Синтаксические конструкции asyncio

Понимание библиотеки **asyncio** мы начнем с практических примеров. Разберемся, какие методы и объекты лучше использовать в конкретных случаях, а позже изучим вопрос устройства и работы более глубоко.

Синтаксический сахар: Async & Await

Это ключевые слова**,** которые мы используем при разработке асинхронных приложений на Python.

async — ключевое слово, через которое мы объявляем асинхронную функцию, короутину:

```
async def some_function():
    #function's logic
    pass
```

await — ключевое слово, через которое мы запускаем асинхронную функцию и передаем управление другим короутинам:

```
await some_function()
```

В обычных функциях асинхронные вызовы использовать нельзя, потому что это приведет к исключению **SyntaxError:**

```
import asyncio



def main():
    await asyncio.sleep(1)

main()
```

```
...
await asyncio.sleep(1)
    ^
SyntaxError: 'await' outside async function
```

Аналогично обстоит дело с вызовом через **await** обычной функции. Без ключевого слова **async** в объявлении получим исключение **ValueError:**

```
import asyncio
import time



def main():
    time.sleep(1)

asyncio.run(main())
```

```
    raise ValueError("a coroutine was expected, got {!r}".format(main))
ValueError: a coroutine was expected, got None
```