زمان‌بندی و مدیریت Taskها در `asyncio`

در برنامه‌نویسی غیرهمزمان با استفاده از `asyncio`، مدیریت زمان‌بندی و اجرای هم‌زمان `Task`ها بخش مهمی از عملکرد است. در اینجا به توضیح مفاهیم مرتبط با زمان‌بندی و مدیریت `Task`ها در `asyncio` می‌پردازیم.

1. استفاده از `asyncio.sleep()` برای تاخیر در توابع غیرهمزمان

`asyncio.sleep()` برای اضافه کردن تاخیر یا معطل کردن اجرای یک تابع غیرهمزمان به مدت زمان مشخص استفاده می‌شود. این متد مشابه `time.sleep()` است، اما با این تفاوت که در زمان خوابیدن، دیگر کدها اجرا می‌شوند و برنامه بلوکه نمی‌شود.

مثال:

```
import asyncio

async def task(name, delay):
    print(f"Task {name} starting.")
    await asyncio.sleep(delay)
    print(f"Task {name} finished after {delay} seconds.")

async def main():
    await asyncio.gather(task("A", 2), task("B", 1))

asyncio.run(main())
```

در اینجا:

- `await asyncio.sleep(delay)` برای ایجاد تاخیر به مدت `delay` ثانیه در هر `task` استفاده می‌شود.
- با استفاده از `asyncio.gather()`، هر دو `task` به صورت همزمان اجرا می‌شوند و هیچ‌کدام از آن‌ها برنامه را بلوکه نمی‌کند.

2. اجرای Taskهای غیرهمزمان با استفاده از `asyncio.create_task()`

`asyncio.create_task()` به شما امکان می‌دهد تا یک Task جدید غیرهمزمان بسازید و آن را به Event Loop اضافه کنید. این روش امکان می‌دهد که Taskهای جدید را به صورت غیرهمزمان و هم‌زمان اجرا کنید.

مثال:

```
import asyncio

async def task(name, delay):
    print(f"Task {name} starting.")
    await asyncio.sleep(delay)
    print(f"Task {name} finished after {delay} seconds.")

async def main():
    task1 = asyncio.create_task(task("A", 2))
    task2 = asyncio.create_task(task("B", 1))

    # صبر کردن برای اتمام هر دو Task
    await task1
    await task2
```

```
asyncio.run(main())
```

در اینجا:

- `asyncio.create_task()` هر `task` را به صورت غیرهمزمان ایجاد می‌کند و آن‌ها را به Event Loop اضافه می‌کند.
- با استفاده از `await task1` و `await task2`، منتظر می‌مانیم تا هر دو Task تکمیل شوند.

3. مدیریت زمان‌بندی با استفاده از `asyncio.wait()` و `asyncio.as_completed()`

در صورتی که بخواهید چندین Task را مدیریت کنید و ترتیب تکمیل آن‌ها را پیگیری کنید، می‌توانید از متدهای `asyncio.wait()` و `asyncio.as_completed()` استفاده کنید.

- **`asyncio.wait()`** : این متد به شما این امکان را می‌دهد که منتظر اتمام تعدادی از Taskها باشید و بعداً آن‌ها را بررسی کنید.
- **`asyncio.as_completed()`** : این متد یک ژنراتور برمی‌گرداند که به ترتیب، نتایج Taskها را که تکمیل شده‌اند، می‌دهد.

مثال با استفاده از `asyncio.wait()` :

```
import asyncio

async def task(name, delay):
    print(f"Task {name} starting.")
    await asyncio.sleep(delay)
    print(f"Task {name} finished after {delay} seconds.")

async def main():
    task1 = asyncio.create_task(task("A", 2))
    task2 = asyncio.create_task(task("B", 1))

    # استفاده از asyncio.wait() برای مدیریت Taskها
    done, pending = await asyncio.wait([task1, task2])
    for task in done:
        print(f"Task {task.get_name()} completed.")

asyncio.run(main())
```

در اینجا:

- `asyncio.wait()` منتظر می‌ماند تا Taskهای `task1` و `task2` اتمام یابند و سپس وضعیت تکمیل آن‌ها را بررسی می‌کند.

مثال با استفاده از `asyncio.as_completed()` :

```
import asyncio

async def task(name, delay):
    print(f"Task {name} starting.")
    await asyncio.sleep(delay)
    print(f"Task {name} finished after {delay} seconds.")
    return f"Result of {name}"
```

```
async def main():
    task1 = asyncio.create_task(task("A", 2))
    task2 = asyncio.create_task(task("B", 1))

    # استفاده از asyncio.as_completed() برای دریافت نتایج به ترتیب
    for completed_task in asyncio.as_completed([task1, task2]):
        result = await completed_task
        print(f"Completed task result: {result}")

asyncio.run(main())
```

در اینجا:

- `asyncio.as_completed()` نتایج Taskها را به ترتیب اتمام آنها باز می‌گرداند. این روش به شما این امکان را می‌دهد که نتایج را به محض تکمیل هر Task دریافت کنید.

نتیجه‌گیری

- **`asyncio.sleep()`** به شما اجازه می‌دهد که تاخیر در توابع غیرهمزمان ایجاد کنید بدون اینکه برنامه را بلوکه کند.
- **`asyncio.create_task()`** برای ایجاد Taskهای غیرهمزمان و اجرای آن‌ها به صورت همزمان استفاده می‌شود.
- **`asyncio.wait()`** و **`asyncio.as_completed()`** به شما این امکان را می‌دهند که چندین Task را مدیریت کنید و نتایج آن‌ها را پیگیری کنید.

این قابلیت‌ها به شما کمک می‌کنند تا کنترل بیشتری روی زمان‌بندی و مدیریت همزمانی در برنامه‌های غیرهمزمان با استفاده از `asyncio` داشته باشید.