**Лекция. Динамическая типизация**

## Динамическая типизация

Продолжая говорить о переменных, сегодня мы рассмотрим динамическую типизацию. Давайте же начнем!

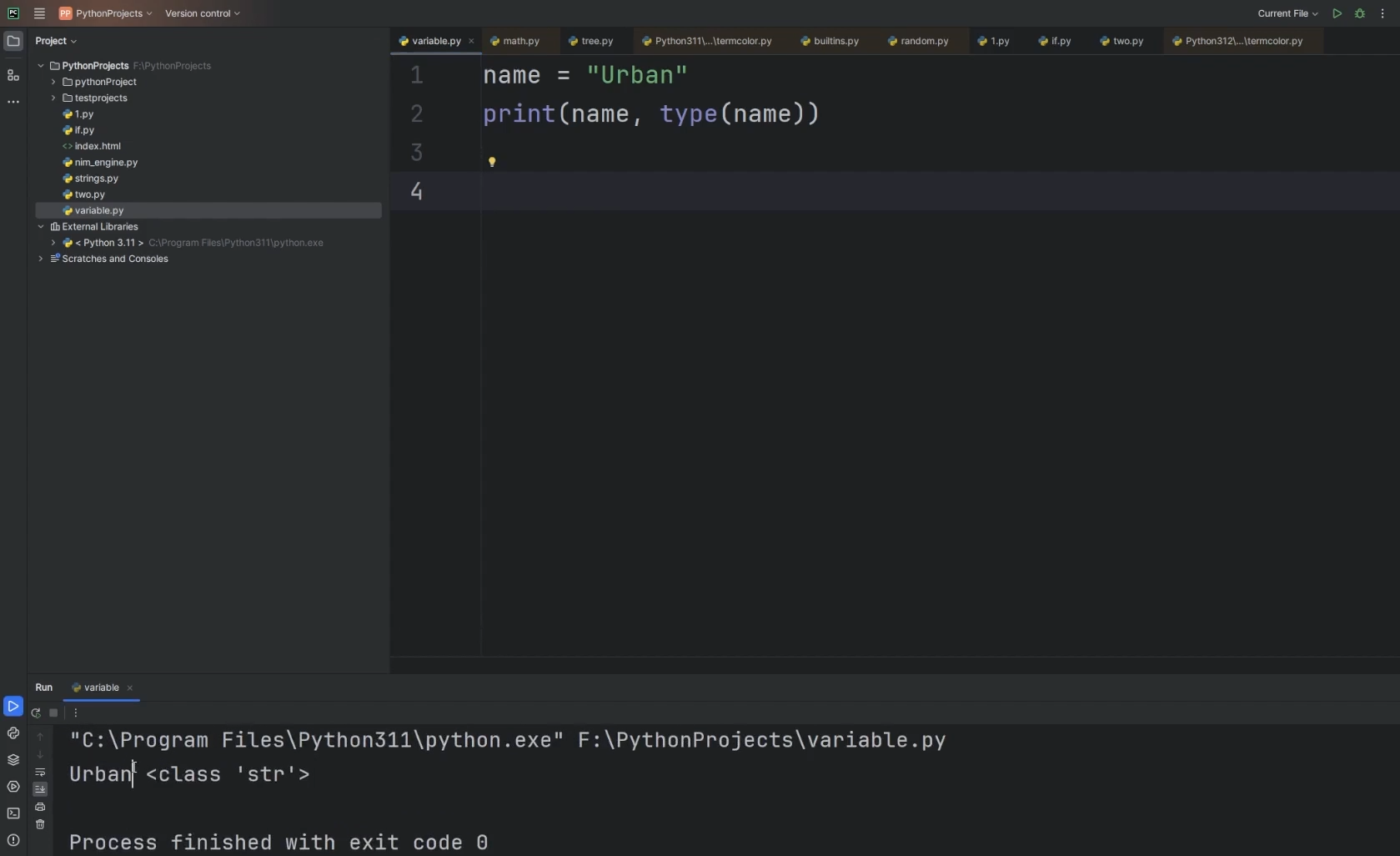
### Адрес памяти. Id объектов

В Python тип данных хранится не в переменной, а в самом объекте, на который она ссылается. Стоит упомянуть, что в Python объектом является все.

Как мы помним, переменная является частью памяти, в которую мы помещаем объект, чтобы использовать позже. В Python работать напрямую с адресами памяти не представляется возможным. Однако, есть одно исключение – функция **id**, которая в определенных реализациях может показать нам адрес памяти объекта. Тем не менее, ее основная и, пожалуй, единственная цель, это проверить ссылаются ли два имени на один и тот же объект, т. к. обратиться по этому адресу средствами Python невозможно.

