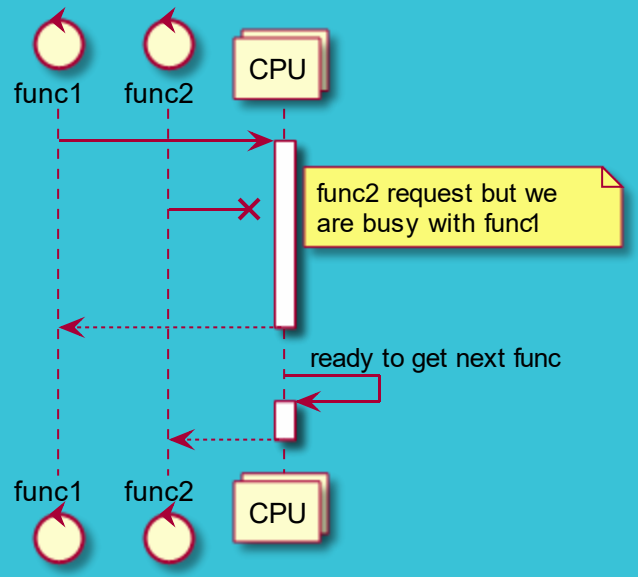
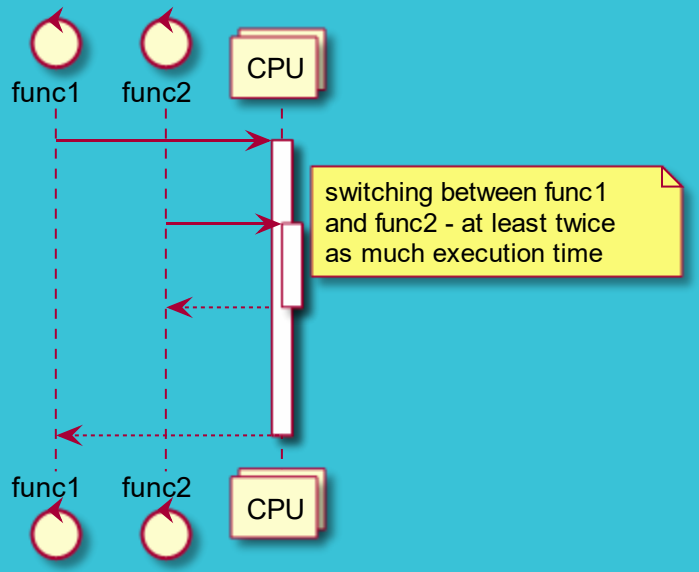
Не синхронный Python

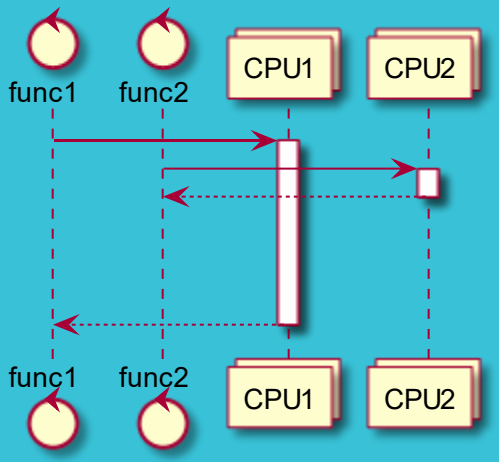
Зачем

- Много ввода-вывода
- Много вычислений
- Фоновая работа

Архитектура







Python GIL

- CPython
- io
- NumPy

Multiprocessing

```
import multiprocessing as mp
import time


def f(name, timeout, queue):
    time.sleep(timeout)
    print('hello', name)
    queue.put(name + ' done!')


queue = mp.SimpleQueue()  # queue for communicating with the processes we will spawn
bob = mp.Process(target=f, args=('bob', 0.3, queue))
bob.start()  # start the process
alice = mp.Process(target=f, args=('alice', 0.1, queue))
alice.start()  # start the process

# wait for processes to complete
bob.join()
alice.join()

# print results from intercommunication object
for result in iter(queue.get, None):
    print(result)
```

Threading

```
import threading
import time
import queue


def f(name, timeout, queue):
    time.sleep(timeout)
    print('hello', name)
    queue.put(name + ' done!')


q = queue.Queue()  # thread-safe queue
bob = threading.Thread(target=f, args=('bob', 0.3, q))
bob.start()  # start the thread

alice = threading.Thread(target=f, args=('alice', 0.1, q))
alice.start()  # start the thread

# wait for threads to complete
bob.join()
alice.join()

# print results from intercommunication object
for result in iter(q.get, None):
    print(result)
```

gevent

```
import gevent
from gevent import monkey; monkey.patch_all()
import time


def f(name, timeout):
    time.sleep(timeout)
    print('hello', name)
    return name + ' done!'


bob = gevent.spawn(f, 'bob', 0.3)
bob.start()  # start the greenlet

alice = gevent.spawn(f, 'alice', 0.1)
alice.start()  # start the greenlet

# wait for greenlets to complete
bob.join()
alice.join()

# print results
print(bob.value)
print(alice.value)
```

asyncio

```
import asyncio


async def f(name, timeout):
    await asyncio.sleep(timeout)
    print('hello', name)
    return name + ' done!'


async def main():
    bob = asyncio.create_task(f('bob', 0.3))  # start the coroutine
    alice = asyncio.create_task(f('alice', 0.1))  # start the coroutine

    # wait for coroutines to complete
    print(await bob)
    print(await alice)

asyncio.run(main())  # implicitly starts the loop
```

Callback hell

```
import asyncio


async def f(name, timeout, on_result):
    await asyncio.sleep(timeout)
    print('hello', name)
    on_result(name + ' done!')


def on_result(msg):
    print(msg)


async def main():
    bob = asyncio.create_task(f('bob', 0.3, on_result))  # start the coroutine
    alice = asyncio.create_task(f('alice', 0.1, on_result))  # start the coroutine

    # wait for coroutines to complete
    await bob
    await alice

asyncio.run(main())  # implicitly starts the loop
```