3.13.5

Quick search Go

`asyncio` — Asynchronous I/O

Hello World!

```
import asyncio

async def main():
    print('Hello ...')
    await asyncio.sleep(1)
    print('... World!')

asyncio.run(main())
```

asyncio is a library to write **concurrent** code using the **async/await** syntax.

asyncio is used as a foundation for multiple Python asynchronous frameworks that provide high-performance network and web-servers, database connection libraries, distributed task queues, etc.

asyncio is often a perfect fit for IO-bound and high-level **structured** network code.

asyncio provides a set of **high-level** APIs to:

- run Python coroutines concurrently and have full control over their execution;
- perform network IO and IPC;
- control subprocesses;
- distribute tasks via queues;
- synchronize concurrent code;

Additionally, there are **low-level** APIs for *library and framework developers* to:

- create and manage event loops, which provide asynchronous APIs for networking, running subprocesses, handling OS signals, etc;
- implement efficient protocols using transports;
- bridge callback-based libraries and code with async/await syntax.

Availability: not WASI.

This module does not work or is not available on WebAssembly. See WebAssembly platforms for more information.

asyncio REPL

You can experiment with an `asyncio` concurrent context in the REPL:

```
$ python -m asyncio
asyncio REPL ...
Use "await" directly instead of "asyncio.run()".
Type "help", "copyright", "credits" or "license" for more information.
>>> import asyncio
>>> await asyncio.sleep(10, result='hello')
'hello'
```

Raises an auditing event `cpython.run_stdin` with no arguments.

Changed in version 3.12.5: (also 3.11.10, 3.10.15, 3.9.20, and 3.8.20) Emits audit events.

Changed in version 3.13: Uses PyREPL if possible, in which case `PYTHONSTARTUP` is also executed. Emits audit events.

Reference

High-level APIs

- Runners
- Coroutines and Tasks
- Streams
- Synchronization Primitives
- Subprocesses
- Queues
- Exceptions

Low-level APIs

- Event Loop
- Futures
- Transports and Protocols
- Policies
- Platform Support
- Extending

Guides and Tutorials

- High-level API Index
- Low-level API Index
- Developing with asyncio

Note: The source code for asyncio can be found in Lib/asyncio/.