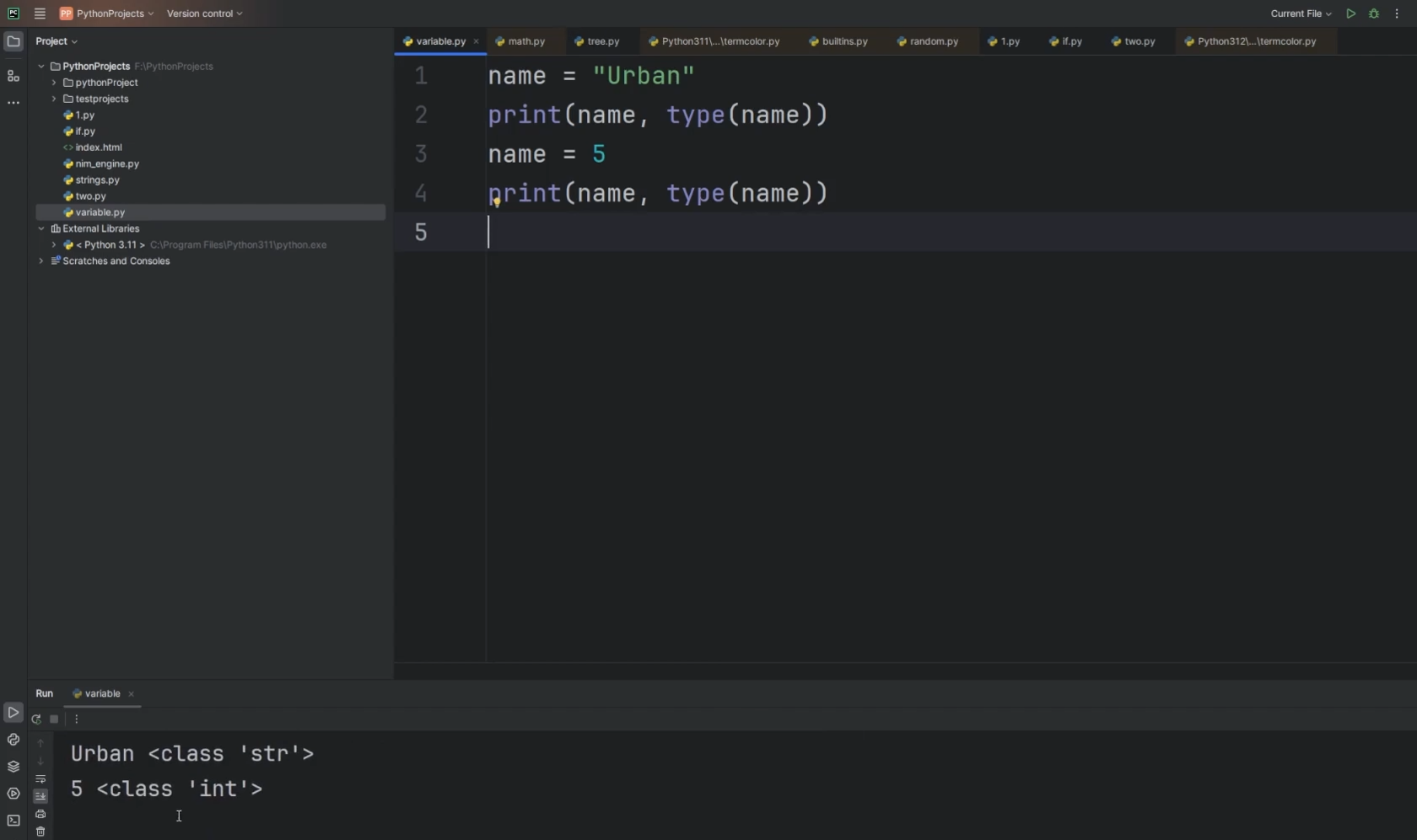
### Как работает динамическая типизация?

Рассмотрим наглядно, у нас есть переменная «name», содержащая строку «Urban». Выведем нашу переменную и воспользуемся командой type, чтобы вывести тип нашего объекта (Рис.1). Соответственно видим, что строка «Urban» это тип str.



