**2023/12/11 00:00|Лекция. Как создать и запустить поток.**

## Как создать и запустить поток.

Допустим, начальник попросил сгенерировать **N-ное количество жанров музыки**, и мы очень удачно нашли для этого сервис, который по url запросу выводит 1 жанр. (Рис. 1)



Рис. 1

**НО!** На этот сервис не придумана возможность вывести сразу несколько жанров. Можно копировать название жанра, записывать и обновлять страницу, но мы делаем по-другому.

В PyCharm нам понадобится библиотека «**requests**», она позволяет обращаться к url. Запишем переменную **URL**. (Рис. 2)

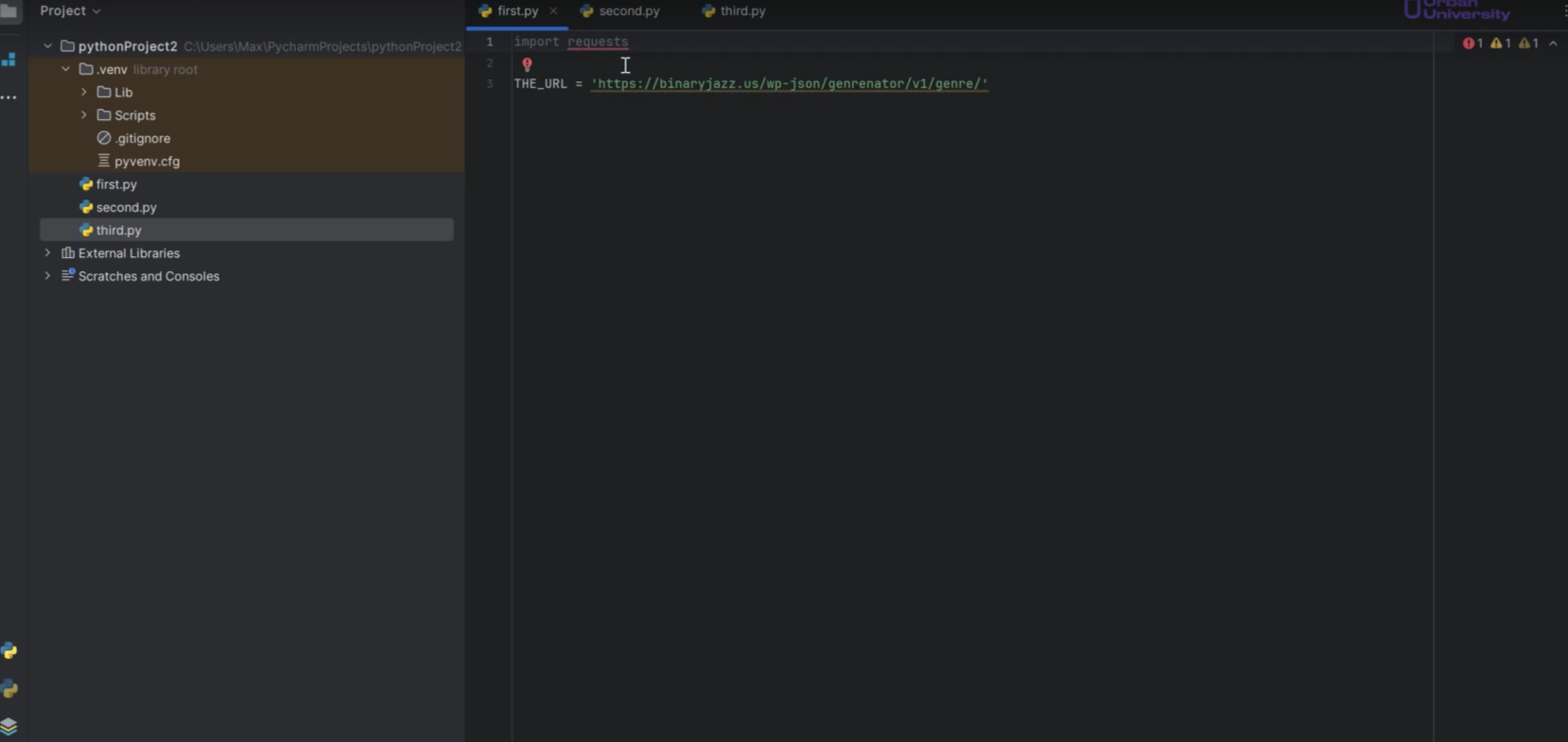


Рис. 2

«**requests**» инсталлируем. Так же нам понадобится переменная «**res**» (сокращенно result) – список жанров, которые мы заберем с этого сайта. (Рис. 3)

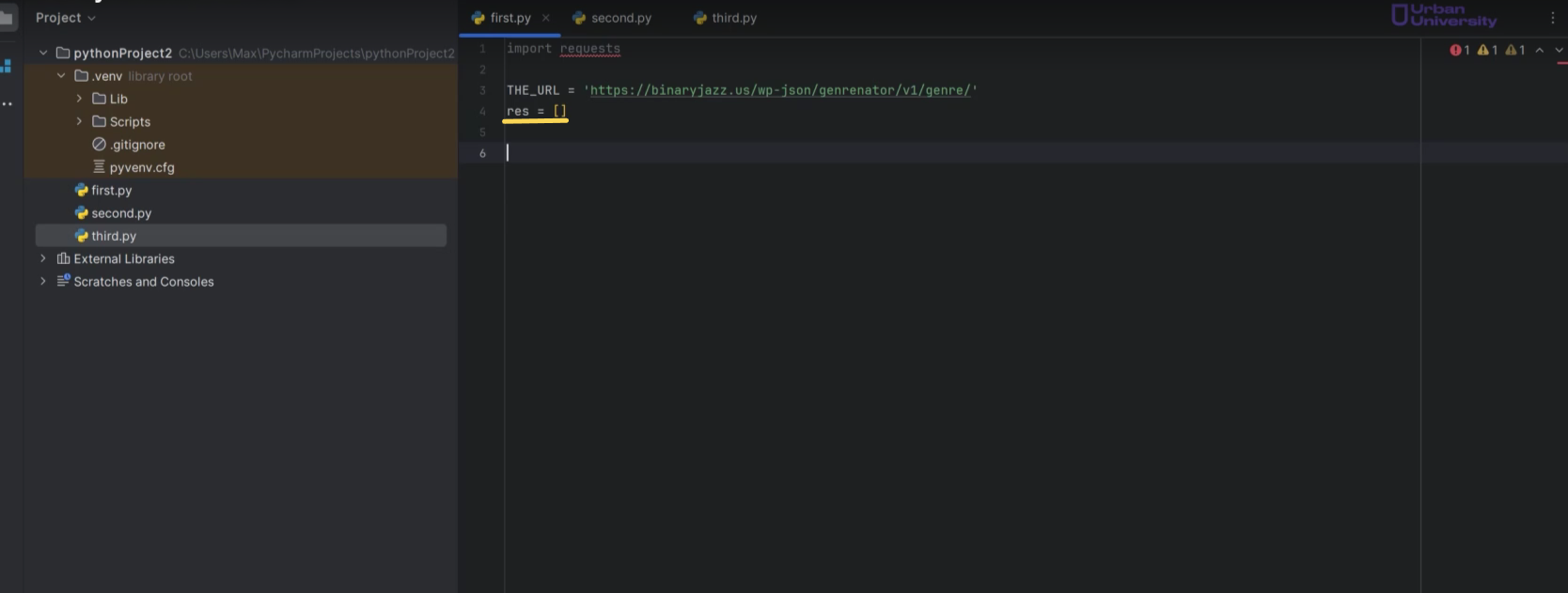


Рис. 3

Будем 20 раз внутри цикла делать запрос на этот сайт. «**requests.get**» позволит делать get запрос на страницу по url, который мы указали в скобках. (Рис. 4)

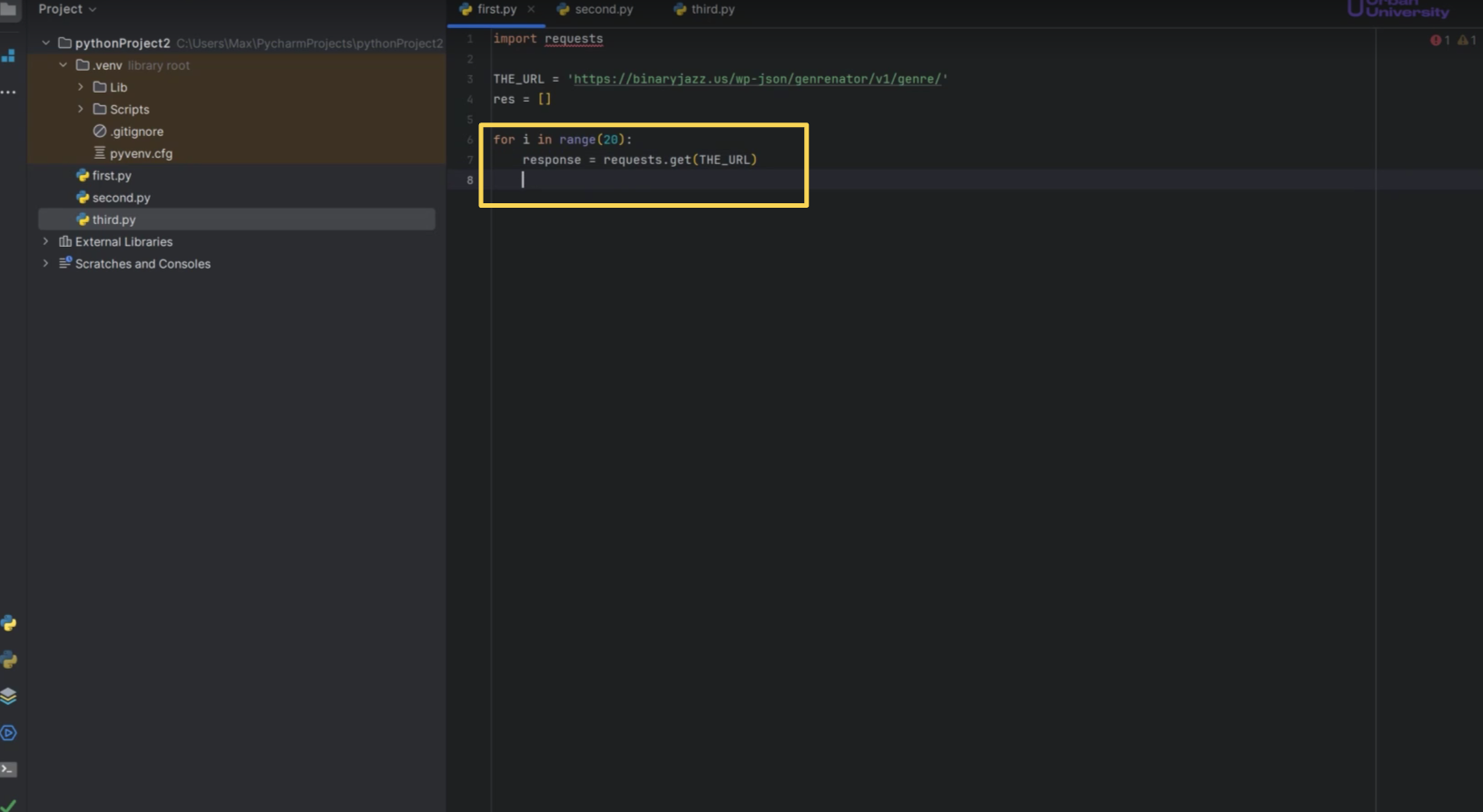


Рис. 4

Получим «**page\_response**» и в нее сложим «**response.json**». «**.json**» позволяет перевести из формата, в котором по интернету передаются файлы, в формат Phyton (в строчку). Этот результат добавляем в словарь «res». (Рис. 5)

