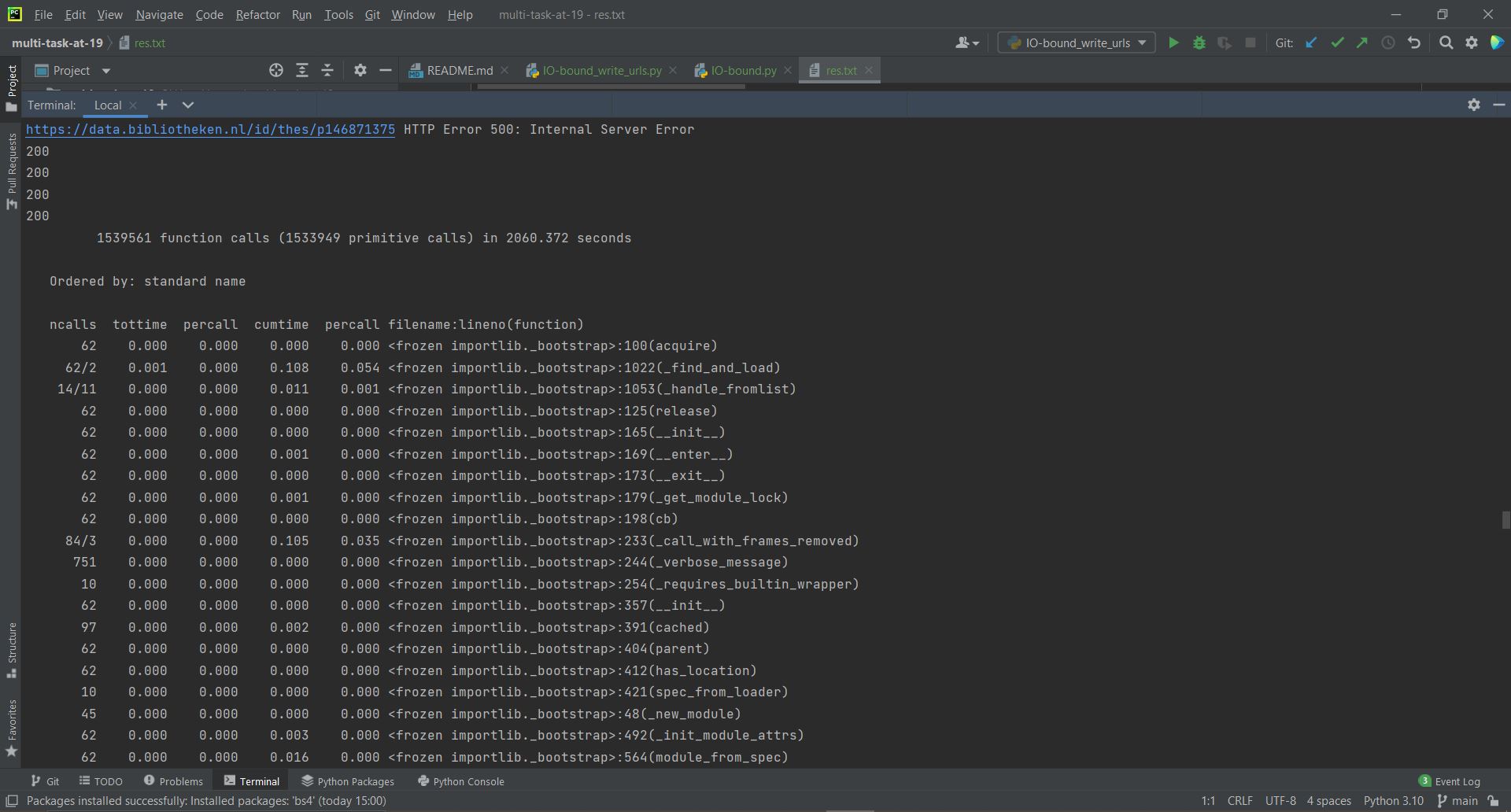
Отчет по работе «Параллелизм и асинхронность»

