Materials: <https://github.com/krishnaik06/Complete-Data-Science-With-Machine-Learning-And-NLP-2024>

**Python**

Tuples- immutable objects (cant modify the entries)

List – can modify the objects

**Lambda function:**

add=lambda a, b :a+b ------------🡪 lambda args: expr

add(5,6)

**Map Function:**

Applies a function to a list of numbers: map(function, numbers)

list(map(lambda x: x\*\*2, numbers))

list(map(square, numbers))

numbers1=[1, 2, 3]

numbers2=[4, 5, 6]

added\_numbers=list(map(lambda x,y:x+y, numbers1, numbers2))

list(map(str.upper, words))

**Filter Function:**

Used to filter elements from a list based on a condition : filter(function, list)

filter(even, numbers)

**os functions:**

os.getcwd() – path

os.mkdir(‘dir name’)

os.listdir(‘.’) – list files in current location

os.path.join(dir\_name, file\_name)

os.path.join(os.getcwd(),dir\_name, file\_name)

os.path.exists(path), os.path.isfile(path),os.path.isdir(path)

**CONDA Commands**

Anaconda- package management

**Jupyter notebook**

**pip install ipykernel**

python -m ipykernel install --user --name your\_existing\_env

**MULTITHREADING AND MULTIPROCESSING**

1. Program: sequence of instruction written in some programming language. Example: google chrome-> .exe ->it’s a program
2. Process: simply an instance of a program that is being executed. (google chrome, when I click this a process(a browser) opens up).

Code segment:

Data segment:

Heap memory: The heap is a large area of memory available for use by the program.

Stack : where the variables of the program are stored

Register: kind of temp memory

Every process has a separate memory space. Since every process has a separate memory space, one process cannot corrupt other process.

Every process has different IO requirements, so we will take some time to switch from one process to another.

1. Threads: a thread is a unit of execution within a process.

Single threaded program:

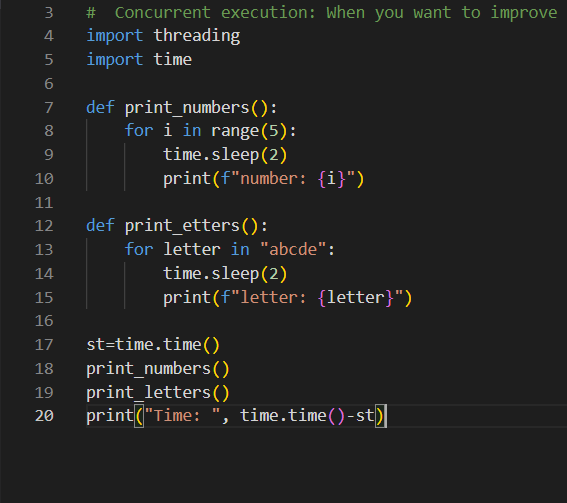
Multithreaded program: all the threads have common code segment, data segment and heap. Every thread has its own stack and register.

* When to use multithreading?

I/O-bound tasks: Tasks that spend more time waiting for I/O operations (e.g., file operations, network requests).

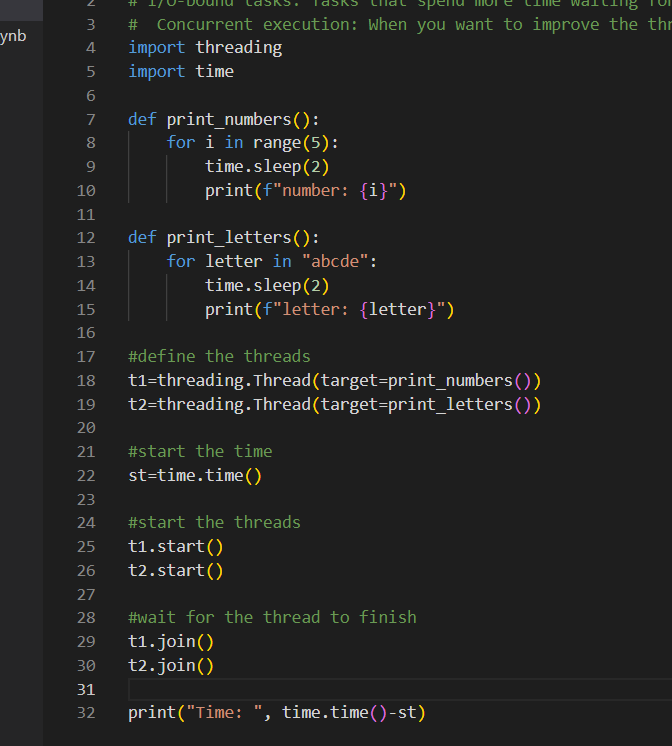
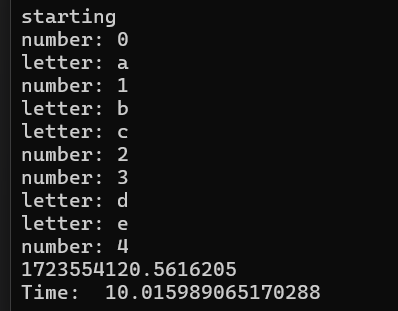
 Concurrent execution: When you want to improve the throughput of your application by performing multiple operations concurrently.

