**2023/11/13 00:00|Лекция. Позиционирование в файле.**

**Позиционирование в файле**

Мы продолжаем работать с файлами, и на сегодняшнем занятии речь пойдёт про позиционирование. Сверху вы могли уже увидеть то, что появился дополнительный импорт «io» (Рис.1). Он нам пригодится немного позднее.

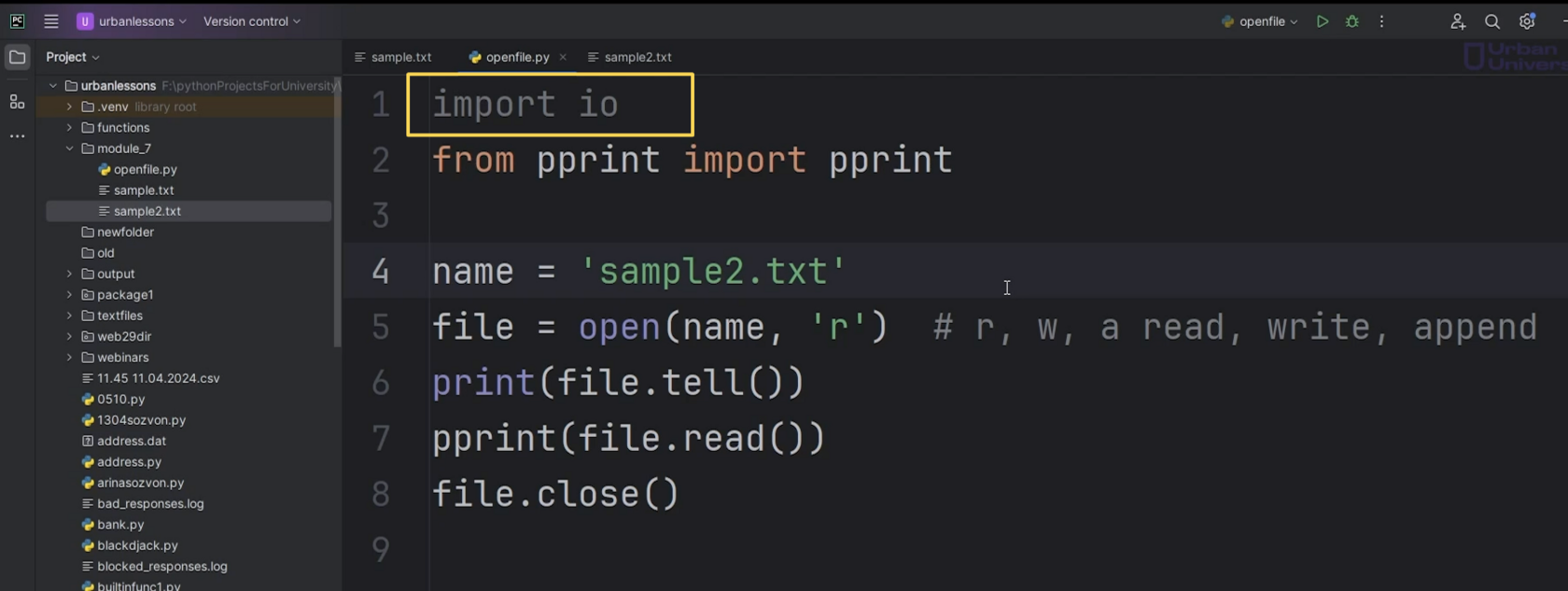


Рис.1

А сейчас давайте вспомним слова, которые мы озвучивали на предыдущем занятии. Мы говорили, что когда работаем с файлами, работаем с неким невидимым курсором. Если хотим считать файл, когда его открываем, курсор появляется в самом начале на первом символе. Однако в процессе работы с файлом он у нас перемещается. Метод «tell» позволяет получить позицию этого курсора, «read» считывает полностью информацию нашего текстового файла. Запускаем, видим, что курсор был в начале, вывели строку, курсор передвинулся на позицию 13 (Рис.2).