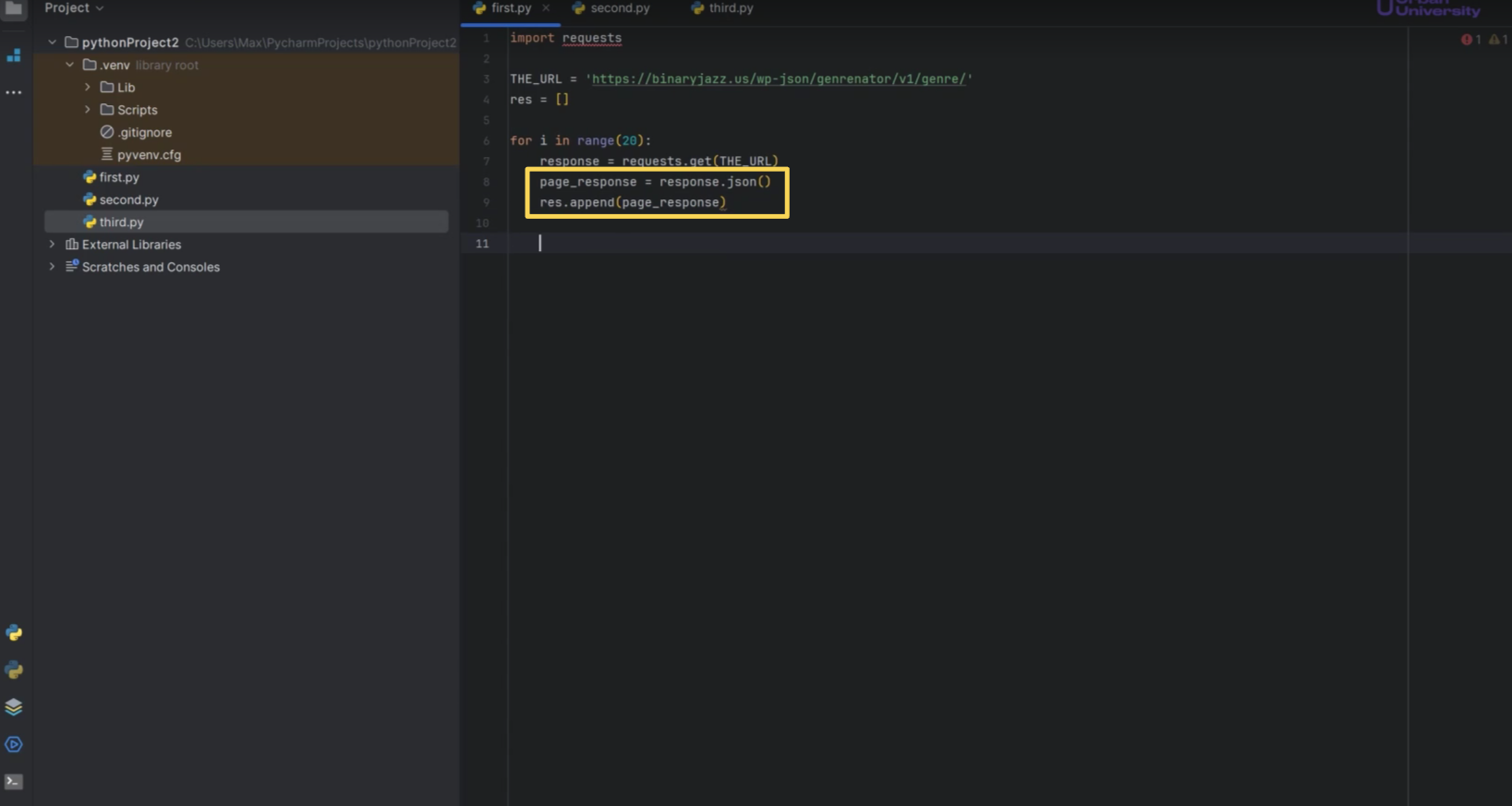


Рис. 5

После этого словарь можно напечатать. Изменим 20 жанров на 10 и запустим. (Рис. 6)

Суть в том, что к строчке «response» он обращается по адресу и есть задержка между тем как мы обратились к сайту и как получили данные. Запрос доходит на сайт, сайт обрабатывает запрос, он возвращается. Во время этих запросов наша программа ничего не делает. Phyton встает на строчке кода («response») и ждет когда вернется ответ, после этого с помощью метода «.json» вычленяем строчку жанра и добавляем его в «res». После этого начинается следующий шаг цикла.

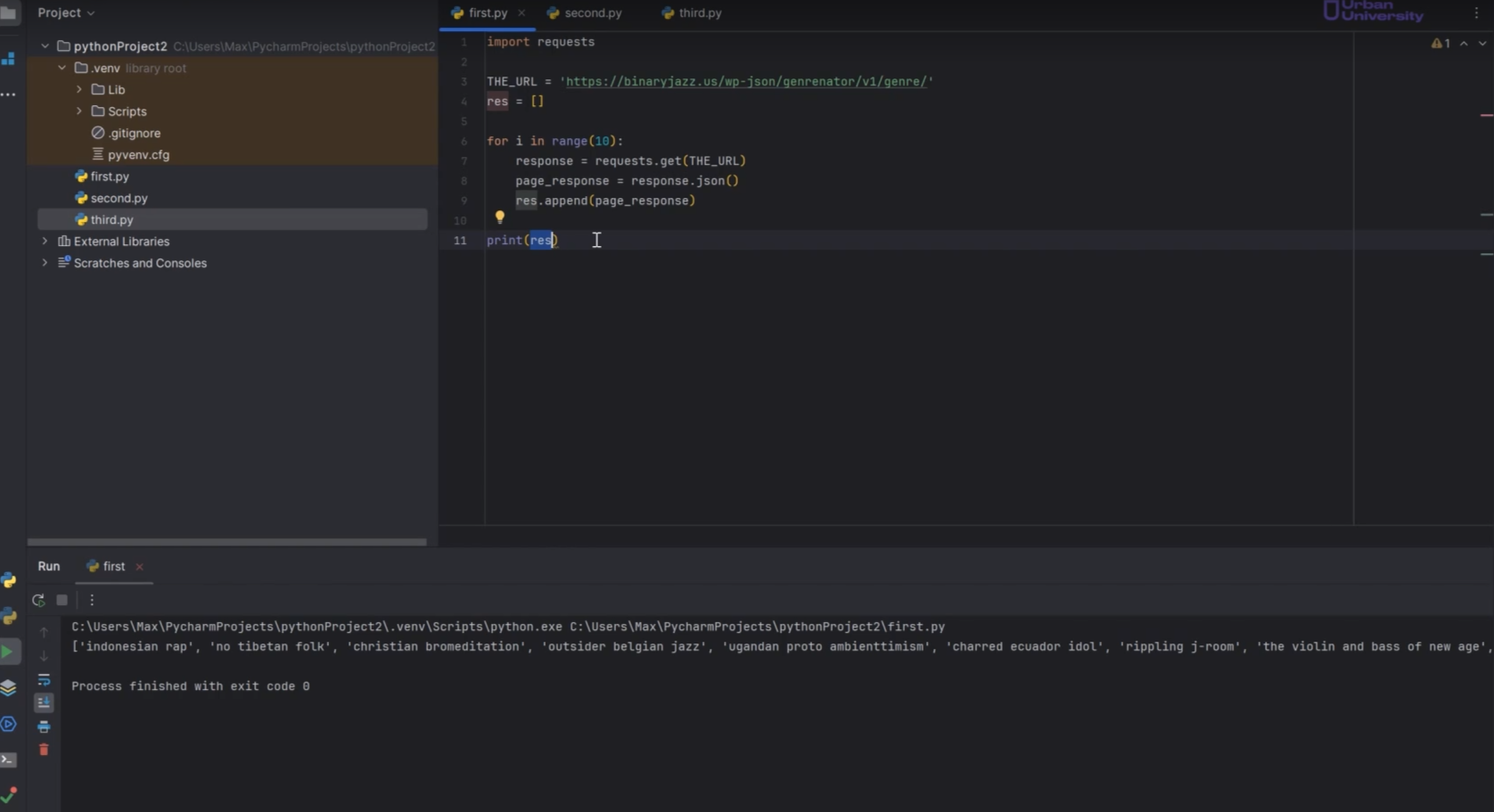


Рис. 6

Воспользуемся библиотекой «**datetime**» чтобы посчитать, как долго это работает. Берем из «datetime» метод «**now**» он запоминает настоящее время и записывает в переменную «**time\_start**». В конце выполнения программы создаем переменную «**time\_end**», которая хранит "**datetime.now**". Между 1 и 2 «datetime.now" проходит промежуток времени, который нам нужен. (Рис. 7). Этот промежуток запишем в «**time\_res**». (Рис. 8)