Usually whenever we run a program only a single thread is used.



Takes 20 sec to execute

When we introduce threading 2 threads to call out 2 different functions.

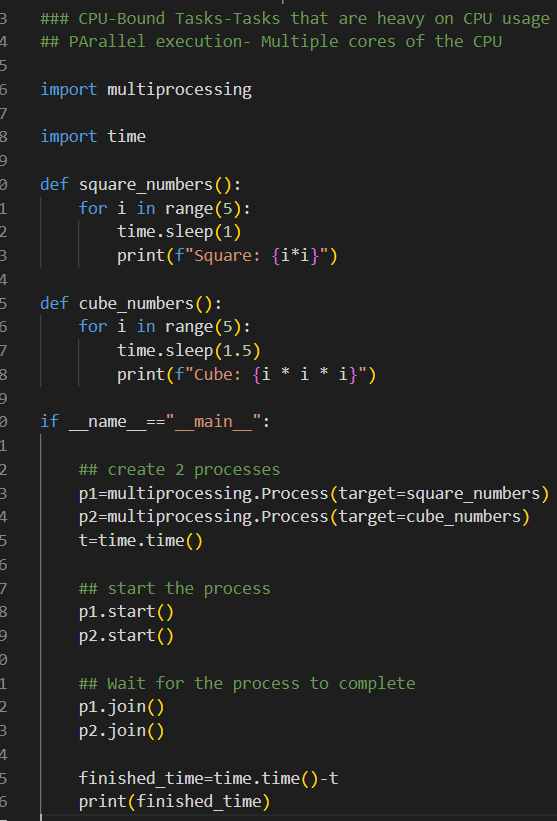
 

It takes 10 sec to execute

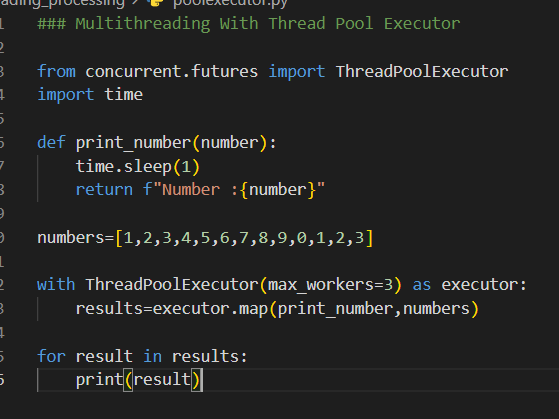
**Multiprocessing – when we require cpu power**

Processes that run in parallel. Useful for cpu bound tasks: mathematical computations, data operations. Parallel execution: use multiple cores of a cpu. **Processes have a separate memory while threads do not share separate memory.**