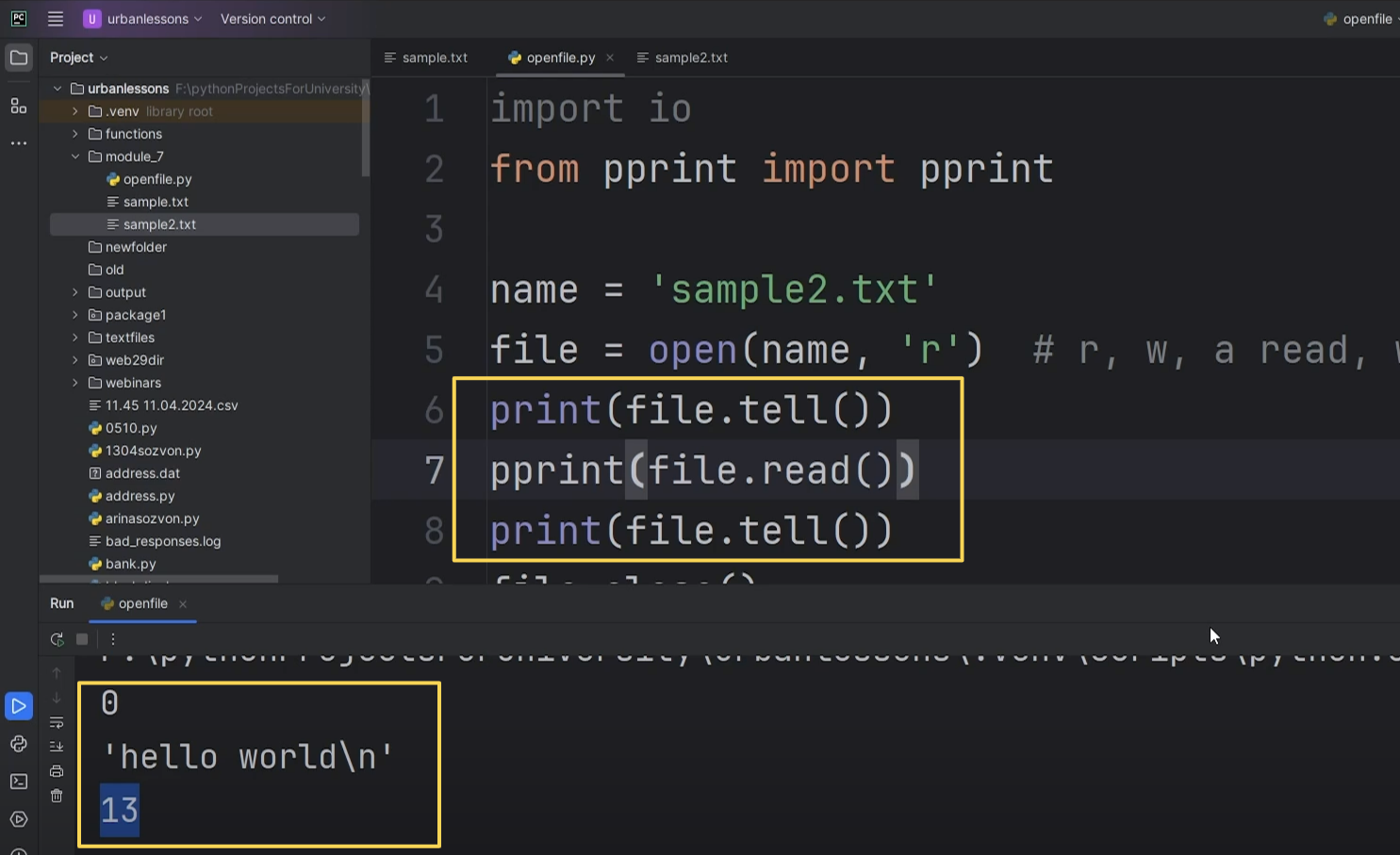


Рис.2

Однако, когда мы работаем с чтением, мы работаем с символами. Если представить наш текстовый документ как большую строку, мы, грубо говоря, считаем с такого-то по такой-то символ. Однако, когда мы работаем с вот этим невидимым курсором, этот невидимый курсор двигается по байтам. И работать с ним надо в байтах. То есть указывать значение передвинуться на такой-то байт. Это не всегда удобно. Часто можно встретить различия для человека, который с этим не знаком. Он может вывести, например, содержимое файла, посмотреть, где находится его курсор, и увидеть, что значение положения курсора не совпадает с количеством символов в текстовом документе. Либо же при попытке передвинуть у него может возникнуть такая ошибка. Если мы сейчас хотим задать новое положение, должны воспользоваться методом «seek» и задать этот самый сдвиг. И здесь я скажу, что хочу сдвинуть файл на 14, то есть передвинуть курсор на 14 символ (Рис.3). Оно сработало.

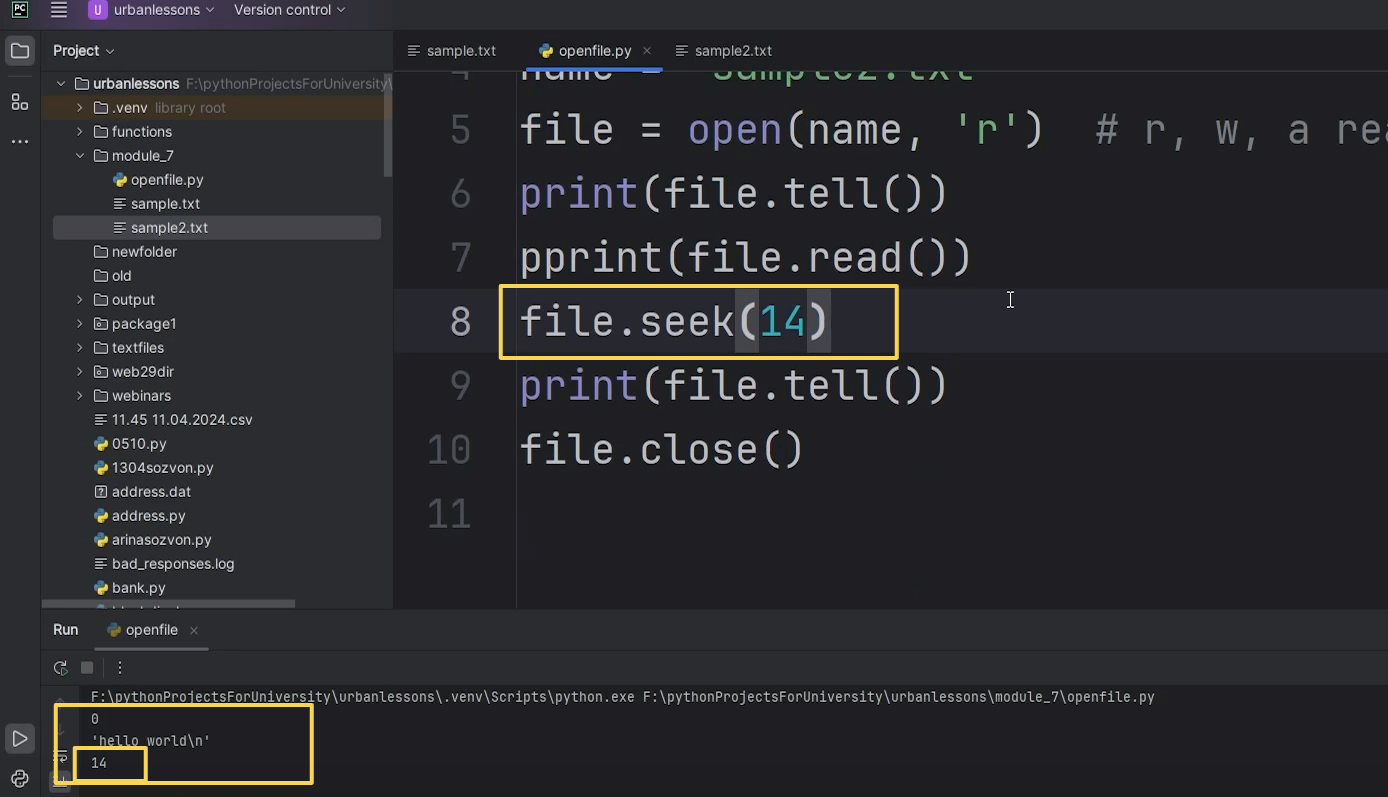


Рис.3

Если я хочу передвинуть курсор на 15, оно сработало. (Рис.4).

