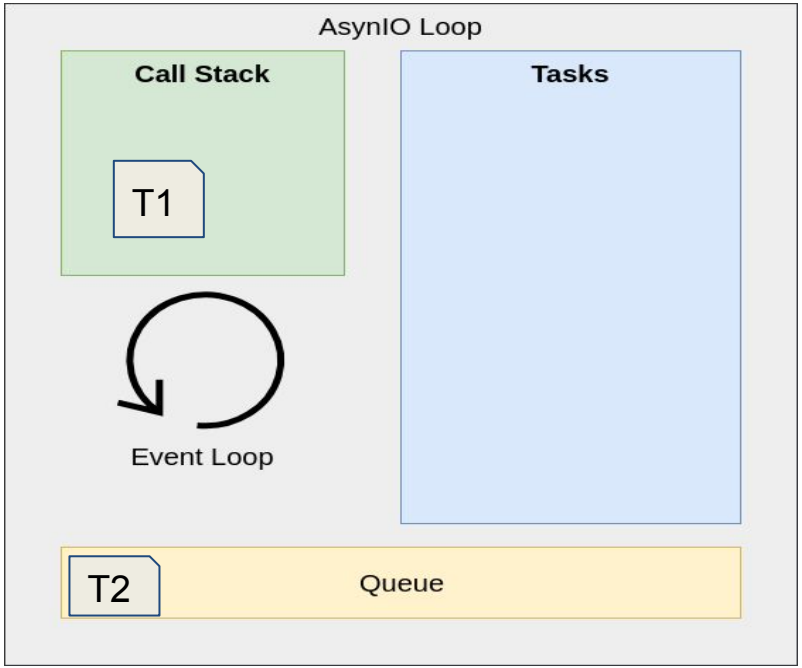



- Quando o Python inicializar o loop, ele vai adicionar a primeira task da **queue** (**T1**) na **Call Stack** para executar;

Adaptado de:
https://www.youtube.com/watch?v=8aGhZQkoFbQ

# Arquitetura Sincrona x AsyncIO

THE DEVELOPER'S CONFERENCE

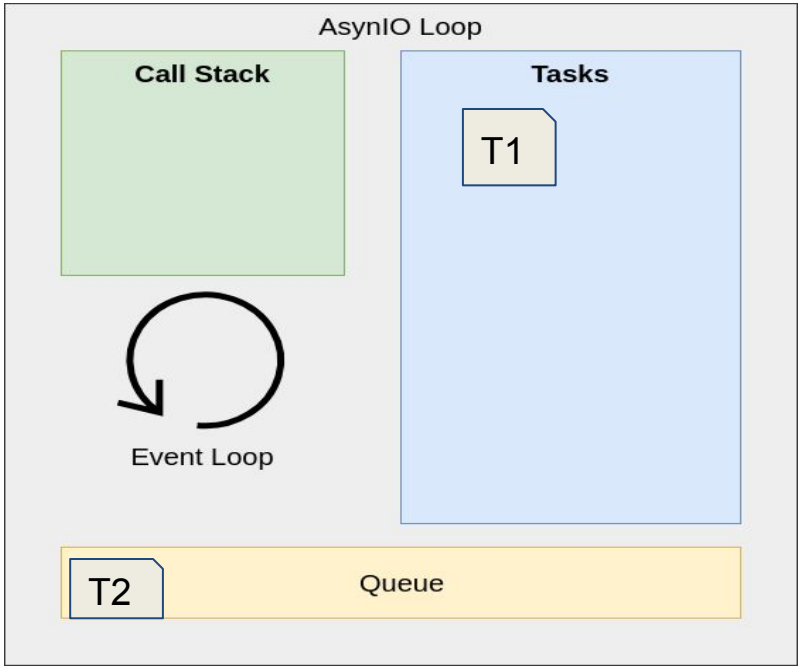



- Quando o Python inicializar o **loop**, ele vai adicionar a primeira task da **queue** (**T1**) na **Call Stack** para executar;
- Então o **loop** vai task **T1** até encontrar um IO (ou awaitable);

