

# Usando asyncio para reescrever processamento sem dependência em código concorrente

---

Alexandre Yukio Harano

São Paulo, 18 de Janeiro de 2018 – Grupy-SP

Núcleo de Informação e Coordenação do Ponto BR (NIC.br)

harano@nic.br

alexandre@harano.net.br

<https://alexandre.harano.net.br/>

1. Motivação
2. Histórico
3. Mecanismos disponíveis
4. Exemplos de Uso

# Motivação

---

**Python é lento.**

# Python é lento.

**Why Python is Slow: Looking Under the Hood**

<https://jakevdp.github.io/blog/2014/05/09/why-python-is-slow/>

**Python é lento?**

# Tempos de Resposta

Ação	Tempo de resposta (ns)
execução de instrução típica	1 ns
recuperar de memória cache L1	0,5 ns
errar predição condicional	5 ns
recuperar de memória cache L2	7 ns
travar/destravar mutex	25 ns
recuperar de memória principal	100 ns
enviar 2K bytes via rede 1Gbps	20 000 ns
ler 1MB sequencialmente da memória	250 000 ns
recuperar de um novo local do disco (busca)	8 000 000 ns
ler 1MB sequencialmente do disco	20 000 000 ns
enviar um pacote dos EUA para a Europa e receber a resposta	150 000 000 ns

**Fonte:** Teach Yourself Programming in Ten Years, por Peter Norvig  
<http://norvig.com/21-days.html#answers>



# Histórico

---



## PEPs relacionadas

**PEP 380** Syntax for Delegating to a Subgenerator (3.3+)

**PEP 3156** Asynchronous IO Support Rebooted: the "asyncio" Module (3.4+)

**PEP 492** Coroutines with `async` and `await` syntax (3.5+)

**PEP 525** Asynchronous Generators (3.6+)

**PEP 530** Asynchronous Comprehensions (3.6+)

# PEP 380 e PEP 492

Python 3.4+ (PEP 380)	Python 3.5+ (PEP 492)
<pre>import asyncio  @asyncio.coroutine def hello_world():     print("Hello World!")  @asyncio.coroutine def main():     yield from hello_world()  loop = asyncio.get_event_loop() loop.run_until_complete(main()) loop.close()</pre>	<pre>import asyncio  async def hello_world():     print("Hello World!")  async def main():     await hello_world()  loop = asyncio.get_event_loop() loop.run_until_complete(main()) loop.close()</pre>

## Baseado em:

**Python 3.4** Hello World Coroutine

<https://docs.python.org/3.4/library/asyncio-task.html#example-hello-world-coroutine>

**Python 3.5** Hello World Coroutine

<https://docs.python.org/3.5/library/asyncio-task.html#example-hello-world-coroutine>



<https://github.com/ayharano/aio-exemplo>

# **Mecanismos disponíveis**

---

# Mecanismos disponíveis

`asyncio.AbstractEventLoop`

Efetua o controle da concorrência das corrotinas.

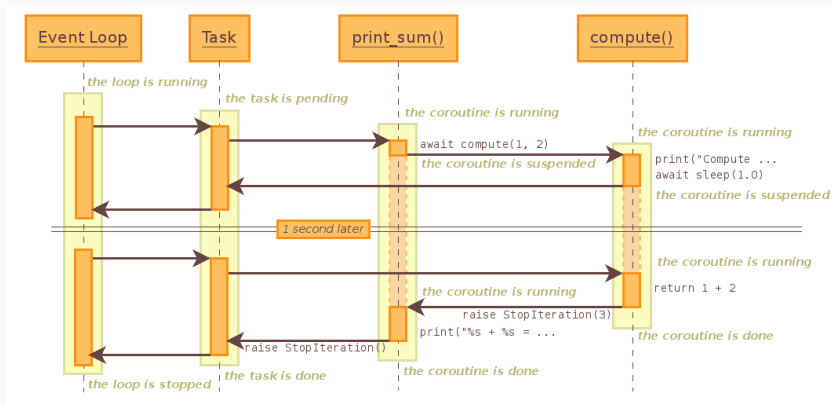
`run_forever()`

Executa continuamente até `stop()` ser chamado.

