Components of an Asynchronous Code

In this lesson, we will learn about the components of the asynchronous code.

WE'LL COVER THE FOLLOWING

Write an Asynchronous Code

To write asynchronous code in python, import the library using `import asyncio`.

Components

Asyncio has 3 main components:

1. coroutines
2. event loop
3. future

Coroutine

A coroutine is the result of an asynchronous function which can be declared using the keyword `async` before def.

```
async def my_function(argument):
  pass
```

When we declare a function using the async keyword the function is not executed; instead, a coroutine object is returned.

To call a coroutine, write

```
my_coroutine = my_function(argument)
```

There are two ways to read the output of an async function from a coroutine. The first way is to use the `await` keyword, which is only possible inside async functions and will wait for the coroutine to terminate and return the result.

```
result = await my_function(argument)
```

The second way is to add it to an event loop.

Event Loop

The event loop is the object which executes our asynchronous code and decides how to switch between async functions. After creating an event loop we can add multiple coroutines to it; these coroutines will all be running concurrently when `run_until_complete` or `run_forever` is called.

```
loop = asyncio.new_event_loop()  # create loop
future = loop.create_task(my_coroutine) # add coroutine to the loop
loop.run_until_complete(future) # add coroutine to the loop concurrently
loop.close() # close the loop
```

Future

A future is an object that works as a placeholder for the output of an asynchronous function, and it gives us information about the function state. A future is created when we add a coroutine to an event loop. There are two ways to this:

```
future = loop.create_task(my_coroutine)
```

The method adds a coroutine to the loop and returns a task which is a subtype of the future.

Now that you have learned the components of asynchronous programming, let’s move on to execute tasks using asynchronous programming let's move on to execute tasks using asynchronous programming.