Adaptado de:
https://www.youtube.com/watch?v=8aGhZQkoFbQ

# Arquitetura Sincrona x AsyncIO

THE DEVELOPER'S CONFERENCE

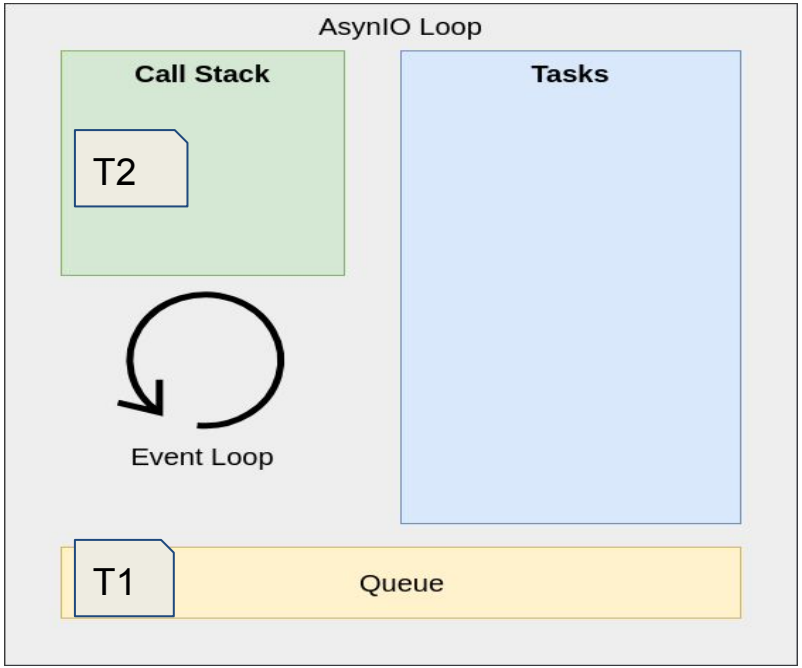



- Quando o Python inicializar o loop, ele vai adicionar a primeira task da **queue** (**T1**) na **Call Stack** para executar;
- Então o **loop** vai task **T1** até encontrar um IO (ou awaitable);
- Enquanto a **T1** estiver aguardando o IO, o **loop** vai pegar a próxima Task (**T2**) para adicioná-la na **Call Stack**;

Adaptado de:
https://www.youtube.com/watch?v=8aGhZQkoFbQ

# Arquitetura Sincrona x AsyncIO

THE DEVELOPER'S CONFERENCE

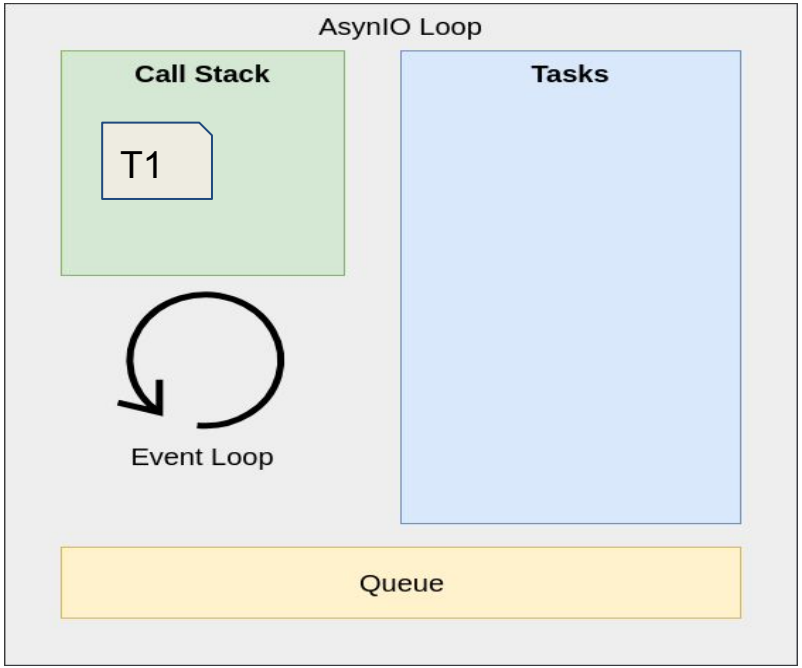



- Quando o Python inicializar o loop, ele vai adicionar a primeira task da **queue** (**T1**) na **Call Stack** para executar;
- Então o **loop** vai task **T1** até encontrar um IO (ou awaitable);
- Enquanto a **T1** estiver aguardando o IO, o **loop** vai pegar a próxima Task (**T2**) para adicioná-la na **Call Stack**;
- Durante a execução do **T2** pelo loop, a **T1** finalizou o IO dela retornando para a **queue**;