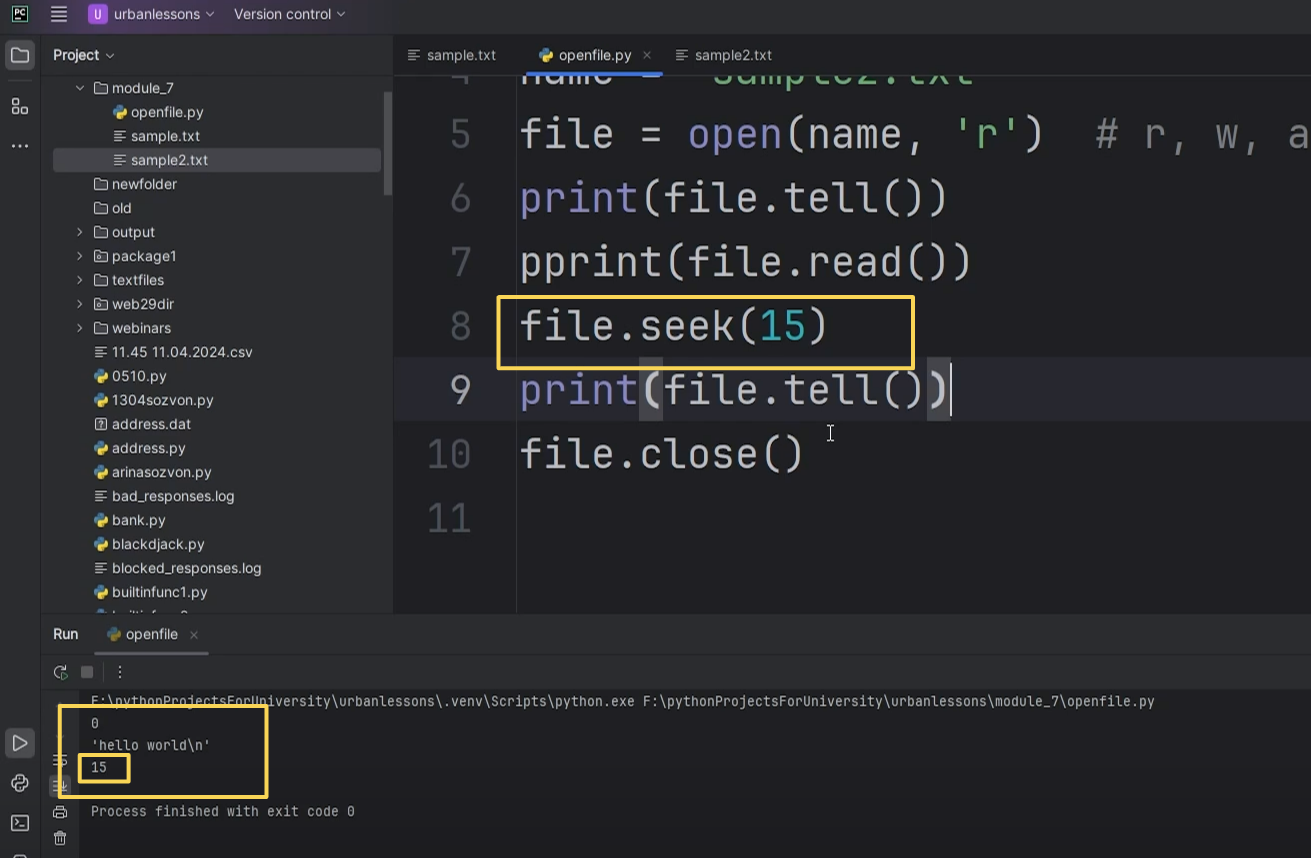


Рис.4

Дальше просто нет содержимого, но курсор все равно двигается. Давайте попробуем что-нибудь вставить из текста. Воспользуемся режимом открытия «а». После того как мы передвинули курсор на 15 символ, давайте что-нибудь запишем в наш файл. Воспользуемся методом «write» и напишем «new text». Запускаем (Рис.5). Видим, что у нас курсор передвинулся на 21 символ.

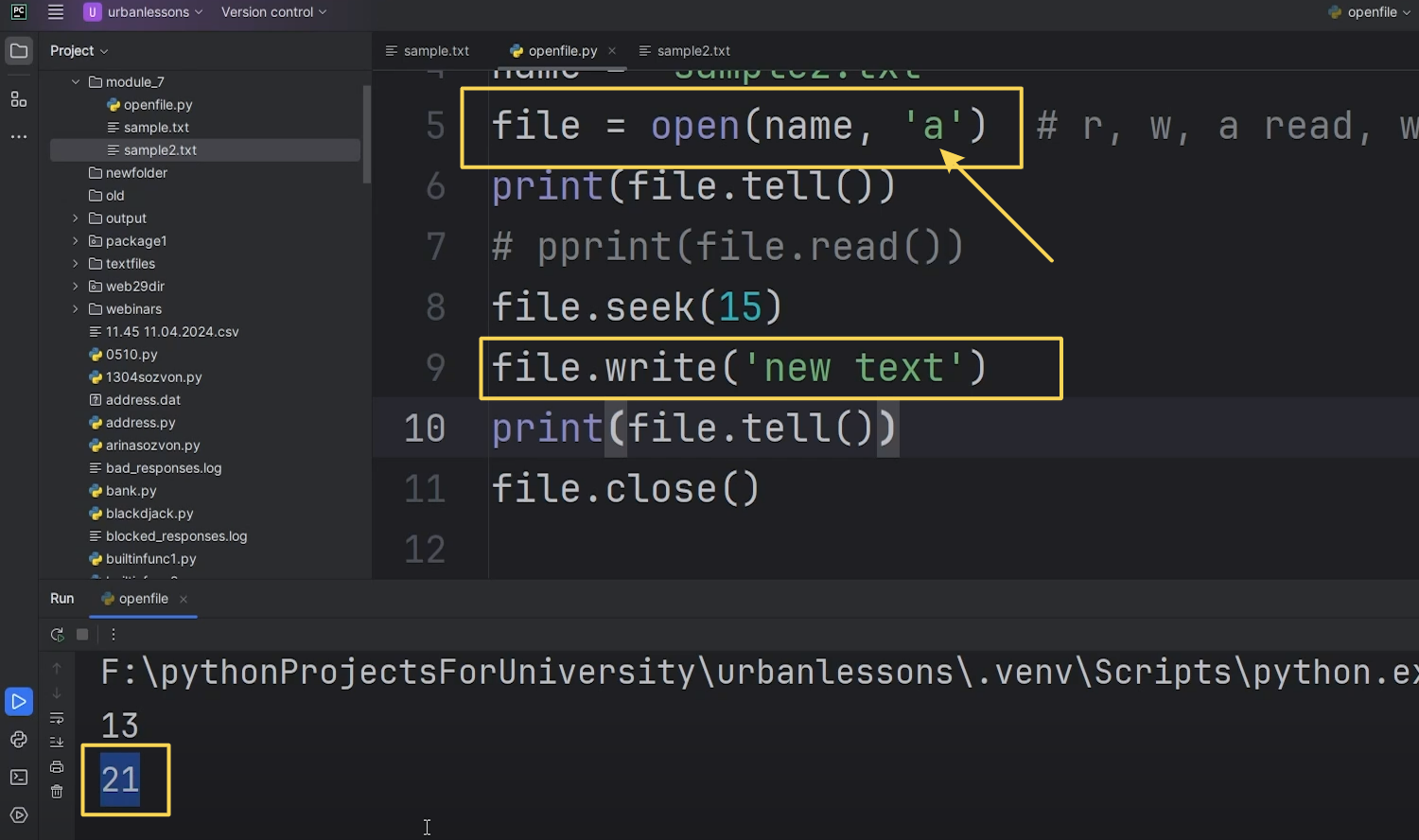


Рис.5

Даже если считать посимвольно, у нас «new» даёт 3 символа, «text» даёт 4, пустой пробел это ещё 1 символ, итого 8. Мы находились на 13 символе, передвинули курсор на 15 и записали 8 символов должны были получить позицию 23, однако здесь позиция 21. Что-то не сходится. Смотрим в «sample2.txt», текст встал куда нужно (Рис.6). То есть, когда мы работаем с нашим курсором, мы работаем именно с байтами. Это нужно запомнить.

