**2023/10/17 00:00|Лекция. Пакеты и способы импорта**

## Пакеты и способы импорта

На сегодняшнем занятии мы с вами поговорим про пакеты и про возможности их импорта. Чем больше, ваш проект, тем больше потребности в наличии пакета и тем больше вероятность, что один пакет будет содержать другой. То есть у вас будет такая сложная многоуровневая структура. Обратите внимание на структуру нашего проекта. Мы подготовили три пакета: «package1», «package2» и «package3» (Рис.1). Эти пакеты, например, «package1» содержит сам пакет «package2». Subpackage или подпакет- это тот пакет, который внутри другого пакета, как бы это странно не звучало (это же очевидно). И «package3»- это подпакет для второго нашего пакета. Каждый из них содержит какие-то файлы.

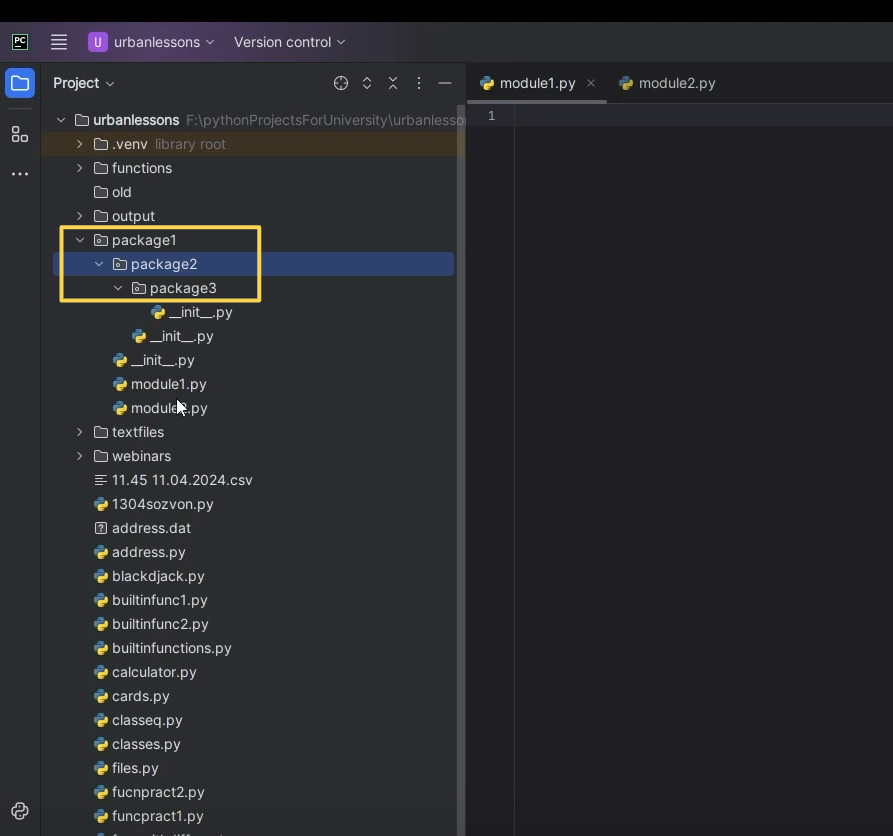


Рис.1

У нас есть два модуля «module1» и «module2», которые находится внутри «package1» (Рис.2). Напоминаем, что главный пакет «package1» находится внутри рабочей директории внутри «urbanlessons».

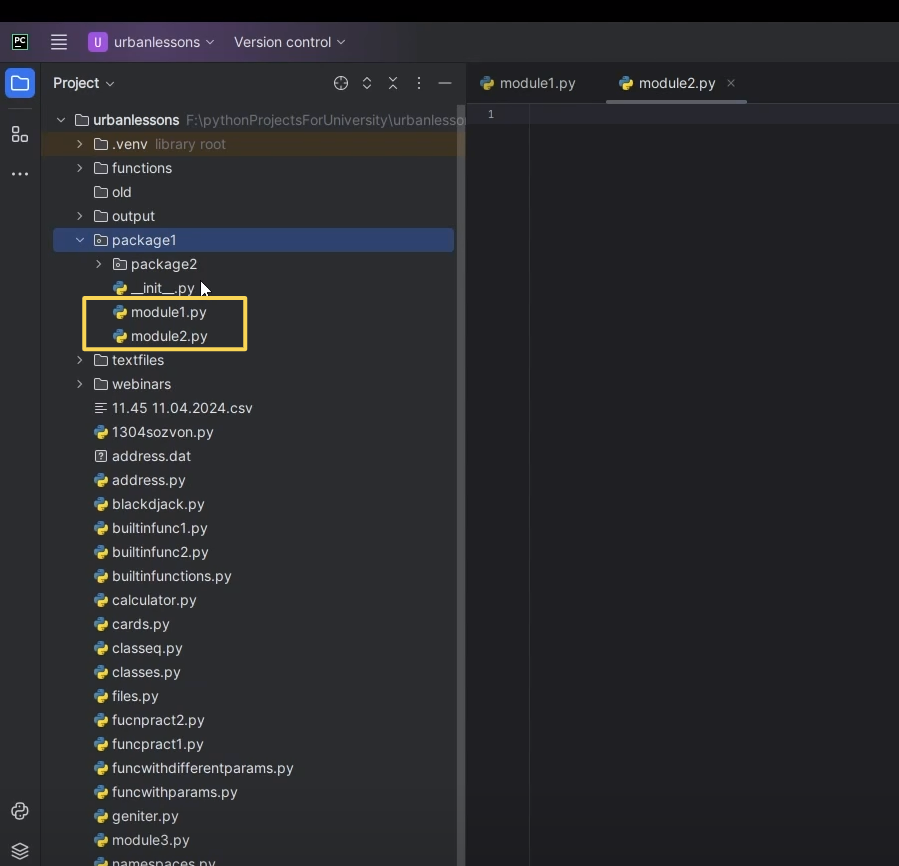


Рис.2

Представим ситуацию: у нас есть первый модуль, в нём есть какая-то функция, допустим, «hello», которая принимает какое-то имя. И выводит, например, (“Hello”, name) (Рис.3)

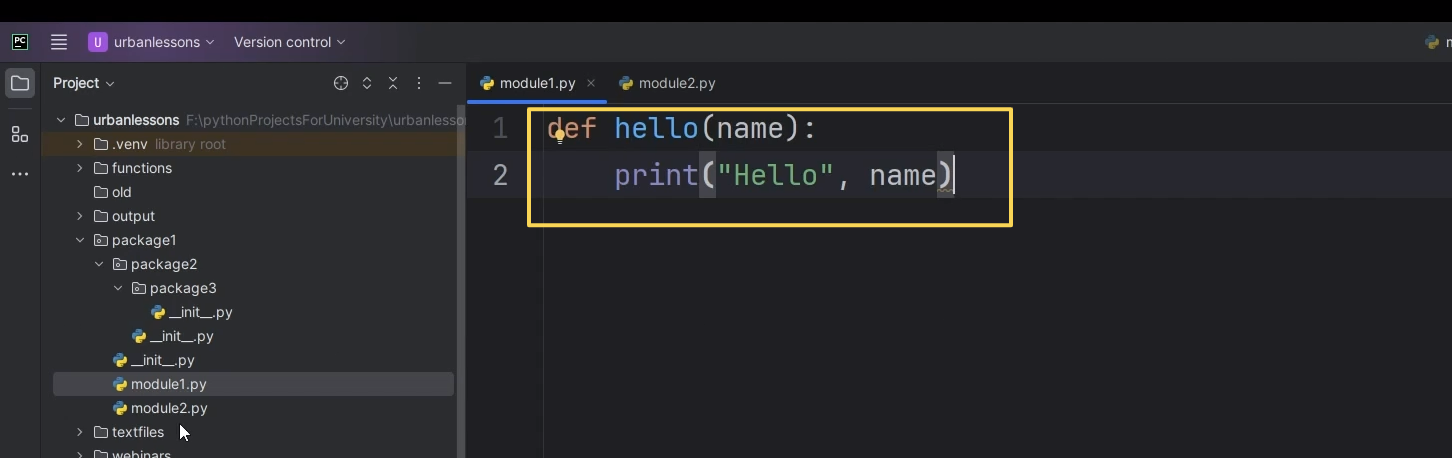


Рис.3

Есть второй модуль и в нем функция «def good\_word», грубо говоря, комплимент какой-то, который точно также принимает имя. Что мы будем делать внутри этой функции? Напишем «ты лучший» и добавим «name» (Рис.4).

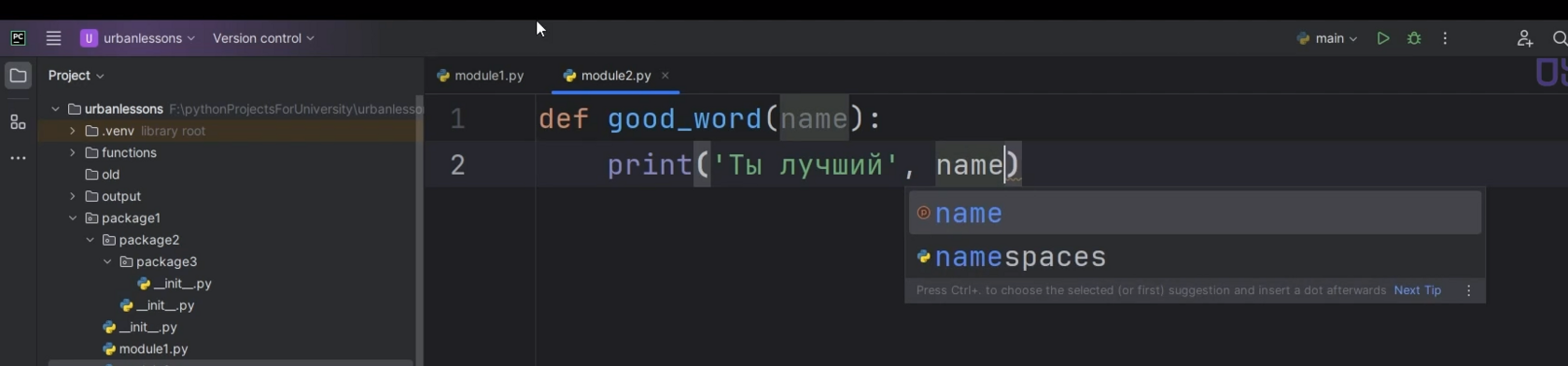


Рис.4

В первом модуле, соответственно, чтобы все было в одном формате, напишем «Привет» (Рис.5).

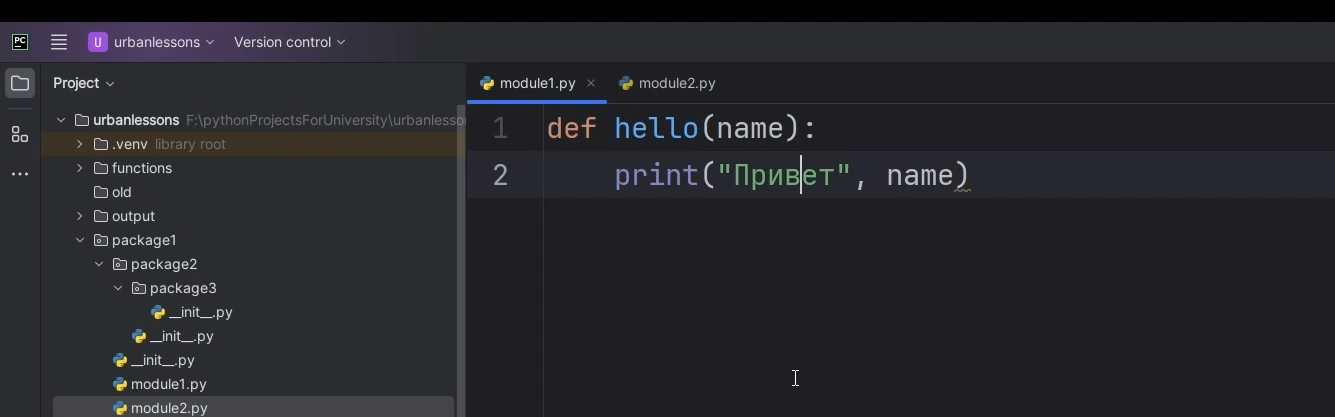


Рис.5

Вот у нас две функции, которые находятся в разных модулях. Давайте во втором модуле кое-что проверим. Создадим такую конструкцию «if \_\_name\_\_==\_\_main\_\_», которую мы рассматривали, и здесь будем вызывать функцию «Урбан», передавая её как аргумент. Запускаем и видим, все работает (Рис.6).

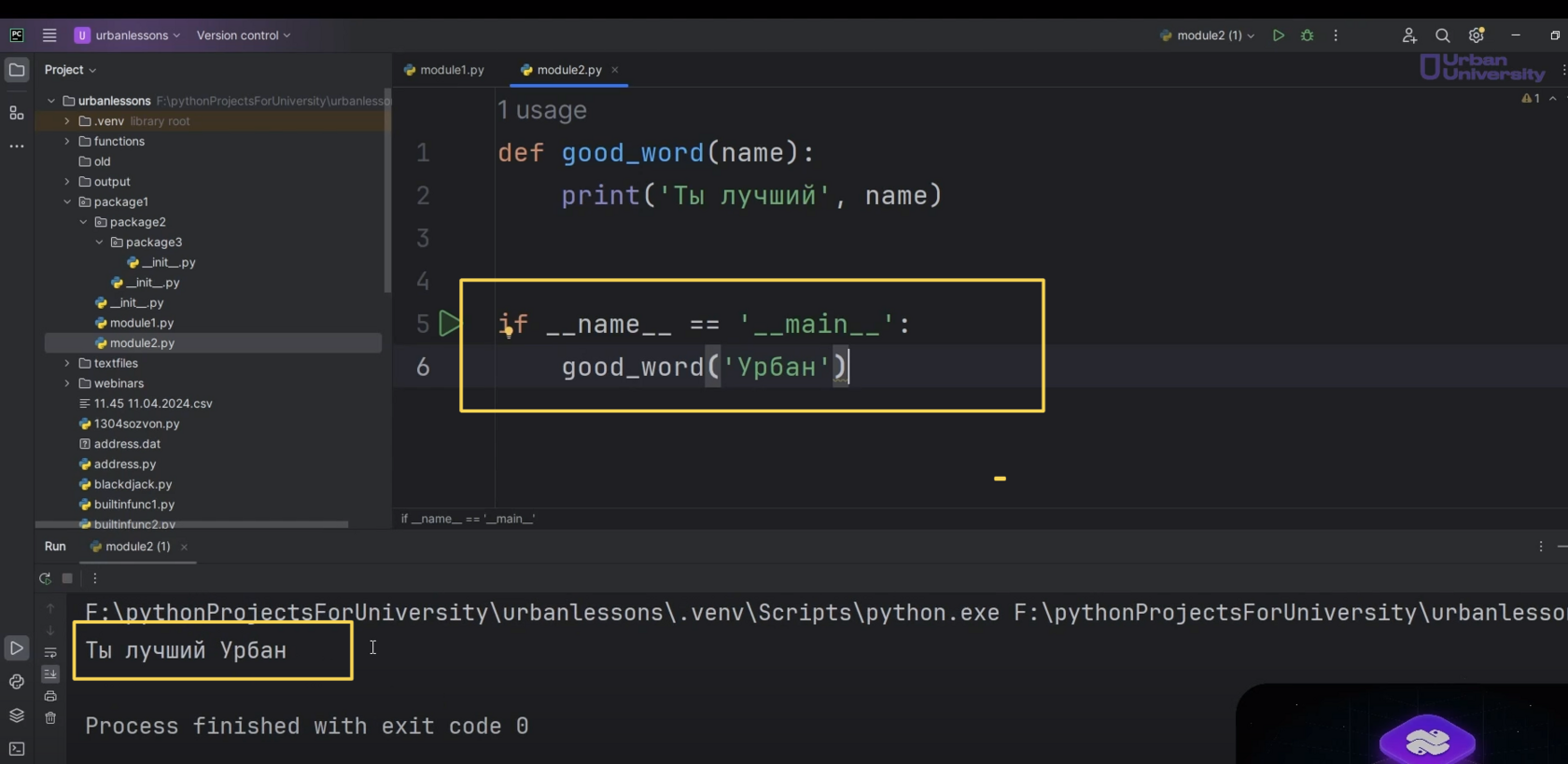


Рис.6

Однако мы хотим, допустим, внутри этой функции «good\_word», вызвать ещё и функцию из первого модуля. Что нам нужно сделать? Естественно, её нужно импортировать. Там у нас функция называется «hello», которая тоже принимает «name». Мы будем сюда его передавать, получать мы будем вот отсюда (Рис.7).

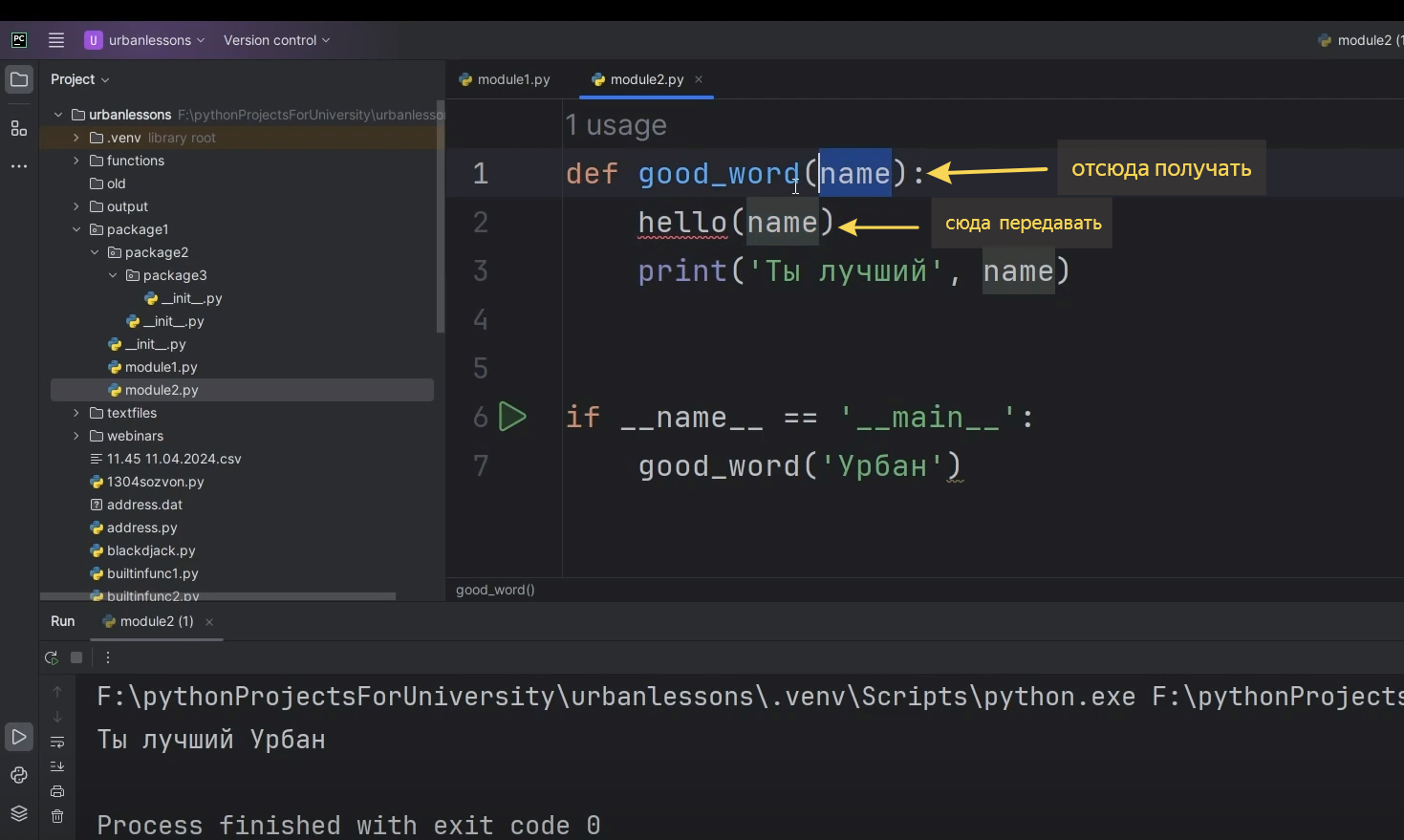


Рис.7

Предварительно, конечно, нужно импортировать. Напишем «from package1.module1 import hello» (Рис.8).

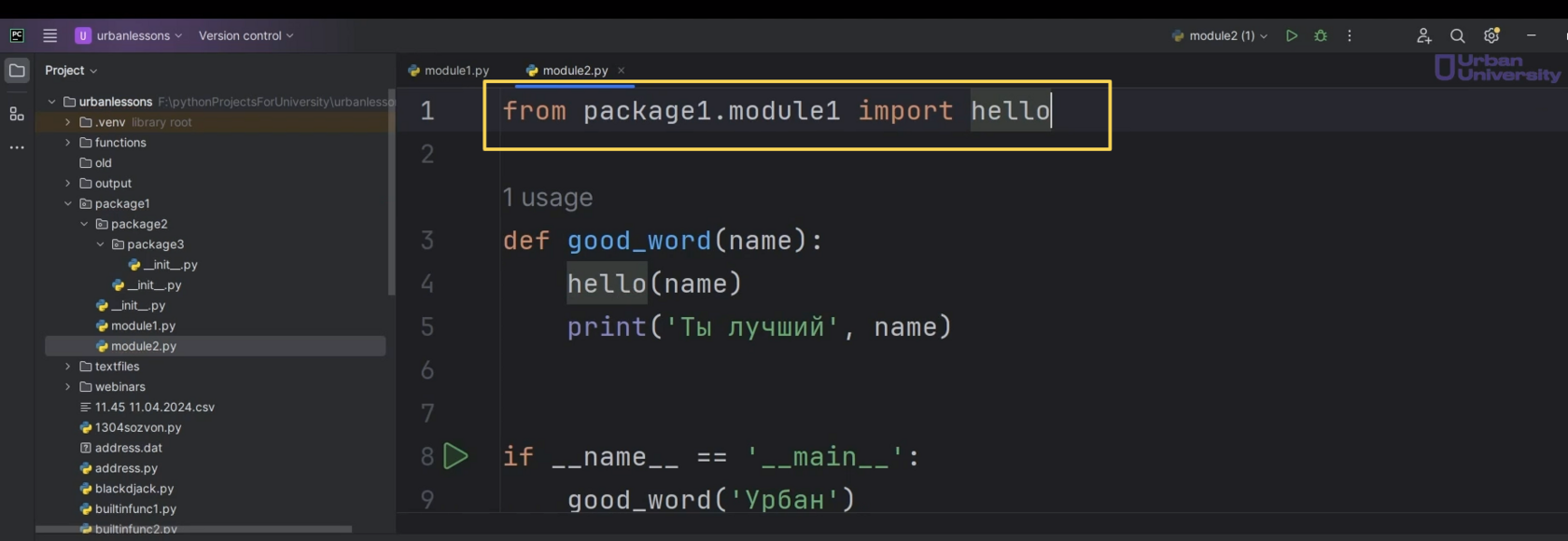


Рис.8

Запускаем (Рис.9). У нас все работает.

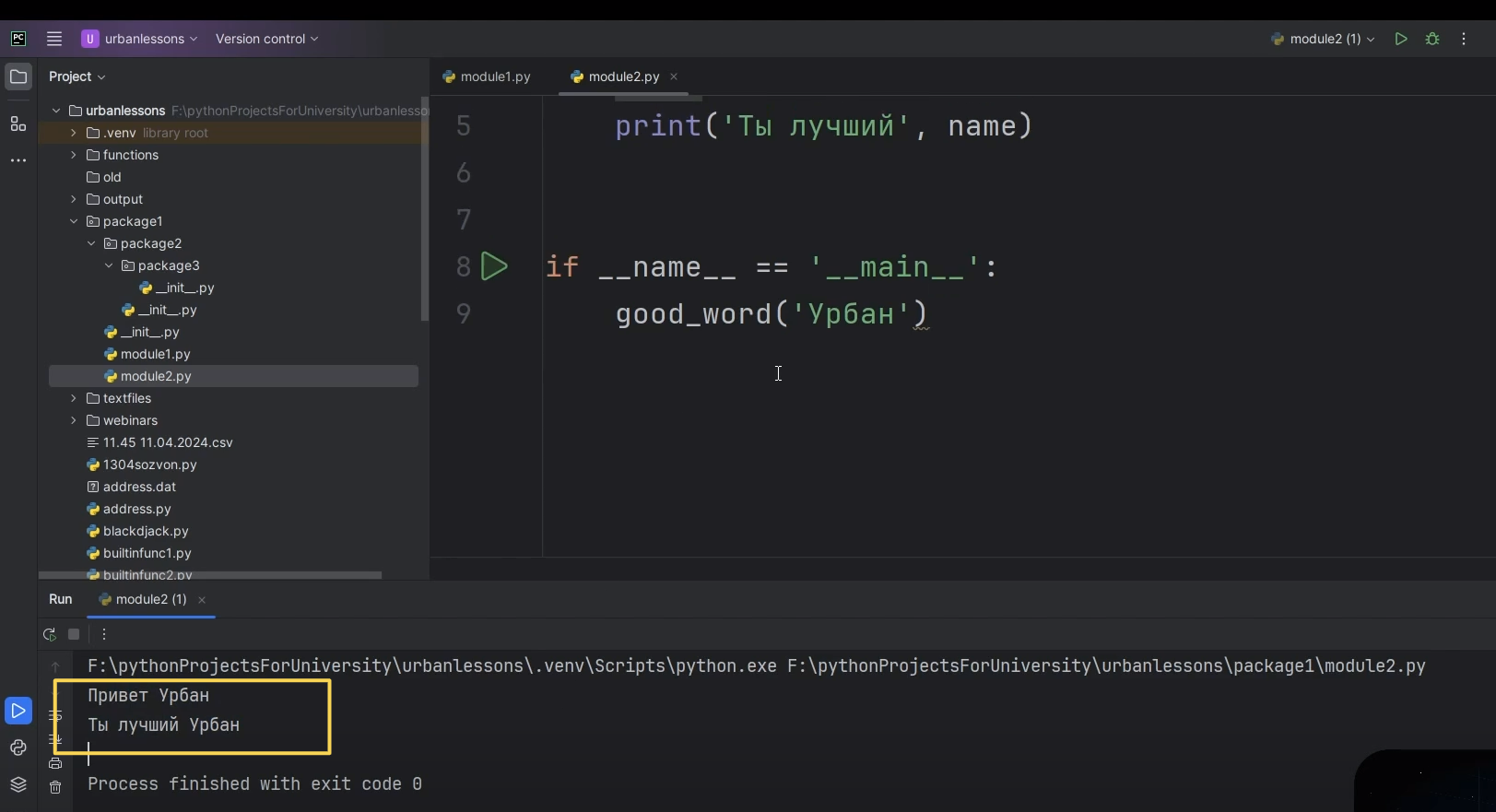


Рис.9

Сейчас мы использовали абсолютный импорт. **Существует два вида импорта:**

1.абсолютный, когда мы полностью указываем путь. То есть изначально мы опираемся на нашу корневую директорию на «urbanlessons»;

2.относительный.

**В чем плюс абсолютного импорта?** Когда мы используем абсолютный импорт, можем писать, например, импорт, там что-то, что-то, что-то- это одно дело. И у нас не возникает никаких ошибок, то есть при импорте мы явно говорим что собираемся взять, откуда берем и так далее. И интерпретатор никогда не путается. Он всегда знает, что мы от него хотим, что просим включить.

Но есть и **минус**. Допустим, наш проект растёт, мы увеличиваем количество пакетов, то есть у нас их все больше появляется, больше уровней вложенности. У нас их становится целая куча, и у нас файлы перемещаются куда-то в глубину, скажем так. Вот теперь они у нас находятся внутри «package3» (Рис.10).

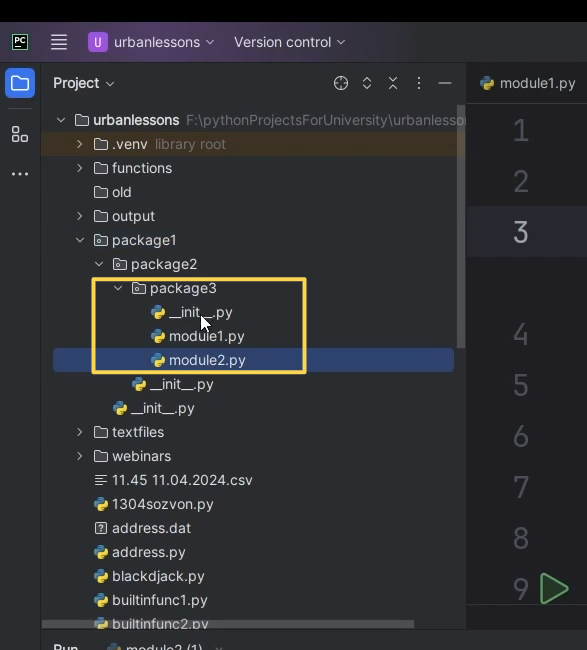


Рис.10

Теперь абсолютный путь до этой функции «hello», которая у нас находится в первом модуле, выглядит иначе. То есть мы сначала берём «package1», потом «package2», потом «package3», потом «module1», и говорим «import hello» (Рис.11).

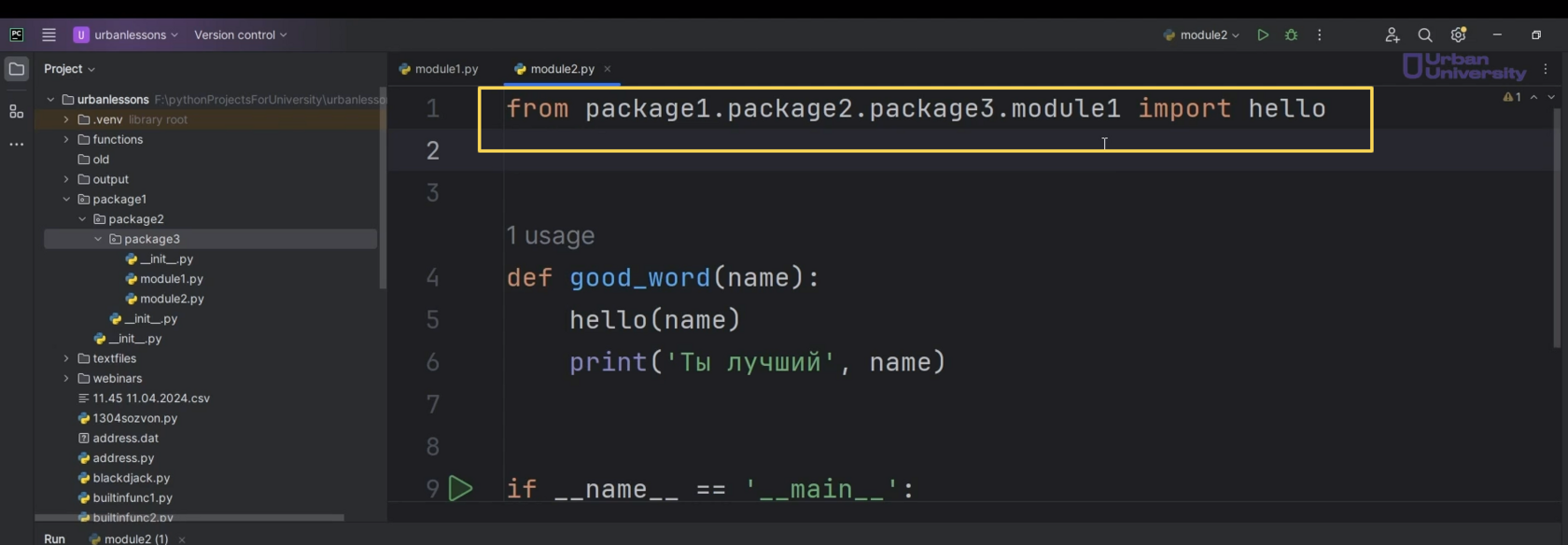


Рис.11

Это не есть хорошо. Импорт у нас длинный. Это немного может сделать написание проблематичным, потому что этих уровней вложенности может быть на самом деле сколько угодно. У вас может быть и 10 пакетов. И, соответственно, у вас строчка будет огромная, и пакеты могут называться не так коротко, допустим, как у нас, могут состоять из нескольких слов. Например «my\_ project\_package\_for\_...» и так далее, то есть это можно продолжать до бесконечности. Естественно, это проблема, потому что слишком длинная строка. Это очень неудобно писать. Это один из недостатков абсолютного импорта.

Но давайте отменим действия. У нас сейчас «module1», «module2», они внутри «package1».

**Что же такое относительный импорт?** Относительный импорт- это использование конструкции «from» с применением точки либо двоеточия (Рис.12).

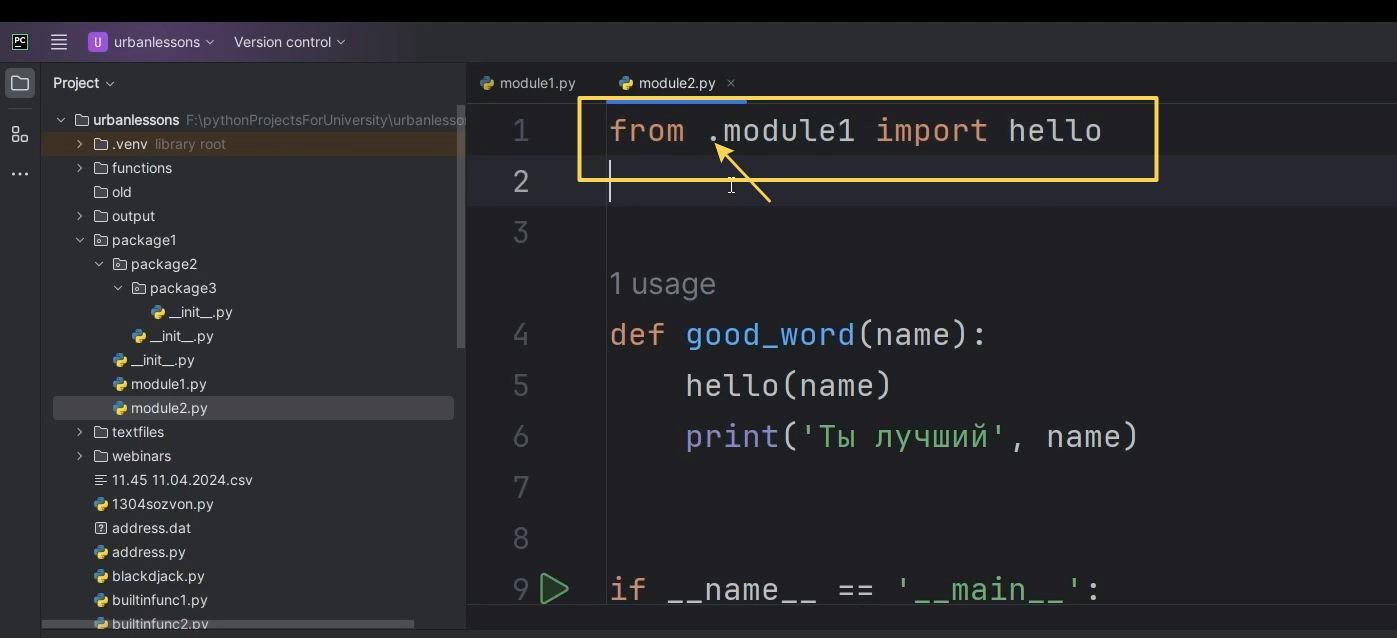


Рис.12

Что у нас обозначает точка? Точка обозначает, что мы берём текущий каталог. Из него, достаём «module1» и импортируем что-то. Двоеточие обозначает, что мы поднимаемся на каталог выше. В нашем случае они у нас находятся в одном каталоге, поэтому точка нам подходит для относительного импорта.

Если мы перемещаем эти два модуля в «package3», они у нас переместились, но ничего не сбилось, потому что здесь действует относительный импорт. Ему не важен путь до точки. Мы берём текущую директорию, главное, чтобы они находились в одном месте. В нашем случае мы используем точку и здесь название нашего модуля и все. При переносе мы видим то, что ничего для них не поменялось (Рис.13).

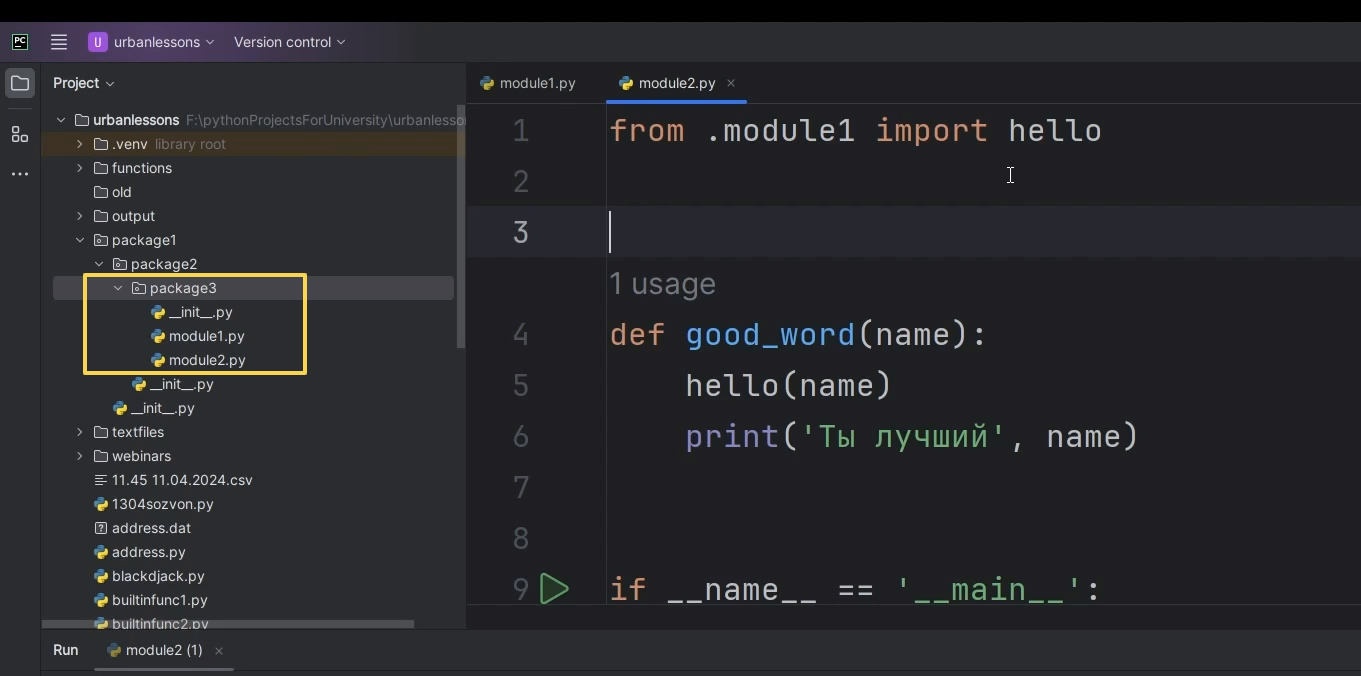


Рис.13

Однако мы хотим, чтобы это все дело работало, запускаем и видим вот ошибку (Рис.14).

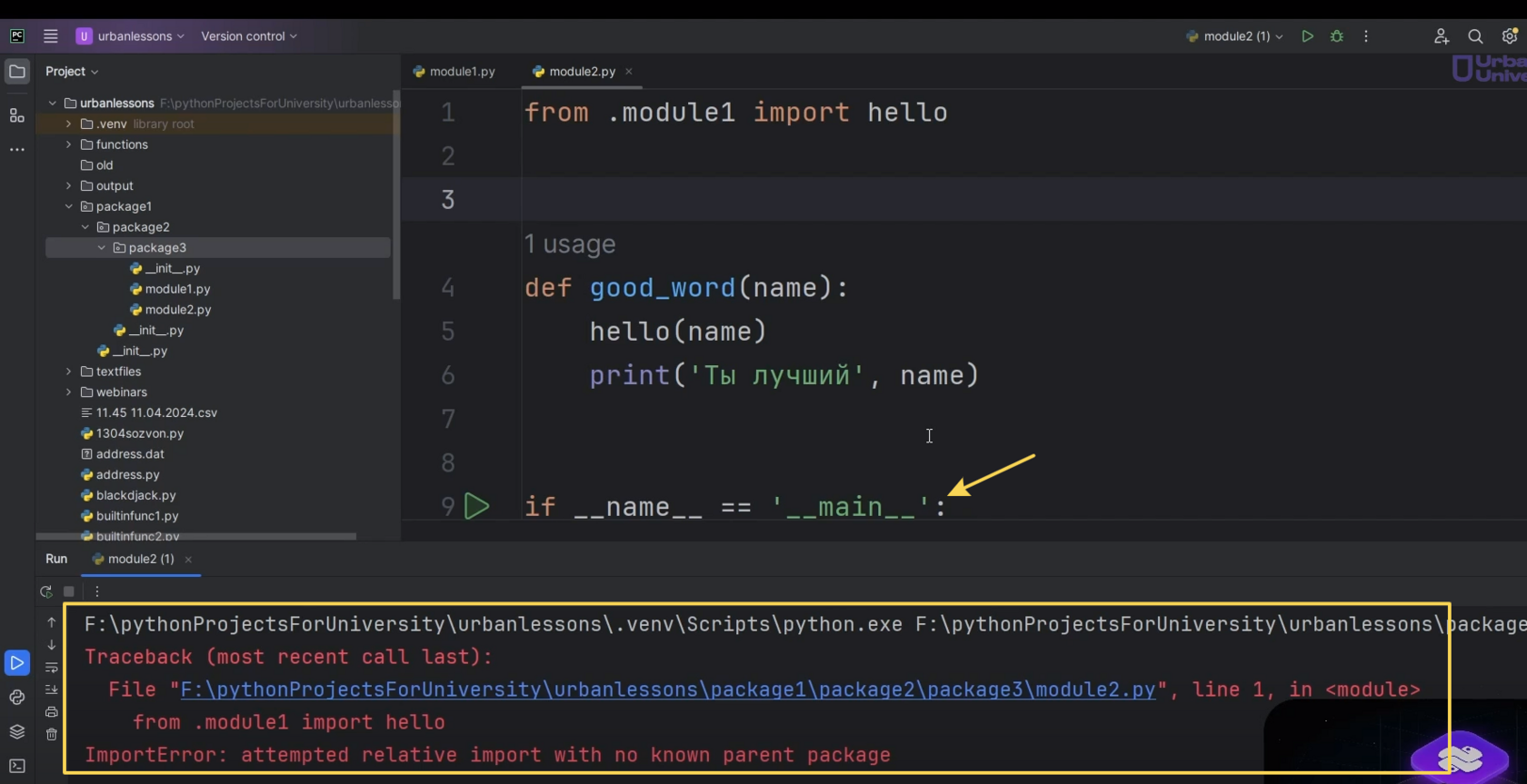


Рис.14

Эта ошибка говорит о том, что наш пакет запускается отсюда как «main». Он не видит банально, где он находится, какой у него родительский пакет. И он не может это дело выполнить.

Есть несколько способов для исправления данной проблемы. Один из них- добавление пакетов директорий в переменные среды, то есть в «path». Но это такой тяжёлый способ. Как правило, для решения таких проблем создают точку входа в программу. Многие из вас, наверное, когда создавали свои первые проекты, видели файл «main.py».

«Main.py» там находился неспроста. Как правило, в любом проекте есть какой-то файл, который находится прям в корневой директории и является входной точкой в нашу программу.

Давайте такой файл и создадим. Назовём его «main.py» (Рис.15,Рис.16).

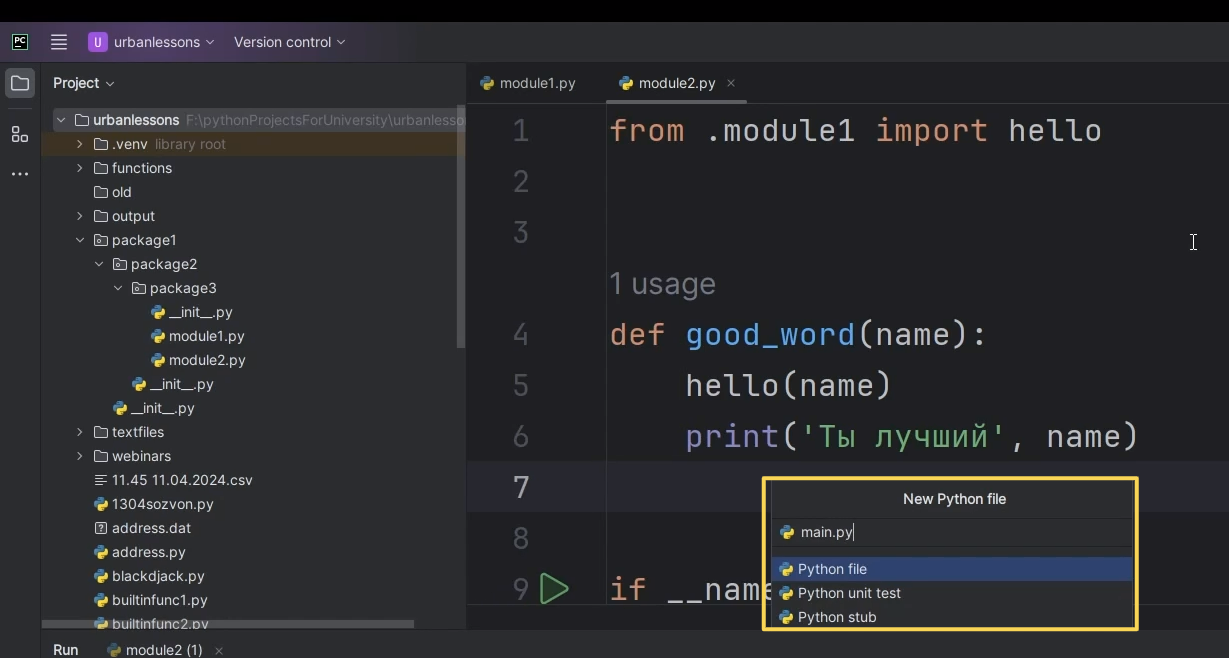


Рис.15

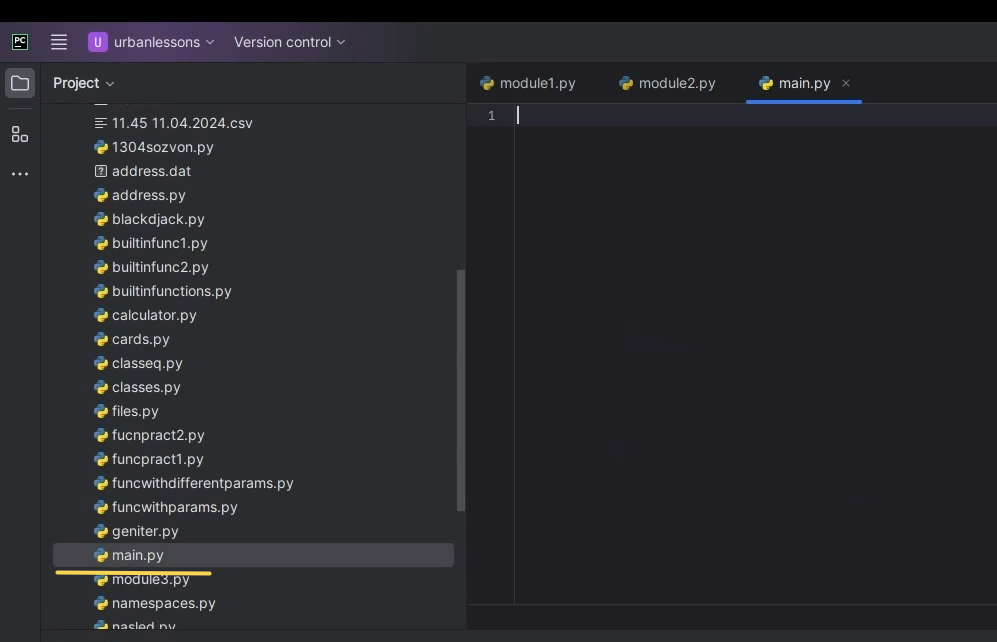


Рис.16

Что мы с вами сделаем? Мы укажем абсолютный путь. Нас интересует наша функция «good\_word». Значит, мы берём наш «package1», берём «package2», оттуда берём «package3», оттуда берём «module2» и оттуда импортируем вот эту функцию «good\_word» (Рис.17).

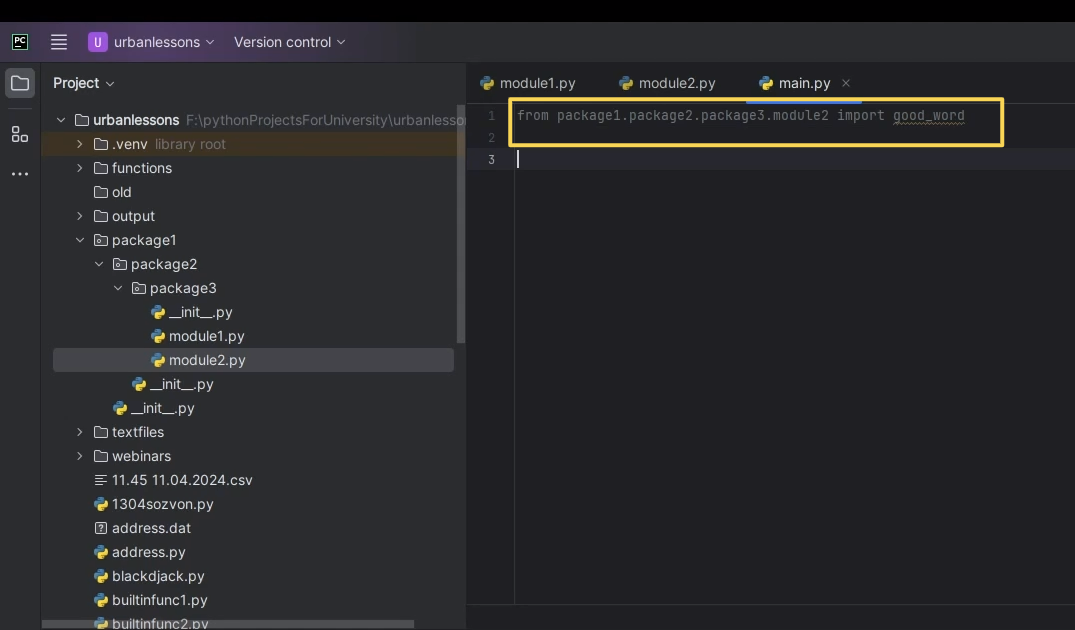


Рис.17

Далее создаём вот такую вот конструкцию «if \_\_name\_\_==\_\_main\_\_», и вызываем функцию «good\_word» и уже передаём в неё какое-то значение (Рис.18, Рис.19).

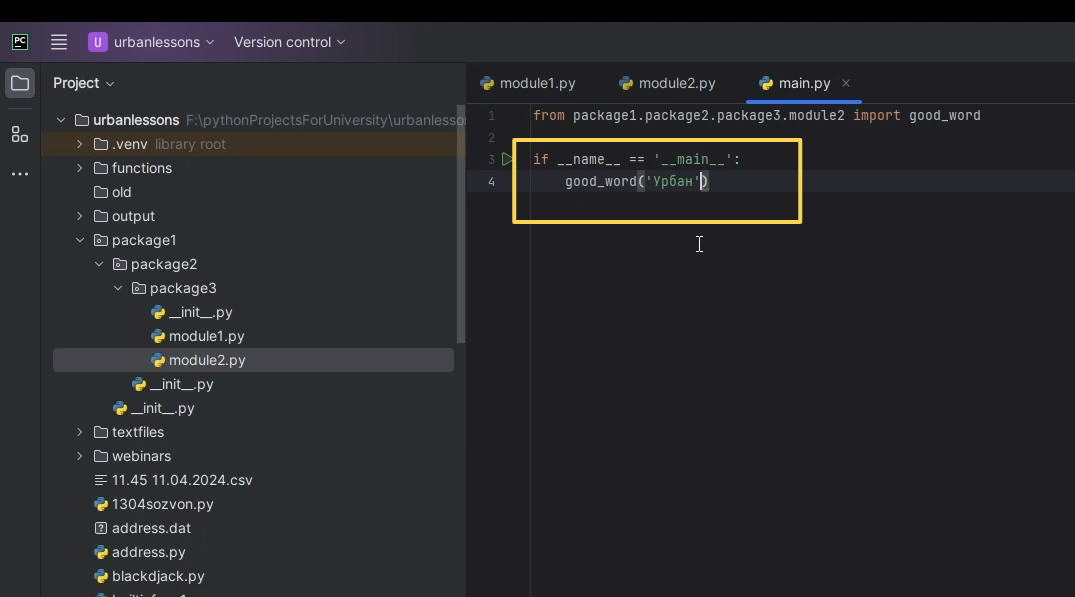


Рис.18



Рис.19

Что у нас происходит? Когда мы запускаем конструкцию, вот такую из основной входной точки в нашу программу, проблема с запуском относительных импортов у нас решается, потому что мы знаем, где наш проект. Мы явно здесь указываем путь. Мы как бы знаем все пути, то есть, естественно, это интерпретатор знает. Ему становится проще понять, где он находится. И проблема относительного импорта у нас пропадает. Однако нужно учитывать тот момент, что когда у нас есть запускающий файл «main.py», который в основной директории проекта, чтобы все работало корректно, мы всегда при любых обстоятельствах указываем абсолютный путь до нашего, соответственно, файла. То есть это полностью с указанием, откуда куда залезли, потом ещё раз куда залезли, ещё и что достали. То есть, если вы видите проблему, связанную вот с таким вот импортом «ImportError», где у нас не видна родительская директория, то подумайте внимательно над вашей структурой проекта (Рис.20). Вероятнее всего, вам необходимо создать какую-то входную точку программы, в вашу программу, в основную директорию и написать там логику, которая будет вызывать нужную вам функцию.

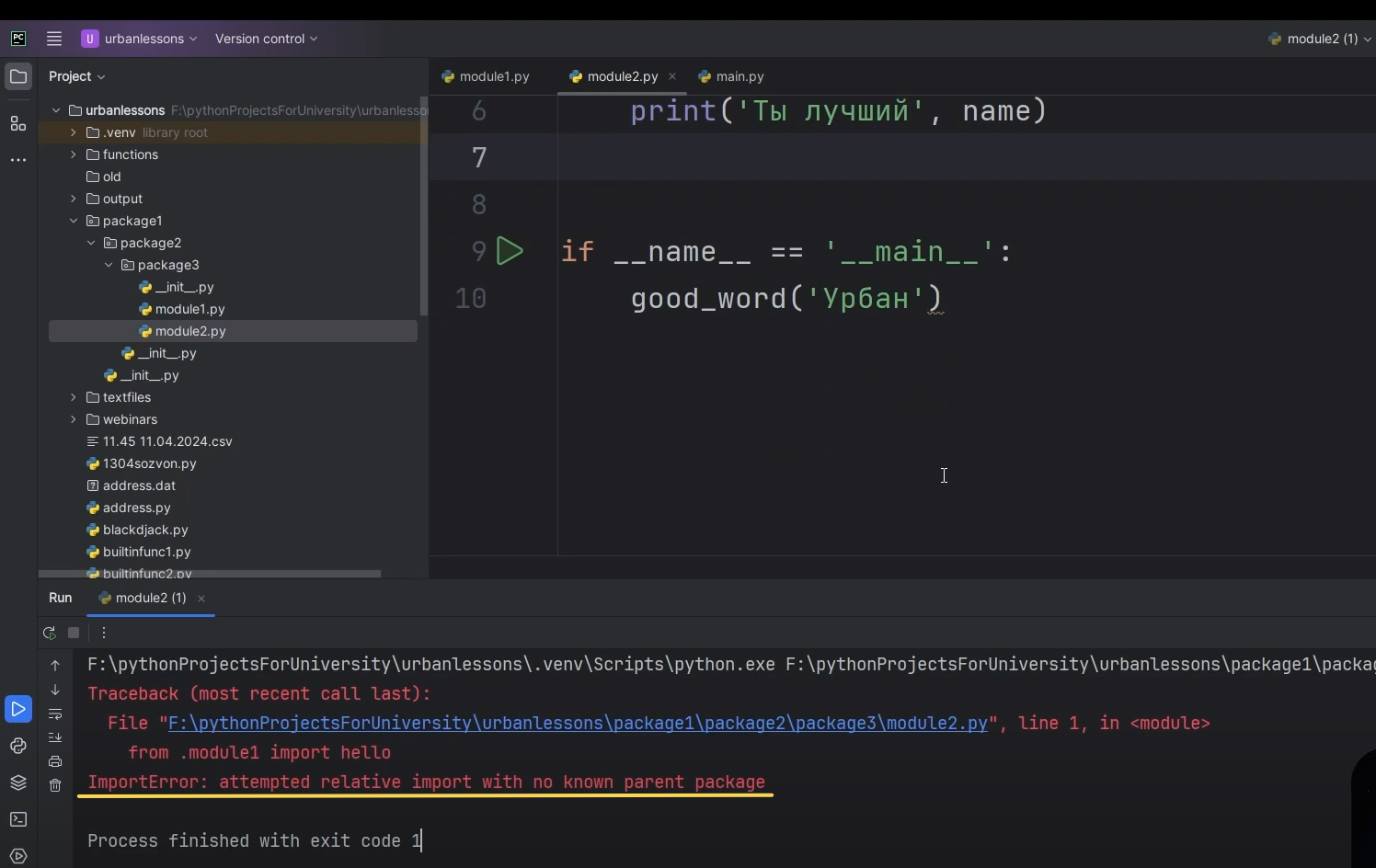


Рис.20

Что у нас там получается? Мы сначала импортируем функцию «good\_word». Указываем полный путь, итак, мы, получается, знаем, относительно чего мы ищем вот этот «module1» и импортируем вторую функцию «hello».

Для тренировки вам рекомендую поэкспериментировать. Попробуйте создать такую же структуру, как у нас, перемещаться между пакетами. Например, взять «module1», переместить не в «package3», а в «package2»- на уровень выше и проверить, как это будет работать. У нас импорт сам заменился (Рис.21).

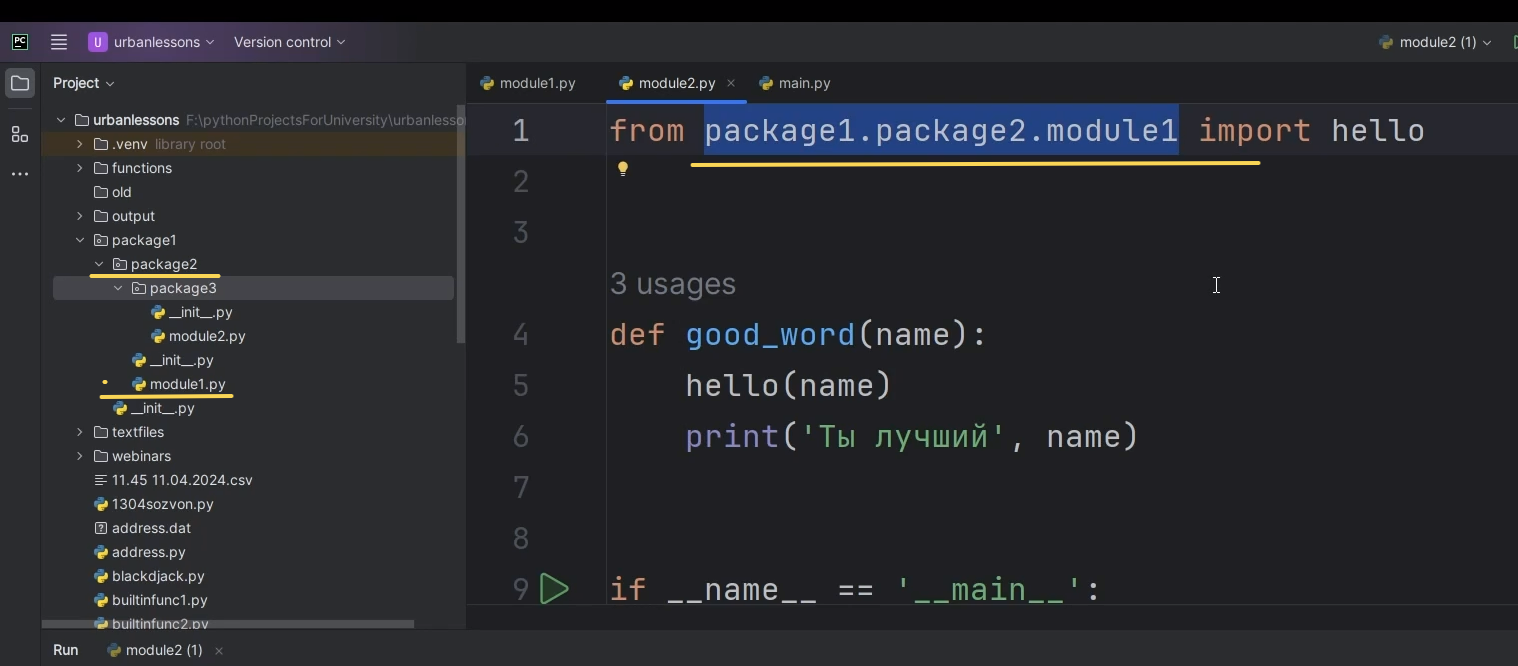


Рис.21

Но вы попробуйте сделать относительный импорт и это запустить.

В принципе, это все, что у нас касается работы с пакетами. Будьте внимательны! Желаем всем успехов!