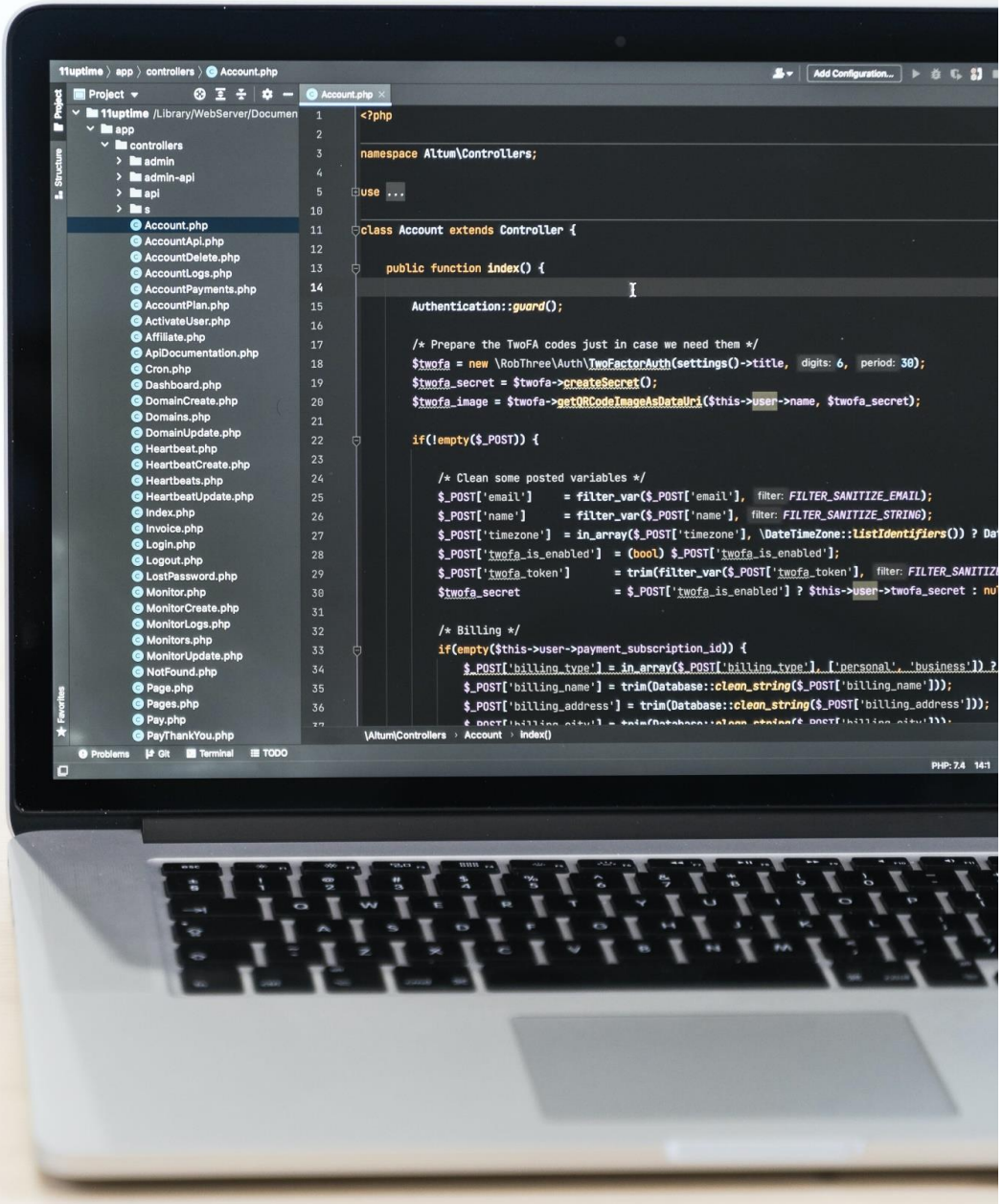
ANALYTICS

Monitoria 5 de Python

Acelerando a programação científica

HDI SEGUROS

ANALYTICS



Problema real da vida do programador

Um código que demora 1 semana para rodar e fazer todas as contas em um Dataframe.

HDI SEGUROS

Mas por que isso acontece?

- ✓ Baixa capacidade de processamento e rede
- ✓ Programamos mal
- ✓ Não usamos a ferramenta certa para a demanda certa

HDI SEGUROS

Python não é uma linguagem de alto desempenho

Temos linguagens de programação muito mais rápidas como:

- Pascal
- C#
- Java
- Ada
- Julia
- Fortran
- Rust
- C++
- C

ANALYTICS

Então por que usamos o Python?