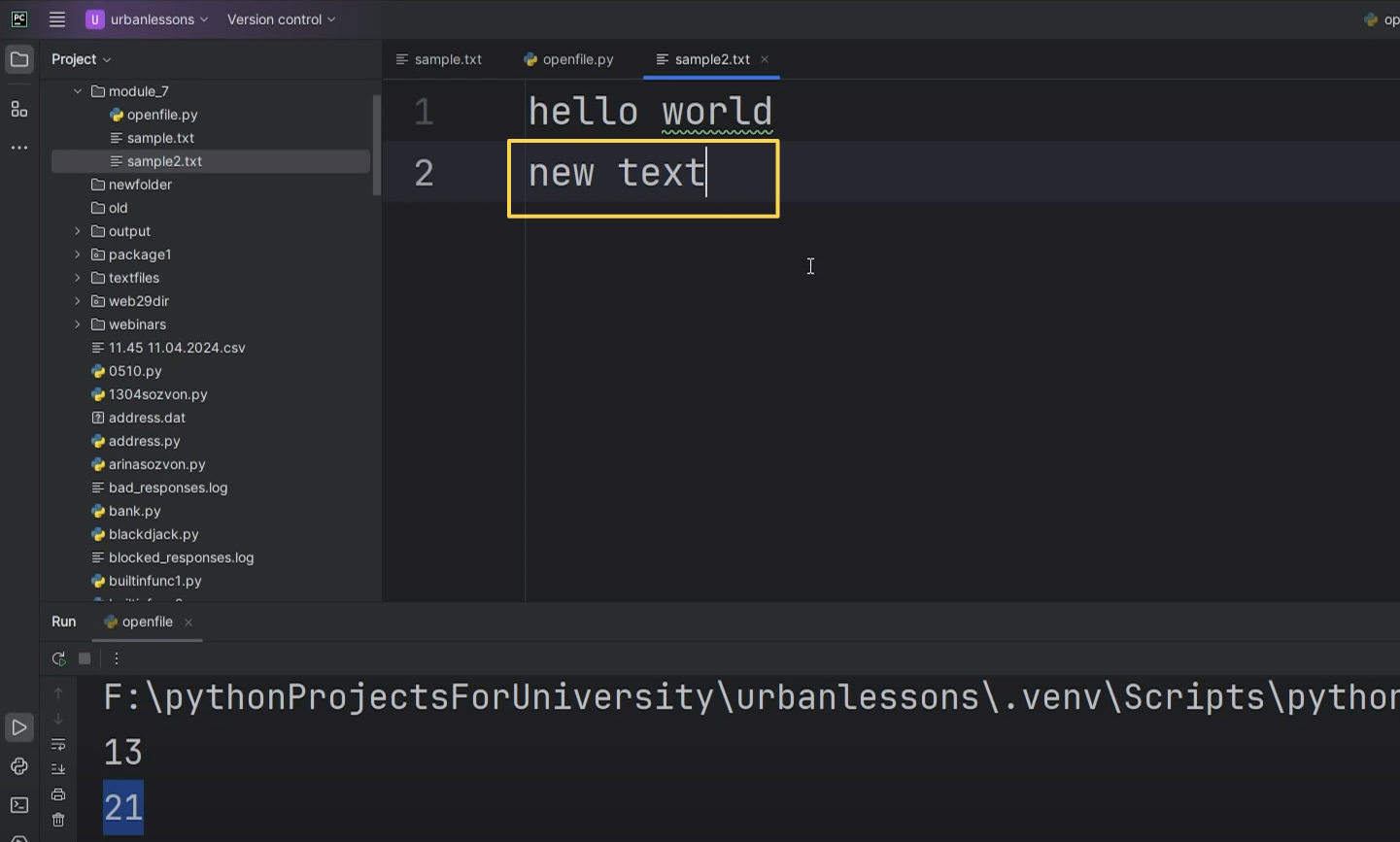


Рис.6

Однако при работе с файлами мы можем встретить некоторые ошибки, связанные с кодировкой, про которую мы говорили на первом занятии. У каждого символа есть своё числовое значение, и компьютер интерпретирует определённое число в соответствующий символ.

Изменим содержимое файла «sample2». Напишем «Привет, я новый текст», «Вторая строка нового текста» (Рис.7).

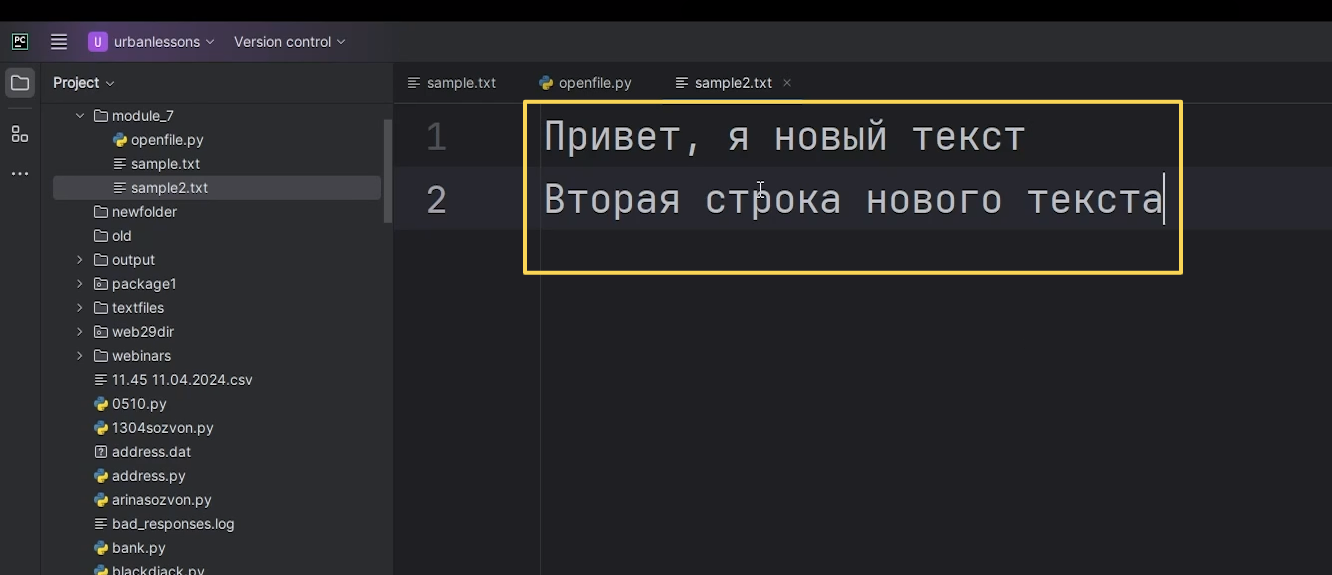


Рис.7

Попробуем считать содержимое нашего файла. Будем открывать файл в режиме чтения. Раскомментируем строчку «pprint(file.read()). Новый текст записывать, соответственно, мы не будем. Просто будем наблюдать за курсором, считывать файлы и снова наблюдать за курсором. Запускаем, получаем ошибку (Рис.8). Видим, что у нас кодировка не может перевести байт формата «0x8f» в соответствующий символ.

