

# ThoughtWorks®

*tudo ao mesmo tempo agora*

---

# PYTHON ASSÍNCRONO

---

*Programação concorrente sem tocar em threads ou callbacks*



**PYTHON**BRASIL 2016

**ThoughtWorks®**

# **LUCIANO RAMALHO**

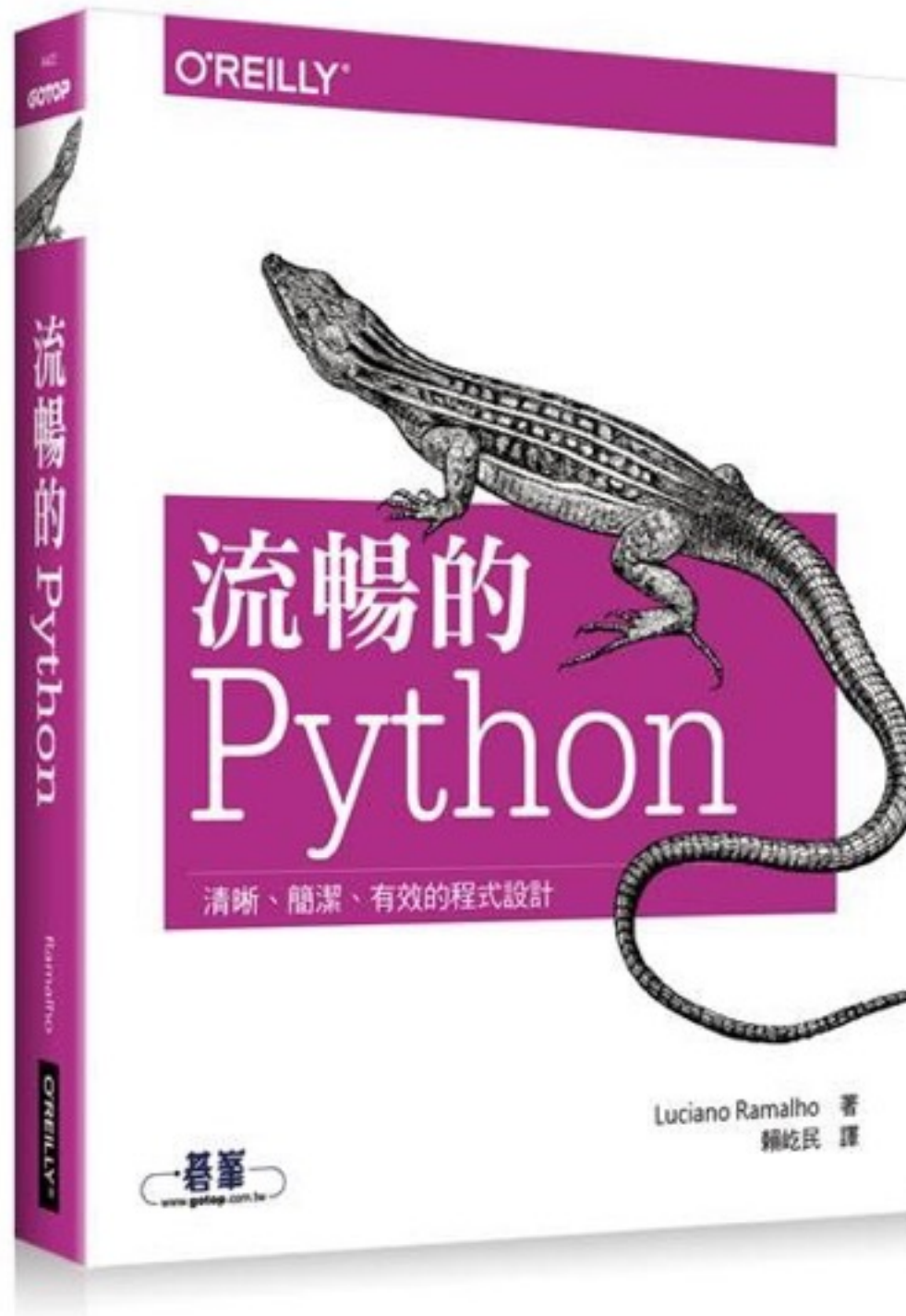
---

*Technical Principal*

---

*@ramalhoorg*  
*luciano.ramalho@thoughtworks.com*

# FLUENT PYTHON, MEU PRIMEIRO LIVRO



**Fluent Python** (O'Reilly, 2015)

**Python Fluente** (Novatec, 2015)

**Python к вершинам  
мастерства\*** (DMK, 2015)

流暢的 **Python**<sup>†</sup> (Gotop, 2016)

also in **Polish, Korean...**

\* *Python. To the heights of excellence*

† *Smooth Python*

# CONCORRÊNCIA

---

Não é o mesmo que *paralelismo*!

# CONCORRÊNCIA X PARALELISMO

---

## Concurrency vs. parallelism

Concurrency is about dealing with lots of things at once.

Parallelism is about doing lots of things at once.

Not the same, but related.

Concurrency is about structure, parallelism is about execution.

Concurrency provides a way to structure a solution to solve a problem that may (but not necessarily) be parallelizable.



