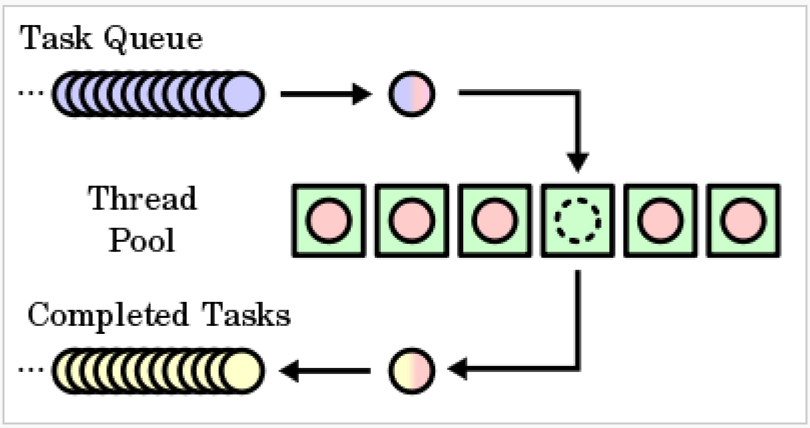
What is Task?

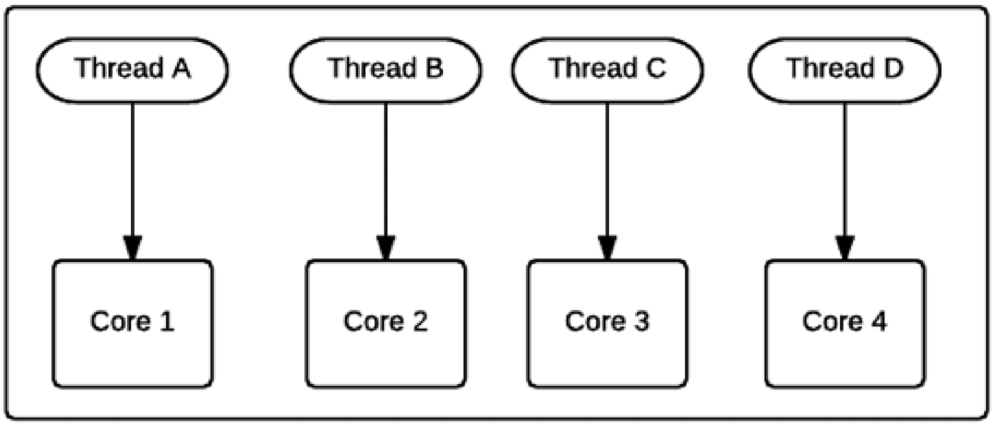
- .NET framework provides Threading.Tasks class to let you create tasks and run them asynchronously.
- A task is an object that represents some work that should be done.
- The task can tell you if the work is completed and if the operation returns a result, the task gives you the result.



A sample thread pool (green boxes) with waiting tasks (blue) and completed tasks (yellow)

What is Thread?

- .NET Framework has thread-associated classes in System.Threading namespace.
- A Thread is a small set of executable instructions.



Why we need Task

- It can be used whenever you want to execute something in parallel. Asynchronous implementation is easy in a task, using' async' and 'await' keywords.

Why we need Thread

- When the time comes when the application is required to perform few tasks at the same time.

How to create Task

```
static void Main(string[] args) {
    Task < string > obTask = Task.Run(() => (
        return“ Hello”));
    Console.WriteLine(obTask.result);
}
```

How to create Thread

```
static void Main(string[] args) {
    Thread thread = new Thread(new ThreadStart(getMyName)
);
    thread.Start();
}
public void getMyName() {}
```

Differences Between Task And Thread

1. The Thread class is used for creating and manipulating a thread in Windows. A Task represents some asynchronous operation and is part of the Task Parallel Library, a set of APIs for running tasks asynchronously and in parallel.
2. The task can return a result. There is no direct mechanism to return the result from a thread.
3. Task supports cancellation through the use of cancellation tokens. But Thread doesn't.
4. A task can have multiple processes happening at the same time. Threads can only have one task running at a time.
5. We can easily implement Asynchronous using 'async' and 'await' keywords.
6. A new Thread()is not dealing with Thread pool thread, whereas Task does use thread pool thread.
7. A Task is a higher level concept than Thread.