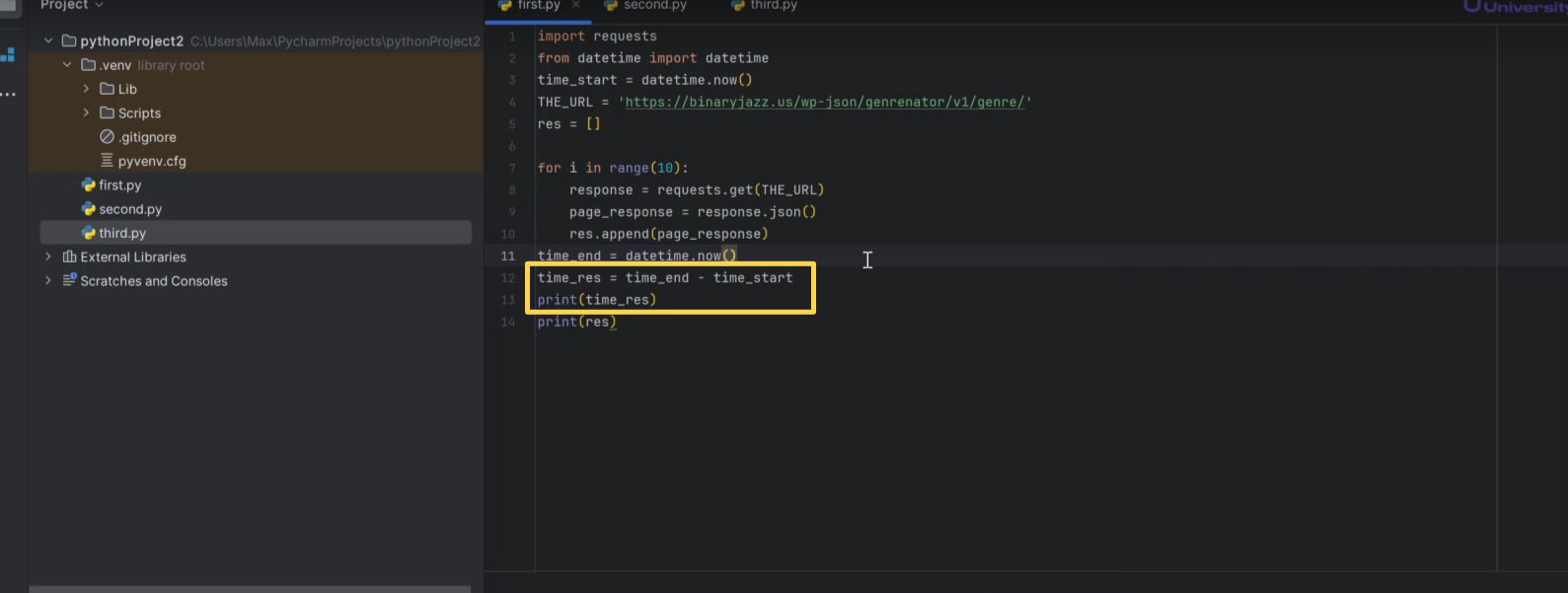


Рис. 7

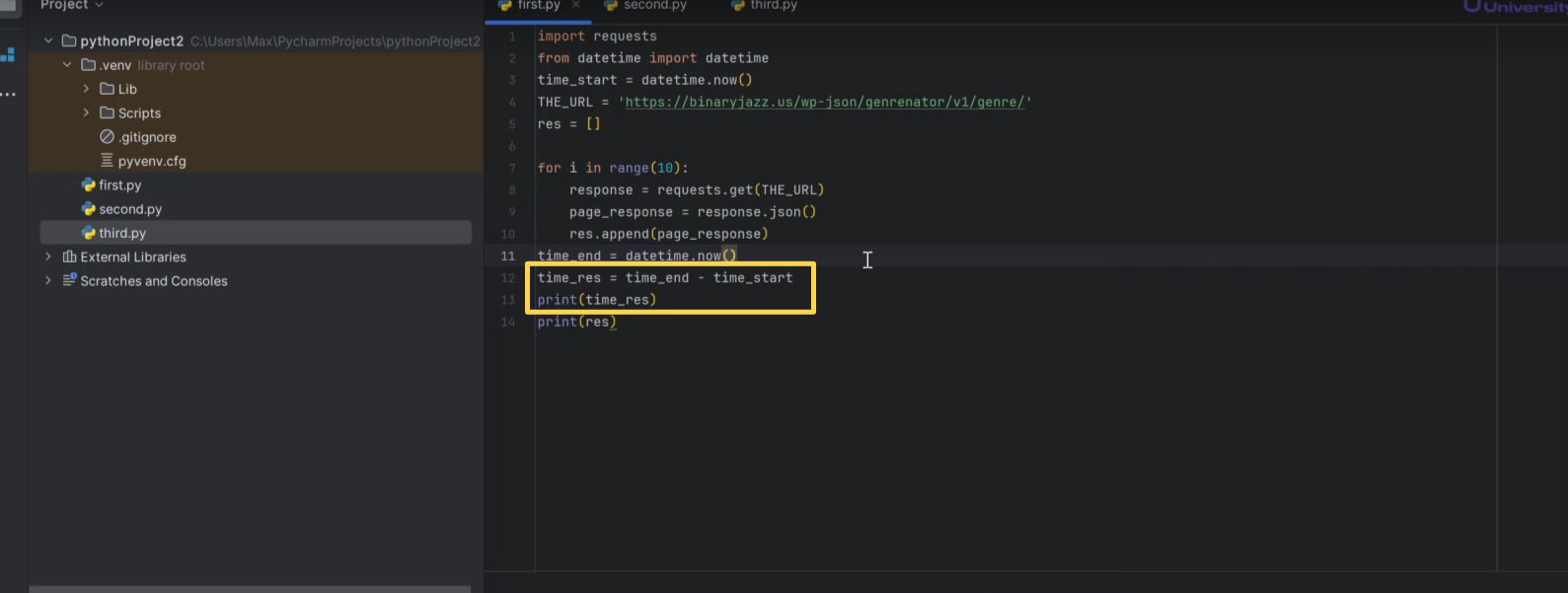


Рис. 8

Запустим. Понадобилось 28 секунд. Для получения 10 вариантов это много. (Рис. 9)

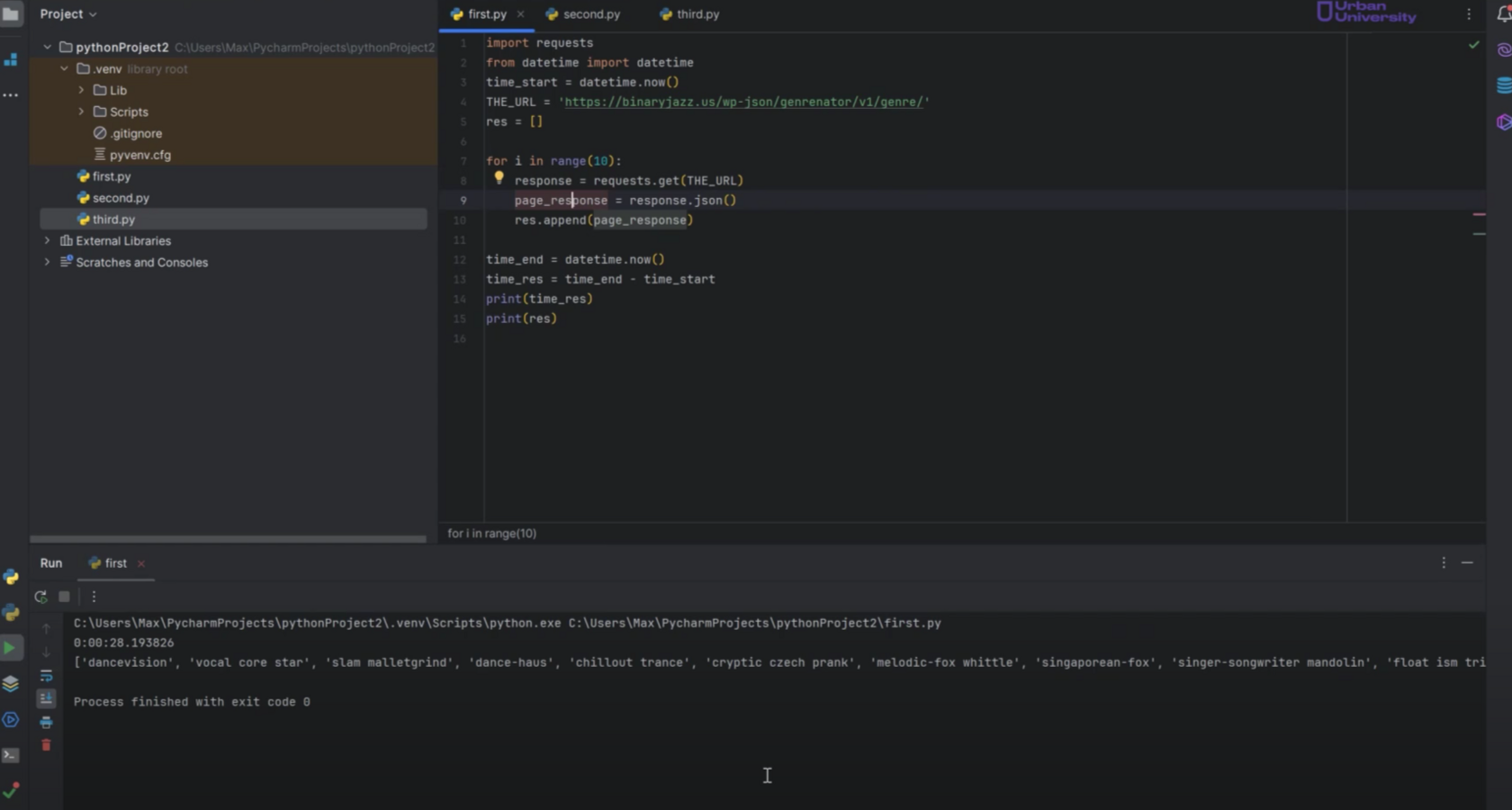


Рис. 9

Как с помощью потока решить эту задачу быстрее?

Импортируем поток, «datetime». (Рис. 10)

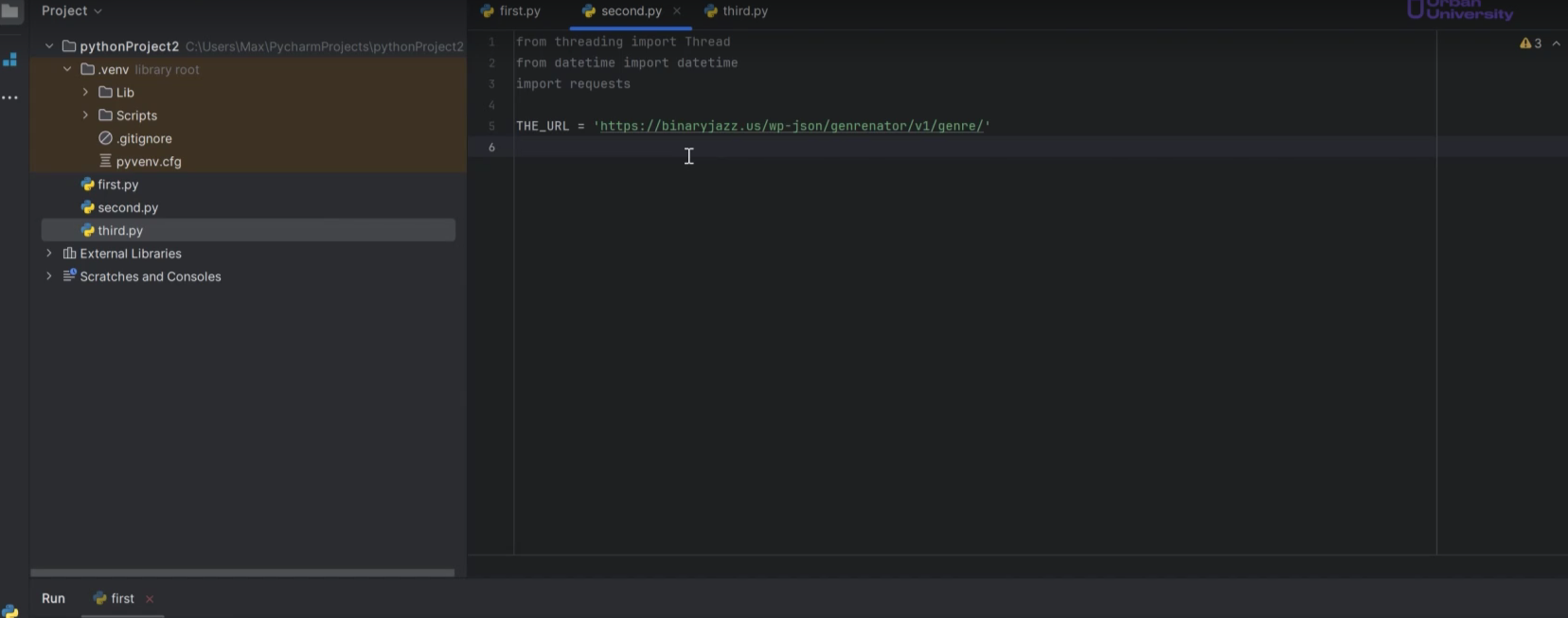


Рис. 10

Представим, что наша программа это 1 человек, который бегает на сервера этого сайта и собирает информацию. Суть в том, что внутри цикла побежал и больше некому бежать на сервера сайта. Потоки позволяют использовать нескольких «человечков».

