**به نام خدای رنگین کمان**

*خب اول از همه باید بدونیم که این مفاهیم برای کسانی هستش که حداقل یکسالی با پایتون کار کردند، و میخان دانش خودشون رو افزایش بدن.*

**: Multithreading in python**

**به صورت خلاصه یعنی ما چندین ترد را همزمان انجام بدیم و لازم نیست از بالا شروع کنی به کامپایل کردن تا پایین.(میتونی بگی اول بالا بعد آخری بعد وسطی و غیره)**

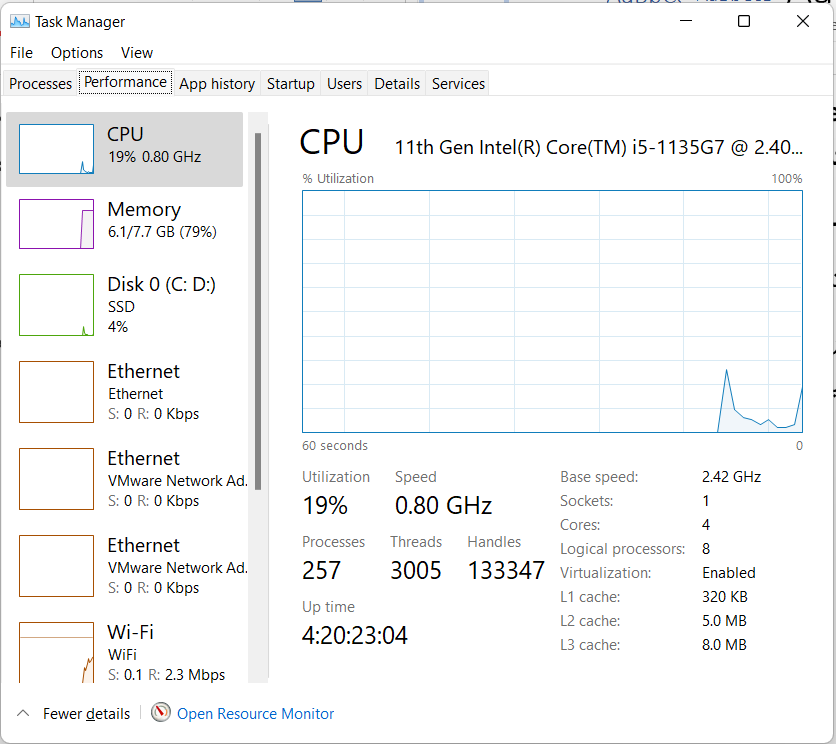
**پروسس (Process) : به هر برنامه که شما روی سیستم عاملتون ران میکنید یک پروسس میگویند، مثلا فایرفاکس یا همین ورد که من دارم توش مینویسم.**

**ترد (Tread) : به کوچک ترین قسمت هر پروسس یک ترد میگویند، مثلا توی فایرفاکس تو میتونی دانلود کنی که این خودش یه ترد هست، میتونی تب های مختلف باز کنی ، میتونی فیلم ببینی و.....**

**هر پراسس حداقل یدونه ترد داره که خوده اون برنامه هستش.**

مثلا ماشین یک پروسس هستش و فرمون و دنده و آینه و .... هر کدوم یک ترد هستند.

نکته : برای دیدن پروسس ها و ترد های ران شده روی سیستمت ، برو توی قسمت تسک منیجر و بعدش برو توی سی پی یو اونجا نوشته :