*Rob Pike - 'Concurrency Is Not Parallelism'*  
[https://www.youtube.com/watch?v=cN\\_DpYBzKso](https://www.youtube.com/watch?v=cN_DpYBzKso)

# GIRANDO 18 PRATOS COM APENAS 2 MÃOS

---

**GIRANDO PR  
ATOS CONCO  
RRÊNCIA MO  
DERNA EM  
PYTHON**



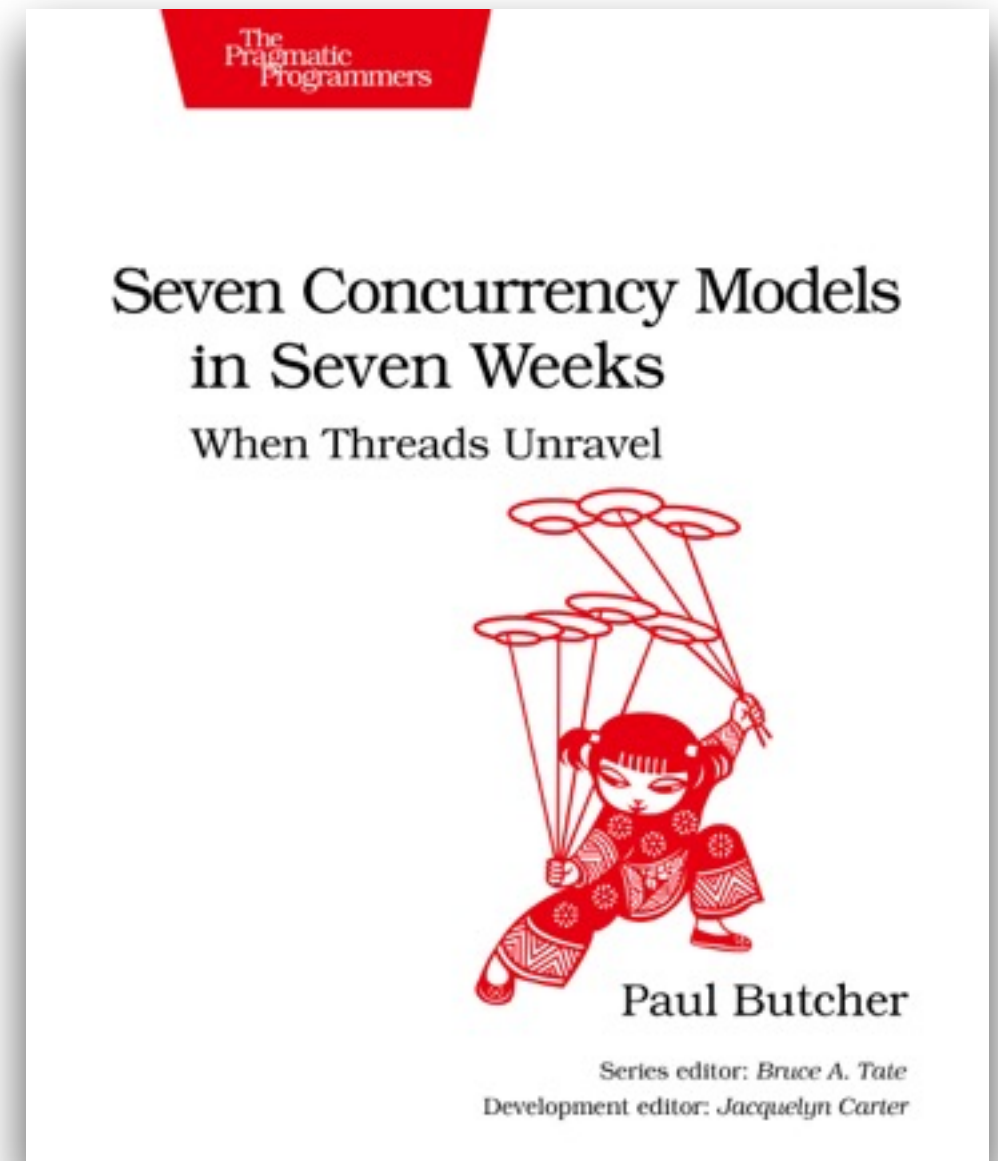
<https://speakerdeck.com/ramalho/await-em-python-3-dot-5>



# DESFIANDO THREADS

---

- *Seven Concurrency Models in Seven Weeks — When Threads Unravel* (Paul Butcher)
- Sequer menciona callbacks, interrupções e outras formas de concorrência em baixo nível
- Cap. 1: problemas com threads
- Demais capítulos: abstrações mais poderosas
  - Actors, CSP, STM, data parallelism...
- Suporte nativo em linguagens
  - Erlang, Elixir, Clojure, Go, Cilk, Haskell...



