Threading module

# Multithreading

## Introduction

Here we are going to introduce the basic concept of multithreading in python. A single threaded program is a program that works on one thing at a time, a multithreaded one is the one that works on multiple stuff simultaneously, basically executing multiple lines of code at the same time.

## Example

In the example below we are going to show a simple implementation of threading.

import time, threading  
  
print**('Program started')**msg = **'Wake up Neo...'**def say\_this**(**msg**)**:  
 time.sleep**(**2**)** print**(**msg**)**thread\_func = threading.Thread**(**target=say\_this, args=**[**msg**])**thread\_func.start**()**print**('Program ended')**

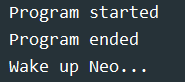
In the example above our output without multithreading would be:

Program started

Wake up Neo…

Program ended

But with threading enabled our say\_this function is executed the last since since it has a sleep() function in it. After the sleep() ends then our function gets executed. And it gets printed after the last print() function.



Notice that target=say\_this not say\_this() this is because our function is waiting to be executed by the Thread. Also if we have \*args or \*\*kwargs we want to pass in we must write them out separately in a list form like showed above.

## Concurrency issues

You can easily create several new threads and have them all running at the same time. But multiple threads can also cause problems called concurrency issues. These issues happen when threads read and write the same variables at the same time, causing threads to trip over each other. Concurrency issues are hart to reproduce consistently, making them hard to debug.

Multithreaded programming is its own wide subject and beyond this small tutorial.

What we have to keep in mind is to avoid concurrency issues, Never let multiple threads read or write the same variable. When you create a new thread make sure its target only uses variables from the local scope. This will avoid hard-to-debug concurrency issues.