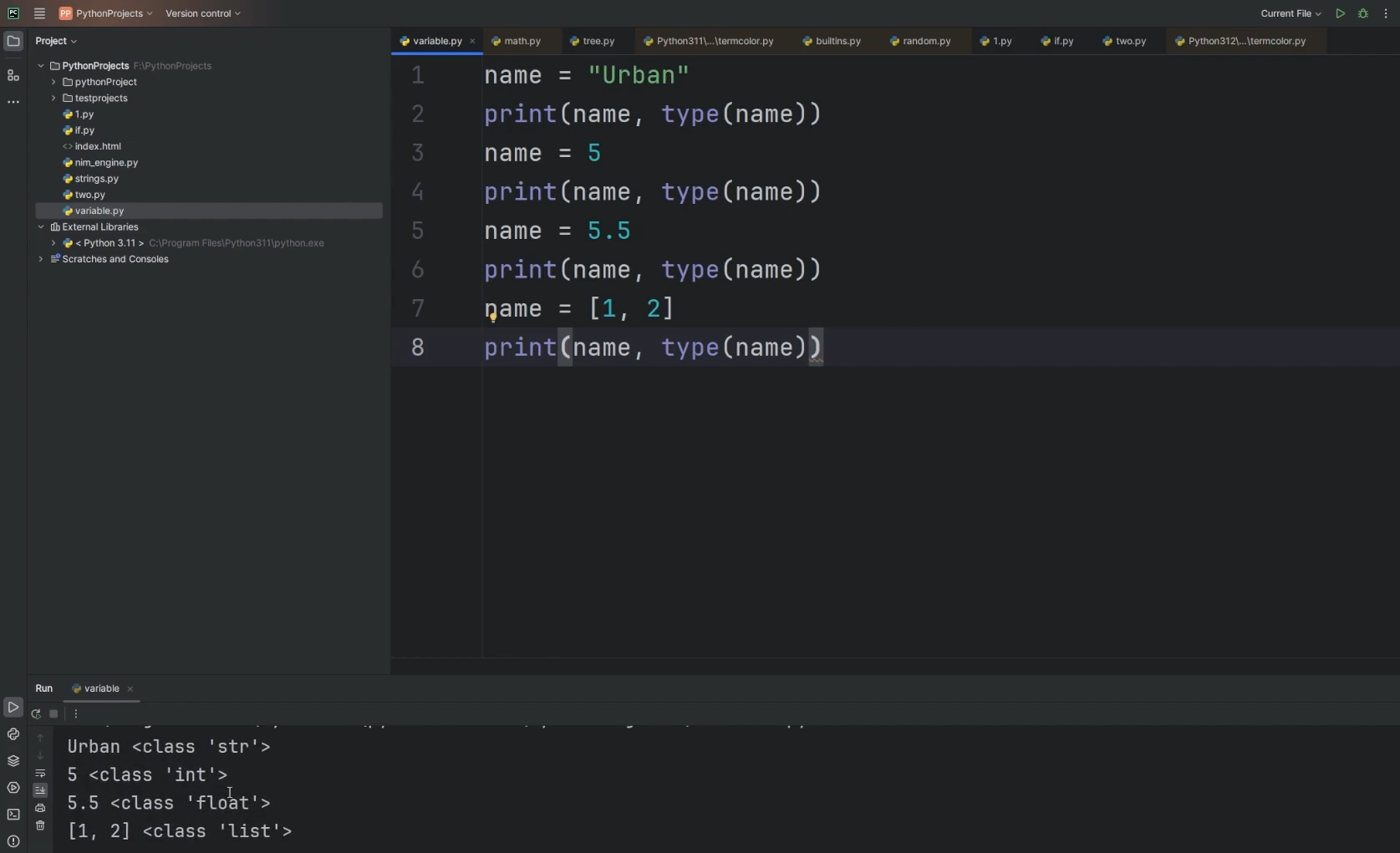
(Рис.1)

Говоря о динамической типизации, мы подразумеваем, что тип данных внутри переменной может меняться. То есть мы можем написать, что теперь «name = 5» и повторить заданную нами раннее инструкцию (Рис.2). Теперь мы видим, что наш объект это число 5, относящееся к типу int. Мы переопределили переменную и ее тип изменился.



(Рис.2)

По тому же принципу мы можем написать «name = 5.5» и теперь наш объект будет типа float, то есть число с плавающей точкой, или же, что «name = [1, 2]», тогда ответом будет тип list, то есть список (Рис.3).