# CONCORRÊNCIA APESAR DA GIL

---

Sim, mesmo com a querida Global Interpreter Lock



# ALTERNATIVAS EM PYTHON

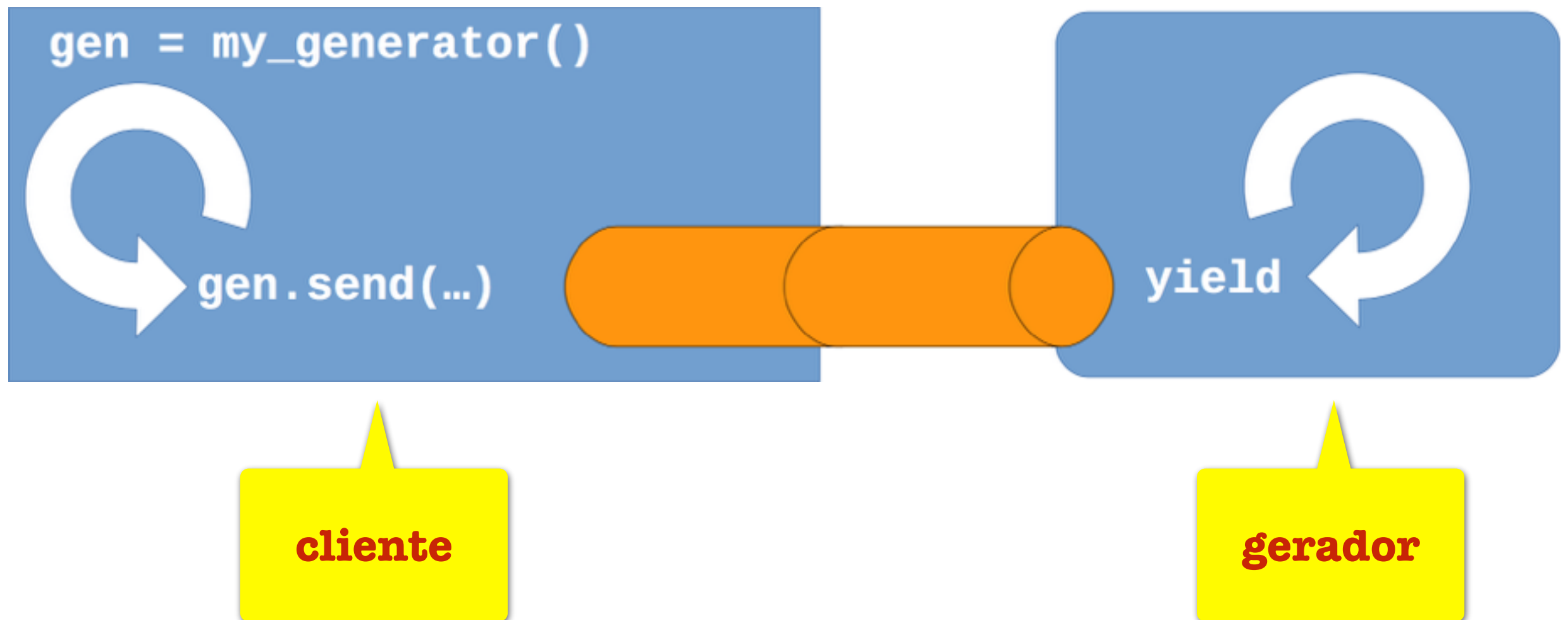
---

- **Threads**: OK para I/O de alto desempenho em escala limitada
  - Ver: Motor 0.7 Beta With Pymongo 2.9 And A Threaded Core  
— A. Jesse Jiryu Davis — <https://emptysqua.re/blog/motor-0-7-beta/>
- *Certas* bibliotecas externas em Cython, C, C++, FORTRAN...
- **Multiprocessing**: múltiplas instâncias de Python
- **Celery** e outros sistemas de distribuição de tarefas
- Callbacks e deferreds em **Twisted**
- **gevent**: greenlets e monkey-patching
- Corrotinas em **Tornado** e **Asyncio**