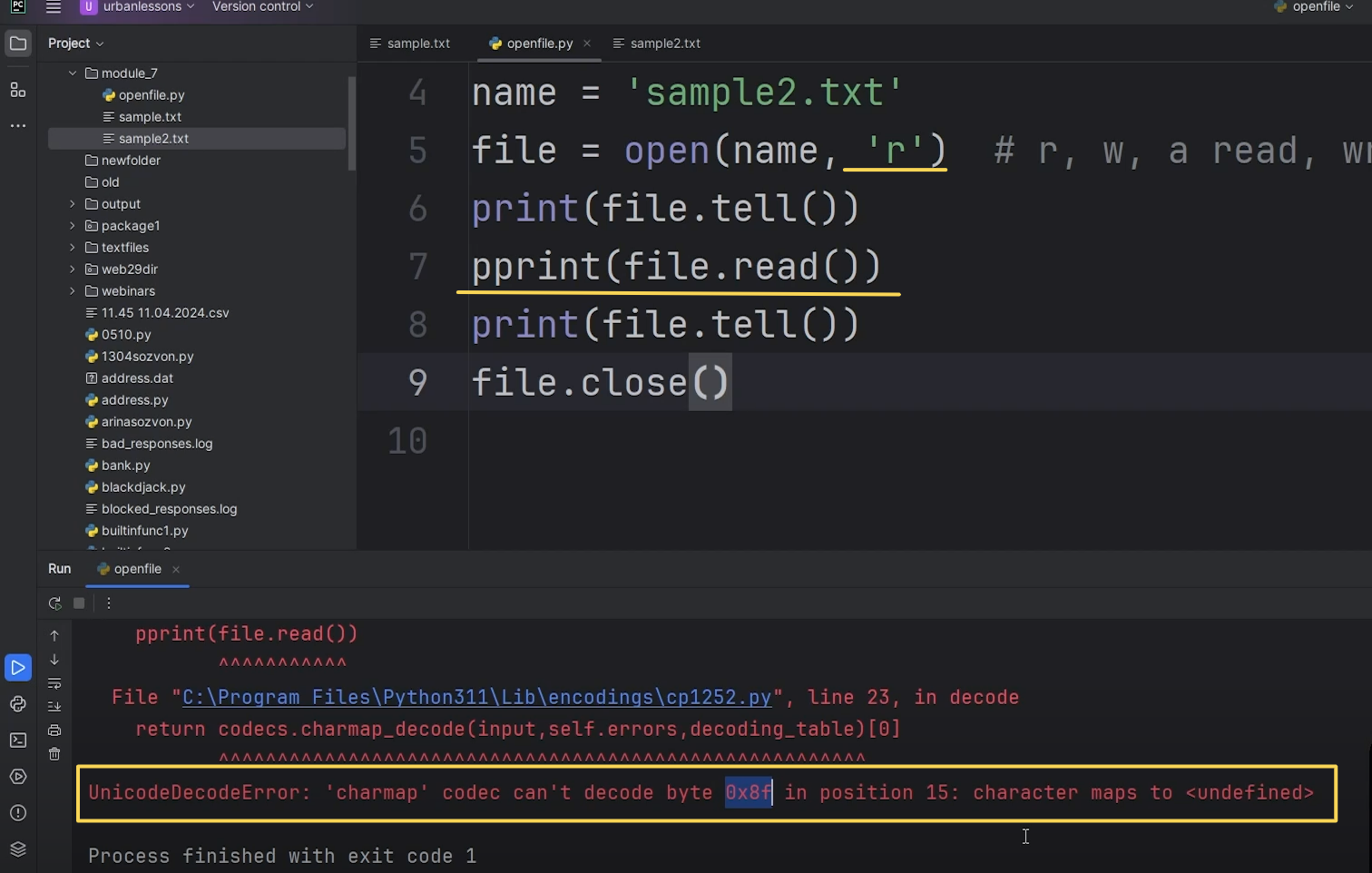


Рис.8

Как бороться с такой ошибкой? Если мы видим ошибку, которая связана с раскодировкой таблицы «Unicode», про которую мы тоже упоминали, то, вероятнее всего, нам нужно добавить параметр «encoding» в используемую нам функцию. Мы используем «open» для открытия файла и в ней надо добавить параметр «encoding» и указать, что мы будем использовать «utf-8» (Рис.9).

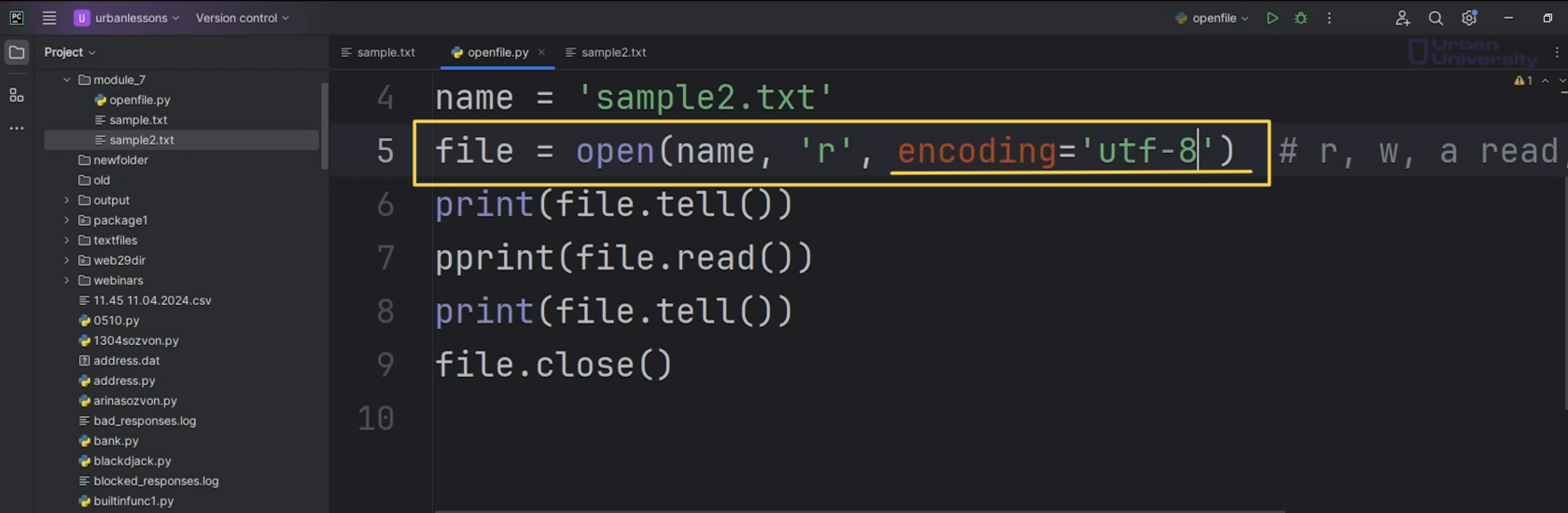


Рис.9

Самые внимательные помнят, что по умолчанию у нас кодировка, в которой открывается файл, является «сp1252», поэтому мы не могли считать символы, которые написаны в «sample2.txt».

Запускаем и видим «Привет, я новый текст\nВторая строка нового текста» (Рис.10).