`run_until_complete(future)`

Executa até `future` ser completado.

# Diagrama sobre corrotinas



**Fonte:** Example: Chain coroutines

<https://docs.python.org/3/library/asyncio-task.html#example-chain-coroutines>

# Mecanismos disponíveis

## `asyncio.Future`

Permite a execução de corrotinas de forma não bloqueante: é usado para armazenar o resultado do cálculo efetuado dentro de uma corrotina quando passado por parâmetro. Pode disparar um *callback* se configurado para tal, mas não é necessário.

`set_result(valor)` usado para armazenar um valor.

`result()` usado para obter valor calculado dentro de uma corrotina. *Só pode ser chamado depois que valor tiver sido atribuído!*

# Mecanismos disponíveis

```
asyncio.ensure_future(coro_or_future, *,  
loop=None)
```

Agenda a execução de uma corrotina.

```
asyncio.shield(arg, *, loop=None)
```

Protege a corrotina passada por parâmetro contra cancelamento da corrotina exterior.

```
asyncio.sleep(delay, result=None, *,  
loop=None)
```

Corrotina para esperar delay segundos.

# Mecanismos disponíveis

```
asyncio.gather(*coros_or_futures, loop=None,  
return_exceptions=False)
```

Agrega os resultados dos `asyncio.Future` passados por parâmetro de acordo com a ordem de instanciação da sequência original (não necessariamente a ordem de término).

```
asyncio.wait_for(fut, timeout, *, loop=None)
```

Espera `timeout` segundos: caso não termine no tempo, cancela `fut` e lança `asyncio.TimeoutError`.



# Mecanismos disponíveis

```
asyncio.wait(futures, *, loop=None,  
timeout=None, return_when=ALL_COMPLETED)
```

Espera completar `asyncio.Future` e corrotinas passadas por parâmetro. A sequência do parâmetro *não pode ser vazia*. São 3 condições de devoluções possíveis:

`FIRST_COMPLETED` assim que algum completar.

`FIRST_EXCEPTION` assim que a primeira `Exception` for lançada. Caso não tenha nenhuma, é equivalente a `ALL_COMPLETED`.

`ALL_COMPLETED` somente quando todos completarem. Opção *default*.

# Mecanismos disponíveis

```
asyncio.wait(futures, *, loop=None,  
timeout=None, return_when=ALL_COMPLETED)
```

Uso:

```
done, pending = await asyncio.wait(fs)
```

Entrega uma tupla de dois valores: done, pending.

Não lança `asyncio.TimeoutError`: os não completos dentro de `timeout` são entregues dentro de `pending`.

## Exemplos de Uso

---

**Cliente/servidor http**   aiohttp

**Cliente/servidor ftp**   aioftp

**Banco de dados**   aiopg  
                      aiomysql

**Outros**   aioredis  
           aiodocker  
           ...

# asyncio boilerplate

```
#!/usr/bin/env python3
""" asyncio working boilerplate for Python 3.5+. """

import asyncio
import sys

async def main(argv, *args, **kwargs):
    return

if __name__ == '__main__':
    _LOOP = asyncio.get_event_loop()
    _LOOP.run_until_complete(main(sys.argv))
    _PENDING = asyncio.Task.all_tasks()
    _LOOP.run_until_complete(asyncio.gather(*_PENDING))
    _LOOP.stop()
    _LOOP.close()
```

Disponível em [https://github.com/ayharano/aio-exemplo/blob/master/aio\\_boilerplate.py](https://github.com/ayharano/aio-exemplo/blob/master/aio_boilerplate.py)



# Lista de tarefas pendentes

## **Propósito** Apresentar feedback visual via CLI

1. Recebe `dict` que mapeia `asyncio.Future` para um rótulo.
2. Inicializa contador em zero.
3. Calcula set de instâncias de `asyncio.Future` pendentes.
4. Inicializa set de esperas de `asyncio.sleep`.

