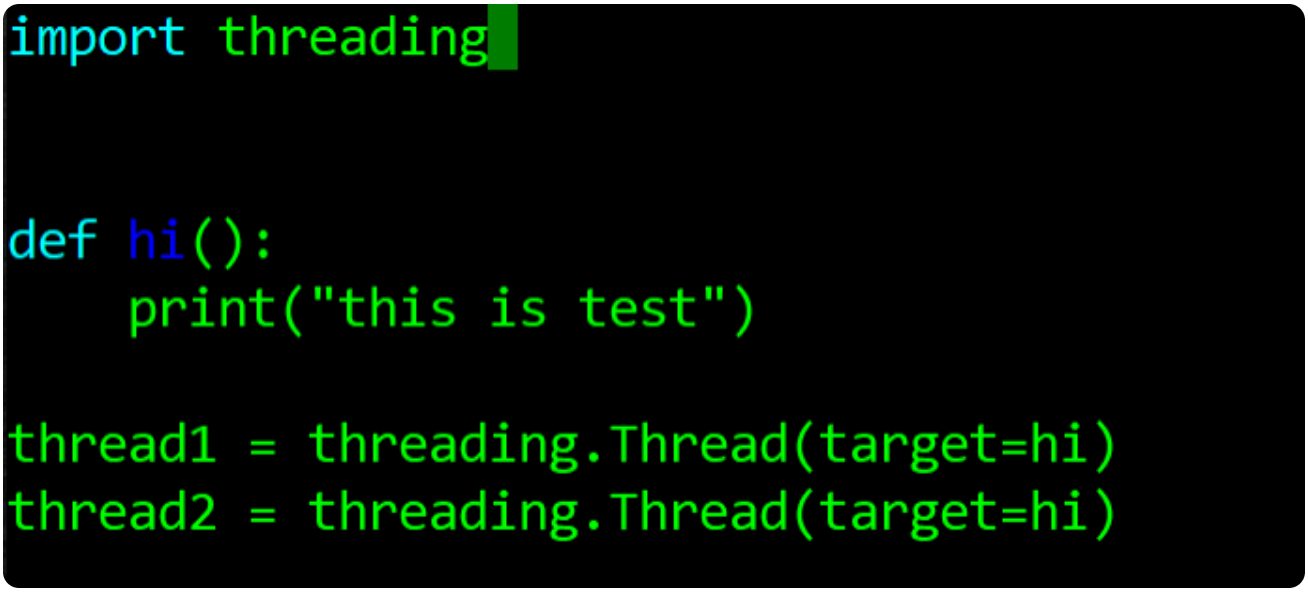
Python Threading

Threads

- ال thread ده عباره اكتر من عملية ممكن اني البروسيسور يقوم بيهم ف وقت واحد
- يبقي اي عمليتين او اكتر بيشتغلو مع بعض ف نفس الوقت دا بنسميه Threading
- عشان نقدر نستخدم ال threads دي لازم نستدعي مكتبتها فالبايثون
- تعالي افهمك



- بص ياباشا بعد ما استدعيت المكتبه بتاعت ال threading عملت فانكشن عادي جدا بتطبع جمله
- `threading.Thread(target=hi)`
  start لوحدها بس خلي بالك كدا مش هتشتغل برضو لازم اعملها thread ف hi الامر ده بقولو حطلي الفانكشن
  `threading.Thread(target=hi).start()`
  كدا انا بقولو اعملي الفانكشن يا باشا
- بس طبعا فالصوره اللي فوق احنا حطيناه ف variable ودا عادي يعني مفيهوش مشاكل
  `thread1.start()`
- تعالي بقي نكريت اكتر من thread ونشوف كدا

```
import threading

def hi():

    print("this is Threading")
```

```
thread1 = threading.Thread(target=hi)
thread2 = threading.Thread(target=hi)
thread3 = threading.Thread(target=hi)

thread1.start()
thread2.start()
thread3.start()
```