Объявим функцию «**func**», которая принимает «**url**» и «**res**». Распишем что было до этого внутри цикла. (Рис. 11)

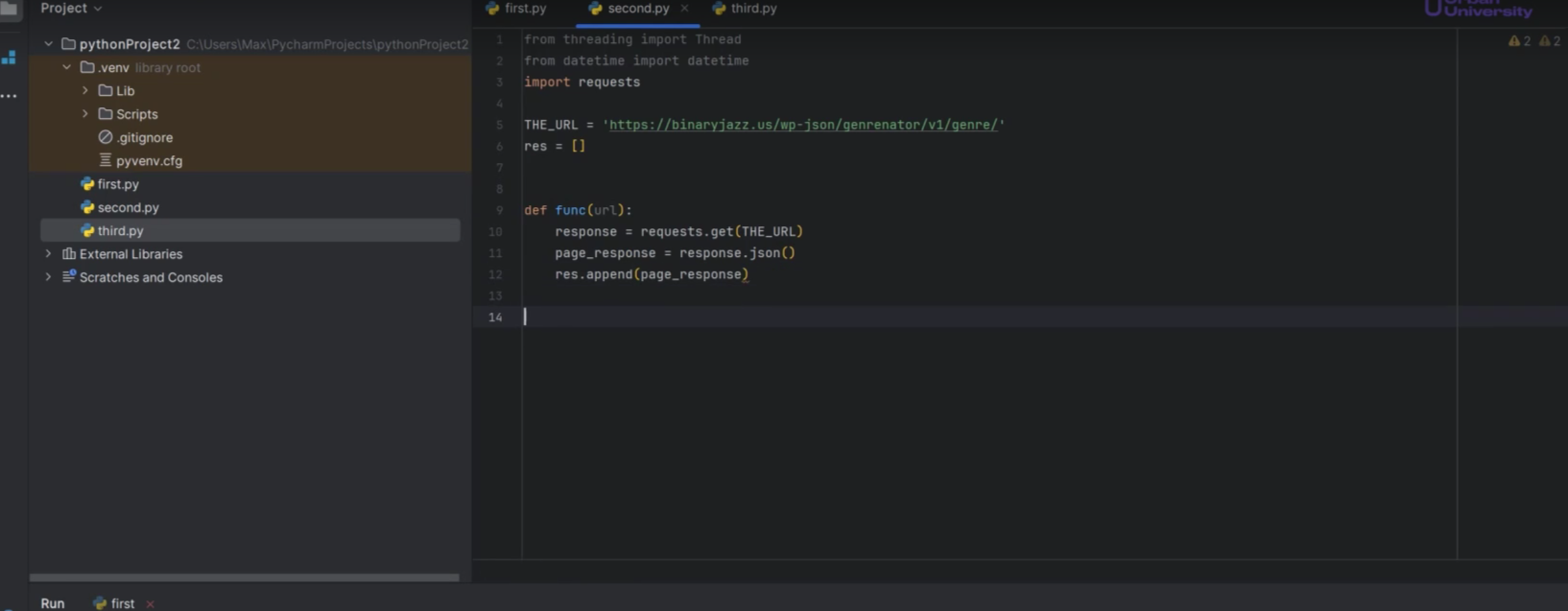


Рис. 11

### Работа с потоками

Чтобы создать поток, обращаемся к классу «**Thread**». «**target**» - функция, которую поток будет выполнять, «**args**» - аргументы, которые будут переданы в эту функцию. (Рис. 12)

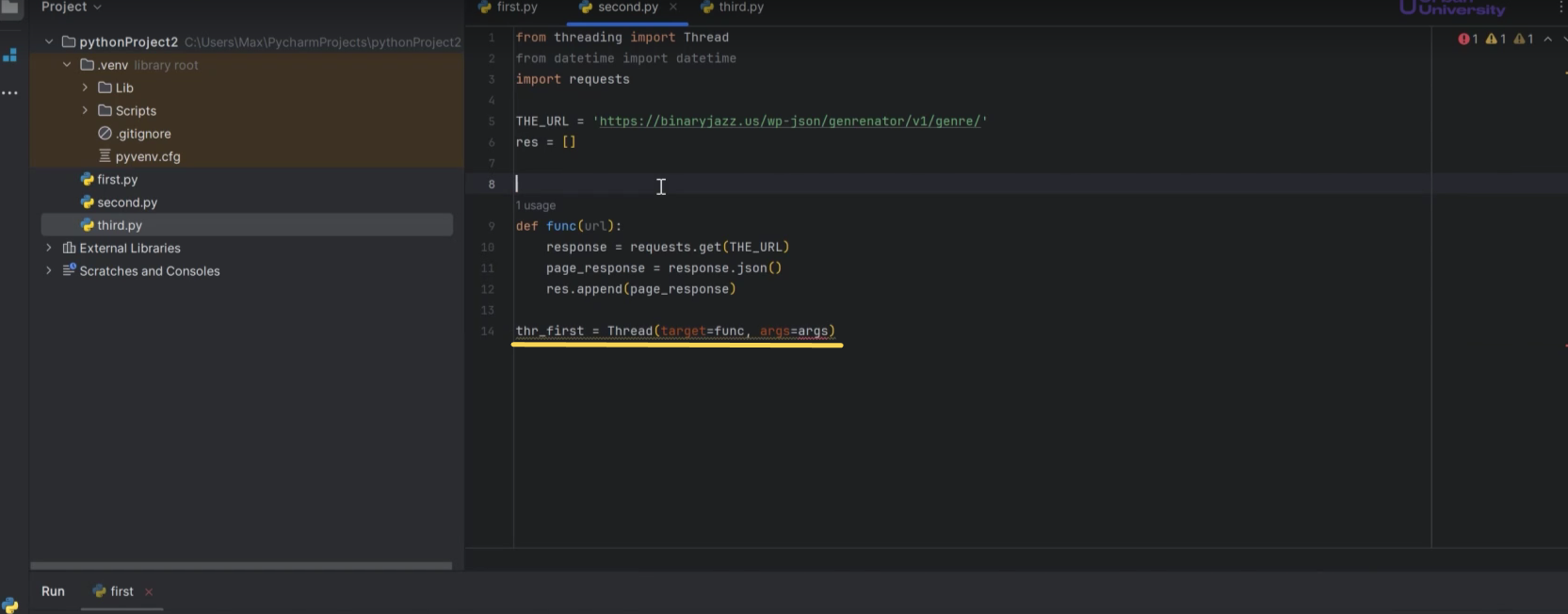


Рис. 12

Каждый поток запускает функцию независимо от остального кода.

Чтобы потоки не шли хаотично есть команды для их контроля. Рассмотрим часть из них. Создадим еще 2 потока и объявляем переменную «**args**» (Рис. 13, 14)