# CONCORRÊNCIA COM CORROTINAS (1)

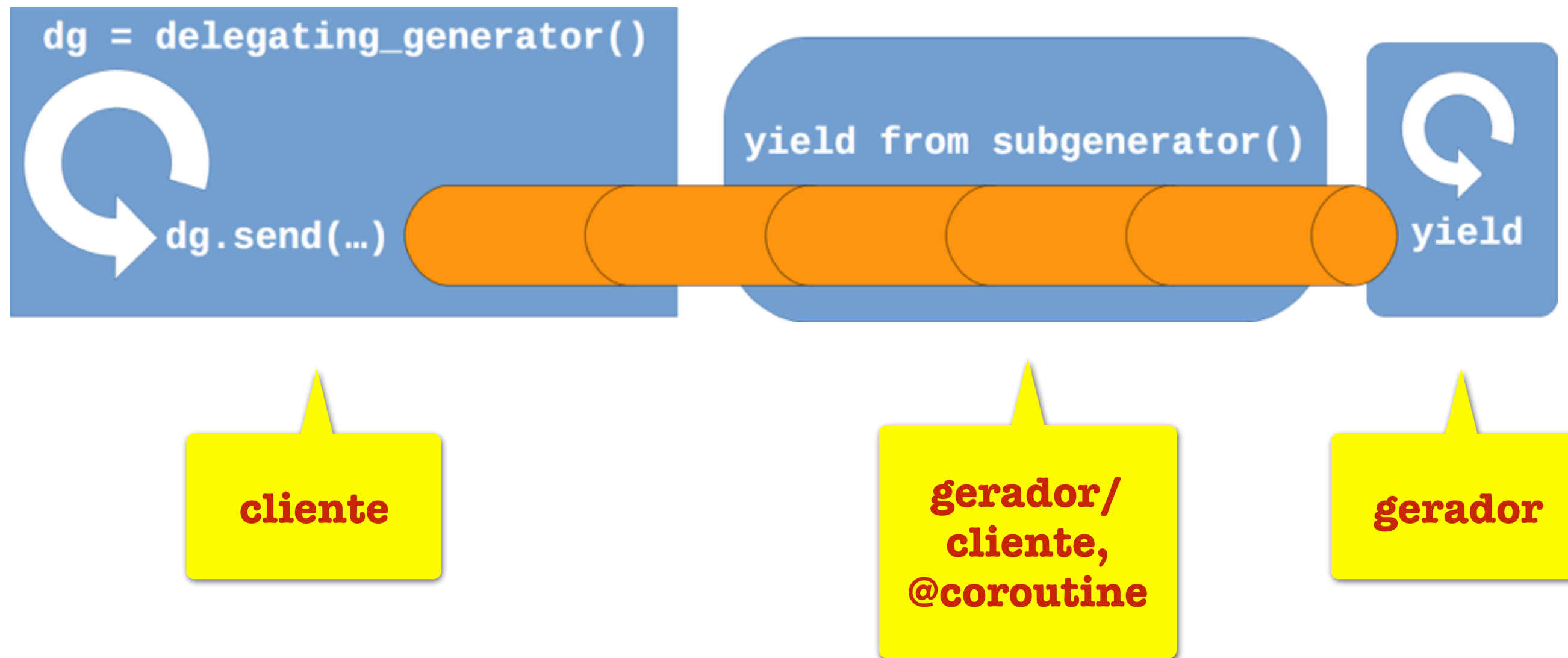
---

- Era uma vez um simples **gerador** que ganhou um método **.send()**



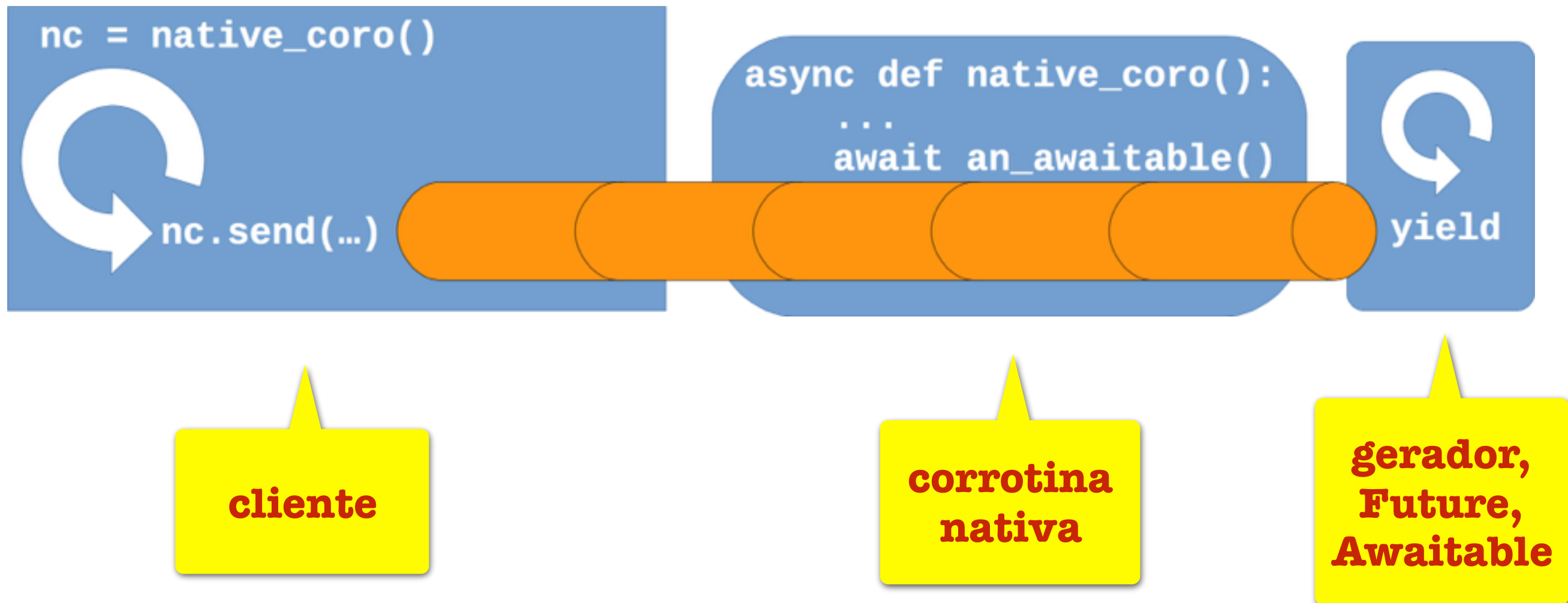
## CONCORRÊNCIA COM CORROTINAS (2)

- Daí a sintaxe **yield from** permitiu que um gerador chamasse outro...



## CONCORRÊNCIA COM CORROTINAS (3)

- E finalmente alguns geradores viraram **corrotinas nativas**.



# ASYNCIO

---

O principal foco de desenvolvimento concorrente em Python hoje

# ASYNCIO

---

- Biblioteca criada por Guido van Rossum (originalmente: Tulip)
  - incorporada à biblioteca padrão no Python 3.4 como asyncio
- **asyncio** tem caráter provisório (*provisional*) no Python 3.5
  - API ainda instável
  - status no Python 3.6...
- Eco-sistema muito ativo
  - por exemplo: <https://github.com/aio-libs/>



# aio-lib

The set of asyncio-based libraries built with high quality for humans

<https://groups.google.com/forum/#!forum/aio-lib>

## Repositories

People 8

Filters

Find a repository...

