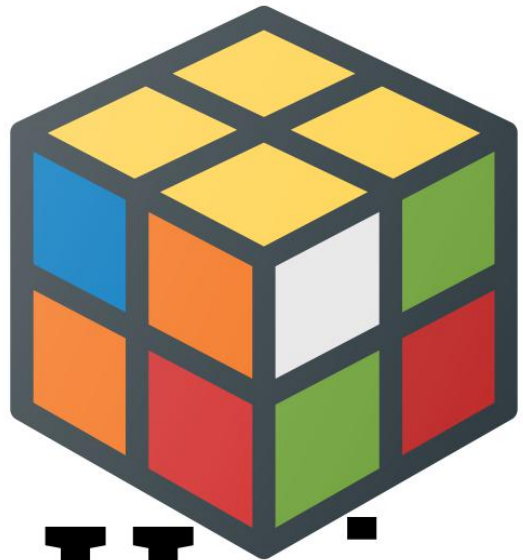


Geek University

Evolua seu lado geek!

www.geekuniversity.com.br

Compreendendo o Asyncio

Compreendendo o Asyncio

O módulo asyncio provê um objeto conhecido como ***Future*** que gerencia a execução e representação do eventual resultado de uma computação.

Compreendendo o Asyncio

O objeto Future possui alguns métodos importantes, dentre eles:

cancel(), que cancela o future;

done(), que retorna True se o future está completo ou cancelado;

result(), que retorna o resultado da computação;

exception(), que retorna qualquer exceção levantada durante a execução;

add_done_callback(funcao), que adiciona uma função a ser executada após o future finalizar;

Compreendendo o Asyncio

Ao executar uma função/método assíncrono, devemos usar SEMPRE a palavra reservada *await* que pausa a execução da função até que esteja completa.

```
1  import asyncio
2
3
4  async def diz_oi_demorado():
5      print('Oi...')
6      await asyncio.sleep(2)
7      print('todos...')
8
9
10
11
12 el = asyncio.get_event_loop()
13 el.run_until_complete(diz_oi_demorado())
14 el.close()
15
```

Compreendendo o Asyncio

O Event Loop para após o future ter finalizado.

```
1  import asyncio
2
3
4  async def diz_oi_demorado():
5      print('Oi...')
6      await asyncio.sleep(2)
7      print('todos...')
8
9
10
11
12 el = asyncio.get_event_loop()
13 el.run_until_complete(diz_oi_demorado())
14 el.close()
15
```

Lembrando que o future nada mais é do que uma corrotina...

Compreendendo o Asyncio

Ainda no conceito de Futures, temos a Task que é uma subclasse de Future e é usada internamente no módulo Asyncio para gerenciar a execução de uma corrotina durante um Event Loop.

Com um event loop podemos criar tasks (taferas) com o método create_task

Compreendendo o Asyncio

Por fim, uma estrutura de dados eficiente para se usar através do módulo Asyncio é a Queue, já conhecida de aulas passadas.

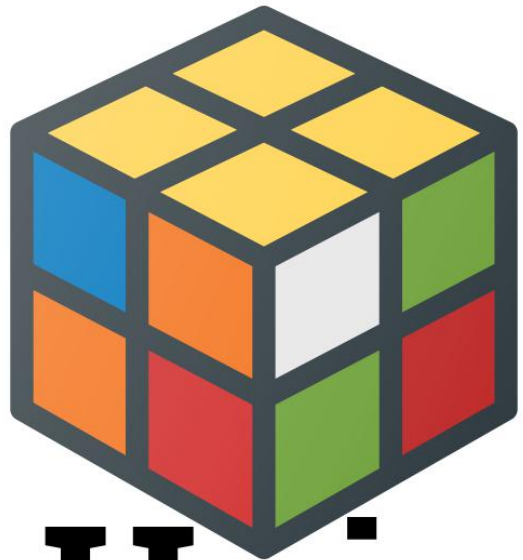
Um código assíncrono é conhecido como non-blocking, ou seja, a execução do código não fica bloqueado por uma operação demorada. A execução continua e volta para buscar o resultado quando a tarefa estiver finalizada.

Compreendendo o Asyncio

Por fim, uma estrutura de dados eficiente para se usar através do módulo Asyncio é a Queue, já conhecida de aulas passadas.

Um código assíncrono é conhecido como non-blocking, ou seja, a execução do código não fica bloqueado por uma operação demorada. A execução continua e volta para buscar o resultado quando a tarefa estiver finalizada.

Vamos ao código...



Geek University

Evolua seu lado geek!

www.geekuniversity.com.br