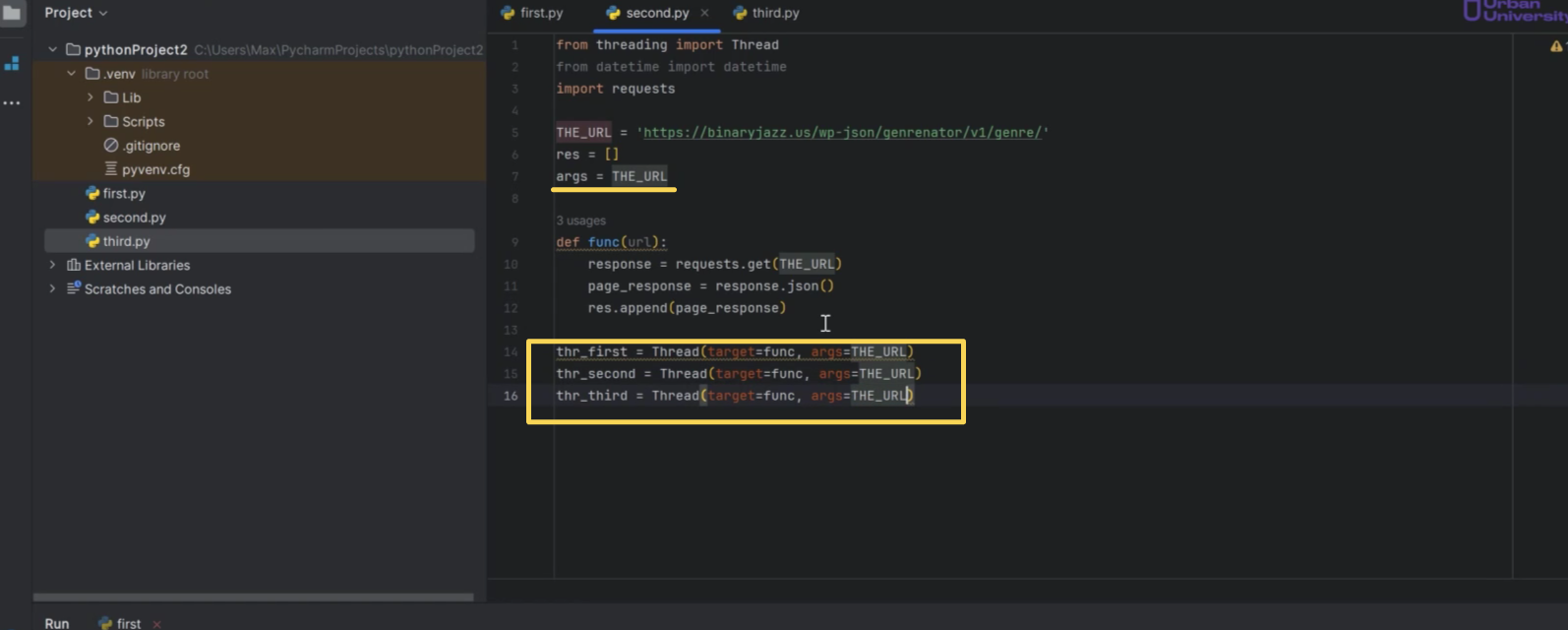


Рис. 13

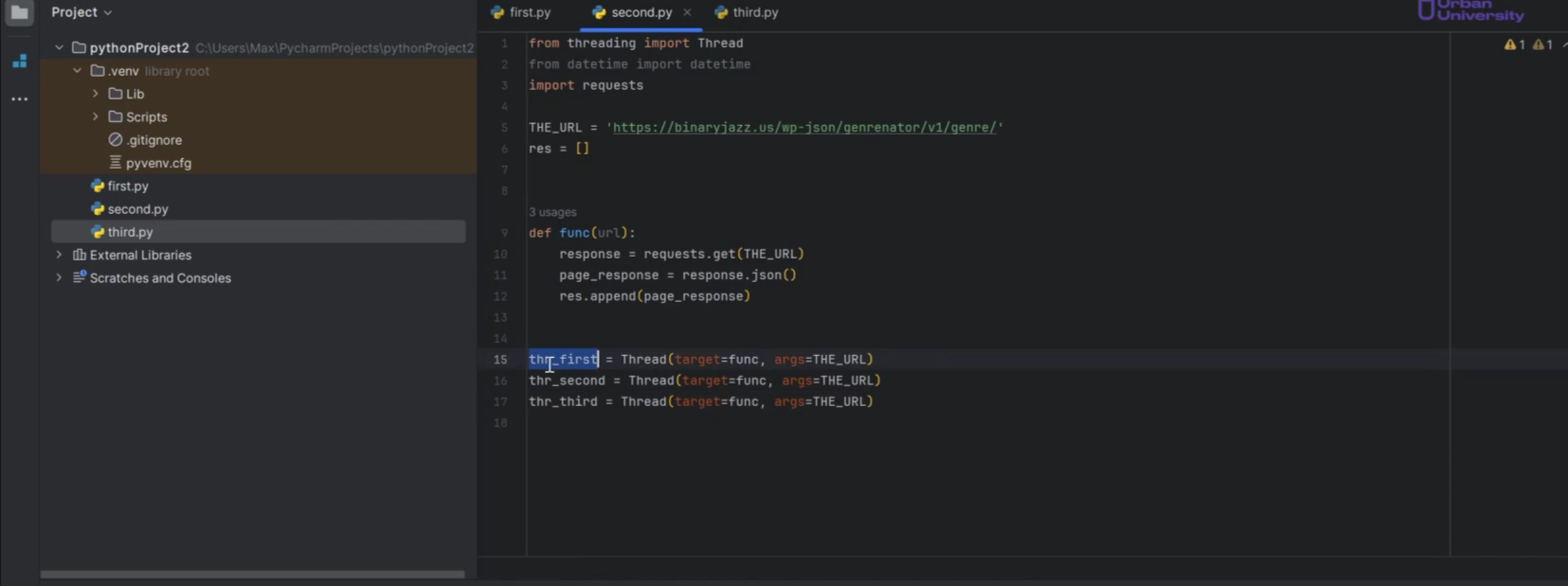


Рис. 14

Чтобы запустить поток, мы его создали, но на «thr\_first» он не запускается. Программа усведомлена, что готовится поток, который будет выполнять функцию. Запустим поток. (Рис. 15)

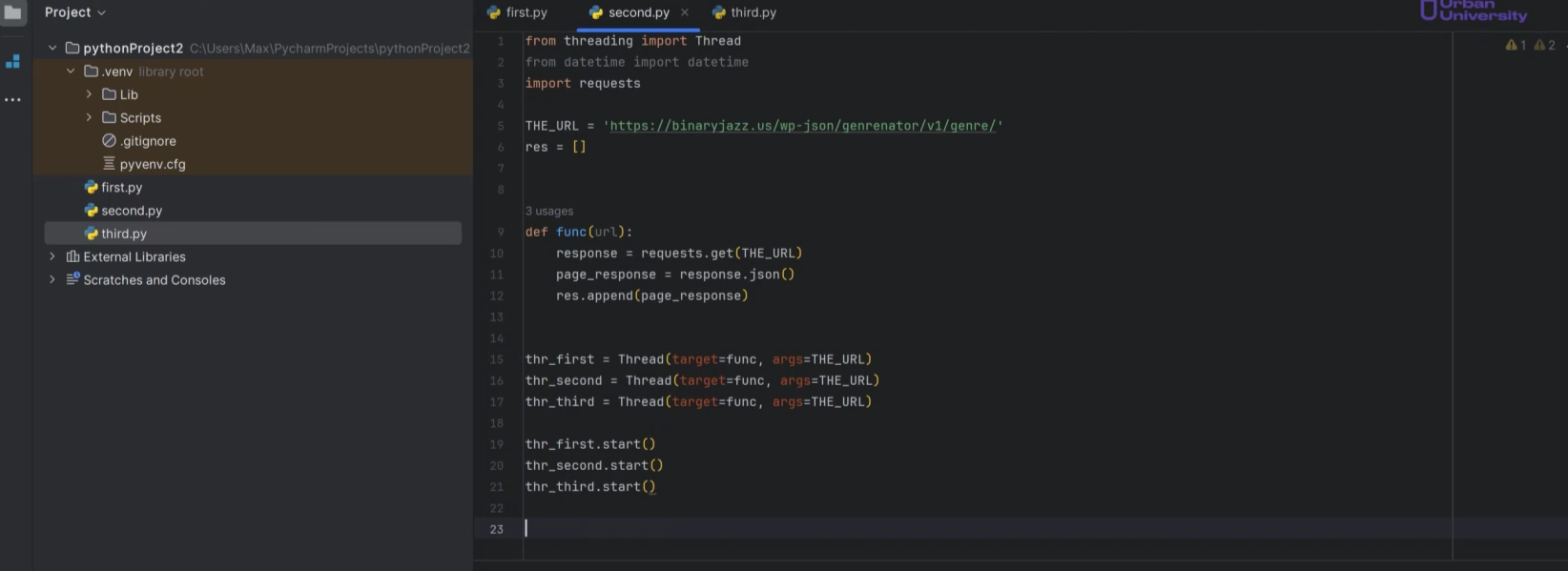


Рис. 15

Теперь нужно дождаться «человечков» пока они не добежали и не заполнили список «res». Если не написать еще одну **конструкцию, которая контролирует конец работы потока**, то поток запустится и основное тело программы оборвется. Программа будет работать, но результат выводиться не будет.

Для этого есть специальный**метод «join»**. Он останавливает работу программы до тех пор, пока не дождется завершения работы потока. (Рис. 16)

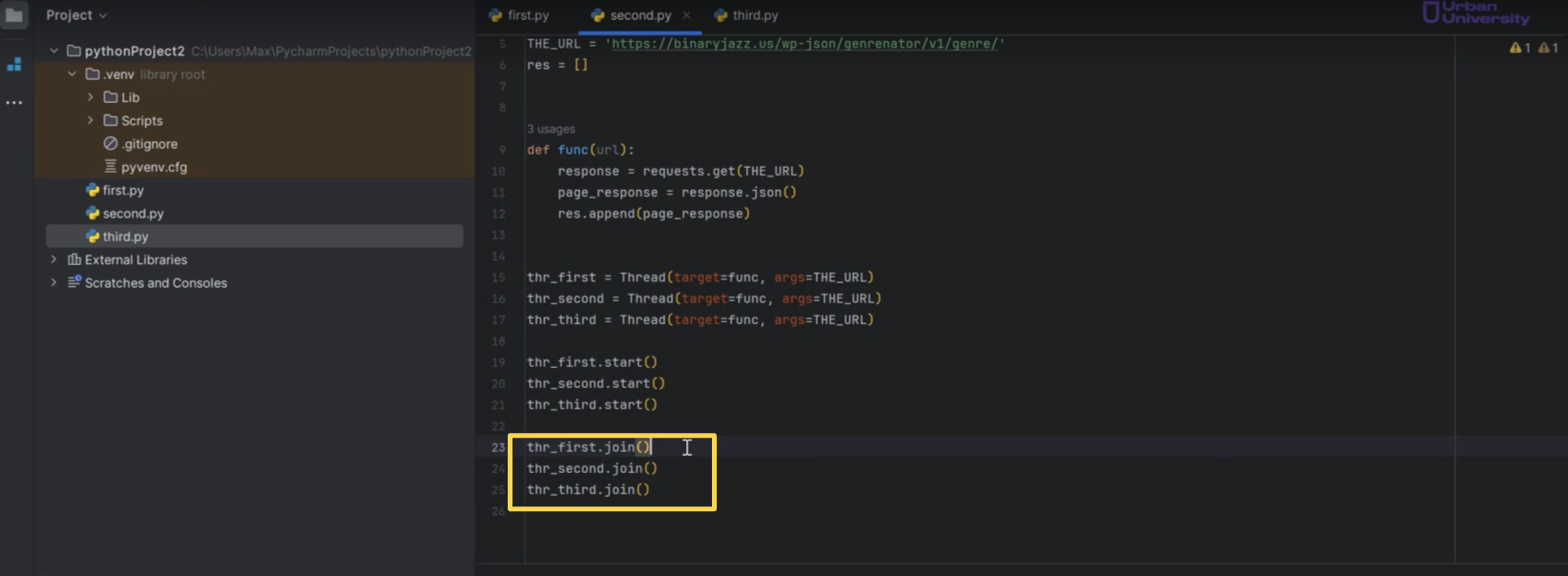


Рис. 16

**Важно!**«Человечки» необязательно прибегают в том порядке, в котором выбежали. Может быть такое, что как только прибегает первый «человечек» окажется что второй и третий уже прибежали и времени много тратиться не будет.