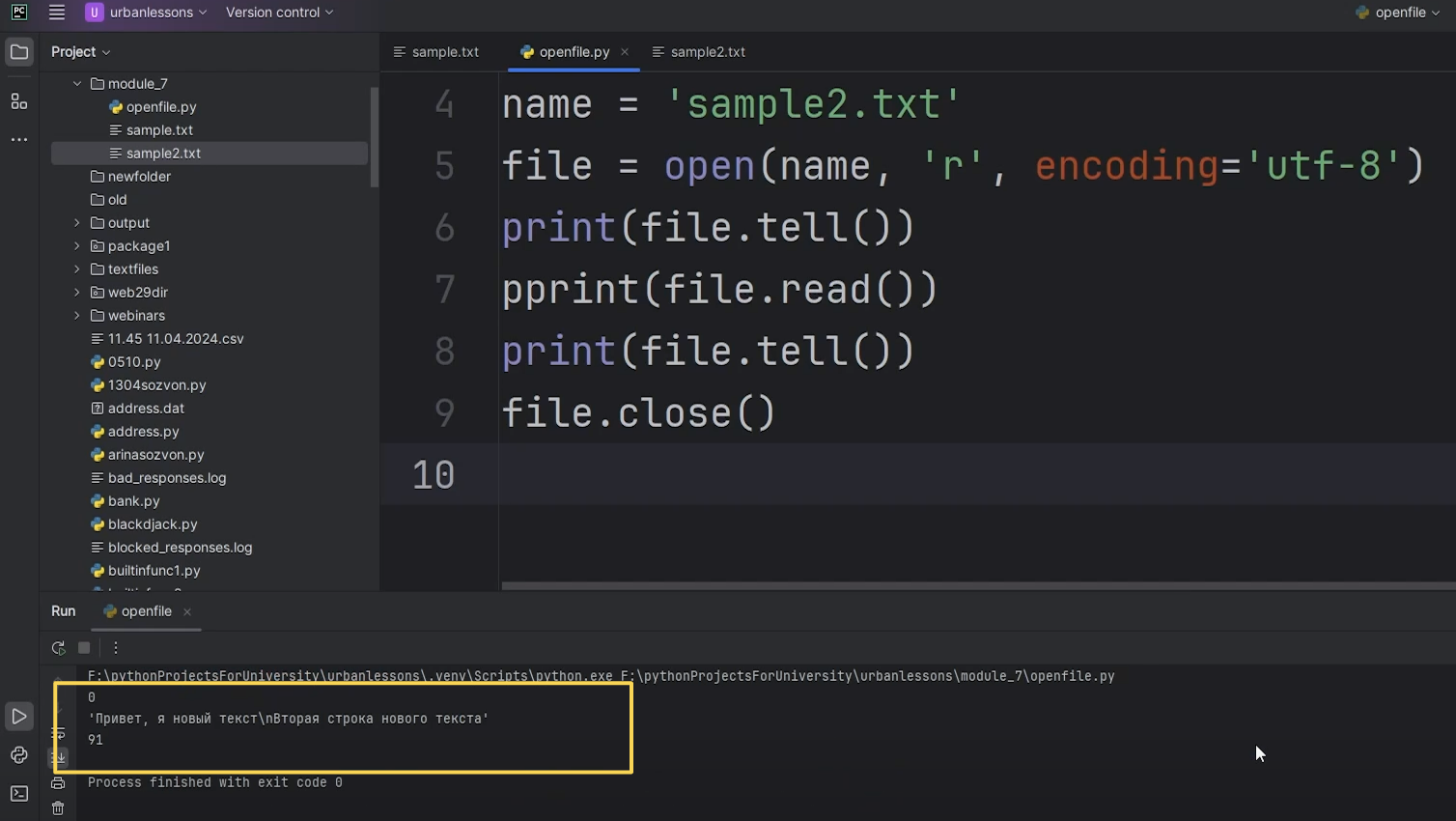


Рис.10

Глядя на это, невольно бросается в голову мысль о том, что как здесь 91 символ? Это потому что тут идёт речь не о символах, а о байтах.

Это такие полезные советы для работы с файлами. Помимо всего прочего, к полезным советам присоединятся ещё некоторые методы. Вы выбрали режим, выбрали кодировку, отлично считали содержимое файла, но хотите узнать, что даёт вам кодировка и режим открытия. Хотя в основном все зависит от режима, в котором вы открываете файл.

Мы не могли записывать файл, который открыт в режиме для чтения, правильно? **Как узнать, можно ли записывать файл или нет?** Для того чтобы это узнать, у нас **есть ряд методов**. Мы можем взять наш файл, воспользоваться методов **«writable»**. Взять наш файл и воспользоваться методом **«readable».**Взять наш файл, воспользоваться методом **«seekable».**

Что эти методы нам дают? Если мы запустим в режиме чтения, мы видим «False, True, True» (Рис.11).