- Grátis
- Popular
- Fácil
- Versátil

HDI SEGUROS

ANALYTICS

A solução é simples:

Acelerar o Python

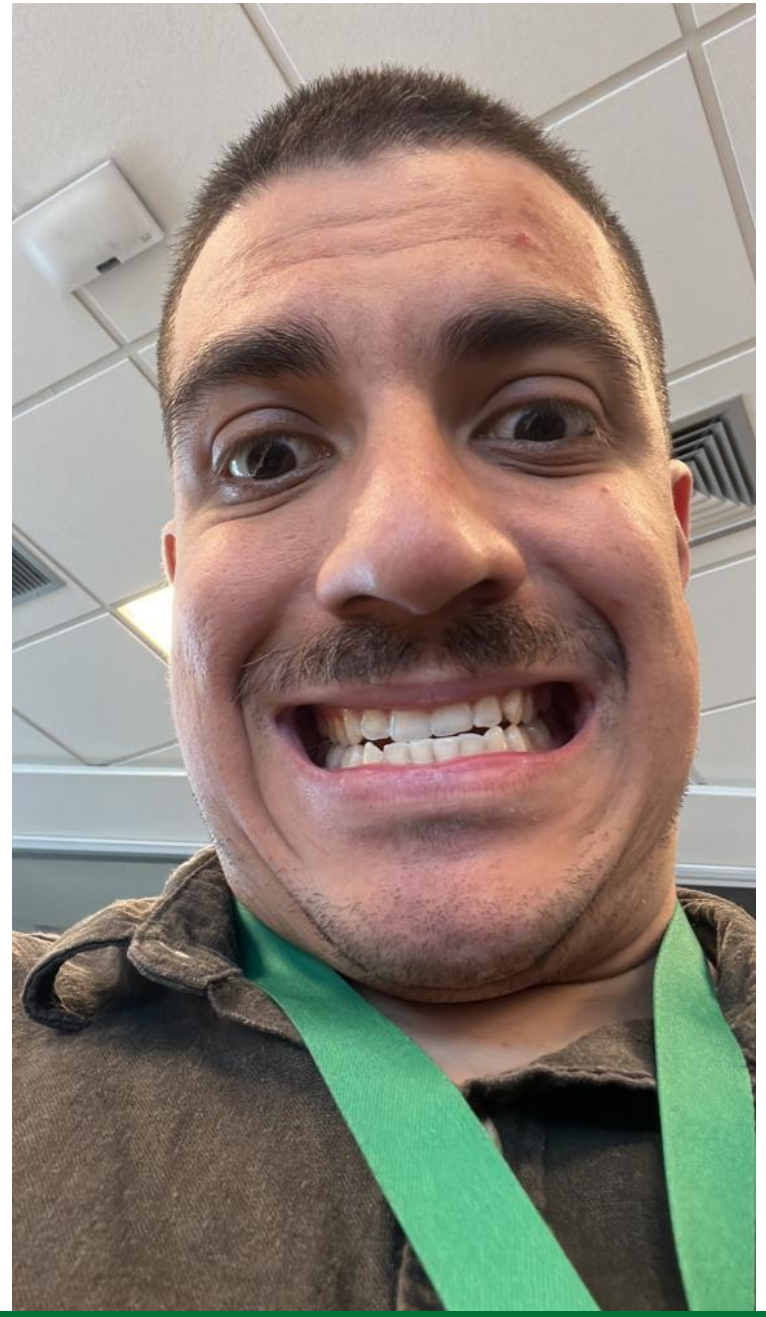
ANALYTICS

ATENÇÃO!!! O PROXIMO SLIDE É O MAIS IMPORTANTE DE TODA A MONITORIA.

PODEM TIRAR PRINT
E GUARDAR

HDI SEGUROS

ANALYTICS



HDI SEGUROS

Ferramentas para acelerar nosso código

JIT

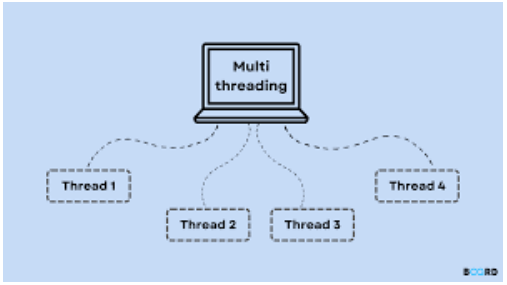
Numba

pypy

FERRAMENTAS DE BIG DATA

PySpark

THREADS



A ideia do Jit é compilar códigos “quentes”

Imagine um trecho de código que seja caro para executar *(sim, estamos falando do seu For infinito).* A partir do momento que ele precisa ser executado várias vezes, esse código é compilado.

ANALYTICS

O que é o processo de compilação?