**IO-BOUND**

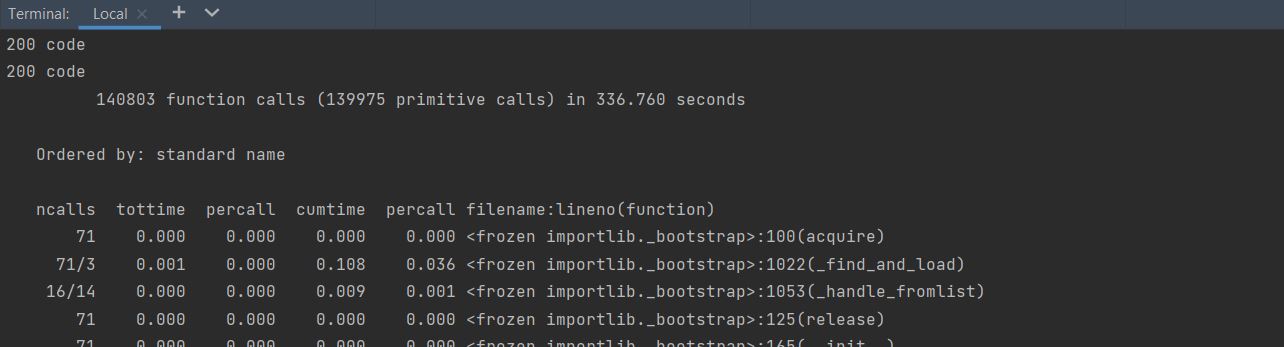
Время работы IO\_bound.py при синхронной проверке ссылок – 2060,4 сек, что приблизительно 34 минуты (очень долго)



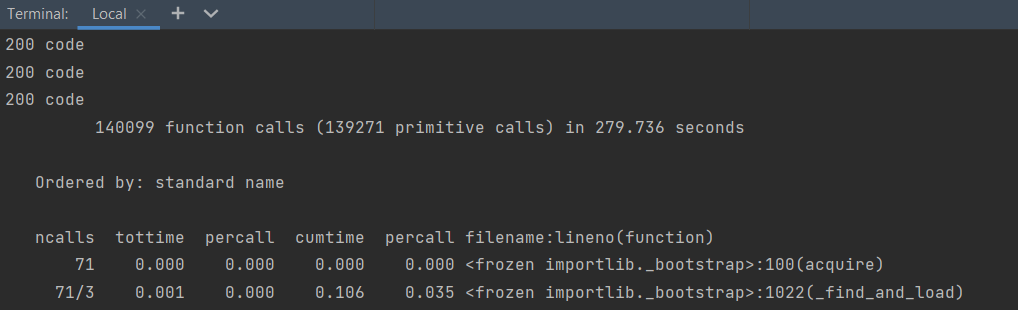
Перепишем код, используя ThreadPoolExecutor.

Время работы, когда количество воркеров(потоков):