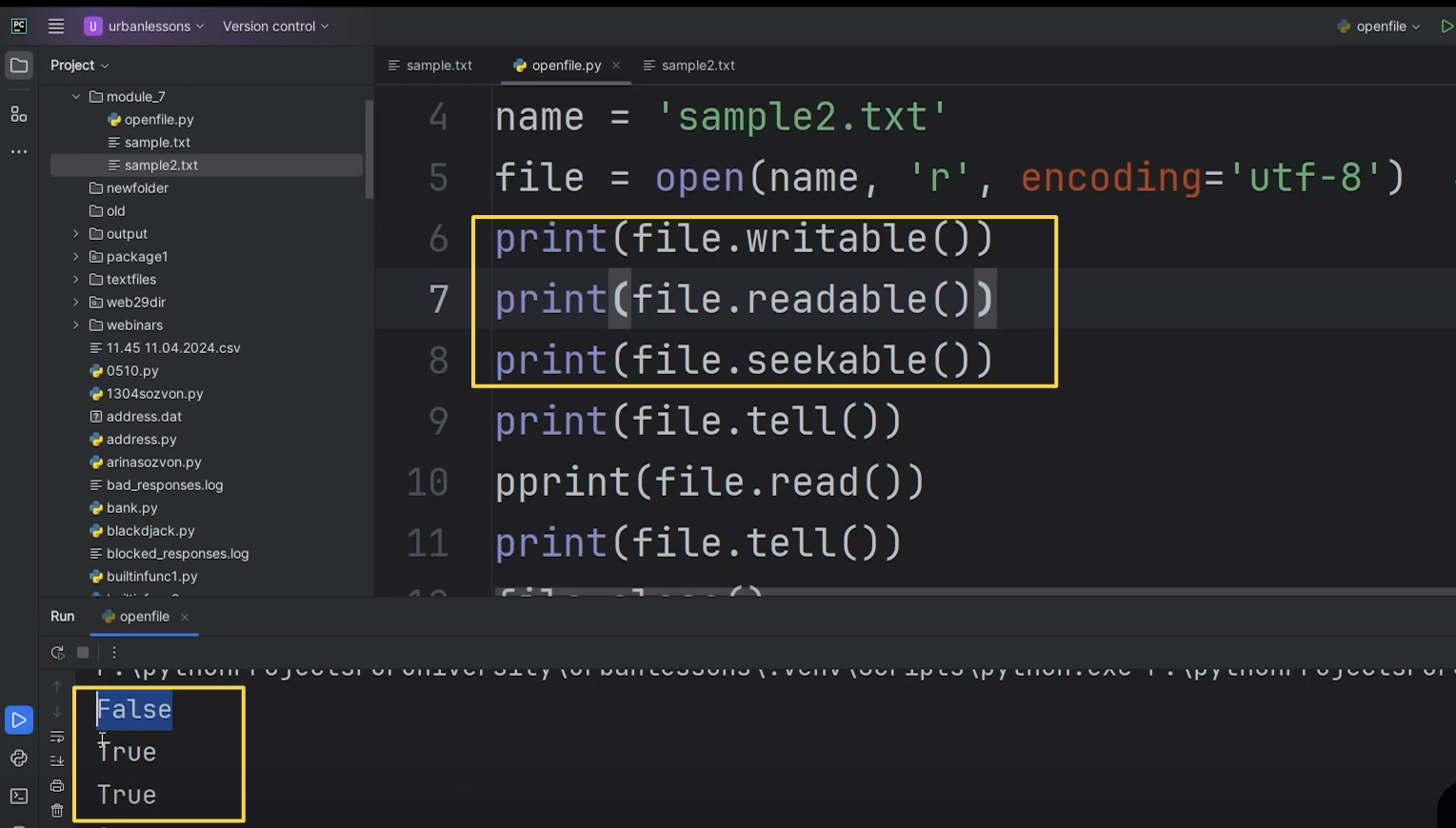


Рис.11

Мы открыли файл в режиме чтения и «writable» вернул нам «False», то есть записать что-то в этот файл мы не можем. Далее мы открыли файл в режиме чтения, и метод «readable» вернул нам «Тrue», то есть мы можем считывать информацию в нашем файле. И следующее «seekable», «seek» мы использовали для того, чтобы перемещать наш курсор. И данный метод будет возвращать «true» или «false» в зависимости от того, есть ли возможность перемещать курсор в нашем файле, либо же этой возможности нет. Все зависит от того, с каким типом файлов вы работаете. Здесь ничего сверхтяжёлого нет.

Наш арсенал команд пополняется полезными методами. Помимо всего прочего, мы можем также выводить информацию об отдельных атрибутах нашего файла. То есть мы можем взять файл и увидеть, что есть «name», «buffer», «closed», «enconding», «errors» (Рис.12).

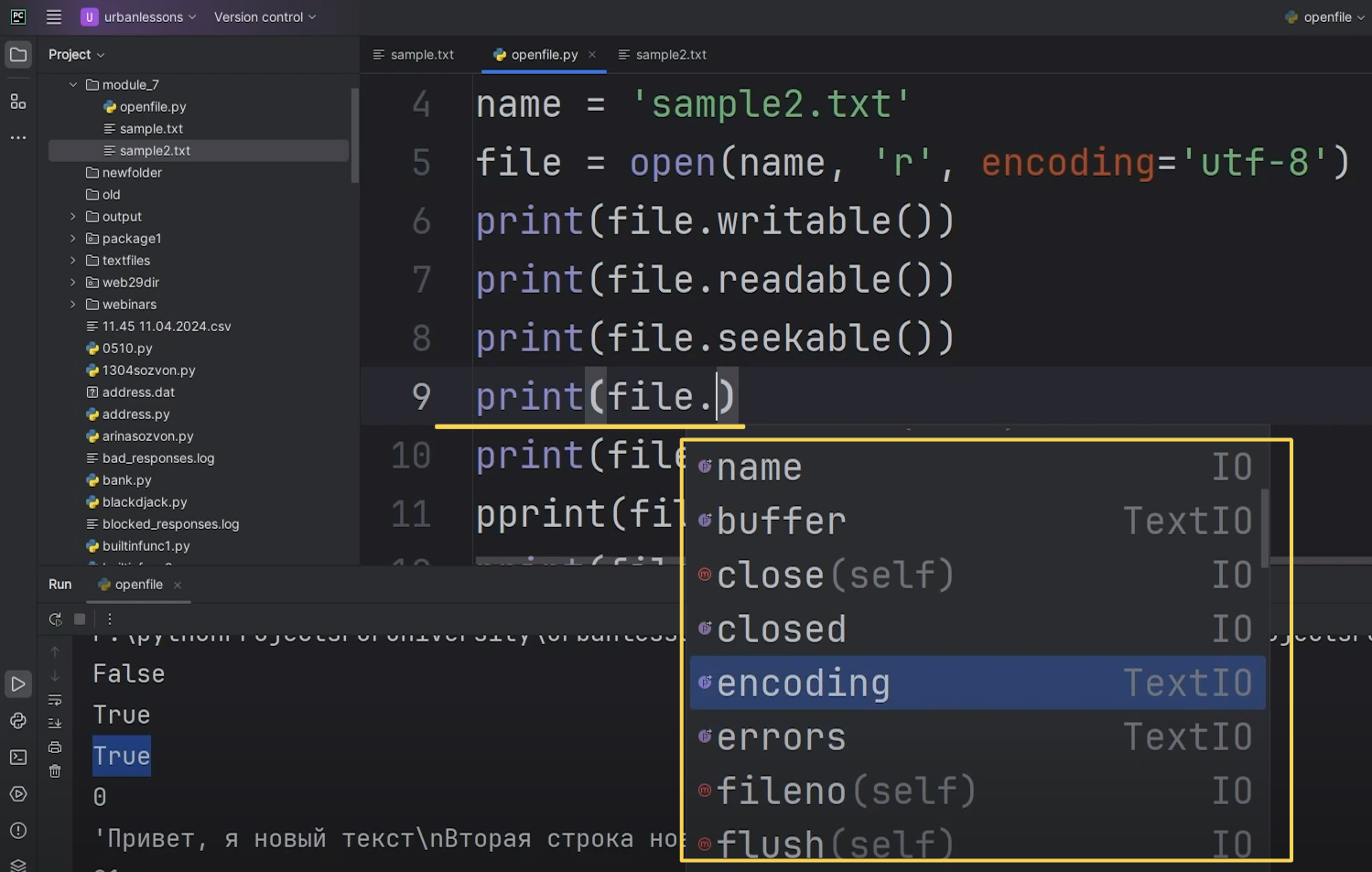


Рис.12

Например, хотим узнать имя файла, выведем атрибут «name» (Рис.13).

