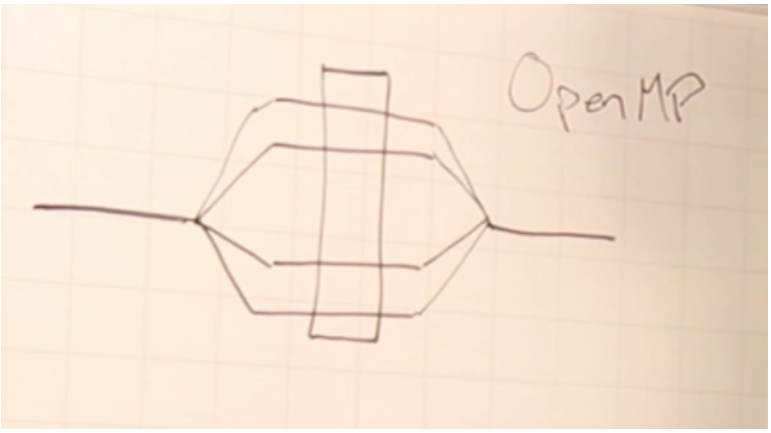
Processes and threads

To understand better parallel computing we'll be discussing the two main types of execution units: processes and threads in more detail.

Let's first look at threads. If you have a parallel program that starts as a single process here we have a single process executing a parallel program. If it uses threads at some point the program starts something called a parallel region that starts multiple threads that will start executing the program.

At this point you will get multiple threads executing the same program. These threads will then execute the instructions in the program in parallel. These threads are short-lived and the main feature is that they share the memory of the parent process, so they are sharing the memory of each other.

Therefore this kind of parallelization is often called shared memory parallelization.At some point in the program execution you might want to close the parallel region and at this point all the threads are closed or joined together and afterwards you will have a serial execution of the program again. These parallel regions you can have multiple of them in the same program so you could open a new parallel region after this and so forth. The main thing here is that you have multiple threads and they are all sharing the same memory space.



In other languages than Python, for example C or Fortran this would be commonly used with something called OpenMP to create the threads and execute them in parallel Unfortunately in Python this kind of the threaded approach is quite limited so let's next look at in more detail something called processes and how to use those in parallel.

So, now instead of having a single process we will have multiple processes in parallel. So we'll have one, two, three, and four processes they are independent but they are executing the same program code. These independent processes they have their own memory space and they are completely independent from each other.

Therefore, to exchange information between these processes. you need to explicitly send and receive messages between the processes. This we will discuss in more detail using something called MPI.