# Lista de tarefas pendentes

## 5. Enquanto existirem `asyncio.Future` pendentes:

5.1 Chama `asyncio.sleep` e armazena no set de esperas.

5.2 Chama `asyncio.wait` com a condição de `FIRST_COMPLETED`.

5.3 Analisa as instâncias de `asyncio.Future` completadas:

- Se tiver `asyncio.Future` com rótulo, armazena o rótulo.
- Registra se a espera mais recente estiver completa.

5.4 Apresenta todos os rotulados completados na ordem.

5.5 Se tiver algum rotulado completo, zera o contador.

Senão:

Se contador completou um ciclo, imprime todos os rótulos dos `asyncio.Future` pendentes com quebra de linha e zera o contador.

Senão, incrementa o contador e imprime ' '.

5.6 Atualiza set de `asyncio.Future` pendentes como interseção das rotuladas.



# Exemplo de apresenta\_pendências.py

```
(aio-exemplo-kaFLRWdV) bash-3.2$ python3 apresenta_pendências.py
..... [faltam A, B, C]
..... [faltam A, B, C]
..... [faltam A, B, C]
..... [faltam A, B, C]
..... A completou!
..... [faltam B, C]
..... [faltam B, C]
..... B completou!
..... [falta C]
..... [falta C]
..... C completou!
!
```



# Estrutura usual de processamento

**Coleta**

**Processamento**

**Análise**

**Apresentação**

# Estatísticas de livros do Machado de Assis

## Coleta

Extração da versão txt dos livros do Machado de Assis disponíveis no Project Gutenberg.

## Processamento

Leitura dos arquivos para memória e extração das linhas de interesse de cada livro.

## Análise

Análise das linhas de acordo com critérios arbitrários.

## Apresentação

Apresentação de estatísticas coletadas ao usuário.

# Project Gutenberg



search for books

- Browse Catalog
- Bookshelves
- Main Page
- Categories
- News
- Contact Info

Project Gutenberg appreciates your donation!

[Donate](#)

- Why donate?

in other languages

- Português
- Deutsch
- Français

hosted by

## Free ebooks - Project Gutenberg

[Book search](#) · [Book categories](#) · [Browse catalog](#) · [Mobile site](#) · [Report errors](#) · [Terms of use](#)

## Some of the Latest Books



## Welcome

**Project Gutenberg** offers over 56,000 free eBooks: Choose among free epub books, free kindle books, download them or read them online. You will find the world's great literature here, especially older works for



<https://github.com/ayharano/aio-exemplo>



O Projeto Gutenberg oferece 56.317 ebooks gratuitos para download.

[Doar](#) [Flattr this!](#)

[Pesquisar](#) [Mais recentes](#) [Termos de uso](#) [Doações?](#) [Móvel](#)

[Ajuda](#)

## Livros: machado de assis (classificados por popularidade)



**Autores**  
2 nomes de autor correspondem a sua pesquisa.



**Classificar por ordem alfabética**



**Classificar por data de publicação**

Exibindo resultados 1-7



**Dom Casmurro (português)**  
Machado de Assis  
142 downloads



**Memórias Postumas de Braz Cubas (português)**  
Machado de Assis  
69 downloads



**Brazilian Tales**  
Machado de Assis, Medeiros e Albuquerque, Henrique Coelho Netto, and Carmen Dolores  
44 downloads



**Histórias Sem Data (português)**  
Machado de Assis  
32 downloads



**Ouncas Borba (português)**  
Machado de Assis  
31 downloads




**A Mão e a Luva (português)**  
Machado de Assis  
30 downloads



**Memorial de Ayres (português)**  
Machado de Assis  
30 downloads

# Project Gutenberg



O Projeto Gutenberg oferece 56.319 ebooks gratuitos para download.


