Programação Assíncrona e Síncrona

Bruno Kobashigawa de Lima, 1810842

Instituto de Ensino Superior Sumaré - Centro Universitário Sumaré (ISES)
São Paulo - SP- Brasil - Departamento de Informática

Resumo
Esse material tem como ênfase focar na explicação das diferenças e exemplos da programação assíncrona e síncrona nos tempos atuais. Com o grande avanço na programação, algoritmos foram criados para realizar um uso concorrente sem precisar recorrer as múltiplas threadings, usando o event loops no lugar e com a simplicidade do python com sua sintaxe fácil.

1 Introdução

Esse artigo irá tratar sobre a diferença e explicação da programação assíncrona e síncrona. Nele terá contigo os seguintes tópicos:

3. A Programação Assíncrona
4. Quando e por que é útil programar assincronamente
5. Tarefas Assíncronas
6. Tarefas Paralelas
7. AsyncIO
8. Coroutines
9. Event Loop
10. Exemplos de Códigos
11. A Programação Síncrona

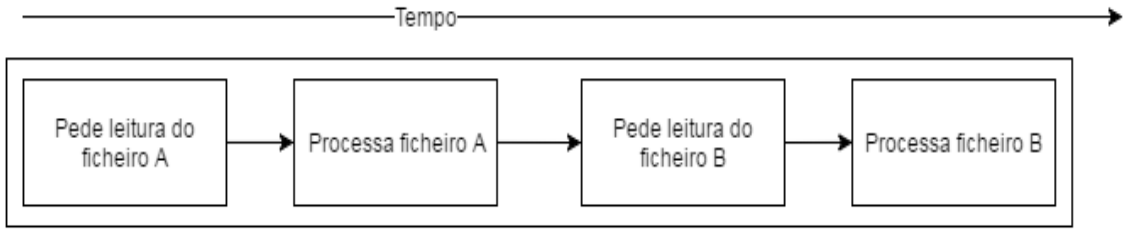
2 Origem do Projeto

Ao longo das minhas pesquisas sobre a linguagem Pyhton, desenvolvi um certo interesse ao tipo de algoritmos feito pela comunidade, em especial a programação Assíncrona e Síncrona. Ambas são de extrema importância no mundo de desenvolvimento, pois cada uma possui seu uso próprio. A programação Síncrona não é novidade, pois usamos frequentemente no ambiente de desenvolvimento.

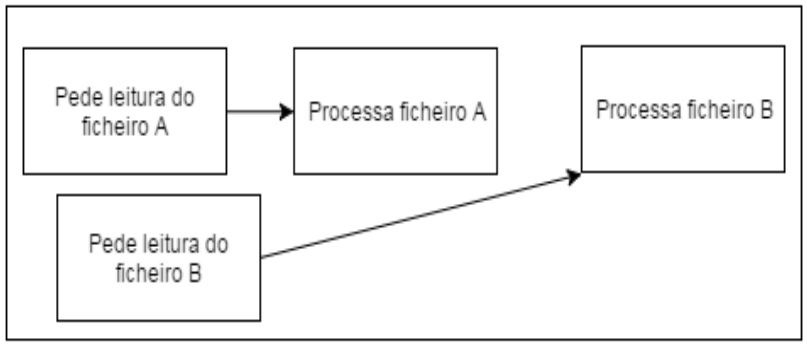
3 A Programação Assíncrona

A programação assíncrona, nada mais é que um código que não necessariamente precisa de uma instrução acabar para poder chamar outra. Esses códigos podem ser rodados em múltiplas funções em um único processo e até uma thread apenas. Resumindo, uma função suspende e outra entra em execução e suspende novamente e outra entra em execução, assim por diante e de uma forma muito rápida.

Para acontecer o poder do código assíncrono, o mesmo deve ser escrito na ordem de assincronia, então deve se escrever as funções com comandos bem definidos e as mesmas devem dar a oportunidade de outra função entrar nessas linhas.



Pedido de I/O sincrono

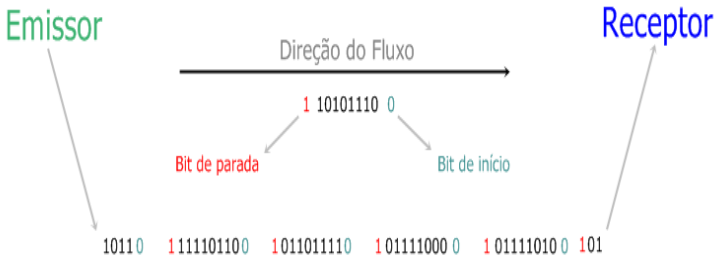


m Pedido de I/O assincrono

4 Quando e por que é útil programar assincronamente?

4.1 Quando uma operação é demorada

Se uma operação é demorada a thread que a está a executar teria que esperar até que ela completasse.



4.2 Para não bloquear a user-interface

Normalmente as user-interfaces processam eventos (cliques em botões, movimento do rato, teclas, ...). Estes mesmos eventos não devem demorar muito tempo para que a user-interface possa continuar a processá-los.

5 Tarefas Assíncronas

Tarefas assíncronas são códigos que disparam uma chamada de instrução e não espera a resposta para poder disparar outra função em cima da mesma. Fazendo isso em uma só thread.

6 Tarefas Paralelas

Já as tarefas paralelas irão separar uma novo thread e isolar a instrução de código para ser executada, no entanto continuará processando de forma síncrona.

7 AsyncIO

AsyncIO é um módulo novo que chegou a nova versão do python (3.4) que possibilita a escrita de códigos assíncronos em uma única thread.

