

Симуляция посадки пассажиров в автобус

Задание: Написать программу, которая симулирует процесс посадки пассажиров в автобус. У каждого пассажира есть своя скорость посадки, и автобус ожидает определенное время(5 сек), прежде чем уехать. Если пассажир не успевает сесть в автобус за отведенное время, он остается и ждет следующего.

Вам предоставлен список словарей (https://github.com/nefelsay/asyncio/blob/main/bus_passengers.py), где каждый словарь представляет пассажира. Каждый пассажир имеет следующие атрибуты:

- Имя
- Возраст
- Скорость (время, необходимое для посадки в автобус)
- Работа

```
passengers = [
    {"Name": "John", "Age": 25, "Speed": 3, "Job": "Engineer"},
    {"Name": "Sarah", "Age": 32, "Speed": 6, "Job": "Doctor"},
    {"Name": "Mike", "Age": 28, "Speed": 4, "Job": "Teacher"},
    {"Name": "Emma", "Age": 30, "Speed": 3, "Job": "Nurse"}, ]
```

Ваша задача - написать асинхронную функцию, которая симулирует процесс посадки каждого пассажира в автобус. Функция должна "ожидать" время, равное скорости посадки пассажира, прежде чем считать пассажира севшим в автобус.

Затем вы должны написать основную асинхронную функцию/точку входа(`main()`), которая создает и запускает задачи для каждого пассажира. Эта функция должна ожидать, пока все пассажиры не сядут в автобус, но только в течение определенного времени (5 секунд). Для установки таймера задачи используйте функцию `asyncio.wait_for(timeout=5)`.

Если какой-либо пассажир не успевает сесть в автобус за это время, его задача отменяется, и он остается ждать следующего автобуса.

В конце вашей программы должны быть выведены имена всех пассажиров, которые сели в автобус, а также имена и профессии тех, кто не успел.

Пример:

```
# Выводить сообщение если пассажир успел сесть в автобус
f'{name} сел в автобус.'

# Если не успел
f'{name} {Job} не успел/а вовремя сесть в автобус.'
```

Напомню:

Функция `asyncio.wait_for()` используется для ожидания завершения задачи или корутины в течение определенного времени. Если задача завершается в течение этого времени, `asyncio.wait_for()` возвращает результат задачи. Если задача не завершается в течение этого времени, `asyncio.wait_for()` автоматически отменяет задачу и вызывает исключение `asyncio.TimeoutError`.

Для решения этой задачи Вам могут пригодиться методы `task.result()` и `task.cancelled()`. О них мы говорили в прошлых уроках курса.

- `task.result()` - возвращает результат задачи, если она уже завершена. В данном случае, это имя пассажира, который сел в автобус. Если задача была отменена или еще не завершилась, `task.result()` вызовет исключение.
- Поэтому важно проверить, что задача не была отменена перед вызовом `task.result()`, это можно сделать с помощью условия `if not task.cancelled()`.

Подсказки:

1. Используйте `asyncio.wait_for()` для ожидания завершения всех задач в течение определенного времени. Вы можете использовать его вместе с `asyncio.gather()` следующим образом: `asyncio.wait_for(asyncio.gather(*tasks), timeout=5)`. Это ожидает, пока все задачи не будут выполнены, но не более 5 секунд на каждую задачу.

Вывод вашей программы должен быть следующим, всего 57 строк:

```
...
...
...
Jacob сел в автобус.
Lucas сел в автобус.
Mike сел в автобус.
Charlie сел в автобус.
Jacob сел в автобус.
Charlie сел в автобус.
Ava сел в автобус.
Mike сел в автобус.
Sarah Doctor не успел/а вовремя сесть в автобус.
Robert Lawyer не успел/а вовремя сесть в автобус.
Sophia Scientist не успел/а вовремя сесть в автобус.
Grace Designer не успел/а вовремя сесть в автобус.
William Writer не успел/а вовремя сесть в автобус.
Ethan Actor не успел/а вовремя сесть в автобус.
Isabella Singer не успел/а вовремя сесть в автобус.
James Lawyer не успел/а вовремя сесть в автобус.
...
...
...
```