[Doar](#) [Flattr this!](#)

[Pesquisar](#) [Mais recentes](#) [Termos de uso](#) [Doações?](#) [Móvel](#)

[Ajuda](#)




















Projeto Gutenberg > [56.319 ebooks livres](#) > [6 por Machado de Assis](#)


## Dom Casmurro by Machado de Assis



[Download](#) [Bibrec](#)

### Faça o download deste ebook

	Formato ?	Tamanho	?	?	?
	<a href="#">Read this book online: HTML</a>	442 kB			
	<a href="#">EPUB (com imagens)</a>	216 kB			
	<a href="#">EPUB (sem imagens)</a>	216 kB			
	<a href="#">Kindle (com imagens)</a>	842 kB			
	<a href="#">Kindle (sem imagens)</a>	842 kB			
	<a href="#">Texto sem formatação em UTF-8</a>	409 kB			
	<a href="#">Mais arquivos...</a>				



# Information About Robot Access to our Pages



## Information About Robot Access to our Pages

**The Project Gutenberg website is intended for human users only.** Any perceived use of automated tools to access the Project Gutenberg website will result in a temporary or permanent block of your IP address. The only exceptions to this rule are below.

### Contents [hide]

- 1 How To Get All Ebook Files
- 2 How to Get Certain Ebook files
- 3 How To Mirror Project Gutenberg
- 4 How To Get Catalog Data

## How To Get All Ebook Files

The best way to have a local up-to-date copy of all files is to setup a private mirror: See: [the Mirroring How-To](#)

## How to Get Certain Ebook files

**wget** is free software and available for Linux, Windows, and Mac OS X at [www.gnu.org/software/wget/](http://www.gnu.org/software/wget/) [↗](#).



# Information About Robot Access to our Pages

Algumas regras sobre a coleta automatizada (pelo Termos de Uso do Project Gutenberg, a partir de 100 livros por dia é considerado automatizado):

- Esperar 2 segundos entre as coletas;
- Usar mirrors para coletar os livros (deveríamos utilizar neste exemplo).



# Estatísticas de livros do Machado de Assis

Foram montadas 4 versões de módulos:

- Versão síncrona e assíncrona.
- Agrupada por etapa e agrupada por livro (afeta processamento e análise).

Funções de coleta, processamento por livro, análise por livro e apresentação foram reunidas por categoria de sincronicidade (

`_base_estatísticas_livro_assíncrono.py` e  
`_base_estatísticas_livro_síncrono.py` )





# Estatísticas de livros do Machado de Assis

Função	Módulos síncronos	Módulos assíncronos
Cliente http	requests	aiohttp
Manipulação de arquivos	pathlib (stdlib)	aiofiles

Além disso, foi utilizado o BeautifulSoup para representação dos arquivos HTML.

# Estatísticas de livros do Machado de Assis

## Exemplo de apresentação de estatísticas de um livro:

[ Estatísticas de 'Memórias Postumas de Braz Cubas' de 'Machado de Assis' ]

Número de linhas sem nenhum caractere visível: 2232

Número de linhas com caractere visível: 6208

Total de linhas: 8440

Número de caracteres visíveis: 300860

Número de sequências contíguas de caracteres visíveis: 61549

Caracter sensível a maiúsculas e minúsculas mais utilizado: 'a' (36332 vezes).

Caracter insensível a maiúsculas e minúsculas mais utilizado: 'a' (36978 vezes).

Sequência sensível a maiúsculas e minúsculas mais utilizada: 'a' (2360 vezes).

Sequência insensível a maiúsculas e minúsculas mais utilizada: 'a' (2537 vezes).

Maiores sequências sensíveis a maiúsculas e minúsculas:

'Constantinopla,--modernas,--em', 'tremula,--coitadinha,--tremula' (30 caracteres).

Maiores sequências insensíveis a maiúsculas e minúsculas

'constantinopla,--modernas,--em', 'tremula,--coitadinha,--tremula' (30 caracteres).



# Estatísticas de livros do Machado de Assis

## Análise via cProfile e pstats

### Condição

Arquivos dos livros já foram previamente coletados em execução anterior.