Adaptado de:
https://www.youtube.com/watch?v=8aGhZQkoFbQ

# Arquitetura Sincrona x AsyncIO

THE DEVELOPER'S CONFERENCE




- Quando o Python inicializar o loop, ele vai adicionar a primeira task da **queue** (**T1**) na **Call Stack** para executar;
- Então o **loop** vai task **T1** até encontrar um IO (ou awaitable);
- Enquanto a **T1** estiver aguardando o IO, o **loop** vai pegar a próxima Task (**T2**) para adicioná-la na **Call Stack**;
- Durante a execução do **T2** pelo loop, a **T1** finalizou o IO dela retornando para a **queue**;
- Quando a **T2** finalizar, a **T1** irá voltar para o **Call Stack** inicializando novamente o processo do **loop**;

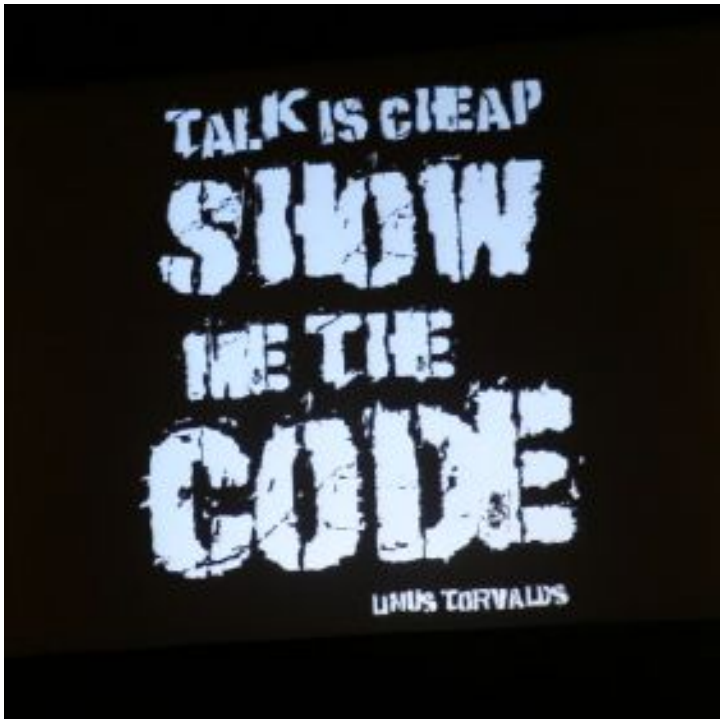
Adaptado de:
https://www.youtube.com/watch?v=8aGhZQkoFbQ

# Como fazer um CLI utilizando AsyncIO

THE DEVELOPER'S CONFERENCE

```python
from argparse import ArgumentParser


def setup_args():
    arg_parser = ArgumentParser(description="CLI Example")
    arg_parser.add_argument(
        "folder", metavar="folder_name", help="Photos Folder")
    arg_parser.add_argument(
        "album", metavar="album_name", help="Album name to be saved")

    return arg_parser.parse_args()


async def main(args):
    pass


if __name__ == "__main__":
    args = setup_args()
    loop = asyncio.get_event_loop()
    loop.run_until_complete(main())
```

# Upload de images para o Google Photos



# Conclusão

THE DEVELOPER'S CONFERENCE

- Com o AsyncIO conseguimos escalar melhor a nossa aplicação, fazendo IO em “paralelo”;
- É muito importante saber utilizar o loop para que possamos pegar o máximo valor dele;
- Sempre devemos projetar o sistema para não travar o loop;
- AsyncIO não é bala de prata;

# Referência

THE DEVELOPER'S CONFERENCE

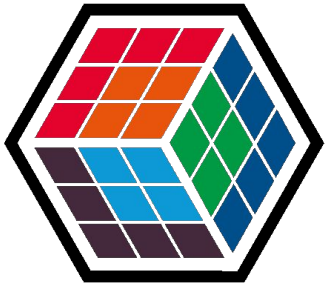
- https://realpython.com/async-io-python/
- https://realpython.com/python-concurrency/
- https://docs.python.org/3/library/asyncio.html
- https://www.youtube.com/watch?v=8aGhZQkoFbQ

THE DEVELOPER'S CONFERENCE

# Obrigado

Github

LinkedIn



# THE DEVELOPER'S CONFERENCE