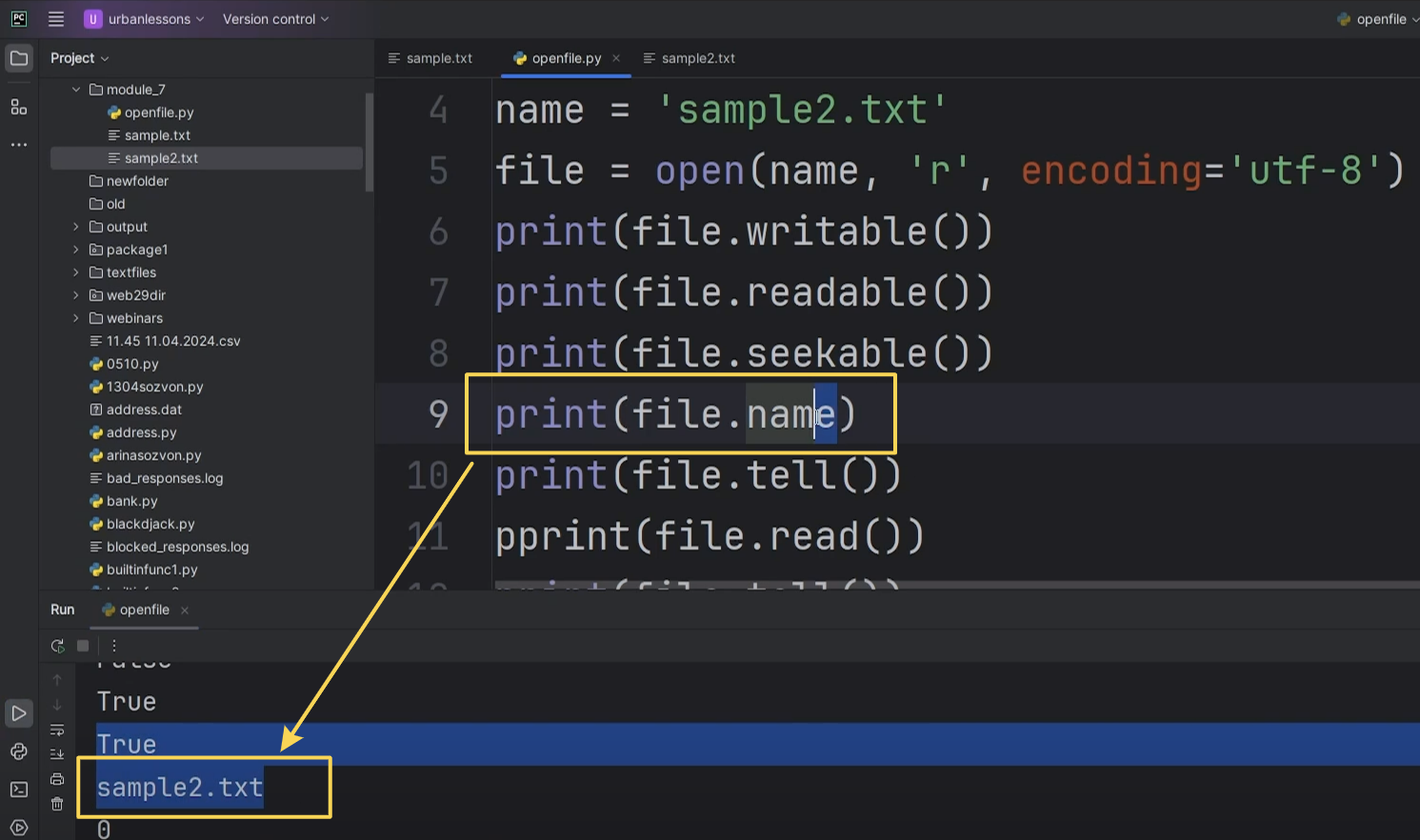


Рис.13

Воспользуемся параметром «buffer». Увидим, что у нас есть «buffer» такой-то (Рис.14).



Рис.14

Воспользуемся атрибутом «closed» и запустим (Рис.15). Видим «false». Файл не закрыт.

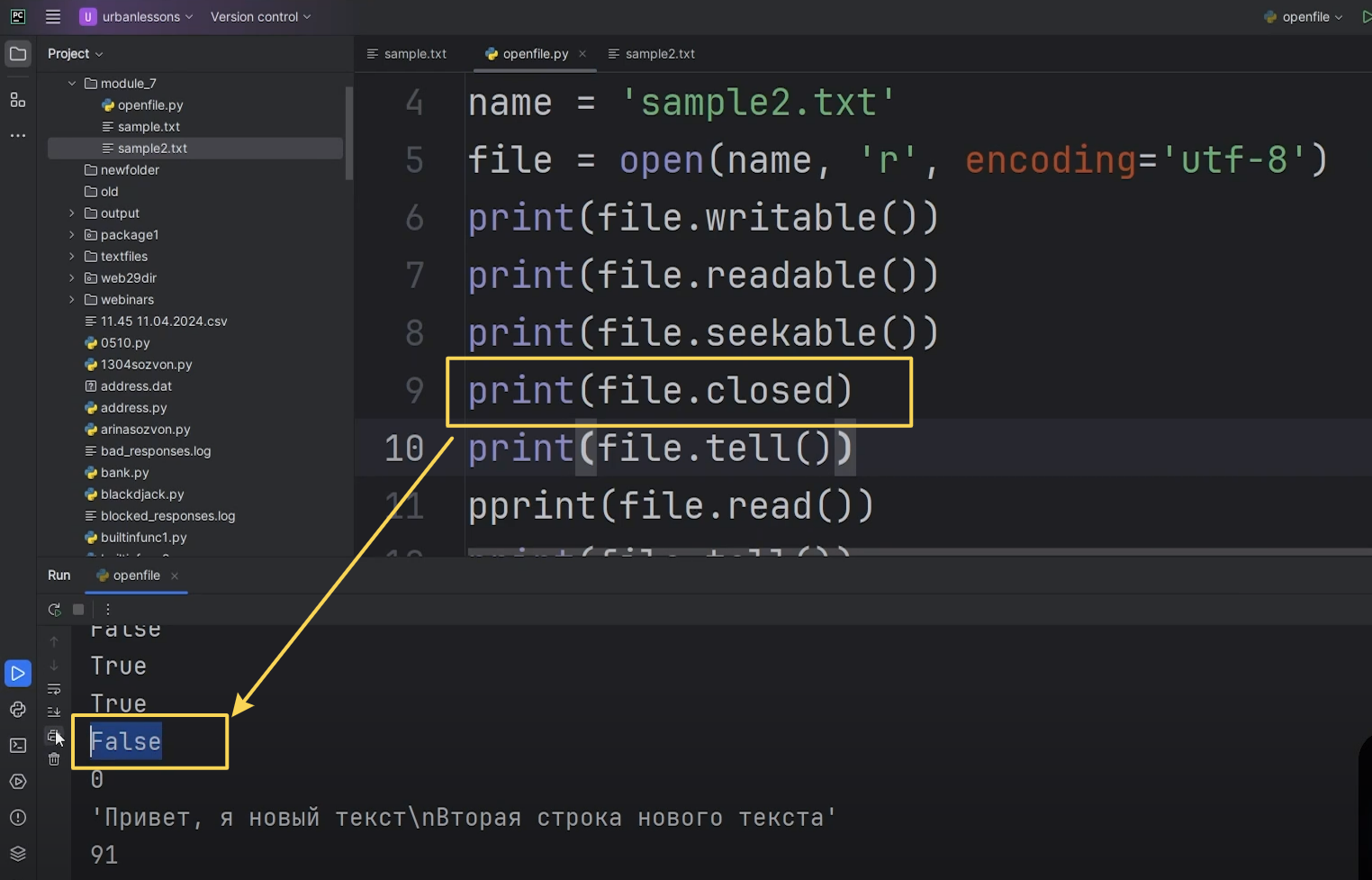


Рис.15

Информацию о файле тоже можно считывать. Раз файл у нас является объектом, каждый объект обладает своими методами и атрибутами. Вспоминаем работу с классами, постепенно применяем знания, которые мы с вами уже получили, для того чтобы сделать нашу работу с файлами максимально комфортной.

На этом наше занятие заканчивается. Всем успехов!