Напечатаем «**res**». (Рис. 17)

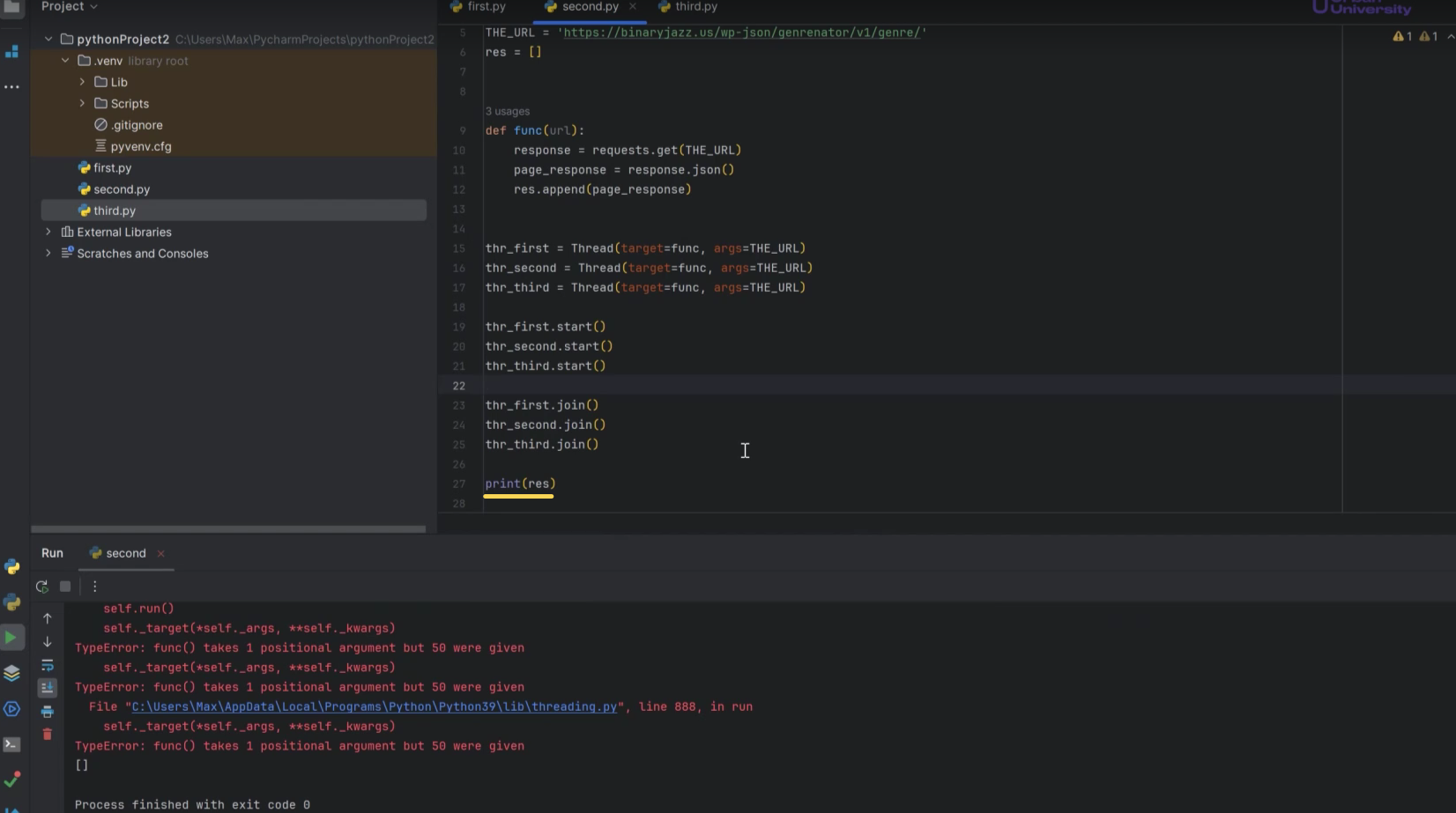


Рис. 17

Ошибка в том, что нужно было **передать кортеж**, а не переменную. (Рис. 18)

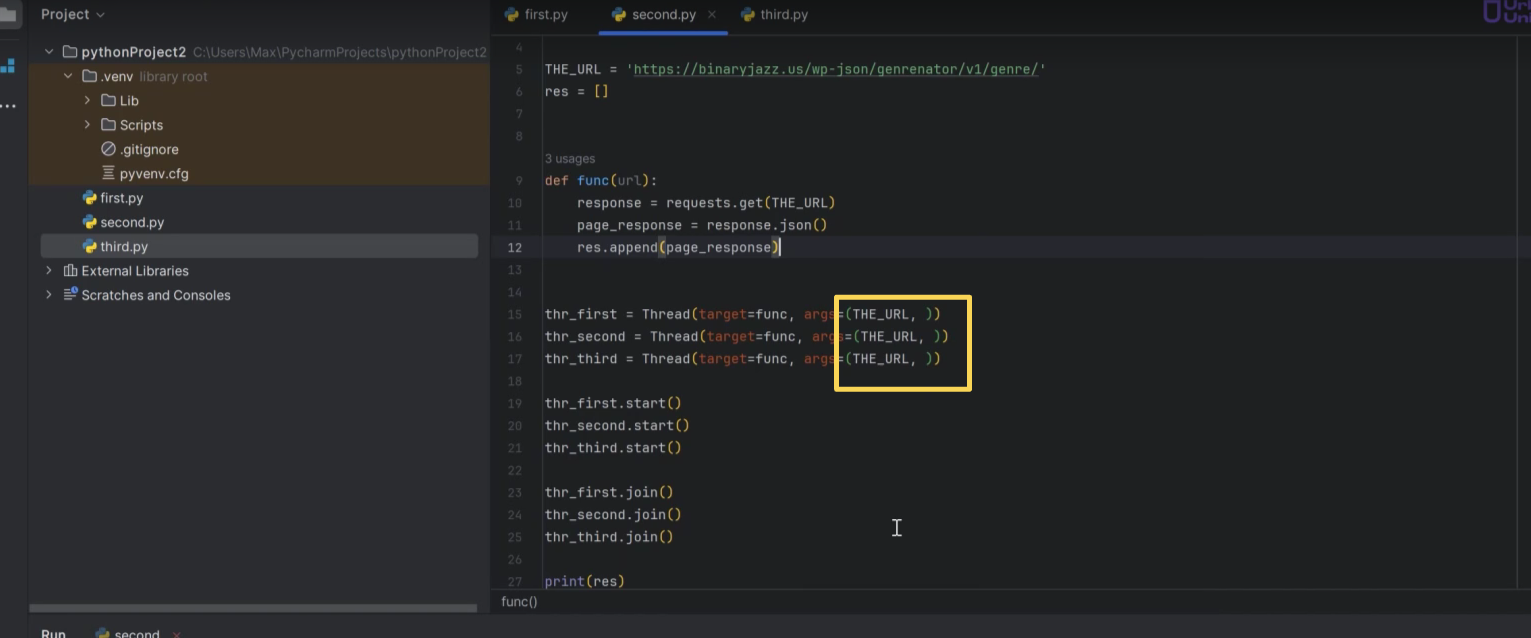


Рис. 18

Запустим. Получим 3 жанра. (Рис. 19)

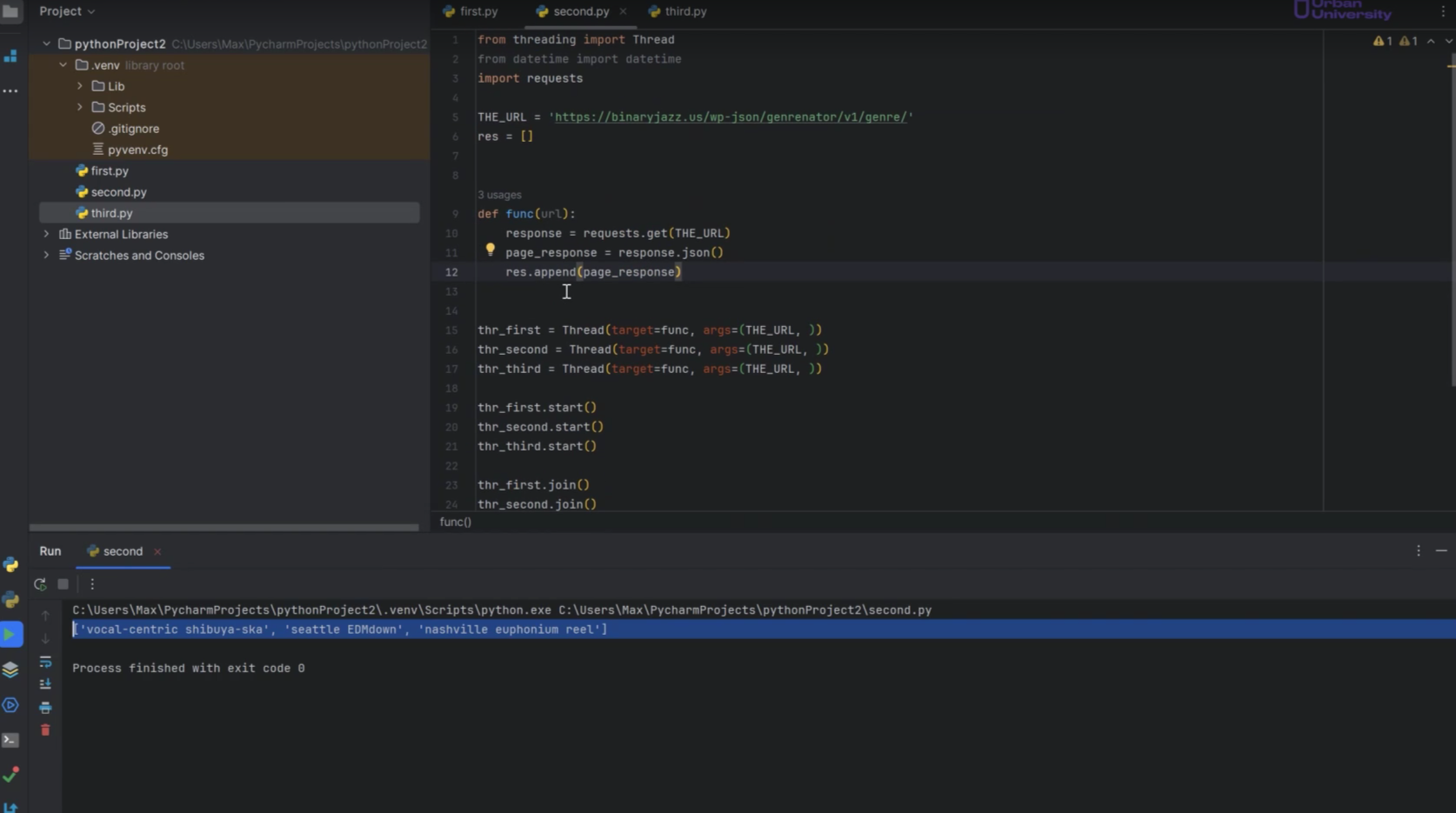


Рис. 19

Замерим скорость выполнения, выведем результат и сравним с предыдущей программой, где не использовались потоки. (Рис. 20, 21)