## aiomysql

Python ★ 197 29

aiomysql is a library for accessing a MySQL database from the asyncio

Updated a day ago

## aiohttp\_admin

Python ★ 24 4

admin interface for aiohttp application

Updated 2 days ago

## aiokafka

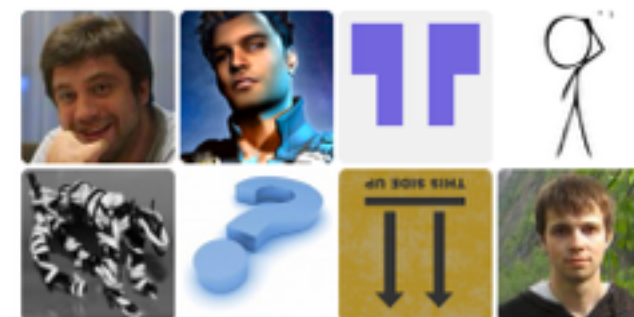
Python ★ 49 12

asyncio client for kafka

Updated 2 days ago

## People

8 >







## aiozmq

Python ★ 175 🔗 23

Asyncio (pep 3156) integration with ZeroMQ

Updated 5 days ago

## aiopg

Python ★ 332 🔗 48

aiopg is a library for accessing a PostgreSQL database from the asyncio

Updated 5 days ago

## sockjs

Python ★ 53 🔗 10

SockJS Server

Updated 5 days ago

## multidict

Python ★ 13 🔗 3

multidict implementation

Updated 5 days ago

## aiobotocore

Python ★ 52 🔗 14

asyncio support for boto3 library using aiohttp

Updated 5 days ago

## aiohttp-debugtoolbar

JavaScript ★ 54 🔗 16

**aioodbc**

Python ★ 39 0

aioodbc - is a library for accessing a ODBC databases from the asyncio

Updated 2 days ago

**aioredis**

Python ★ 196 41

asyncio (PEP 3156) Redis support

Updated 2 days ago

**aiohttp\_session**

Python ★ 39 24

Provide sessions for aiohttp.web

Updated 3 days ago

**aiohttp\_jinja2**

Python ★ 49 19

jinja2 template renderer for aiohttp.web

Updated 3 days ago

**yarl**

Python ★ 33 4

Yet another URL library

Updated 4 days ago

**aiozmq**

Python ★ 175 22

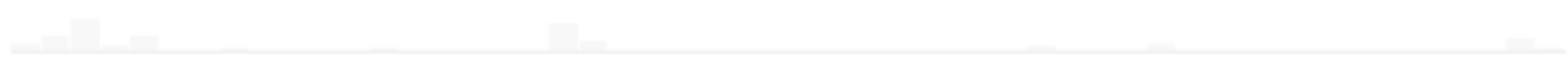


## [aiohttp\\_debugtoolbar](#)

JavaScript ★ 54 🔒 16

aiohttp\_debugtoolbar is library for debugtoolbar support for aiohttp

Updated 7 days ago



## [aiohttp\\_cors](#)

Python ★ 21 🔒 6

CORS support for aiohttp

Updated 7 days ago



## [janus](#)

Python ★ 31 🔒 1

Thread-safe asyncio-aware queue

Updated 7 days ago



## [aiomcache](#)

Python ★ 35 🔒 7

Minimal asyncio memcached client

Updated 7 days ago



## [aiorwlock](#)

Python ★ 17 🔒 1

Synchronization primitive RWLock for asyncio (PEP 3156)

Updated 7 days ago



Minimal asyncio memcached client

Updated 7 days ago

## [aiorwlock](#)

Python ★ 17 🔒 1

Synchronization primitive RLock for asyncio (PEP 3156)

Updated 7 days ago

## [aiosmtpd](#)

Python ★ 20 🔒 5

A reimplementation of the Python stdlib smtpd.py based on asyncio.

Updated 8 days ago

## [aiohttp\\_mako](#)

Python ★ 9 🔒 2

mako template renderer for aiohttp.web

Updated 8 days ago





## aiocouchdb

Python ★ 32 🔒 8

CouchDB client built on top of aiohttp (asyncio)

Updated 25 days ago

## async\_timeout

Python ★ 5 🔒 1

asyncio-compatible timeout class

Updated 28 days ago

## sphinxcontrib-asyncio

Python ★ 4 🔒 1

Sphinx extension to add asyncio-specific markups

Updated on Apr 15

## aioppspp

Python ★ 9 🔒 2

IETF PPSP RFC7574 in Python/asyncio

Updated on Feb 3

## aiorest

Python ★ 76 🔒 9

REST interface for server based on aiohttp (abandoned)

Updated on Apr 30, 2015

# LOOP DE EVENTOS PLUGÁVEL

---

- **asyncio** inclui um loop de eventos próprio
- API **AbstractEventLoopPolicy** permite substituir esse loop de eventos por outro que implemente **AbstractEventLoop**
  - **AsyncIOMainLoop** implementado pelo projeto **Tornado**
  - Loop de eventos de toolkits para GUI: **Quamash** (PyQt4, PyQt5, PySide)
  - Loops de eventos baseados na biblioteca **libuv**, base do Node.js

- Implementada com bindings em Cython sobre **libuv**
  - libuv é a biblioteca de I/O orientada a eventos do **Node.js**, usada também como biblioteca externa em outras linguagens: C++, Lua, Julia, Python, Java, Go, PHP, C# etc.
- Criada por Yuri Selivanov, que criou a sintaxe **async/await**
  - PEP 492 — Coroutines with async and await syntax



# INCORPORANDO UVLOOP

uvloop example · ramalho/t... x +

GitHub, Inc. (US) | <https://github.com/ramalho/tudo-agora/commit/e1abc22e4e8de8178e2d0e45d3c70c13071301e6> | vs code source repo