	Módulo	Tempo (s)	Chamadas de função
estatísticas_livro_síncrono_agrupado_por_etapa		4,995	6854857
estatísticas_livro_síncrono_agrupado_por_livro		5,329	6854846
estatísticas_livro_assíncrono_agrupado_por_etapa		9,406	12267885
estatísticas_livro_assíncrono_agrupado_por_livro		9,030	12268402

Observação: exemplo de uso de cProfile e pstats em

[http://stefaanlippens.net/python\\_profiling\\_with\\_pstats\\_interactive\\_mode/](http://stefaanlippens.net/python_profiling_with_pstats_interactive_mode/)



<https://github.com/ayharano/aio-exemplo>

# Estatísticas de livros do Machado de Assis

Por que as versões assíncronas foram piores nesse caso?

- Os livros já estavam coletados.
- Mesmo nas versões assíncronas, as regras de robôs foram respeitadas, então a coleta é sequencial.
- Uma vez que os livros estejam em memória, nesse caso, o uso de `for` sequencial é mais vantajoso, pois são menos chamadas de função (instanciação de `asyncio.Future` e `asyncio.ensure_future`).
- Caso estivesse projetado o uso de outros recursos de I/O externo (disco, BD, rede, etc), o resultado poderia ser outro.



# Obrigado!

**Alexandre Yukio Harano**

**harano@nic.br**

**alexandre@harano.net.br**

**<https://alexandre.harano.net.br/>**

# Referências i



Łukasz Langa.

## **Thinking in Coroutines - PyCon 2016.**

Palestra – Vídeo – Slides, Maio 2016.

Último acesso em 18/1/2018.



Luciano Ramalho.

## ***Fluent Python.***

O'Reilly Media, Agosto 2015.

# Referências ii



Brett Cannon.

## **How the heck does async/await work in Python 3.5?**

<https://snarky.ca/how-the-heck-does-async-await-work-in-python-3-5/>, Fevereiro 2016.

Último acesso em 18/1/2018.



Laura F. D.

## **asyncio: A dumpster fire of bad design.**

<https://veriny.tf/asyncio-a-dumpster-fire-of-bad-design/>,  
Dezembro 2017.

Último acesso em 18/1/2018.



<https://github.com/ayharano/aio-exemplo>

# Referências iii



The Python Software Foundation.

**PEP 380 — Syntax for Delegating to a Subgenerator.**

<https://www.python.org/dev/peps/pep-0380/>,

Fevereiro 2009.

Último acesso em 18/1/2018.



The Python Software Foundation.

**PEP 3156 — Asynchronous IO Support Rebooted: the "asyncio" Module.**

<https://www.python.org/dev/peps/pep-3156/>,

Dezembro 2012.

Último acesso em 18/1/2018.



<https://github.com/ayharano/aio-exemplo>



# Referências iv



The Python Software Foundation.

**asncio — Asynchronous I/O, event loop, coroutines and tasks.**

<https://docs.python.org/3/library/asncio.html>,  
Março 2014.

Último acesso em 18/1/2018.



The Python Software Foundation.

**PEP 492 — Coroutines with async and await syntax.**

<https://www.python.org/dev/peps/pep-0492/>, Abril  
2015.

Último acesso em 18/1/2018.



<https://github.com/ayharano/aio-exemplo>

# Referências v



The Python Software Foundation.

## **PEP 525 — Asynchronous Generators.**

<https://www.python.org/dev/peps/pep-0525/>, Julho 2016a.

Último acesso em 18/1/2018.



The Python Software Foundation.

## **PEP 530 — Asynchronous Comprehensions.**

<https://www.python.org/dev/peps/pep-0530/>, Setembro 2016b.

Último acesso em 18/1/2018.



<https://github.com/ayharano/aio-exemplo>



Jake VanderPlas.

## **Why Python is Slow: Looking Under the Hood.**

<https://jakevdp.github.io/blog/2014/05/09/why-python-is-slow/>, Maio 2014.

Último acesso em 18/1/2018.



<https://github.com/ayharano/aio-exemplo>