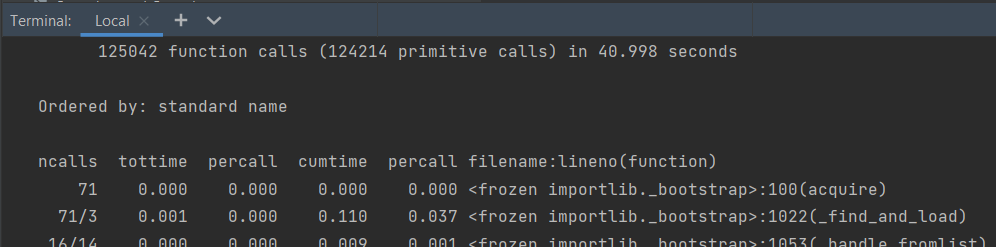
5 – 336,76 секунд



10 – 279,7 секунд



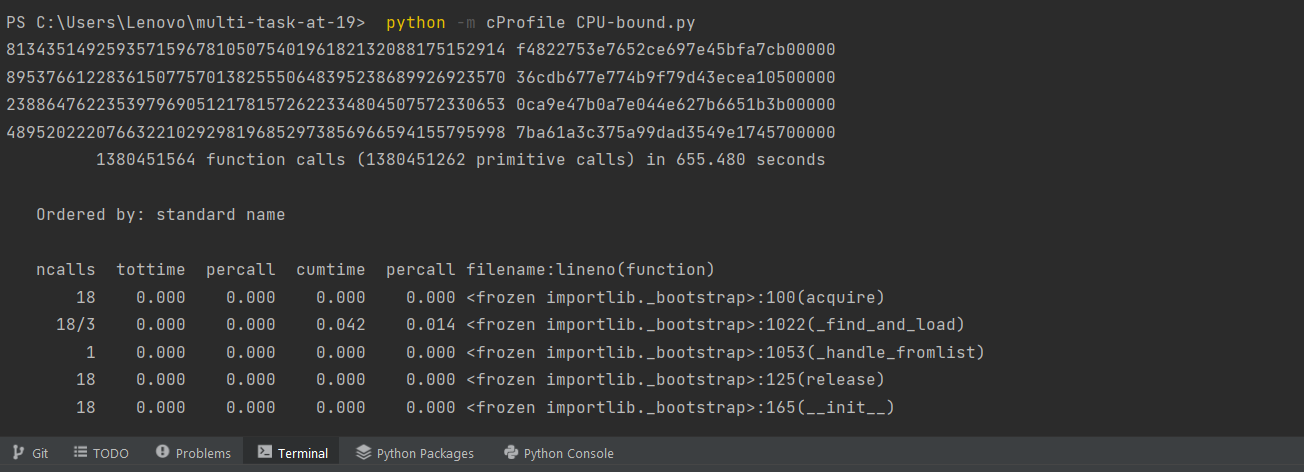
100 – 40,9 секунд



Таким образом, увеличение количества потоков не влечет за собой увеличение используемой памяти и загрузки процессора, и при этом дает выигрыш в скорости (времени) выполнения кода.

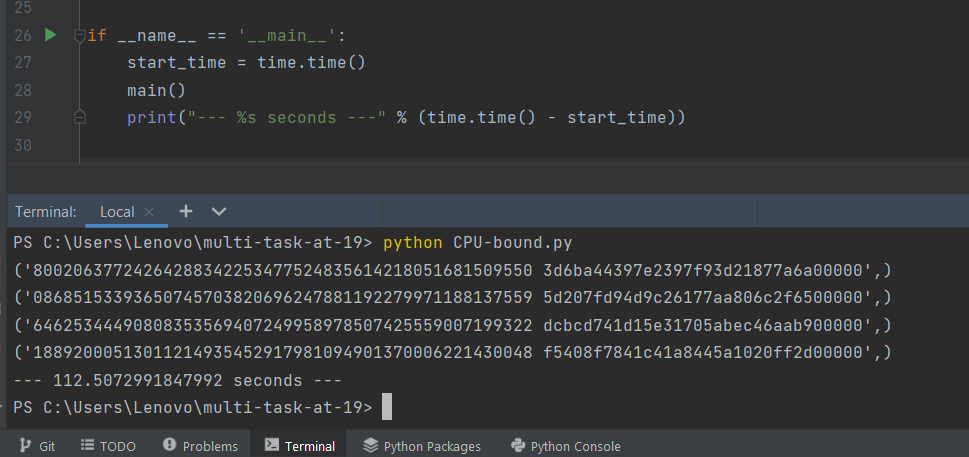
**CPU-BOUND**

Время генерации четырех монет – 655 секунд



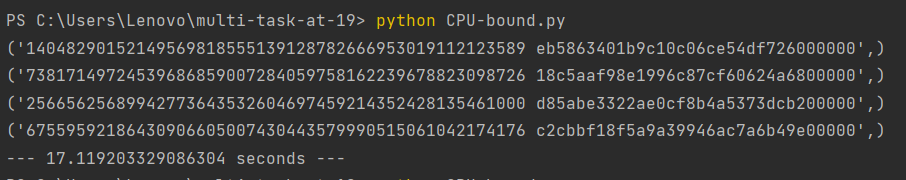
**Ускоримся за счет использования ProcessPoolExecutor.**

При max\_workers=2 время работы – 112,5 секунд.