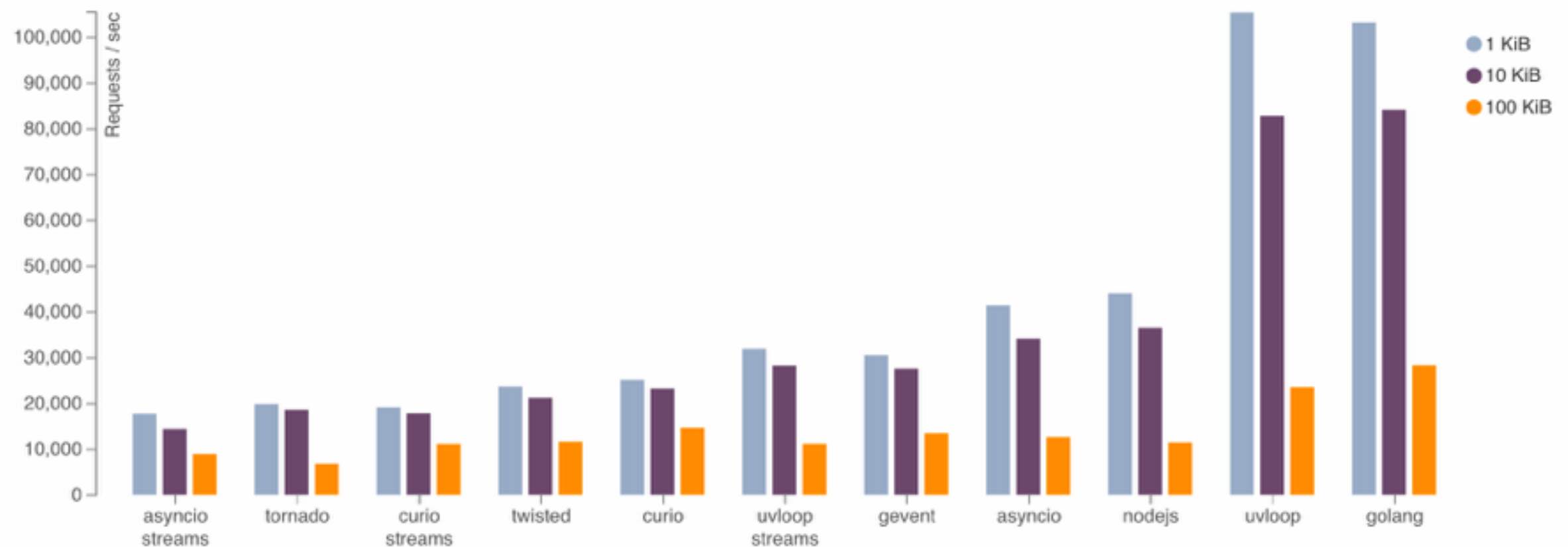
@@ -1,6 +1,6 @@			
1	"""Download flags of top 20 countries by population	1	"""Download flags of top 20 countries by population
2		2	
3	-Asynchronous version	3	+Asynchronous version with uvloop
4		4	
5	Sample run::	5	Sample run::
6		6	
@@ -15,6 +15,7 @@			
15		15	
16	import asyncio	16	import asyncio
17	import aiohttp	17	import aiohttp
		18	+import uvloop
18		19	
19	POP20_CC = ('CN IN US ID BR PK NG BD RU JP '	20	POP20_CC = ('CN IN US ID BR PK NG BD RU JP '
20	'MX PH VN ET EG DE IR TR CD FR').split()	21	'MX PH VN ET EG DE IR TR CD FR').split()
@@ -49,7 +50,8 @@ def save_flag(img, filename):			
49		50	
50		51	
51	def download_many(cc_list):	52	def download_many(cc_list):
52	- loop = asyncio.get_event_loop()	53	+ loop = uvloop.new_event_loop()
		54	+ asyncio.set_event_loop(loop)
53	task_list = [download_one(cc) for cc in cc_list]	55	task_list = [download_one(cc) for cc in cc_list]
54	super_task = asyncio.wait(task_list)	56	super_task = asyncio.wait(task_list)
55	done, _ = loop.run_until_complete(super_task)	57	done, _ = loop.run_until_complete(super_task)

# DESEMPENHO DO UVLOOP

## TCP

This benchmark tests the performance of a simple echo server with different message sizes. We use 1, 10, and 100 KiB packages. The concurrency level is 10. Each benchmark was run for 30 seconds.

See also the full TCP benchmarks [report](#).



Fonte: **uvloop: Blazing fast Python networking** — Yury Selivanov — 2016-05-03

<https://magic.io/blog/uvloop-make-python-networking-great-again/>

# CORROTINAS NATIVAS

---

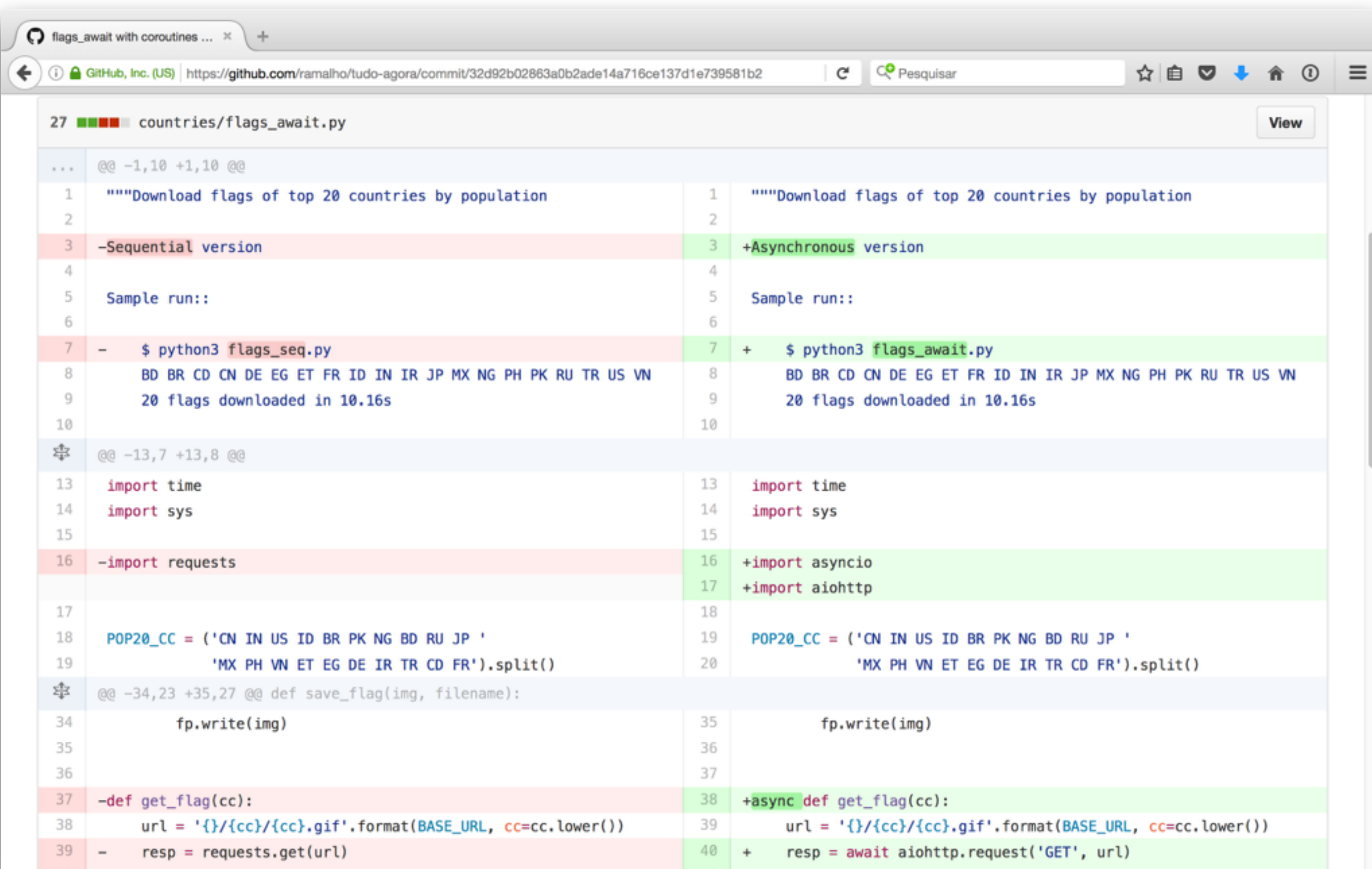
Finalmente, suporte sintático para concorrência em Python!