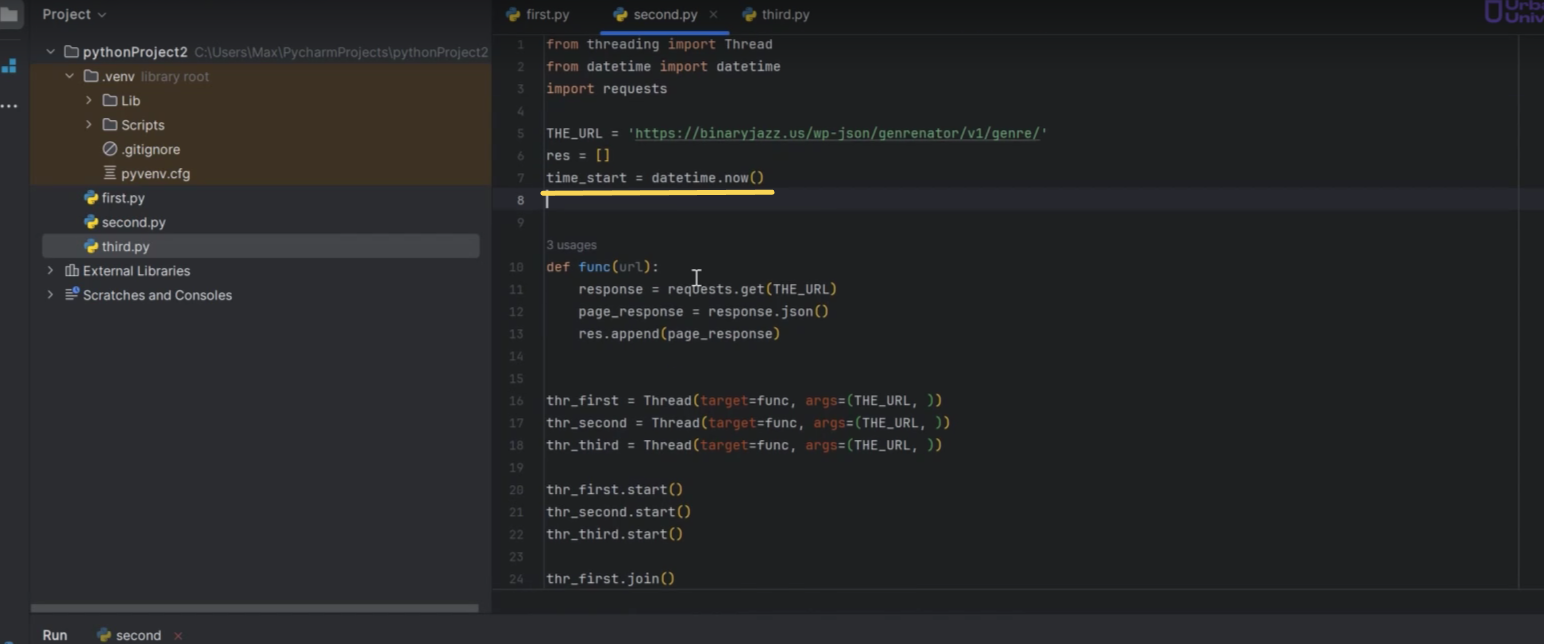


Рис. 20

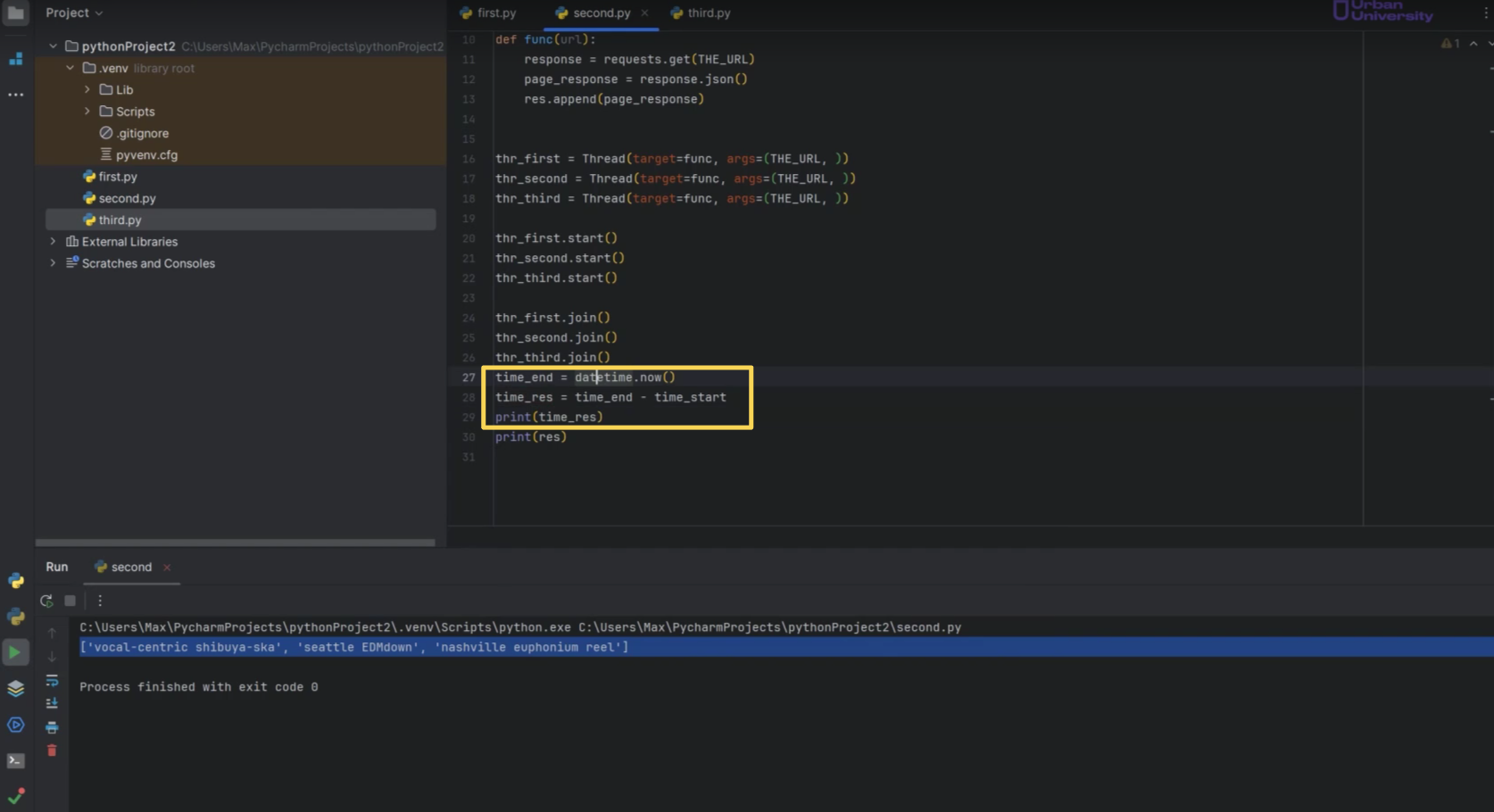


Рис. 21

Изменим количество жанров на 3. (Рис. 22)

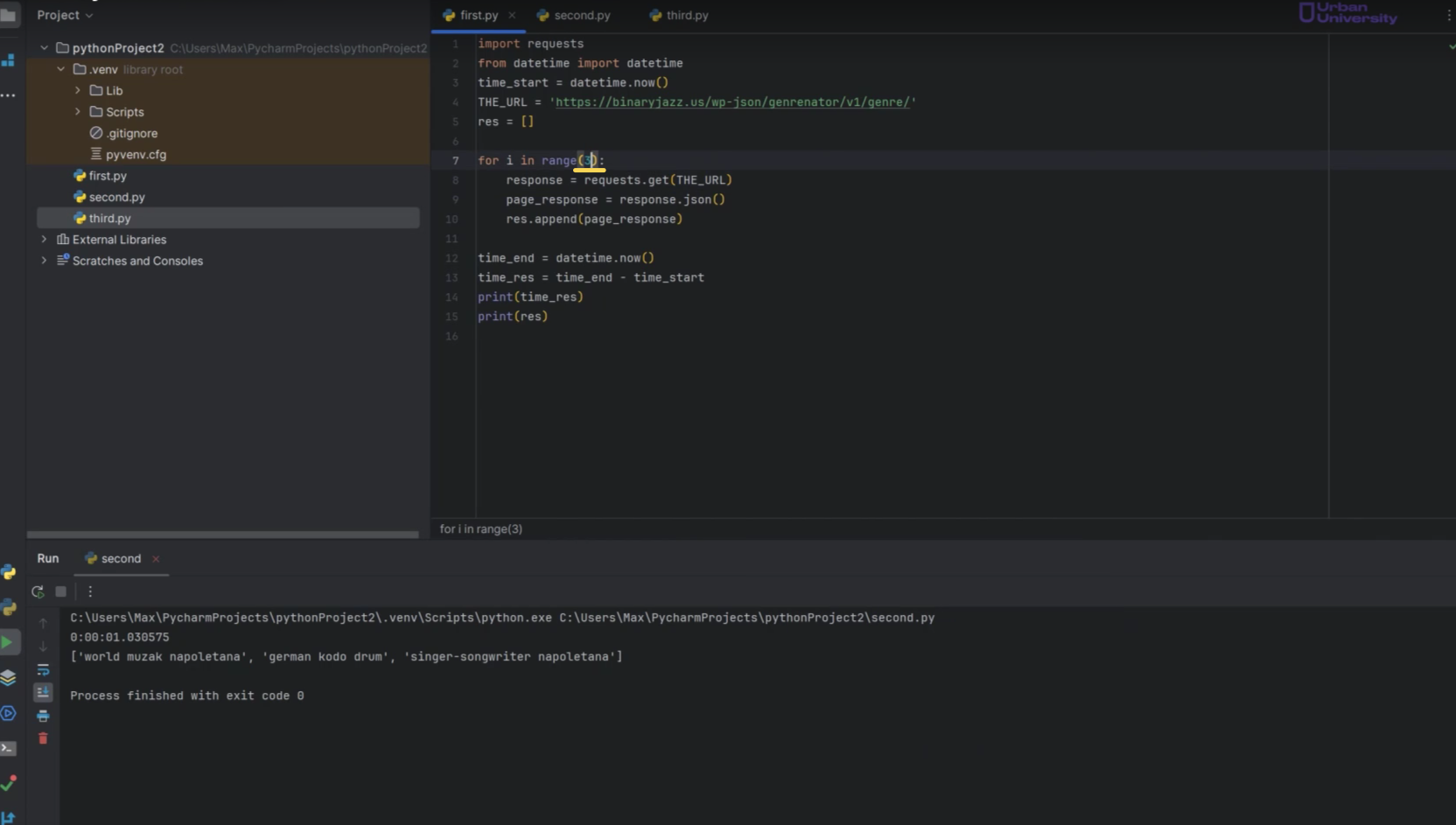


Рис. 22

В первой программе выполнялось 2.6 с, во второй около секунды. (Рис. 23, 24)