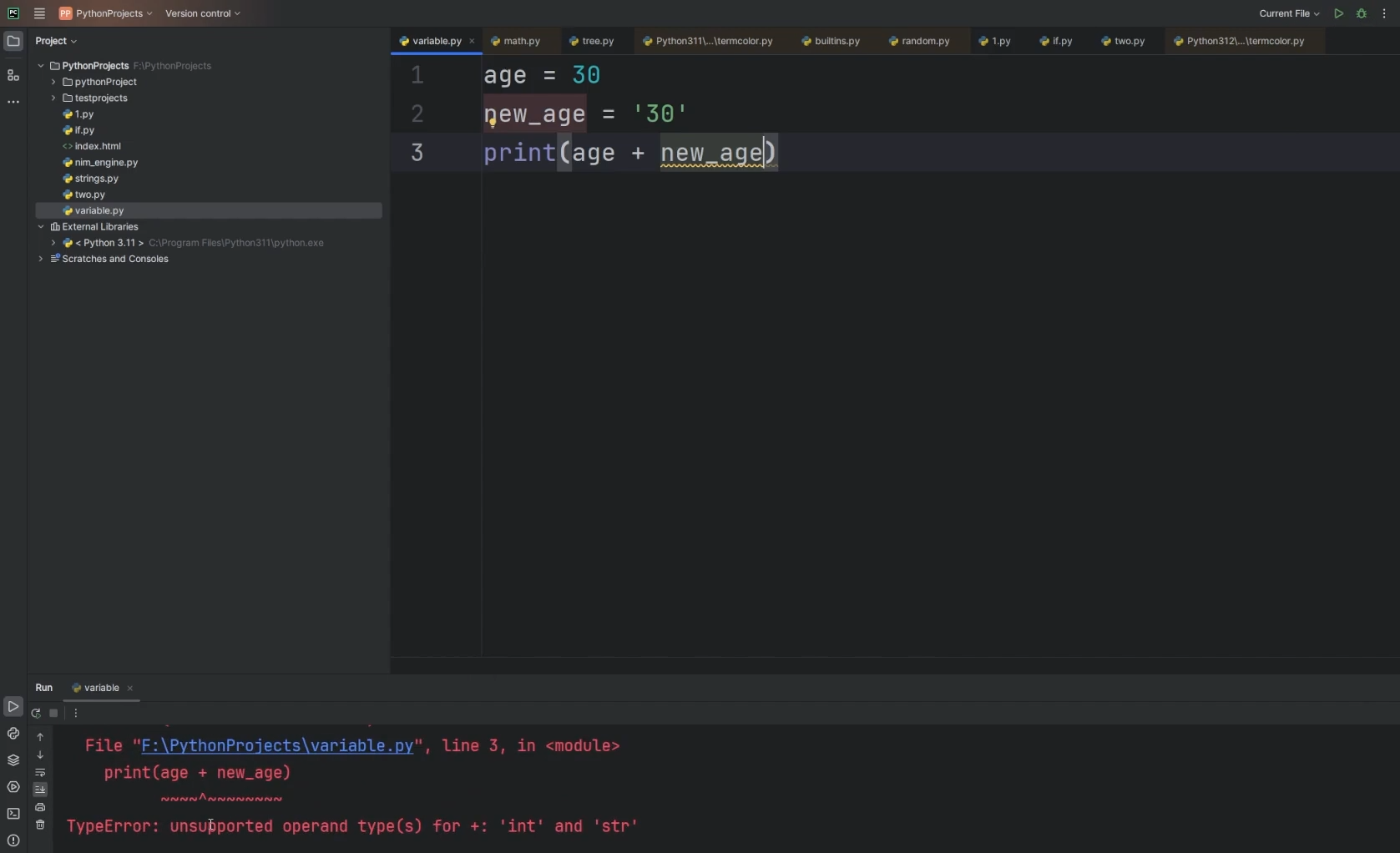


(Рис.3)

В каждом из этих случаев мы видим, что одна и та же переменная «name» ссылается на объекты разных типов.

Если совсем углубляться, то можно сказать, что Python не совсем воспринимает такое понятие, как переменная. Чтобы полностью передать смысл, правильнее будет сказать, что это имя или же идентификатор, ссылающийся на объект.

Если подытожить, то мы должны запомнить, что в процессе работы мы можем изменять тип данных внутри переменной. Однако, нельзя забывать и о том, чтобы при определенных операциях, например при математическом сложении, оба объекта были одного типа. То есть, чтобы после смена типа данных при операциях не возникло конфликта. Как в случае, если мы сложим переменную «age», содержащую число 30, и переменную «new\_age», содержащую строку «30», и получим ошибку (Рис.4).



(Рис.4)

### Интересные ссылки

<https://www.youtube.com/watch?v=R4WF9xad_EI> - 10 вопросов Python-разработчику