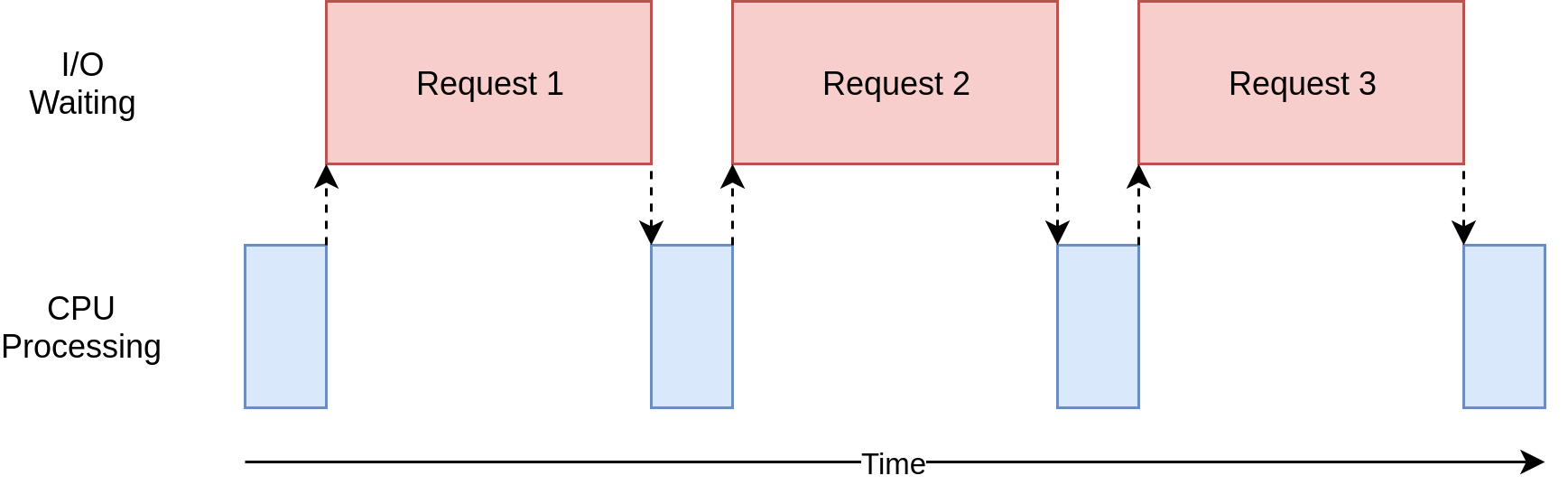
در پایتون نمیتونیم به اون صورت واقعی مالتی تردینگ رو پیاده سازی کنیم و پایتون توانایی اجرای دو ترد را در یک زمان را ندارد، بلکه مشخص میکنیم کدوم ترد اول شروع بشه ، بعدش صبر کنه تا یکی دیگه ران بشه و......

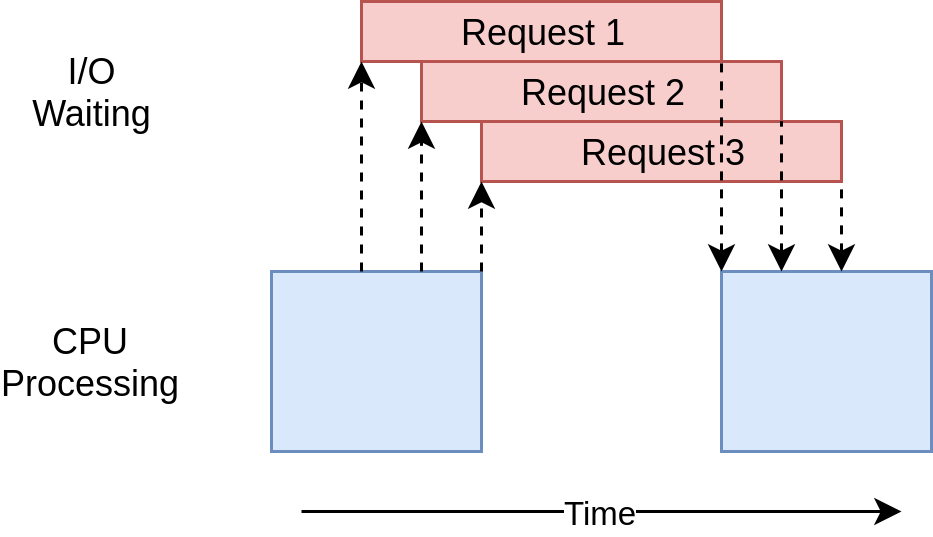
 برای multi-threading در پایتون از ماژول threading  استفاده میشود و ما در پایتون میتوانیم به صورت I/O bound ترد هارا مدیریت کنیم، در صورتی که برنامه شما به شکل I/O bound باشد زمانی را که صرف منتظر ماندن برای رسیدن پاسخ میکند را میتواند به کار دیگری مشغول شود.