- هنا انا عملت threading حد هييجي يقولي ايوا ايه المشكله يعني منتا عملت لنفس الفانكشن thread تلت مرات ايه الابداع
- اقولك ياغبي مهو هنا الفانكشن دي هتتنفذ تلت مرات ف نفس الوقت
- ماهو الطبيعي اصلا اني لما البروسيسور يجي يشتغل كل عملية بتاخد PID خاص بيها وبتستني دورها لحد ما اللي قبليها تخلص وتخش هنا بقي انا عملتهم كلهم ف وقت واحد فهمت؟

```
this is Threading
this is Threading
this is Threading
```

- حد هيقولي ياعم انا مش مقصدك اقولك تعالي كدا اوريك حاجه

```
import threading
import time

def hi():

    print(time.ctime())

thread1 = threading.Thread(target=hi)
thread2 = threading.Thread(target=hi)
thread3 = threading.Thread(target=hi)

thread1.start()
thread2.start()
thread3.start()
```

بص كدا الكود ده انا قولتلو اطبعلي الوقت اللي اتعملت فيه الفانكشن وطبعا هنا هيطبع الوقت بتاع كل فانكشن تعالي اوريك هيطلع ايه

```
Sun Feb 16 01:36:41 2025
Sun Feb 16 01:36:41 2025
Sun Feb 16 01:36:41 2025
```

- شوف طلع ايه؟ التلت فانكشنز اتنفذو ف نفس الوقت بالظبط عشان انا حاططهم ف thread

- حد هيجي يقولي برضو مش مصدق اقولك تعالي بص الكود ده والناتج بتاعه

```
import threading
import os

def hi():

    print(os.getpid)

thread1 = threading.Thread(target=hi)
thread2 = threading.Thread(target=hi)
thread3 = threading.Thread(target=hi)

thread1.start()
thread2.start()
thread3.start()
```

- هنا انا بقولو اطبعلي الرقم بتاع كل عملية هتتعمل

```
8520
8520
8520
```

- الاه بص جابلك ايه؟ دا التلت عمليات بنفس ال PID يعني معني كدا انها عملية واحده وبتتنفذ مره واحده اقولك اه

```
thread1.join()
thread2.join()
thread3.join()
```

- في فانكشن اسمها join دي بتوقف كل حاجه لحد ما الthread يخلص شغله
- الـ `join()` بتوقف تنفيذ الـ main thread لحد ما الـ thread المستهدف يخلص شغله. يعني بتضمن إن الthread يكمل قبل ما الكود يكمل

For Loop with Threads

```
import threading

def hi():

    for i in range(1000):

        print(i)

thread1 = threading.Thread(target=hi)
thread2 = threading.Thread(target=hi)
thread3 = threading.Thread(target=hi)

thread1.start()
thread2.start()
thread3.start()

thread1.join()
thread2.join()
thread3.join()
```

- انا عملت فانكشن بتعمل for loop بتعد من 0 لحد 1000 وهنا طبعا عملت 3 threads يعني العملية دي هتتعمل ف نفس الوقت تلت مرات
- فالعادي لو بتعمل فانكشن عاديه وعملتلها call اول فانكشن هتخلص وبعد كدا تبدأ تاني وتالت فانكشن ورا بعض
- انما دي هتلاقي بيعد من 0 ل 1000 مكرره تلت مرات
- طبعا مش هعرف اجبلك الناتج عشان طويل بس حاول تستوعبها

Functions with Arguments

- احنا اتعلمنا ازاي نعمل فانكشن ونستدعيها جوا ال thread بس لو تلاحظ احنا بنحط الفانكشن من غير القوسين مش زي لما بعمل call عادي للفانكشن

`()hi`

`thread1 = threading.Thread(target=hi)`

- بص فال thread محطناش القوسين
- طب لو انا عامل فانكشن ب params اباصيلها ال argument ازاي؟ هقولك تعالي بص فالكود ده كدا

```
import threading
import time

def printed(name):

    print(f"Hello mr {name}")

yourname = "Mohammed Tantawy"

thread1 = threading.Thread(target=printed, args=(yourname,))
thread1.start()
```

عشان احط ال argument بحط

`args=(yourname, )`

- خلي بالك لازم الarguement يتحط بين قوسين ولازم احط ال comma حتي لو مش هحط argument تاني