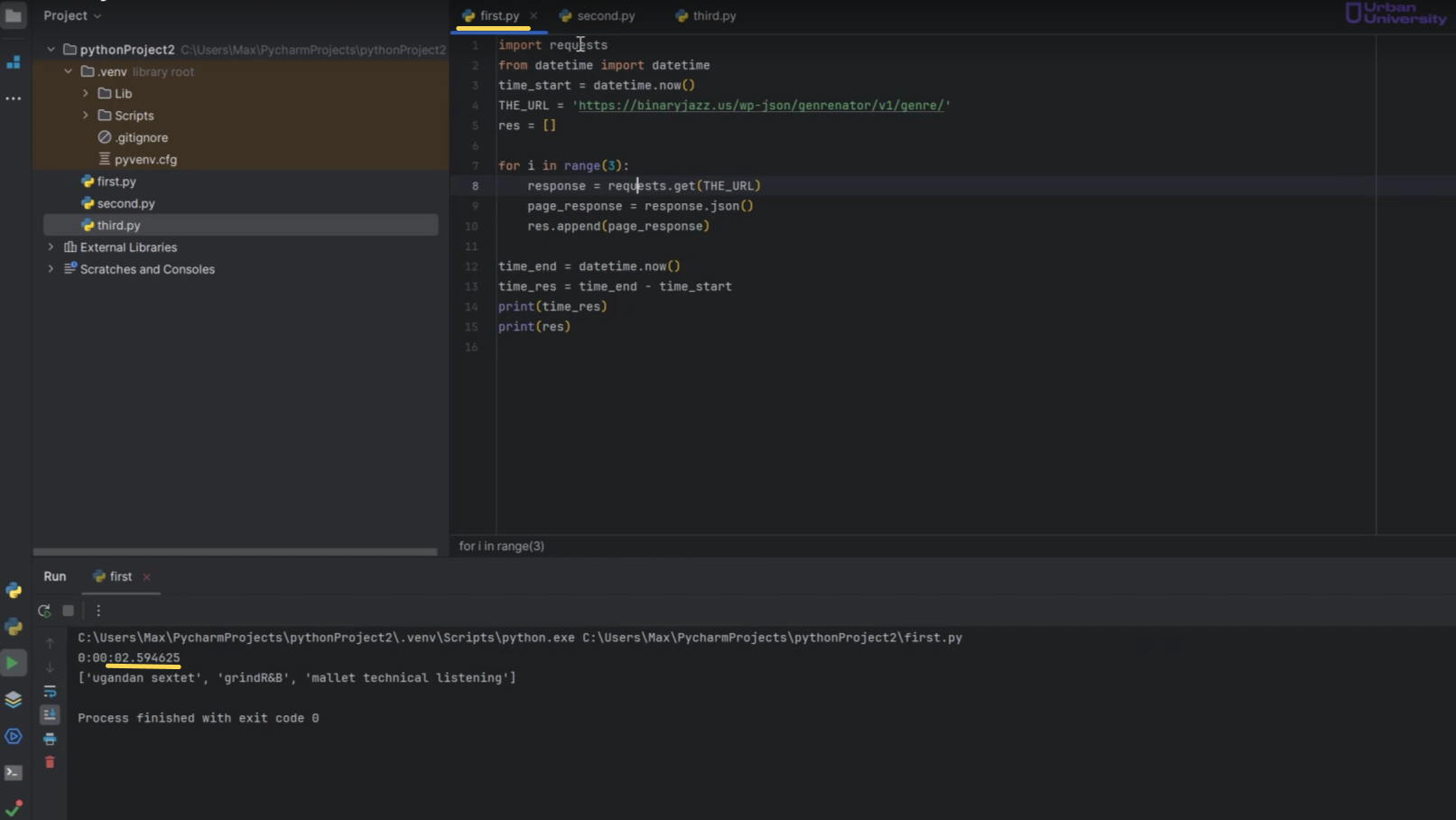


Рис. 23

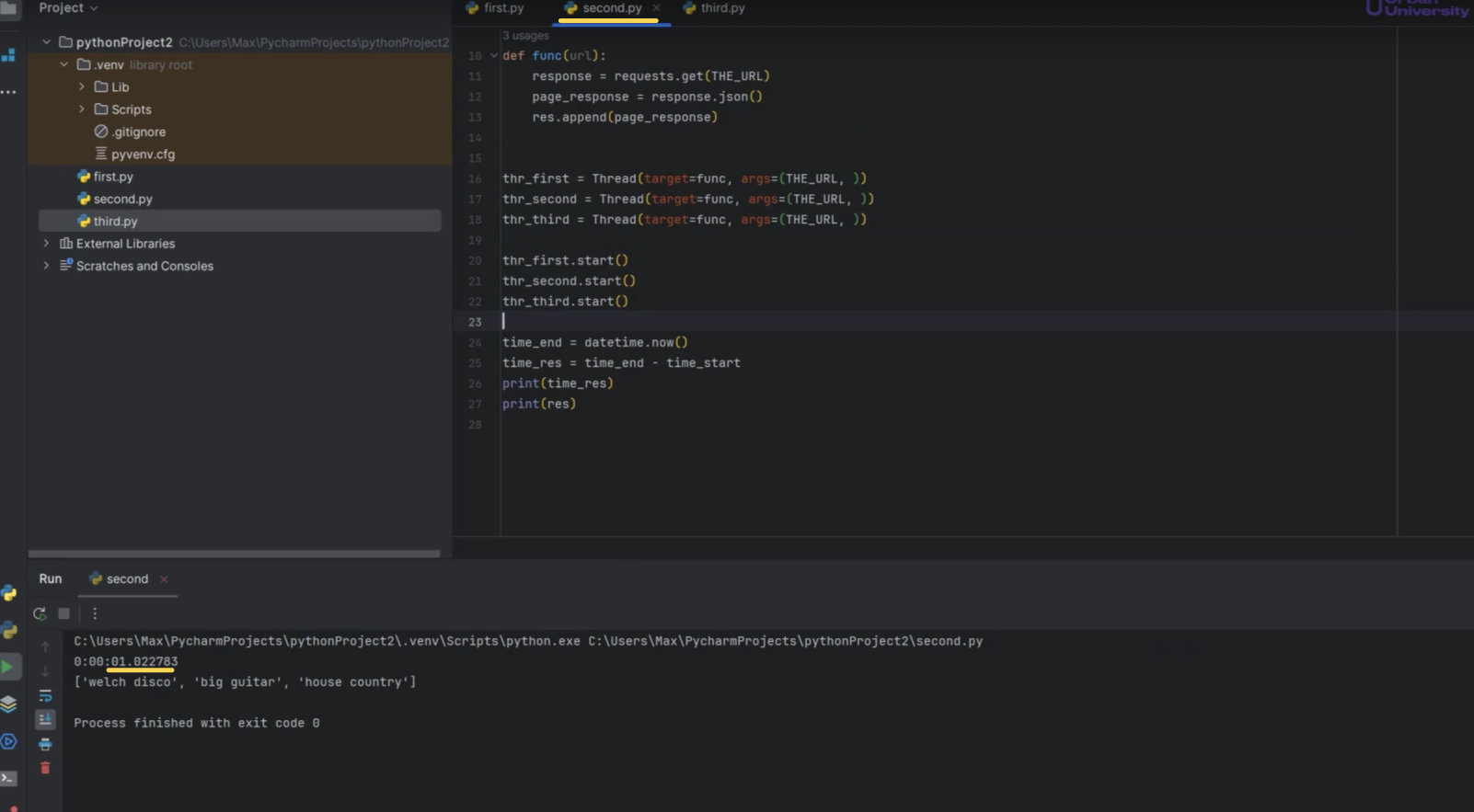


Рис. 24

Что произойдет если не дожидаться потоков? Программа сработает невероятно быстро и безрезультатно. (Рис. 25)

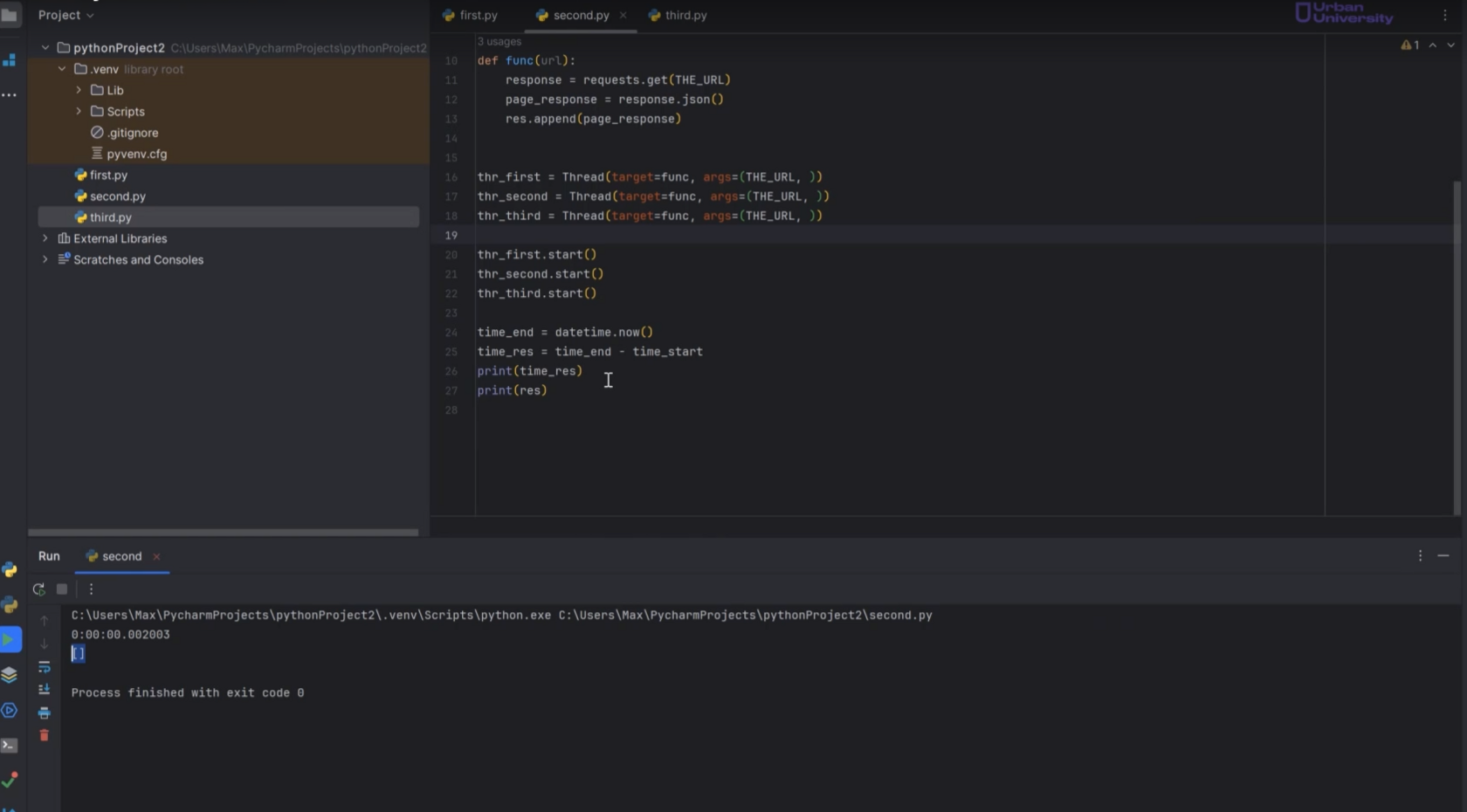


Рис. 25