Concurrency :

It means different parts of a program execute independently

Threads

We can use threads to run codes simultaneously.

Creating threads

- The thread::spawn function is used to create a new thread. The spawn function takes a closure as parameter. The closure defines code that should be executed by the thread.
- The new thread will be stopped when the main thread ends.
- The thread::sleep function forces a thread to stop its execution for a short duration, allowing a different thread to run.
- the main thread is printed first, even though the print statement from the spawned thread appears first in the code

```
use std::thread;
use std::time::Duration;

fn main() {
   thread::spawn(|| {
      for i in 1..10 {
         println!("hi number {} from the spawned thread!", i);
         thread::sleep(Duration::from_millis(1));
      }
   });
   for i in 1..5 {
      println!("hi number {} from the main thread!", i);
      thread::sleep(Duration::from_millis(1));
   }
}
```

Join handles

Use this for run thread completely

```
use std::thread;
use std::time::Duration;

fn main() {
   let handle = thread::spawn(|| {
      for i in 1..10 {
         println!("hi number {} from the spawned thread!", i);
         thread::sleep(Duration::from_millis(1));
      }
   });
   for i in 1..5 {
      println!("hi number {} from the main thread!", i);
      thread::sleep(Duration::from_millis(1));
```

```
    }
    handle.join().unwrap();
}
```