# NOVA SINTAXE ASYNC DEF

---

- **PEP 492**: Novas palavras reservadas em Python 3.5
- **async def** para definir *corrotinas nativas*
- **await** para delegar para objetos **Awaitable**
  - somente em corrotinas nativas
  - **Awaitable** ou "*Future-like*":
    - Instâncias de **Future**, **Task** e derivados
    - corrotinas nativas (**async def...**)
    - geradores-corrotinas decoradas (@types.coroutine)
    - objetos que implementam **\_\_await\_\_** (devolve um iterator)

# ANTES E DEPOIS (1)



The screenshot shows a GitHub diff view for the file `countries/flags_await.py`. The diff compares a previous version (left) with a newer version (right). The changes are highlighted with red and green backgrounds.

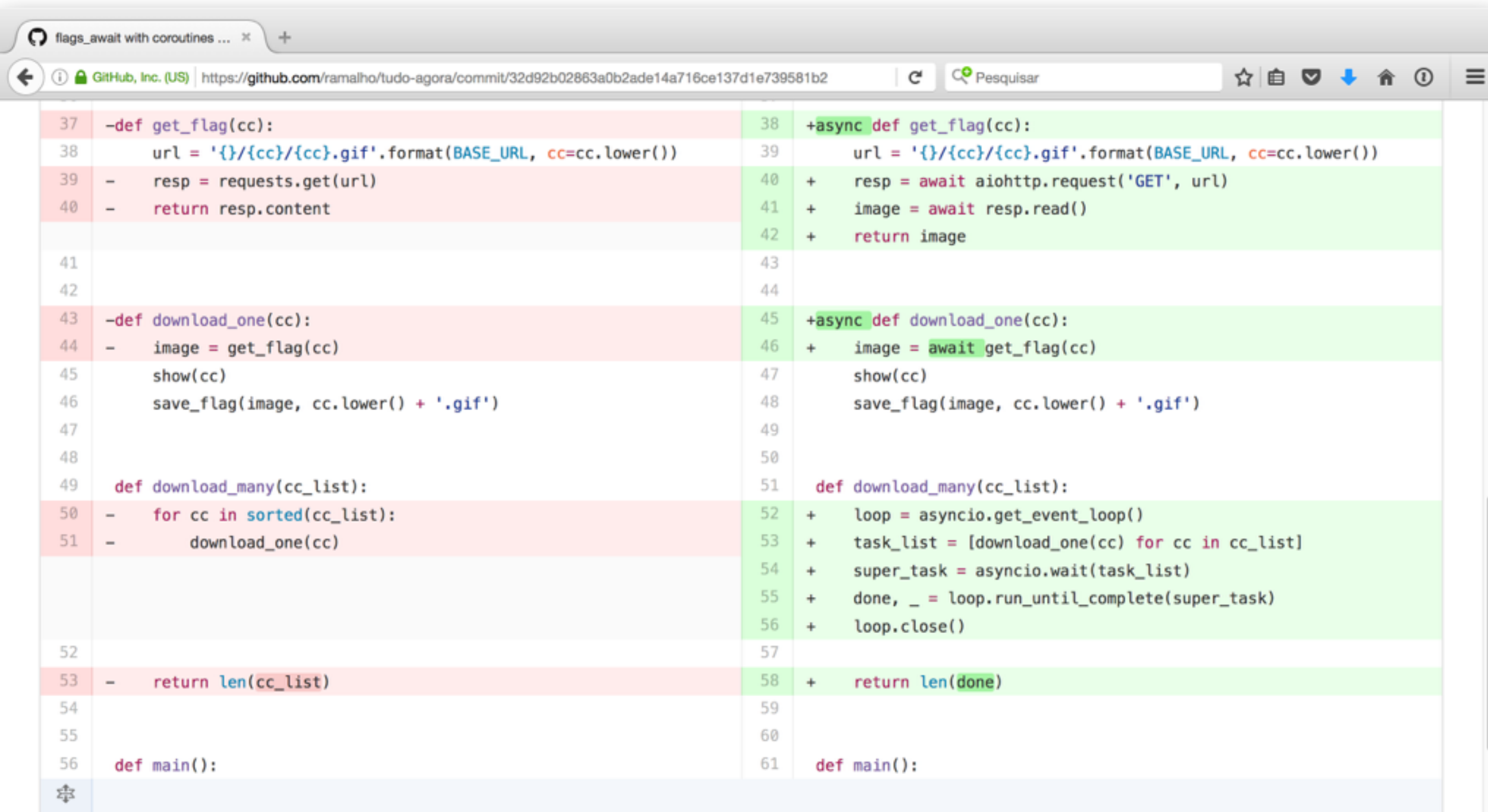
**Left Side (Previous Version):**

```
... @@ -1,10 +1,10 @@
1  """Download flags of top 20 countries by population
2
3  -Sequential version
4
5  Sample run::
6
7  - $ python3 flags_seq.py
8      BD BR CD CN DE EG ET FR ID IN IR JP MX NG PH PK RU TR US VN
9      20 flags downloaded in 10.16s
10
11 @@ -13,7 +13,8 @@
13  import time
14  import sys
15
16  -import requests
17
18  POP20_CC = ('CN IN US ID BR PK NG BD RU JP '
19             'MX PH VN ET EG DE IR TR CD FR').split()
20
21 @@ -34,23 +35,27 @@ def save_flag(img, filename):
34      fp.write(img)
35
36
37  -def get_flag(cc):
38      url = '{}/{cc}/{cc}.gif'.format(BASE_URL, cc=cc.lower())
39      - resp = requests.get(url)
```