Compilação é o ato / processo de traduzir um programa feito em uma linguagem de alto nível para uma linguagem de máquina, para que suas instruções sejam executadas pelo processador.

Ou seja, cria o executável de um programa escrito em uma linguagem de alto nível.

HDI SEGUROS

ANALYTICS

Vamos a prática

Nesse exemplo vamos usar uma sequência de Fibonacci não recursiva para capturamos o tempo de execução do código.

In [4]:
```
import timeit
from numba import njit

def fibonacci(n):
    a,b=0,1
    for _ in range(0, n-1):
        c=a+b
        a=b
        b=c
    return a

%timeit fibonacci(10000)
```

```
1.64 ms ± 22.1 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

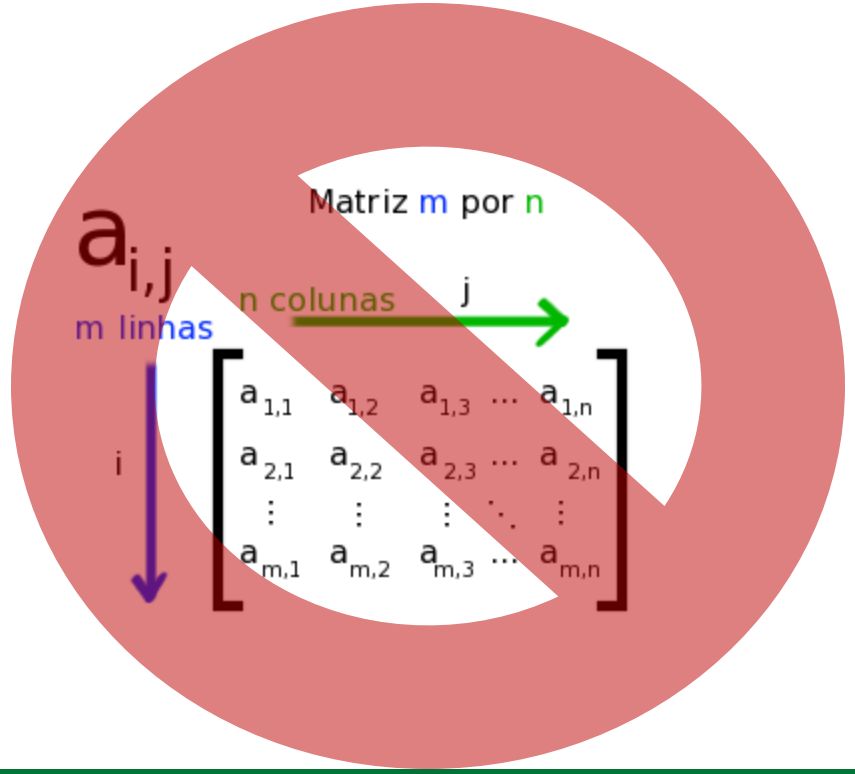
In [5]:
```
@njit
def fibonacci(n):
    a,b=0,1
    for _ in range(0, n-1):
        c=a+b
        a=b
        b=c
    return a

%timeit fibonacci(10000)
```

```
6.89 µs ± 1.15 µs per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

HDI SEGUROS

Importante

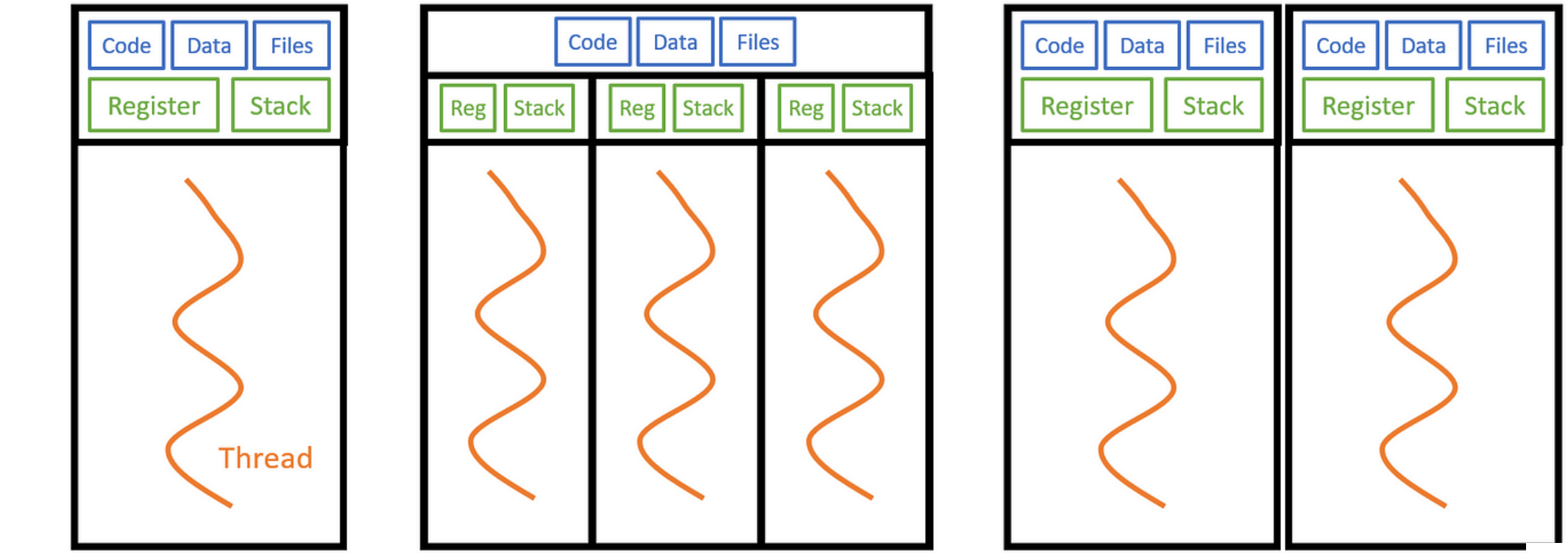
- O numba não funciona bem para contas com dimensões maiores que 3. Ou seja, ele não é bom para cálculos matriciais. Usem com moderação.