اگر برنامه شما CPU bound باشد و همواره CPU شما مشغول کار باشد استفاده کردن از multi-threading نه تنها باعث افزایش سرعت نخواهد شد بلکه باعث افزایش فشار بروی CPU میشود و در نتیجه با کندی سرعت مواجه خواهید شد.

: I/O boundتوجه داشته باشید که فقط وقتی کاره I/O boundداشتید نیاز به multi-treading دارید.

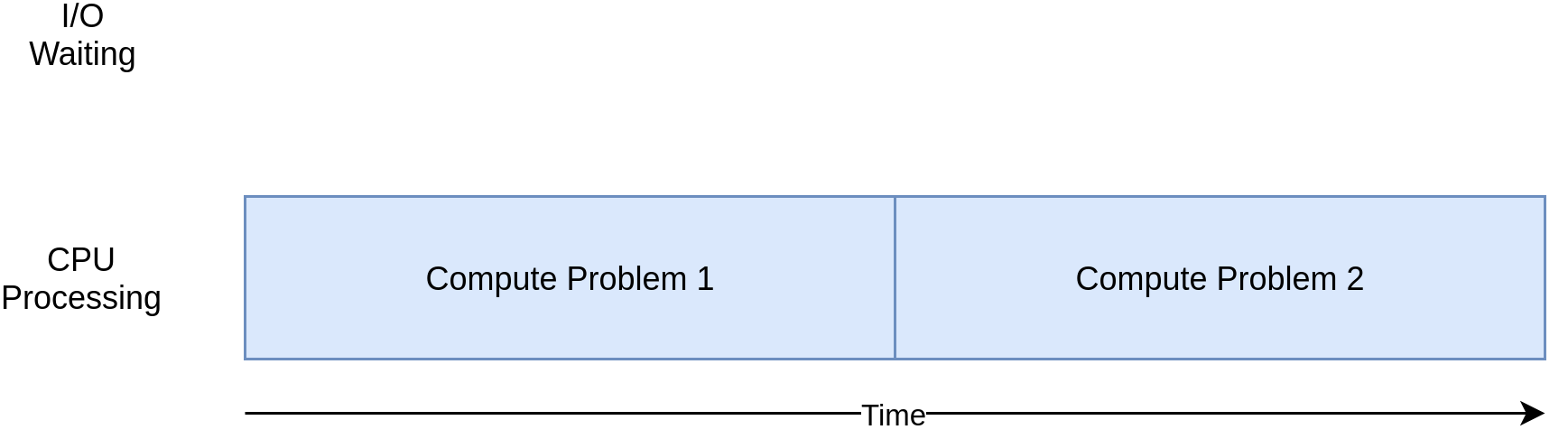
مثلا وقتی که یه درخواستی رو ارسال کردید و منتظر پاسخ اون هستش سی پی یو : (برای مثال کار کردن روی سیستم یکی دیگه با استفاده از سیستم خودت)





ما اینجا زمان هایی که سی پی یو منتظر یک عملی بوده رو استفاده کردیم و یک درخواست دیگه را ارسال کردیم.

: CPU boundدر اینجا کلا سی پی یو ما در حال استفاده و کار کردن هست و منتظر نمیماند که اینجا اگه multi-treadingبزنی بدتر کند میشی، پس در این حالت ما ازmulti-processing استفاده میکنیم که در این آموزش ما نیست.



**پیاده سازی با استفاده از صورت میگیره که دو تا روش داره :**

1. **با استفاده از ارسال یک شئ قابل فراخوانی به تابع سازنده (--init--).**

*passing a callable object to the constructor.*



بعد از استفاده از مالتی تریدینگ :



که در این حالت 3 ثانیه صرفه جویی شد.

1. **بازنویسی متودrun() توی ساب کلاسی که میسازی.**

*. overriding the run () method in a subclass*

**

بدون برنامه ما باید 8 ثانیه طول میکشید ولی الان به 5 ثانیه کاهش یافته است.