**Right Side (New Version):**

```
1  """Download flags of top 20 countries by population
2
3  +Asynchronous version
4
5  Sample run::
6
7  + $ python3 flags_await.py
8      BD BR CD CN DE EG ET FR ID IN IR JP MX NG PH PK RU TR US VN
9      20 flags downloaded in 10.16s
10
11 @@ -13,7 +13,8 @@
13  import time
14  import sys
15
16  +import asyncio
17  +import aiohttp
18
19  POP20_CC = ('CN IN US ID BR PK NG BD RU JP '
20             'MX PH VN ET EG DE IR TR CD FR').split()
21
22 @@ -34,23 +35,27 @@ def save_flag(img, filename):
35      fp.write(img)
36
37
38  +async def get_flag(cc):
39      url = '{}/{cc}/{cc}.gif'.format(BASE_URL, cc=cc.lower())
40      + resp = await aiohttp.request('GET', url)
```

# ANTES E DEPOIS (2)



```
37 -def get_flag(cc):
38     url = '{}/{cc}/{cc}.gif'.format(BASE_URL, cc=cc.lower())
39 -    resp = requests.get(url)
40 -    return resp.content

41
42
43 -def download_one(cc):
44 -    image = get_flag(cc)
45     show(cc)
46     save_flag(image, cc.lower() + '.gif')
47
48
49     def download_many(cc_list):
50 -        for cc in sorted(cc_list):
51 -            download_one(cc)

52
53 -    return len(cc_list)

54
55
56 def main():

38 +async def get_flag(cc):
39     url = '{}/{cc}/{cc}.gif'.format(BASE_URL, cc=cc.lower())
40 +    resp = await aiohttp.request('GET', url)
41 +    image = await resp.read()
42 +    return image

43
44
45 +async def download_one(cc):
46 +    image = await get_flag(cc)
47     show(cc)
48     save_flag(image, cc.lower() + '.gif')
49
50
51     def download_many(cc_list):
52 +        loop = asyncio.get_event_loop()
53 +        task_list = [download_one(cc) for cc in cc_list]
54 +        super_task = asyncio.wait(task_list)
55 +        done, _ = loop.run_until_complete(super_task)
56 +        loop.close()

57
58 +    return len(done)

59
60
61 def main():
```

0 comments on commit 32d92b0

Lock conversation



Write

Preview

AA B i

“ <> 🔗

⋮ ⋮ ⋮

↩ @ 🌟



# CORROTINAS NATIVAS EM AÇÃO

---

```
38  async def get_flag(cc):
39      url = '{}/{cc}/{cc}.gif'.format(BASE_URL, cc=cc.lower())
40      resp = await aiohttp.request('GET', url)
41      image = await resp.read()
42      return image
43
44
45  async def download_one(cc):
46      image = await get_flag(cc)
47      show(cc)
48      save_flag(image, cc.lower() + '.gif')
49
50
51  def download_many(cc_list):
52      loop = asyncio.get_event_loop()
53      task_list = [download_one(cc) for cc in cc_list]
54      super_task = asyncio.wait(task_list)
55      done, _ = loop.run_until_complete(super_task)
56      loop.close()
57
58      return len(done)
```



# MAIS SUPORTE SINTÁTICO

---

- Ainda no **PEP 492**:
  - **async with**:  
invoca métodos especiais assíncronos **`__aenter__*`** e **`__aexit__*`**
    - **`*`**: devolvem objetos **`Awaitable`**
  - **async for**:  
invoca métodos especiais **`__aiter__`** e **`__anext__*`**
    - **`__aiter__`**: devolve iterador assíncrono (que implementa **`__anext__*`**)

# EXEMPLO COM COMANDOS NOVOS

---

```
1  import asyncio
2  import aiopg
3
4  dsn = 'dbname=aiopg user=aiopg password=passwd host=127.0.0.1'
5
6  async def go():
7      async with aiopg.create_pool(dsn) as pool:
8          async with pool.acquire() as conn:
9              async with conn.cursor() as cur:
10                 await cur.execute("SELECT 1")
11                 ret = []
12                 async for row in cur:
13                     ret.append(row)
14                 assert ret == [(1,)]
15
16  loop = asyncio.get_event_loop()
17  loop.run_until_complete(go())
```

# AINDA MAIS SUPORTE SINTÁTICO

---

- Novidades do **Python 3.6**:
  - **PEP 525**: Asynchronous Generators
  - **PEP 530**: Asynchronous Comprehensions (Python 3.6)

# CONCLUSÃO

---

The End

# MINHA OPINIÃO

---

- Eco-sistema ainda jovem: bibliotecas em evolução
  - alguns exemplos do Fluent Python agora geram avisos
- **asyncio** com sua política extensível é uma base sólida para construir o futuro
  - loops de eventos externos demonstra isso (ex. **uvloop**, **pyuv**)
- Experimente **Python 3.5** antes de partir para Go, Elixir, Clojure ou Node 😈!

# THE ONE (ABSTRACT) LOOP

---

*One Loop to rule them all, One Loop to find them,  
One Loop to bring them all and in liveness bind them.*

# ¿PREGUNTAS?

ThoughtWorks®

**LUCIANO RAMALHO**

---

*Technical Principal*

---

@ramalhoorg  
luciano.ramalho@thoughtworks.com

ThoughtWorks®