ANALYTICS

Multithreading

A tradução de thread é linha



Single Processor Single Thread

Single Processor Multithread

Multiprocessing

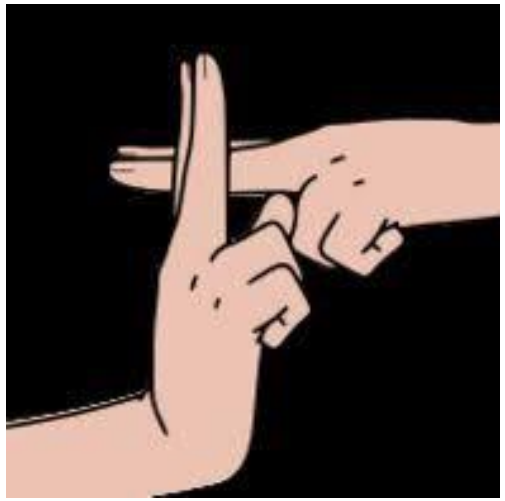
HDI SEGUROS

O que é Multithreading?

- Multithreading é a capacidade de um programa ou sistema operacional de permitir mais de um usuário ao mesmo tempo, sem exigir várias cópias do programa em execução no computador. O multithreading também pode lidar com várias solicitações do mesmo usuário.

- Cada solicitação do usuário para um programa ou serviço do sistema é rastreada como um thread com uma identidade separada. À medida que os programas funcionam em nome da solicitação inicial do thread e são interrompidos por outras solicitações, o status do trabalho da solicitação inicial é rastreado até que o trabalho seja concluído. Neste contexto, um usuário também pode ser outro programa.

- **Assim temos que colocar mais de uma camada de processamento para trabalhar por nós**



Exemplo:

```
In [15]: from time import sleep

def tarefa1():
    x=0
    while x<10:
        print("tarefa 1")
        x+=1
        sleep(0.5)

def tarefa2():
    y=0
    while y<10:
        print("tarefa 2")
        y+=1
        sleep(0.5)

tarefa1()
tarefa2()
```

Saída:

```
tarefa 1
tarefa 1
tarefa 1
tarefa 1
tarefa 1
tarefa 1
tarefa 1
tarefa 1
tarefa 1
tarefa 1
tarefa 2
tarefa 2
tarefa 2
tarefa 2
tarefa 2
tarefa 2
tarefa 2
tarefa 2
tarefa 2
tarefa 2
```

ANALYTICS

In [14]:
```
import threading
from time import sleep

def tarefa1():
    x=0
    while x<100:
        print("tarefa 1")
        x+=1
        sleep(0.5)

def tarefa2():
    y=0
    while y<100:
        print("tarefa 2")
        y+=1
        sleep(0.5)

threading.Thread(target=tarefa1).start()
tarefa2()
```

```
tarefa 1tarefa 2

tarefa 2tarefa 1

tarefa 1tarefa 2

tarefa 1tarefa 2

tarefa 1tarefa 2

tarefa 2tarefa 1

tarefa 2tarefa 1

tarefa 2tarefa 1

tarefa 1tarefa 2

tarefa 2tarefa 1
```

HDI SEGUROS

ANALYTICS

FIM

- Na próxima Monitoria falaremos de PySpark

Muito Obrigado!

HDI SEGUROS