8 Coroutines

Essas são rotinas cooperativas, que concordam em parar a sua execução fazendo com que outra rotina entre em ação, até a mesma fazer a primária voltar. Fazendo assim uma cooperação.

9 Event Loop

É um loop que pode registrar tarefas para serem executadas, executa-las e pausa-las e então pode retomar em execução a qualquer momento.

10 Exemplo de códigos

Um exemplo do uso dos códigos assíncronos será realizado em um mecanismo simples.

```
import asyncio

    def print_and_repeat(loop):
        print('Hello World')
        loop.call_later(2, print_and_repeat, loop)

    loop = asyncio.get_event_loop()
    loop.call_soon(print_and_repeat, loop)
    loop.run_forever()
```

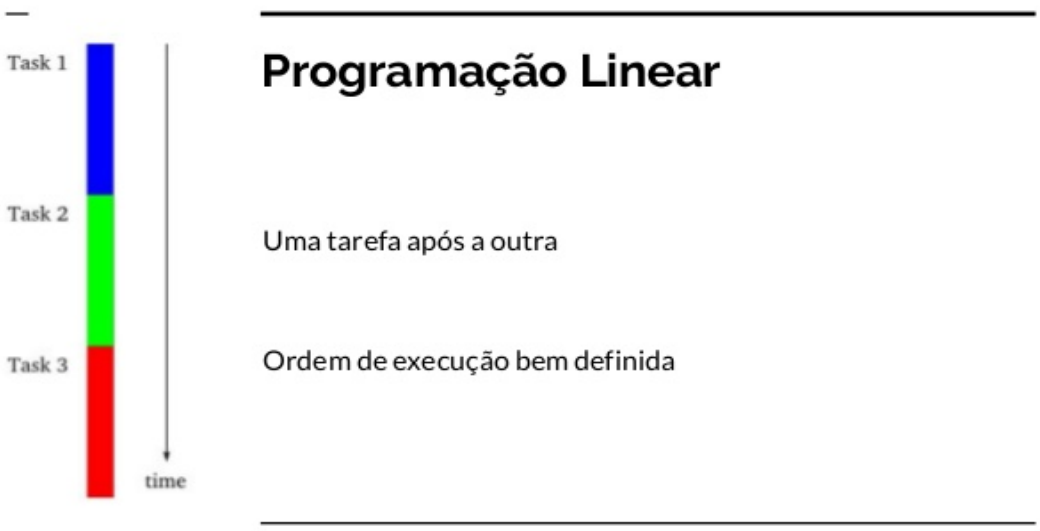
A linha *importasyncio* irá chamar o módulo de assincronia, possibilitando a programação assíncrona. A variável Loop contém o loop de eventos, uma vez que chamamos $asyncio.get_eventloop()$ e esse retorna o loop de eventos atual. Na linha seguinte, chamamos o método $call_soon$ para agendar a chamado de um método. $Call_soon$ adiciona a chamada da função $print_and_repeat$, definida anteriormente. O segundo parâmetro de $call_soon$ é o parâmetro para $print_and_repeat$. Assim o $loop.call_soon(print_and_repeat, loop)$ adiciona ao loop de eventos a chamada de função $print_and_repeat$, passando o loop para primeiro parâmetro.

Resultado é o seguinte:

```
Hello World
Hello World
Hello World
Hello World
Hello World
```

11 A Programação Síncrona

Em termos simples, programação síncrona é fazer uma coisa a seguir à outra. Podemos dizer que é a forma padrão de ser feita, um comando realizado apenas com outra instrução feita.



m

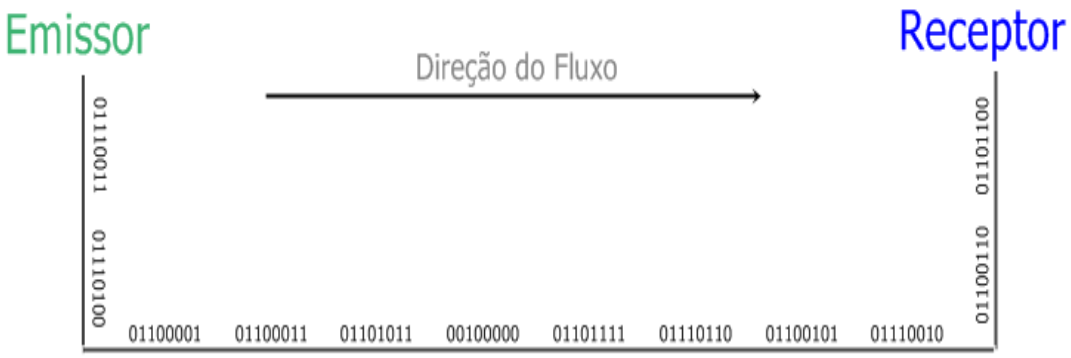
Exemplo de um código síncrono:

```
python = "Ola"
anaconda = "Mundo"

print(python, anaconda)
```

Resultado:

```
Ola mundo

Process finished with exit code 0
```

12 Exemplo Gráfico Síncrono



m

13 Referências

Aylan, 2018 | Devmedia | www.devmedia.com.br/programacao-assincrona-com-python/40258

Comunidade de usuários, 2016,2017 | StackOverflow |

Menezes, Nilo, 2014 | Nilo Menezes > Blog | blog.nilo.pro.br/posts/2014-06-28-python-asyncio-metodos-assincronos-em-python/

Oliveira Wisner, 2018 | Medium | medium.com/@wisner.oliveira/desvendando-programa