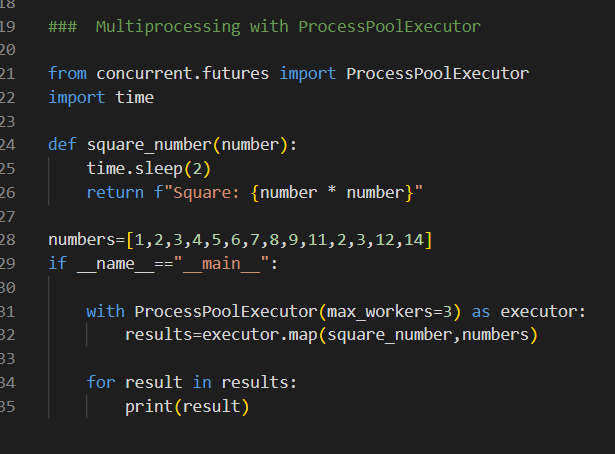
I/O-bound tasks are operations in computing where the performance is primarily limited by the speed of input/output operations rather than the speed of the CPU. These tasks involve waiting for data to be read from or written to an external source, such as a hard drive, network, or database, rather than performing computations. For I/O bound tasks threading otherwise processing.

****

**ThREAD Pool Executor**

****

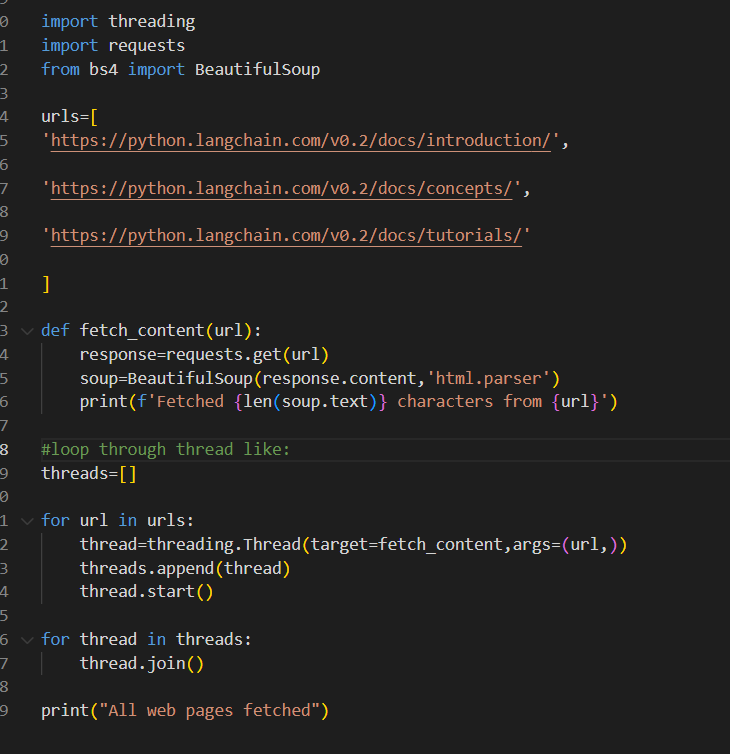
**Process Pool Executor**

****

**Example:**

**Webscrapping with threading**

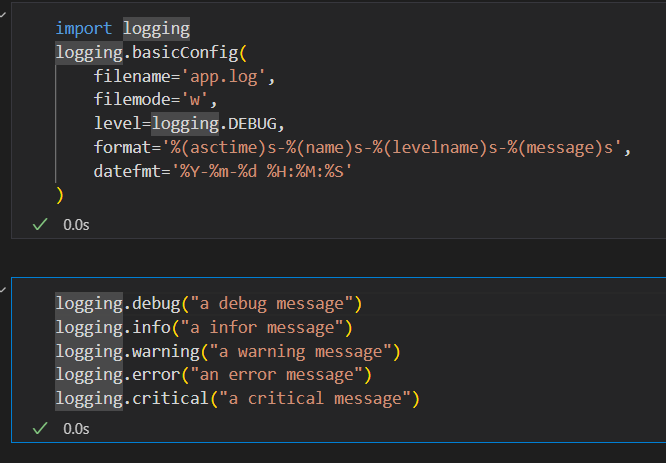
Web scraping often involves making numerous network requests to fetch web pages. These tasks are I/O-bound because they spend a lot of time waiting for responses from servers. Multithreading can significantly improve the performance by allowing multiple web pages to be fetched concurrently.



Start all the threads and join all of them at the end.

**Calculating Factorial of large numbers**

**LOGGING**



If the filename where logging is defined is called logger.py then we can use the logging object anywhere as from logger import logging

Multiple loggers:

