**2023/10/12 00:00|Лекция. Практика по функциям**

## Практика по функциям

На сегодняшнем занятии мы проведём небольшую практику, посвящённую функциям. **Нашей задачей будет написать три функции:**

1. Первая - максимум в списке, то есть функцию, которая будет находить максимальный элемент в списке;
2. Вторая будет считать количество чётных чисел в переданном списке;
3. Третья функция будет возвращать нам уникальный список.

Эти задачи можно решить с помощью встроенных функций, с которыми мы познакомились ранее, но наша задача попрактиковаться и решить, то есть написать данные функции без использования встроенных.

#### Начнем с первой задачи — это максимум в списке.

Создадим функцию, назовём её find\_max()(рис.1). Она будет принимать какой-то список list и данная функция должна найти максимальный элемент. Для начала определимся, что для перебора элементов списка мы будем использовать цикл for. Прежде чем мы будем использовать цикл for, мы создадим переменную, которая будет содержать какое-либо значение. Назовём эту переменную max и можем задать ей значение, например 0(рис.1). Нам нужно любое значение, но брать лучше какое-либо минимальное. Если в списке будет находиться значение меньше, то есть все элементы списка будут содержать значение меньше, чем в переменной max, тогда функция отработает некорректно. Чтобы этого избежать, мы можем взять любой элемент из переданного списка. Возьмём самый первый(рис.2).

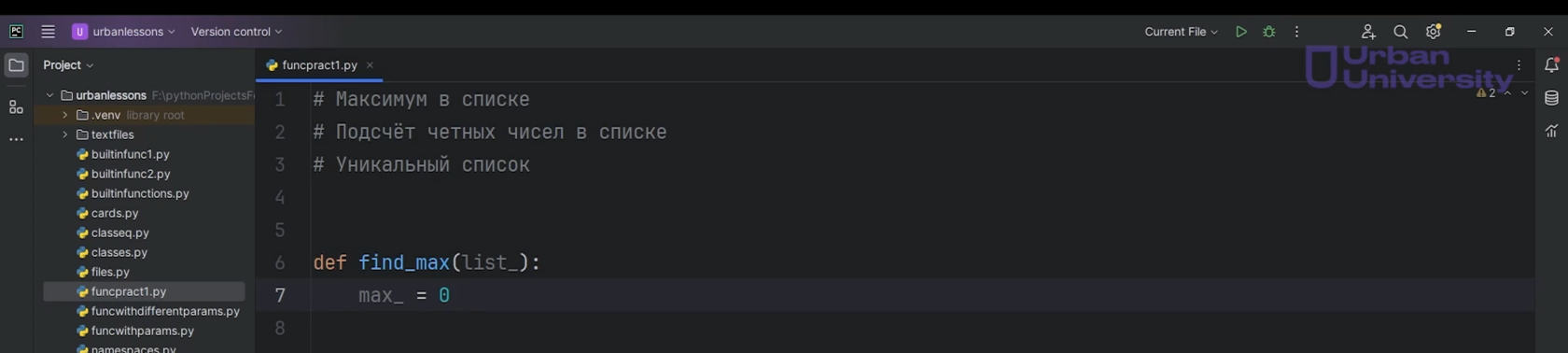


Рис.1

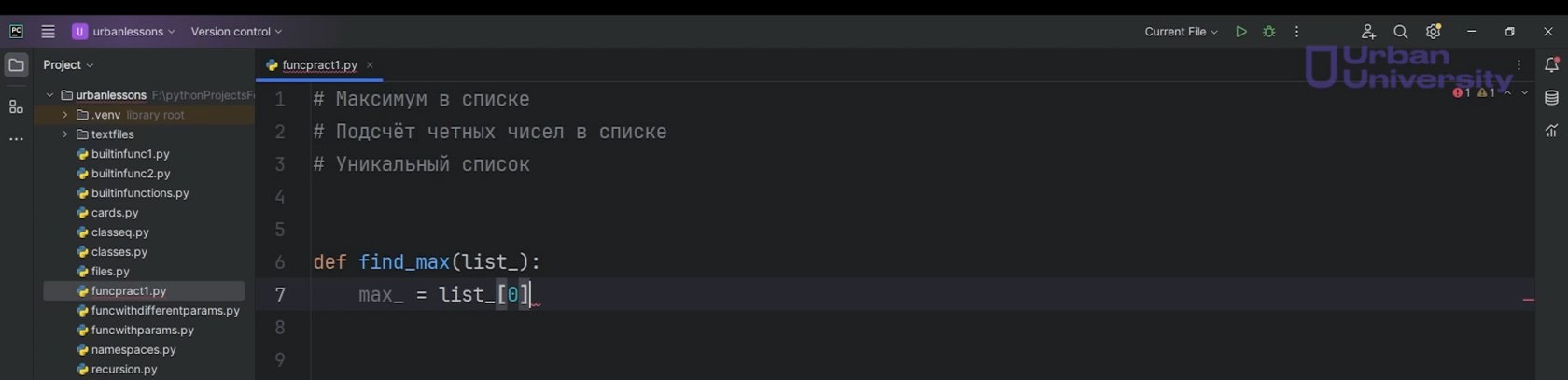


Рис.2

Затем циклом for мы пробежимся по данному списку. Далее мы будем проверять, что если элемент из этого списка больше, чем элемент max, то мы перезаписываем эту переменную max на данный элемент. Это будет происходить до тех пор, пока элементы в списке не закончатся. После цикла мы уже вернём данный элемент(рис.3).

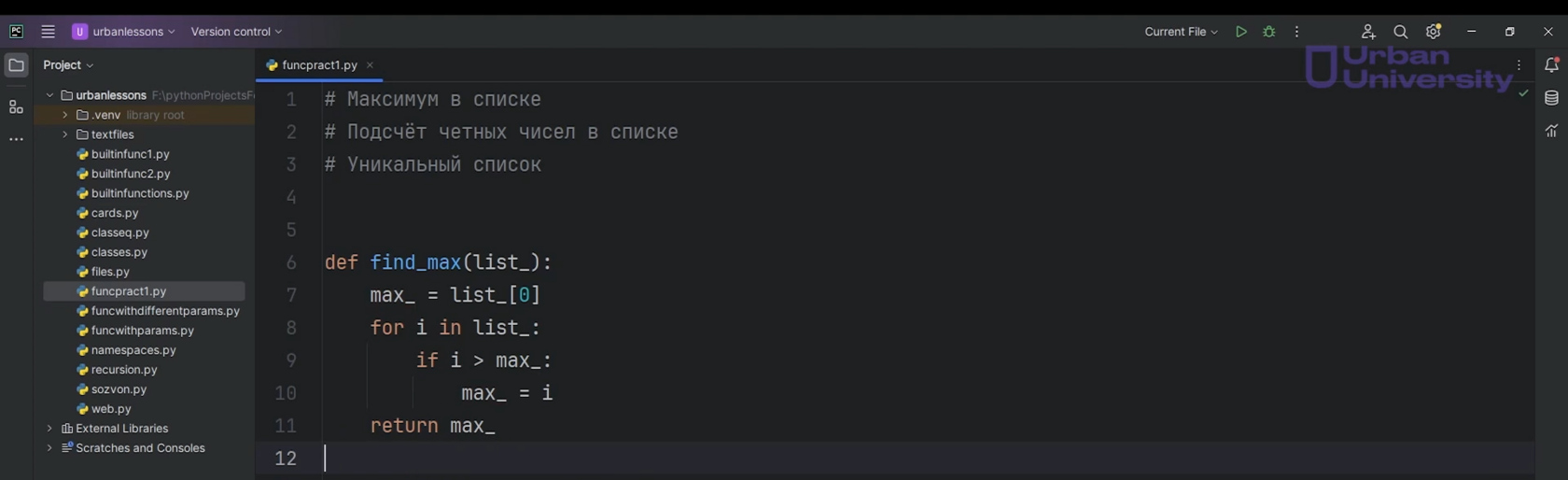


Рис.3

Давайте проверим, как она работает. Вызовем нашу функцию find\_max() и передадим сюда список из абсолютно случайных значений(1, 54, 12, -1, 2, 4). При запуске получаем результат 54(рис.4).

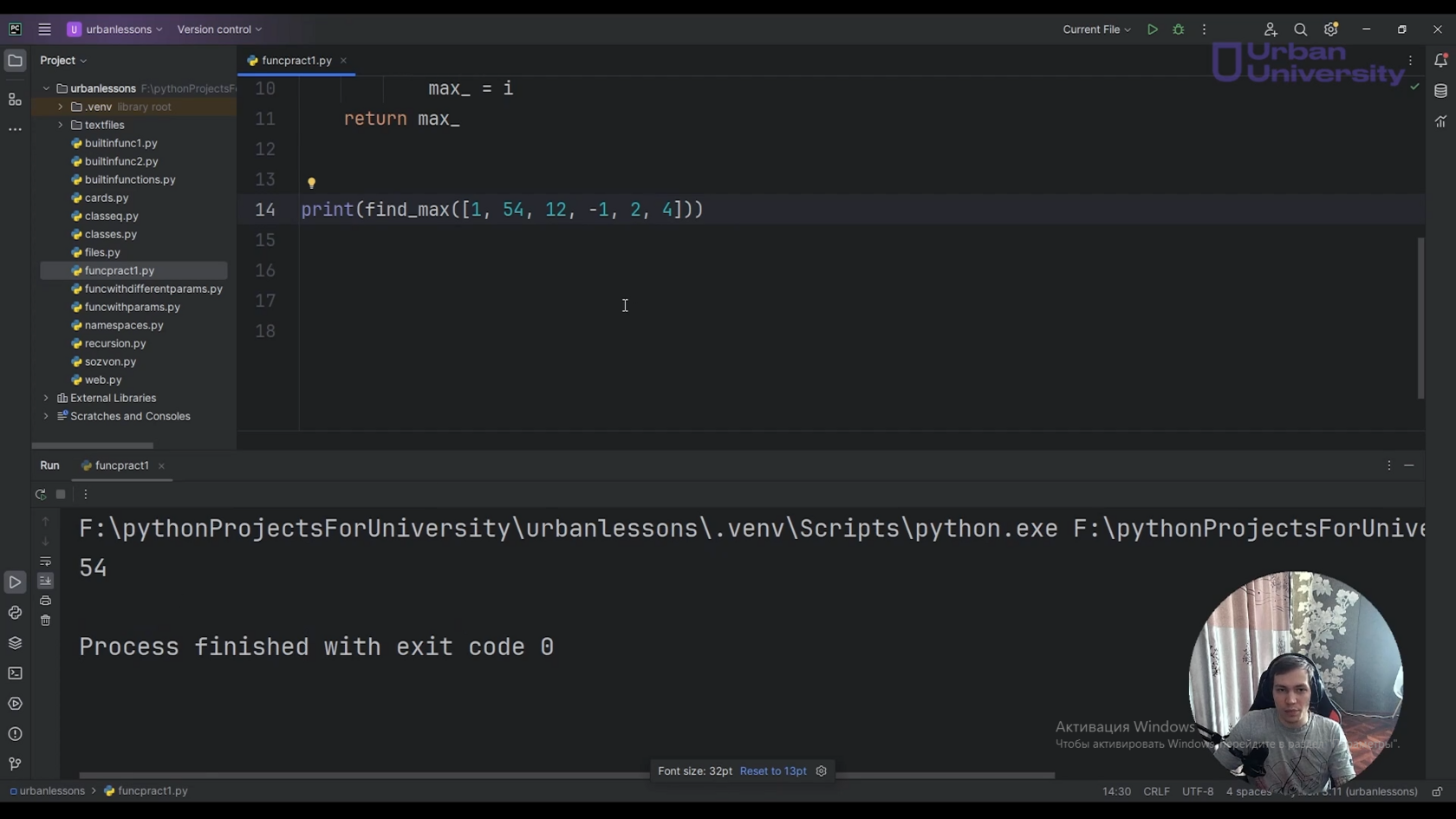


Рис.4

Также из других значений(1, 1, 1, 1, 0) получаем 1(рис.5). Поставили 2 среди значений, получили 2(рис.6), поставили 5, получили 5(рис.7). Всё работает.

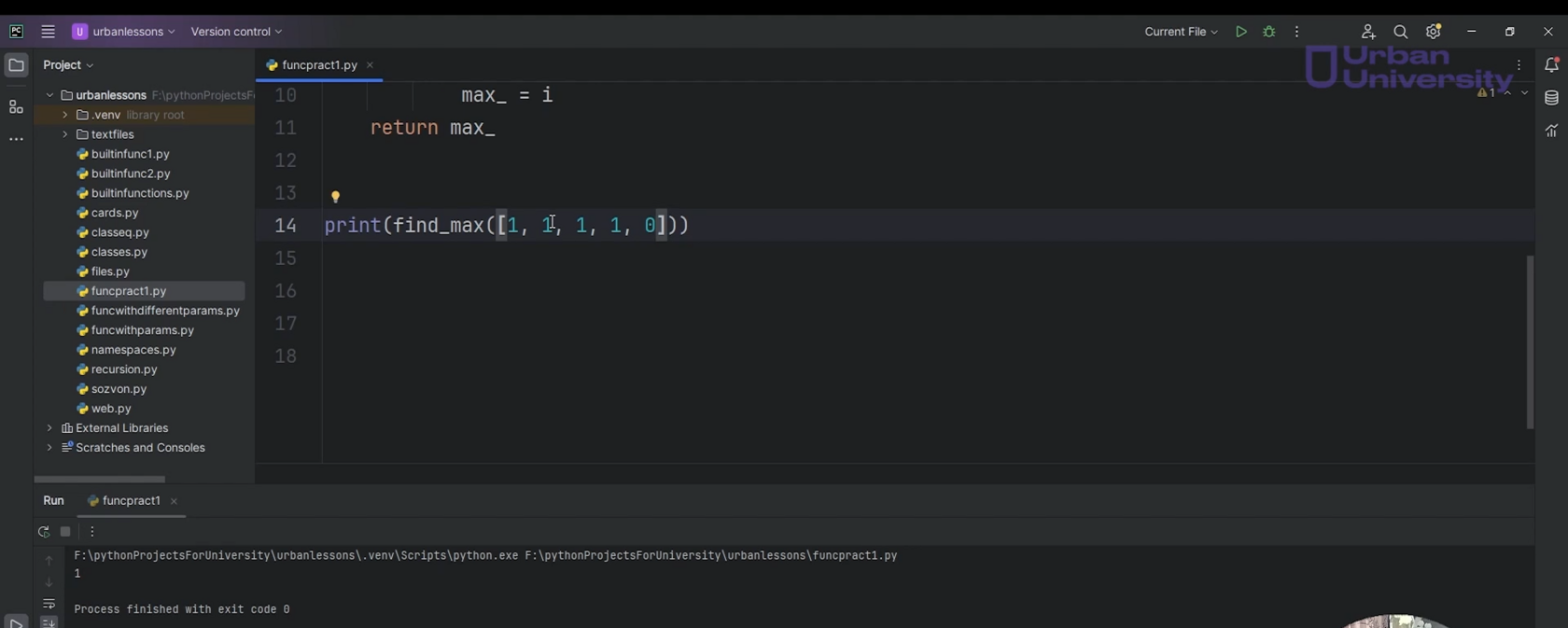


Рис.5



Рис.6

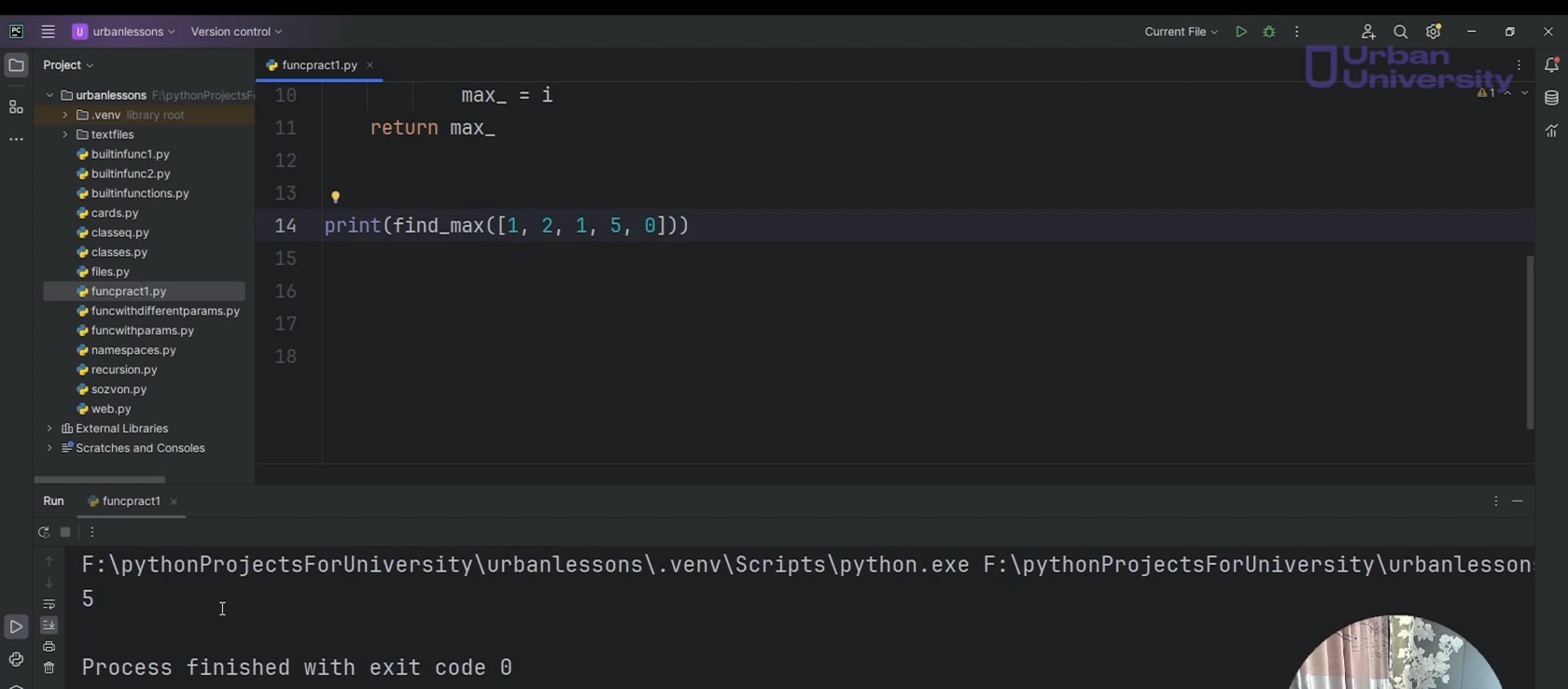


Рис.7

**Задумка такая:** мы просто взяли какой-то элемент из списка, пробежались по этому списку, по каждому элементу и каждый элемент сравнивали с элементом в нашем списке. Если он оказывался больше, мы его перезаписывали. Соответственно так мы пробежимся по каждому элементу и каждый элемент сравним со значением max. В конечном счёте, после того как цикл завершит свою работу, мы получим желаемый результат. Это на самом деле не самая сложная задача.

#### Следующая функция будет по подсчету четных чисел в списке.

Назовем ее count\_even()(рис.8), она тоже будет принимать какой-либо список.

**Здесь задумка такая:** нам нужно что-то считать, значит, нам понадобится переменная, которая будет хранить этот самый счётчик. Данную переменную назовём counter, и изначально он равен 0(рис.8), потому что изначально в списке может быть 0 чётных чисел.

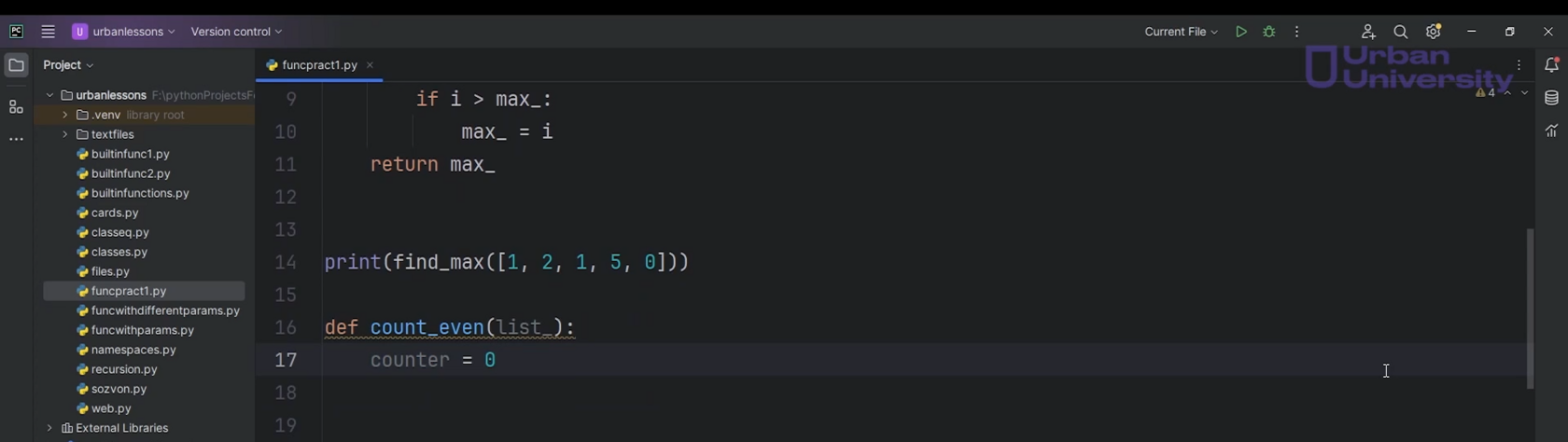


Рис.8

Затем мы будем продвигаться по каждому элементу из переданного списка. Наша задача проверять, будет ли этот элемент являться чётным числом, то есть мы добавим условие, в котором будем брать этот элемент и проверять остаток от деления на 2. Если остаток от деления на 2 будет равен 0, то мы будем увеличивать наш счётчик на 1. После того как наш цикл отработает, мы вернём этот счётчик(рис.9).

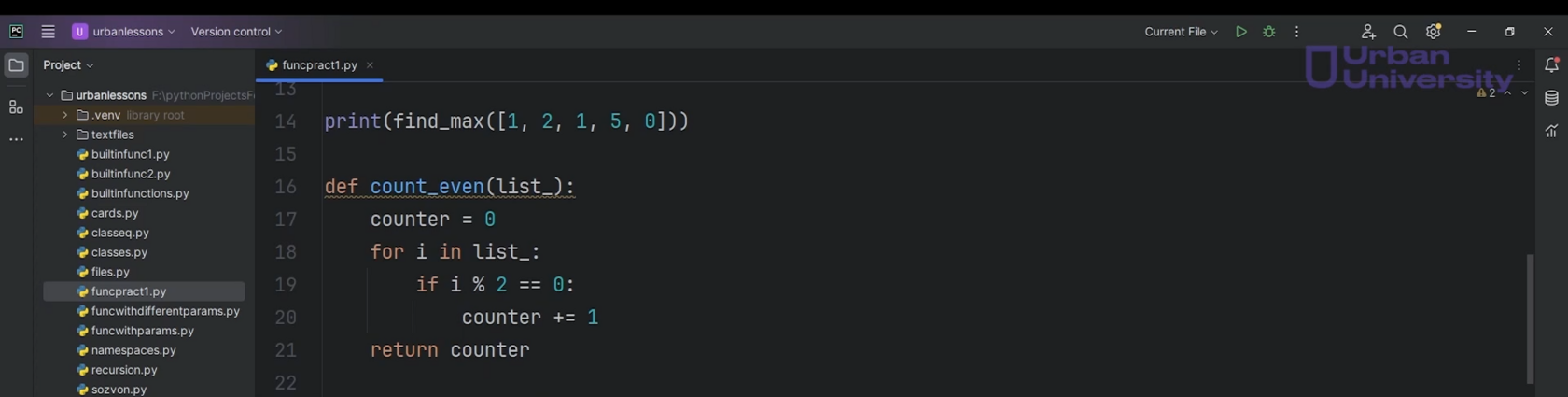


Рис.9

Посмотрим, как это будет работать. Вызываем функцию, передаем значения(2, 2, 3, 4, 2, 1, 0) и получаем 5(рис.10).

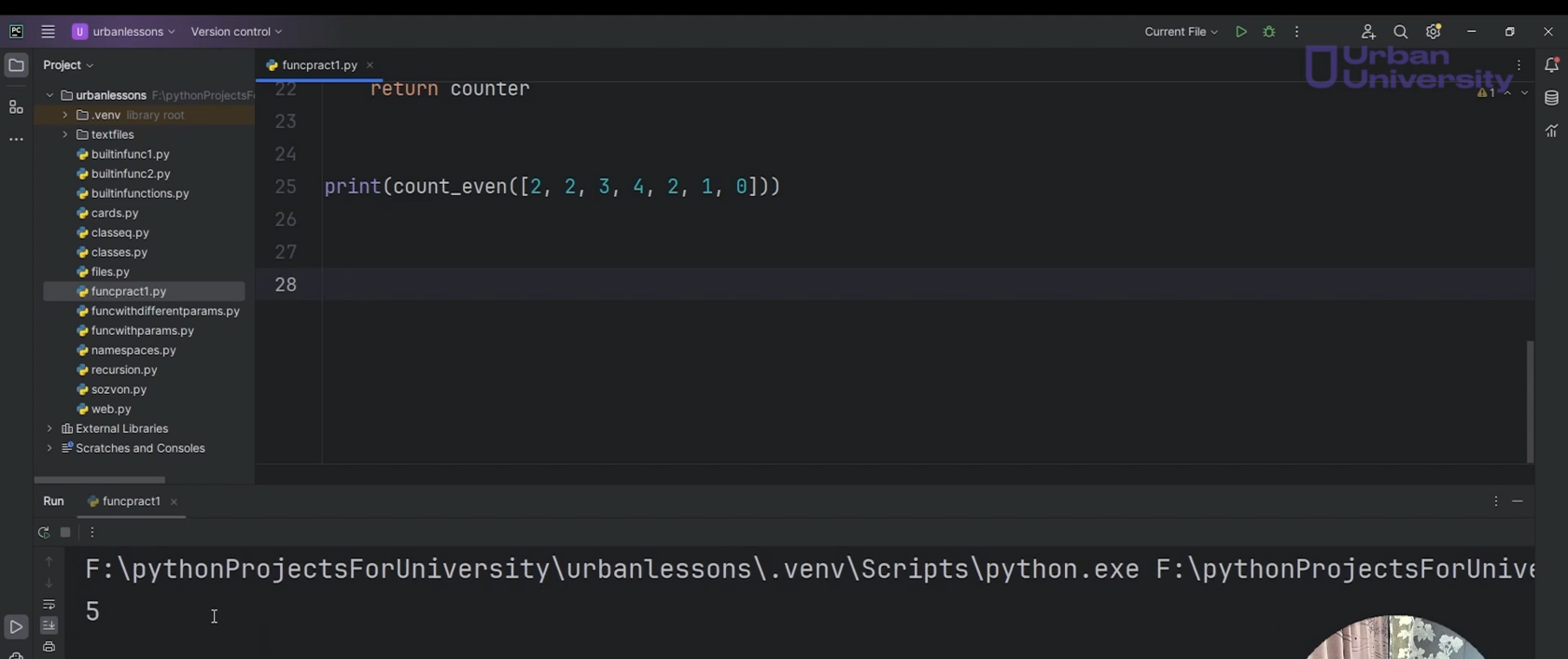


Рис.10

2, 2, 4, 2 - четные и ноль тоже считается чётным числом. Его можно убрать и получим 4(рис.11).

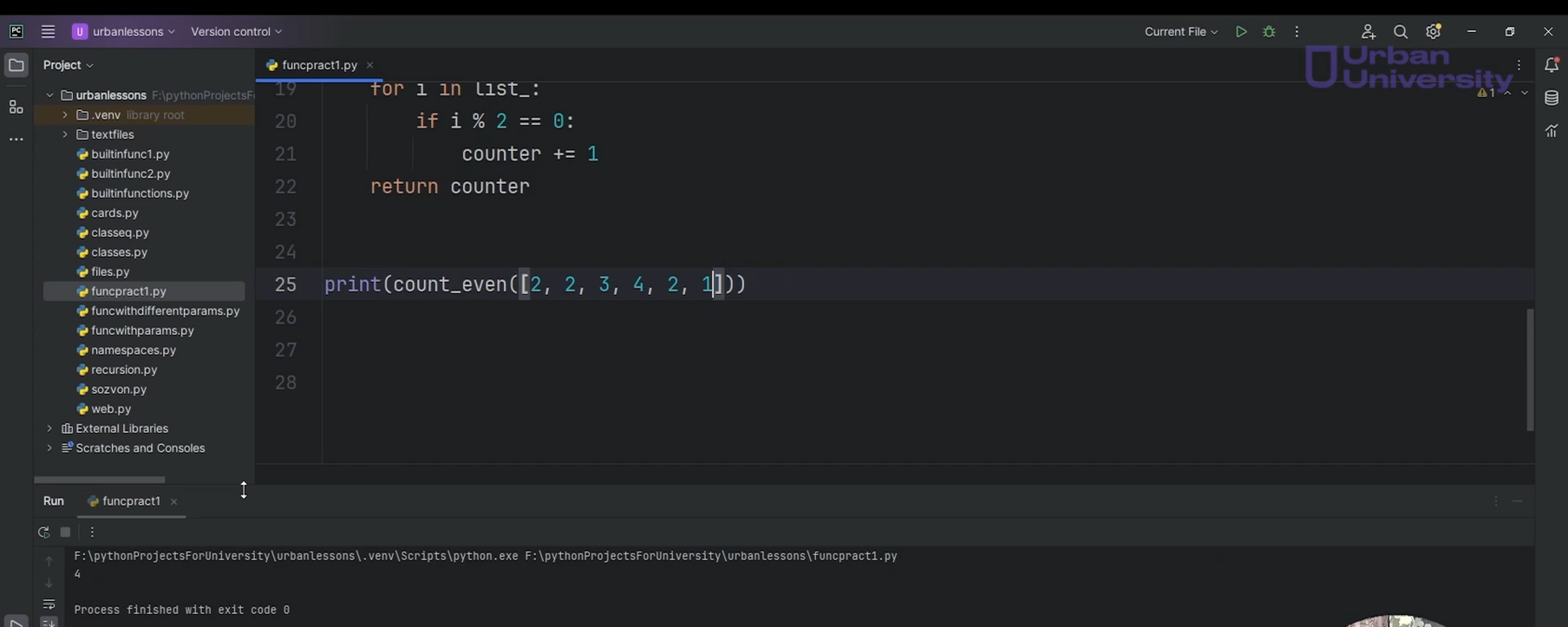


Рис.11

При необходимости мы можем добавить условие в цикле for: если i = 0, то continue(рис.12).

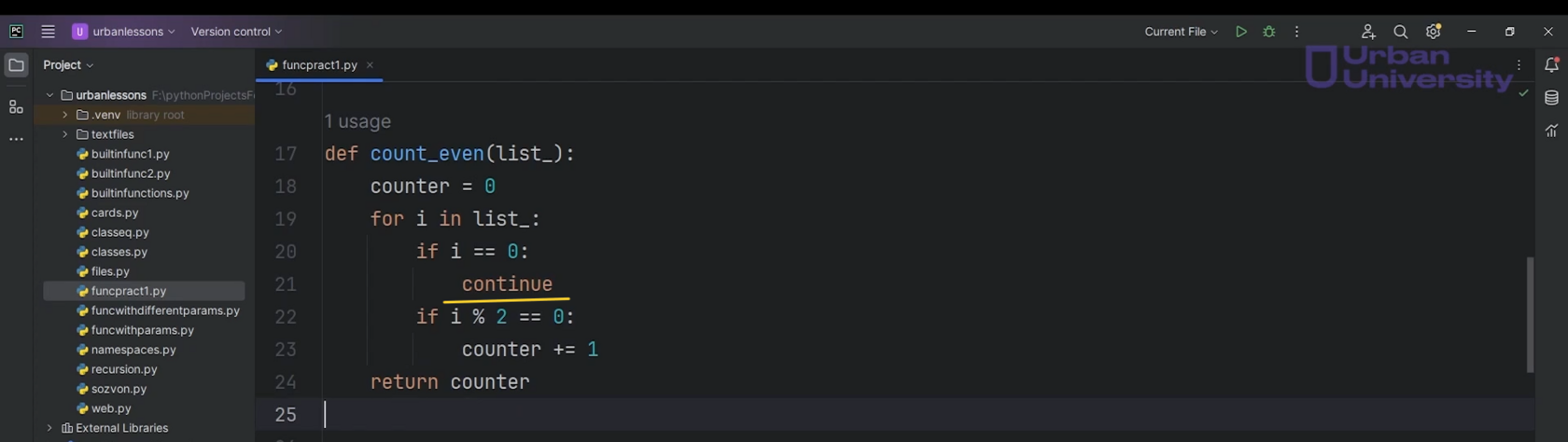


Рис.12

При запуске получим тот же самый результат. Здесь снова 0 не входит в расчёты(рис.13).

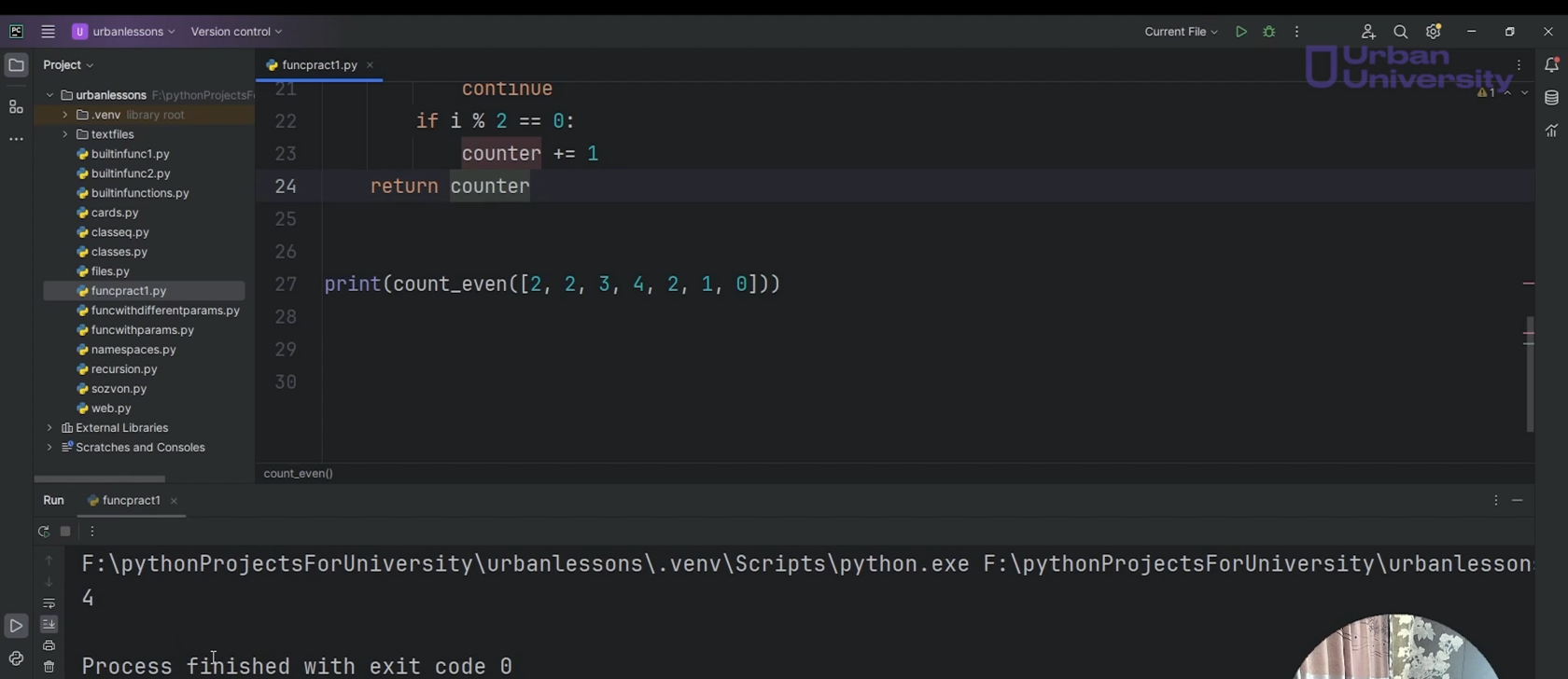


Рис.13

Можно сделать таким образом. Можно также написать pass и elif. Ha самом деле однозначного способа решения для этих задач нет. Их можно решать множеством способов. То, что мы пишем сейчас это всего лишь один из десятка возможных вариантов решения. В этом-то и вся прелесть программирования. То, что для решения одной задачи не всегда есть одно конкретное решение. В принципе, эти две функции - они достаточно простые.

#### Третье — уникальный список.

Давайте посмотрим. Сделаем функцию unique()(рис.14), которая будет точно также принимать список.

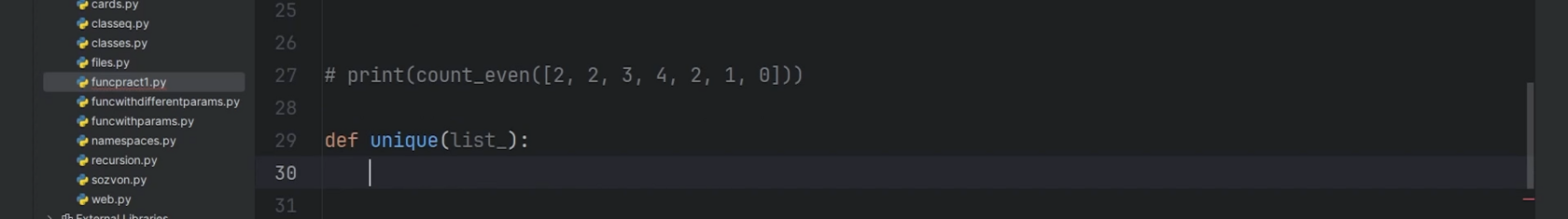


Рис.14

Здесь особенность в том, что когда мы решаем задачи, где нам нужно из исходных данных что-то вернуть, что-то получить, желательно создавать дополнительные хранилище, то есть создавать новые переменные, чтобы не трогать исходные данные. Поэтому мы создадим переменную new\_list с новым списком. Затем с помощью цикла for, мы пробежимся по переданному списку(рис.15).

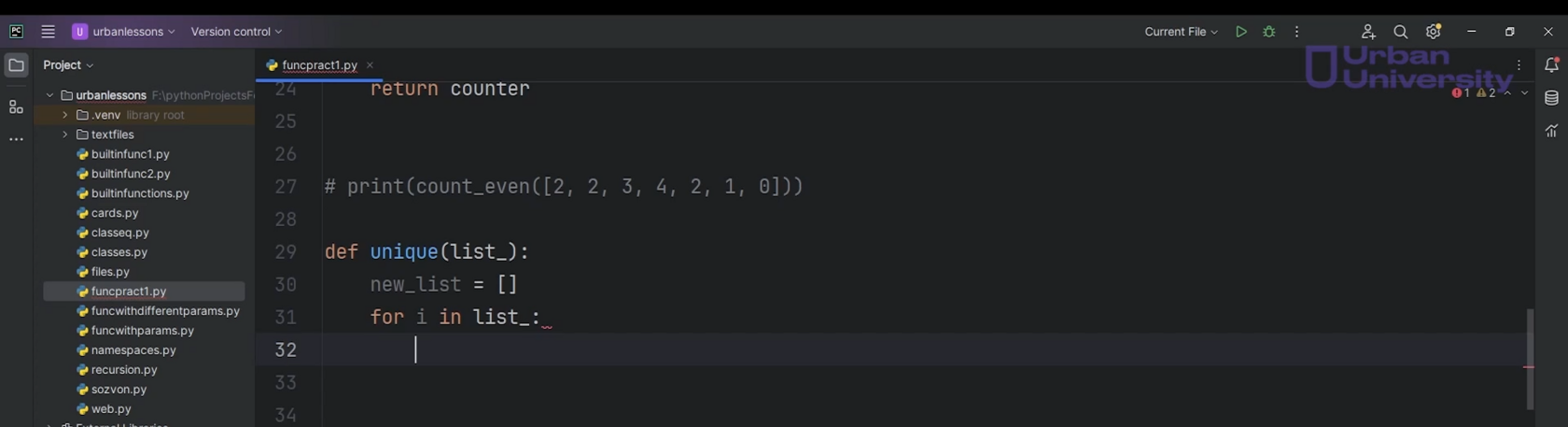


Рис.15

**Тут задумка очень простая:** мы будем пробегать по каждому элементу списка и проверять есть ли этот элемент в новом списке. Если в новом списке его нет, то мы будем добавлять. Если он есть, то мы с ним ничего не будем делать. Если элемент i не в новом списке, тогда мы берём новый список new\_list.append(i), добавляем туда этот элемент. После того как данный цикл завершит свою работу, мы вернём новый список(рис.16).

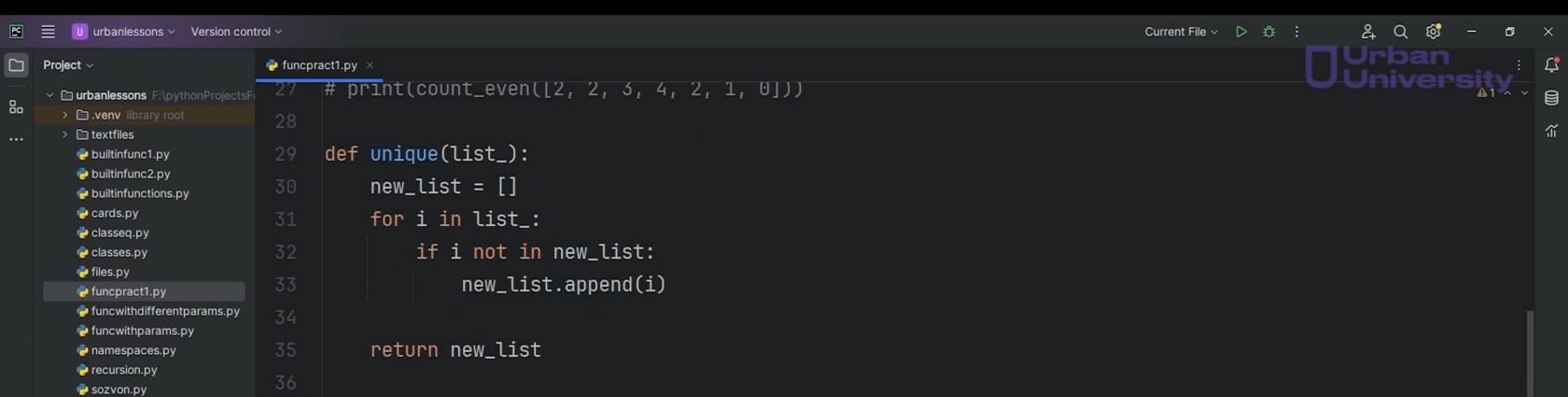


Рис.16

Давайте проверим. Вызовем функцию unique() и здесь передадим список с элементами от 1 до 8 два раза(рис.17). То есть, по сути, мы должны увидеть в ответе только один список, который будет содержать элементы: 1, 2, 3, 4, 5, 6, 7, 8.

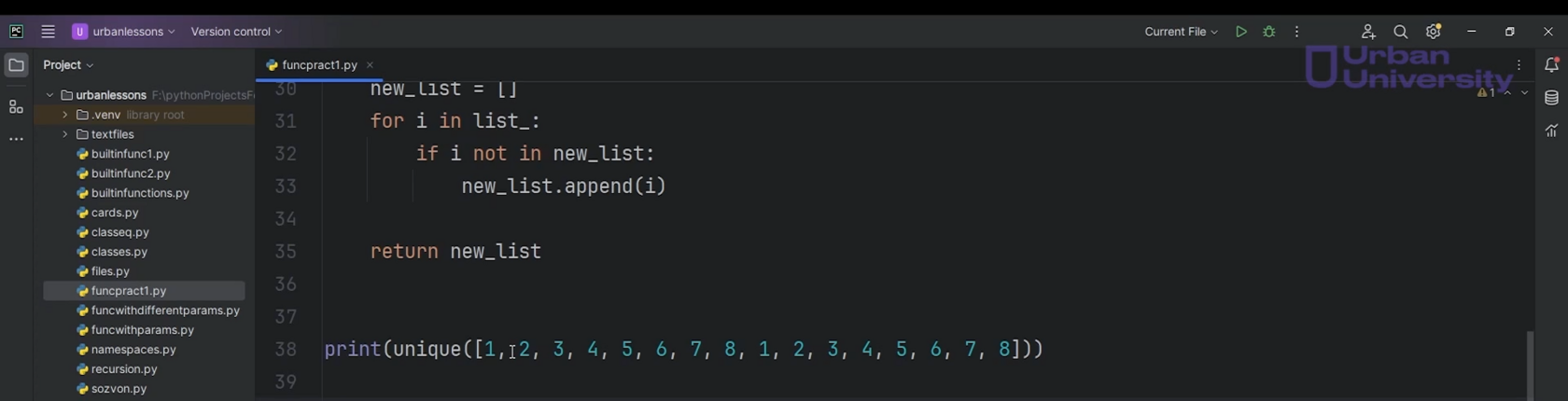


Рис.17

Запускаем, видим новый список(рис.18). Работает. Это можно было бы решить с помощью перевода во множество и потом множество переделать обратно в список, но наша задача - набить руку и потренироваться.

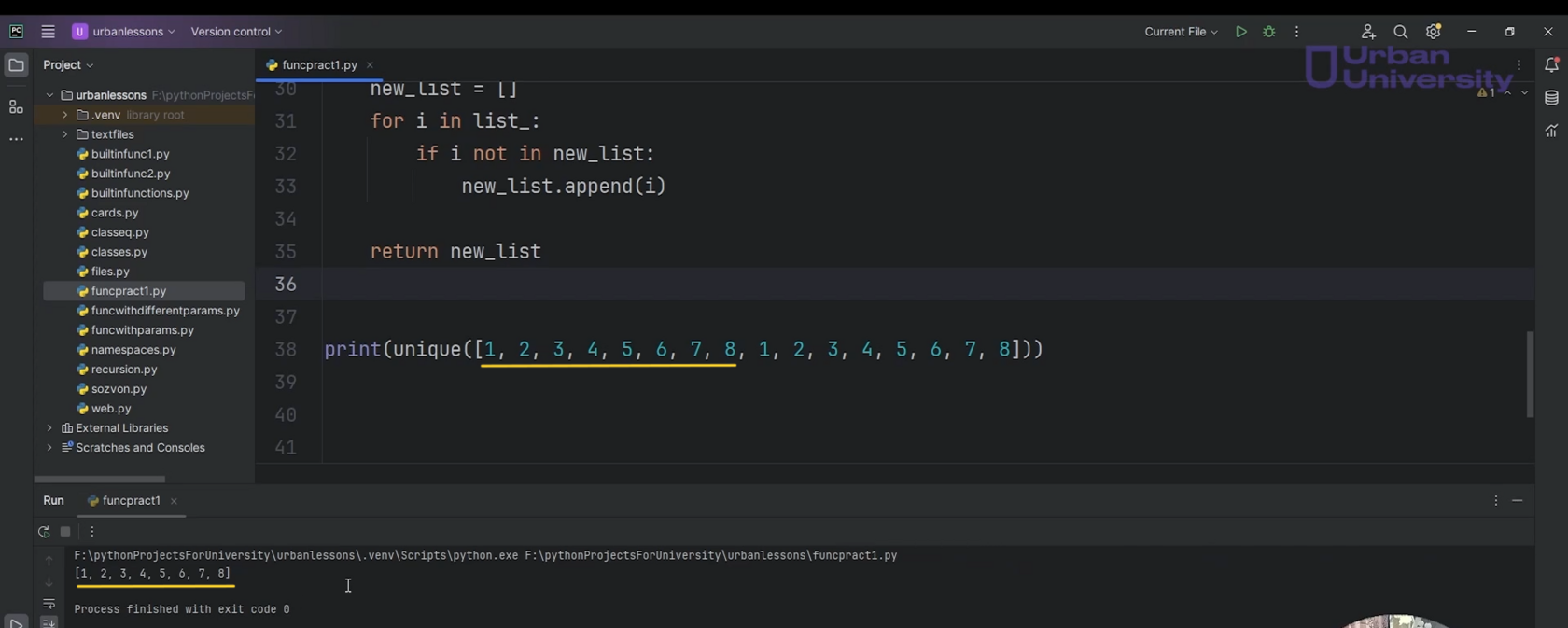


Рис.18

**Работать с базовыми конструкциями очень важно, и навыки работы с ними сильно пригодятся в будущем при решении сложных задач.**