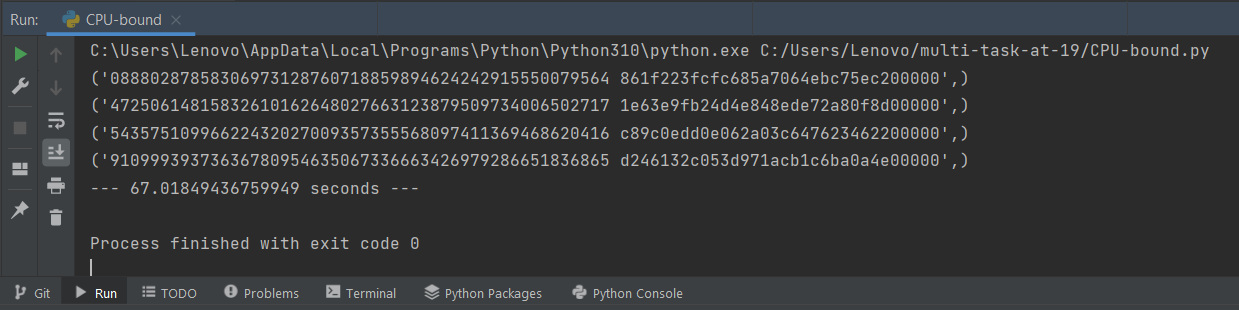
Увеличим max\_workers до 4

Время работы – всего 17 секунд

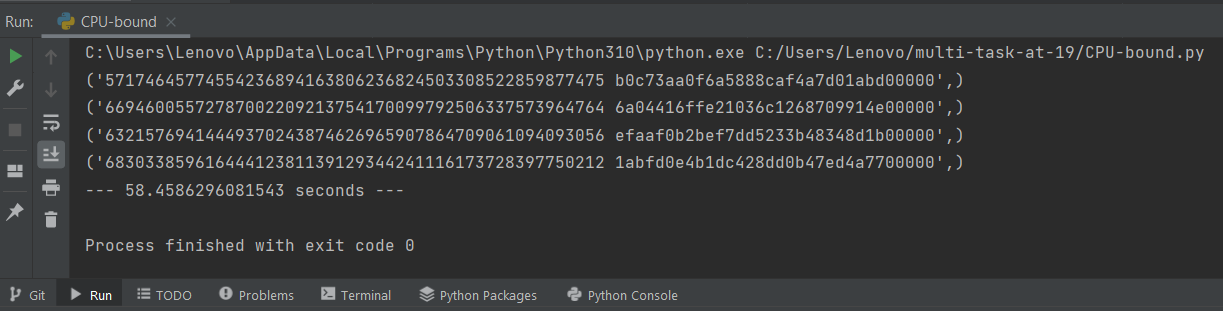


Но при еще большем увеличении воркеров, например до 5, 10, 100, большого прироста производительности мы не увидим, так как физических ядер всего 4.

Время работы при max\_workers = 5 – 67 секунд

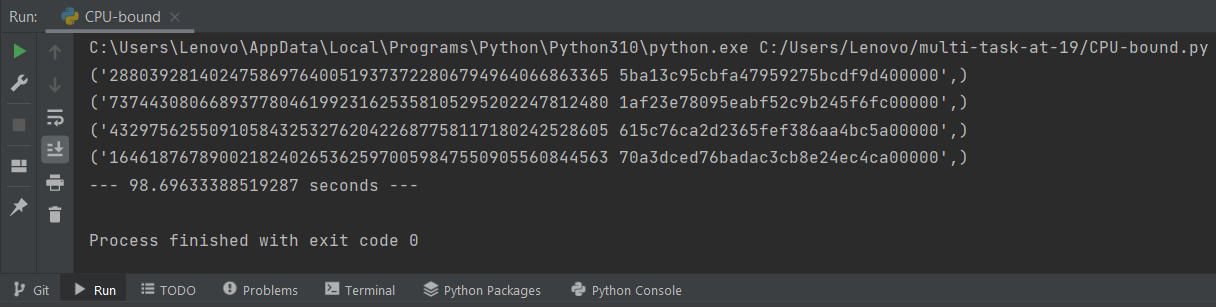


max\_workers = 10 – 58 секунд



max\_workers = 100 нельзя установить, так как максимальное значение воркеров 61

время при 61 воркере – 98 секунд



Увеличение количества процессов влечет за собой увеличение используемой памяти и загрузки процессора, но при этом дает выигрыш в скорости (времени) выполнения кода, если количество процессов не превышает числа физических ядер.