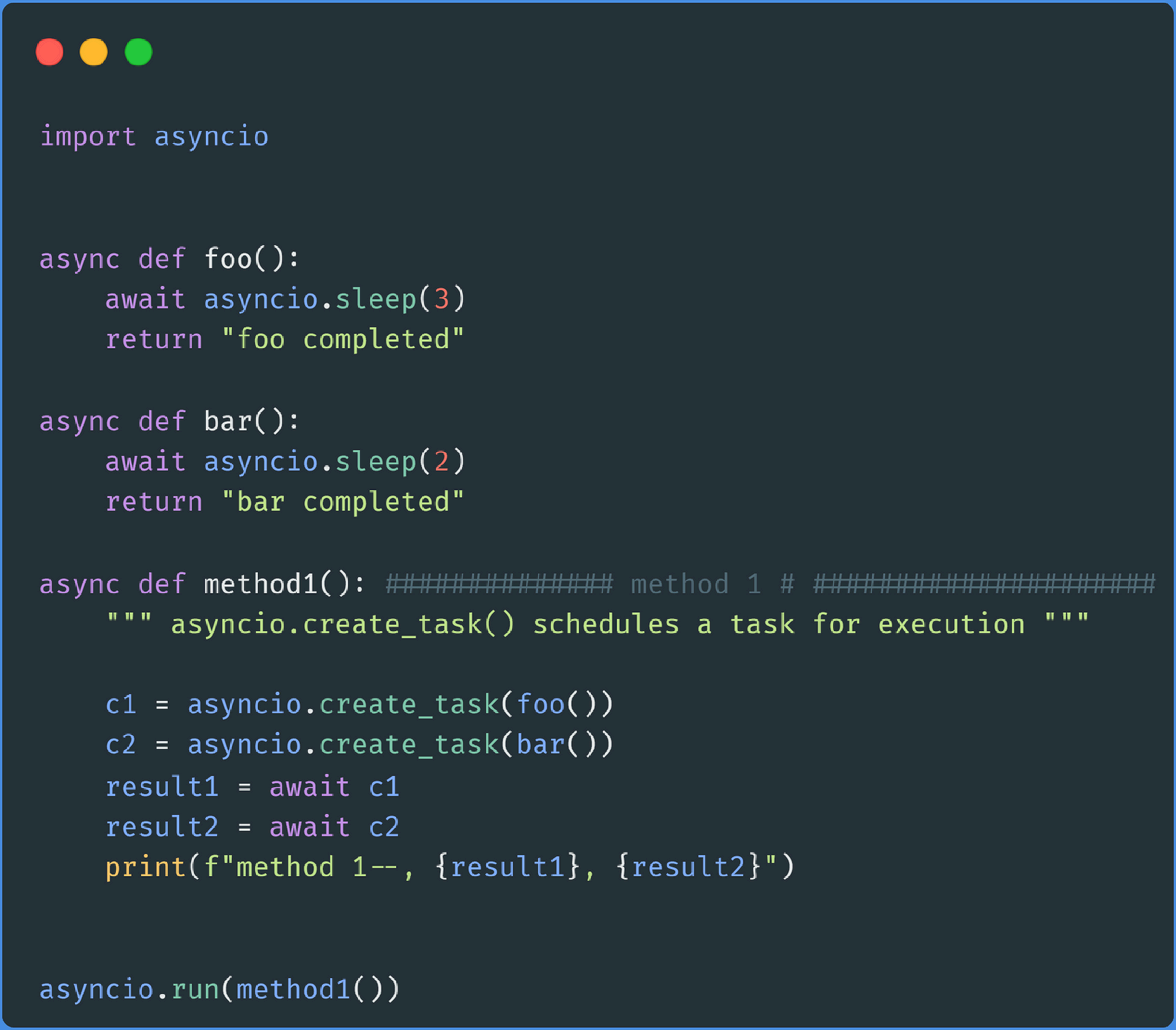
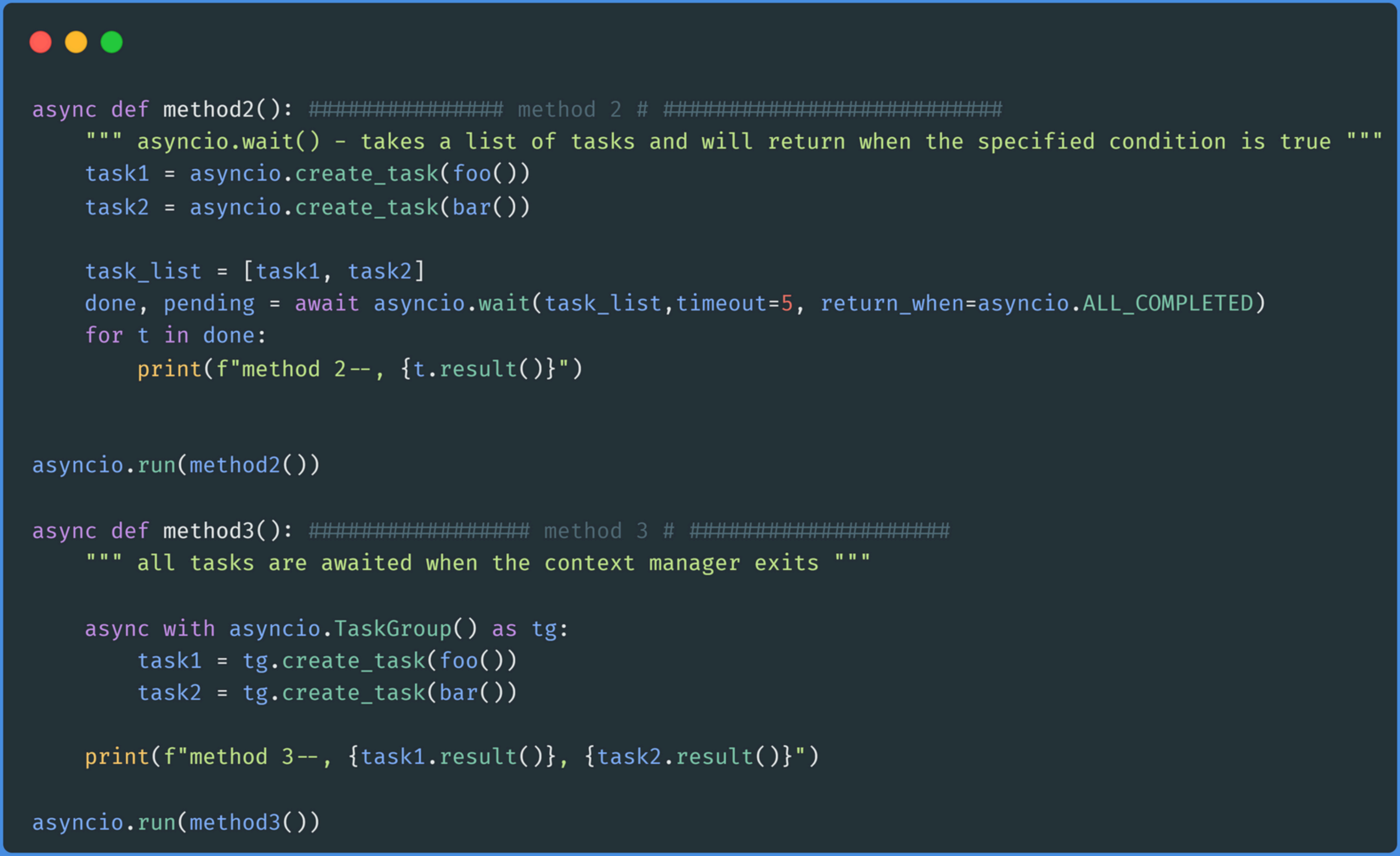
PYTHON - ASYNC TASKS

multiple ways to run async tasks



```python
import asyncio


async def foo():
    await asyncio.sleep(3)
    return "foo completed"

async def bar():
    await asyncio.sleep(2)
    return "bar completed"

async def method1(): ############## method 1 # #####################
    """ asyncio.create_task() schedules a task for execution """

    c1 = asyncio.create_task(foo())
    c2 = asyncio.create_task(bar())
    result1 = await c1
    result2 = await c2
    print(f"method 1--, {result1}, {result2}")


asyncio.run(method1())
```



```python
async def method2(): ############### method 2 # ##########################
    """ asyncio.wait() - takes a list of tasks and will return when the specified condition is true """
    task1 = asyncio.create_task(foo())
    task2 = asyncio.create_task(bar())

    task_list = [task1, task2]
    done, pending = await asyncio.wait(task_list,timeout=5, return_when=asyncio.ALL_COMPLETED)
    for t in done:
        print(f"method 2--, {t.result()}")


asyncio.run(method2())

async def method3(): ################# method 3 # ####################
    """ all tasks are awaited when the context manager exits """

    async with asyncio.TaskGroup() as tg:
        task1 = tg.create_task(foo())
        task2 = tg.create_task(bar())

    print(f"method 3--, {task1.result()}, {task2.result()}")

asyncio.run(method3())
```