Tasks

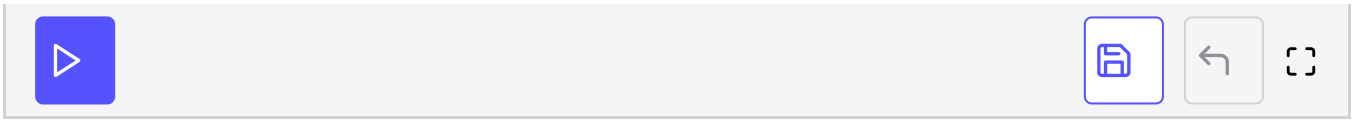
Tasks are a subclass of a Future and a wrapper around a coroutine. They give you the ability to keep track of when they finish processing. Because they are a type of Future, other coroutines can wait for a task and you can also grab the result of a task when it's done processing. Let's take a look at a simple example:

```
import asyncio


async def my_task(seconds):
    """
    A task to do for a number of seconds
    """
    print('This task is taking {} seconds to complete'.format(
        seconds))
    asyncio.sleep(seconds)
    return 'task finished'


if __name__ == '__main__':
    my_event_loop = asyncio.get_event_loop()
    try:
        print('task creation started')
        task_obj = my_event_loop.create_task(my_task(seconds=2))
        my_event_loop.run_until_complete(task_obj)
    finally:
        my_event_loop.close()

    print("The task's result was: {}".format(task_obj.result()))
```



Here we create an asynchronous function that accepts the number of seconds it will take for the function to run. This simulates a long running process. Then we create our event loop and then create a task object by calling the event loop object's **create_task** function. The **create_task** function accepts the function that we want to turn into a task. Then we tell the event loop to run until the task completes. At the very end, we get the result of the task since it

has finished.

Tasks can also be canceled very easily by using their **cancel** method. Just call it when you want to end a task. Should a task get canceled when it is waiting for another operation, the task will raise a **CancelledError**.

Wrapping Up

At this point, you should know enough to start working with the asyncio library on your own. The asyncio library is very powerful and allows you to do a lot of really cool and interesting tasks. You should check out http://asyncio.org/ which is a curated listing of various projects that are using asyncio. It is a wonderful place to get ideas for how to use this library. The Python documentation is also a great place to start from.