**2023/10/24 00:00|Лекция. Классы и объекты**

## Классы и объекты

На данном занятии мы познакомимся с классами. Это понятие новое для вас, но с ней потихоньку начинаем погружаться в сферу объектно-ориентированного программирования.

Раньше мы с вами писали код в виде прямых инструкций. Они выполнялись по порядку, то есть в оперативном стиле, познакомились с функциями, с функциональным стилем программирования. Сейчас пришло время, наконец-то познакомиться с объектно-ориентированным стилем программирования.

Класс можно представить как некую инструкцию. На ее основе создаются объекты. Для того чтобы создать класс, нам достаточно написать ключевое слово class(рис.1). После него пишем непосредственно название нашего класса.

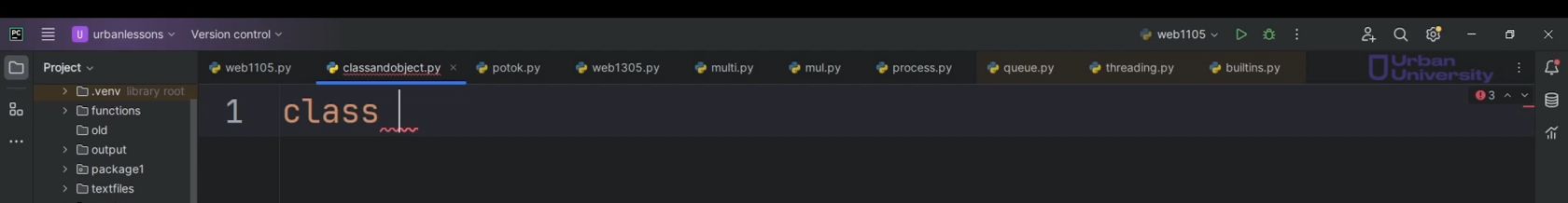


Рис.1

Представим, что у нас есть люди, их может быть много. Каждый обладает какими-то уникальными индивидуальными характеристиками, но в целом обо всех людях можно сказать, что у человека есть имя, например, есть возраст. Давайте создадим класс Human, на основе которого будем создавать много объектов людей(рис.2). По сути, класс будет выступать в роли некого описания, в роли инструкции. На основе этой инструкции у нас будут появляться объекты. Каждый из этих объектов будет по-своему уникален: будет иметь свой адрес в памяти, свои уникальные характеристики и так далее. Про класс ещё можно сказать, что создается собственный тип данных.

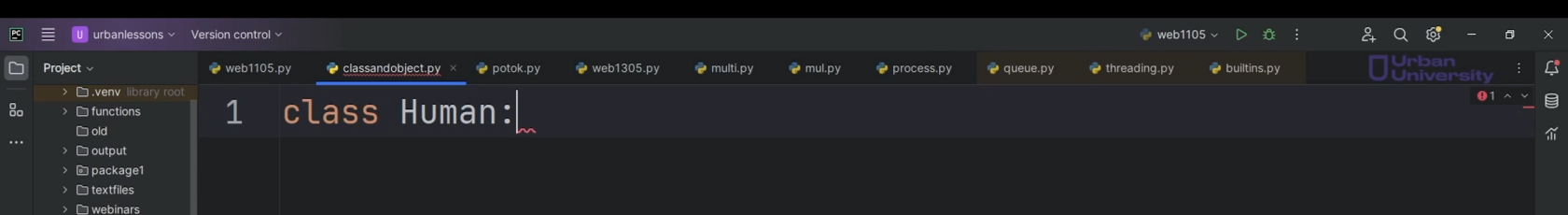


Рис.2

**Пример.** Создадим переменную den — объект класса Human(экземпляр класса)(рис.3). Чтобы создать объект, необходимо написать название класса и открыть, закрыть скобочки. Теперь давайте воспользуемся функцией type и выведем тип у этой переменной(рис.4). Обратите внимание, что данная переменная относится к типу Human. Если применить эту функцию по отношению, например, к 5, то увидим класс int(рис.5), str, list, dict и т.д. До этого мы работали с классами, которые были встроены уже в Python. Сейчас же создали что-то своё. В будущем можно будет переопределять различные методы, как бы делать класс более гибким, но сейчас наша задача просто познакомиться с этой концепцией. Таким образом можно создать собственный тип данных.

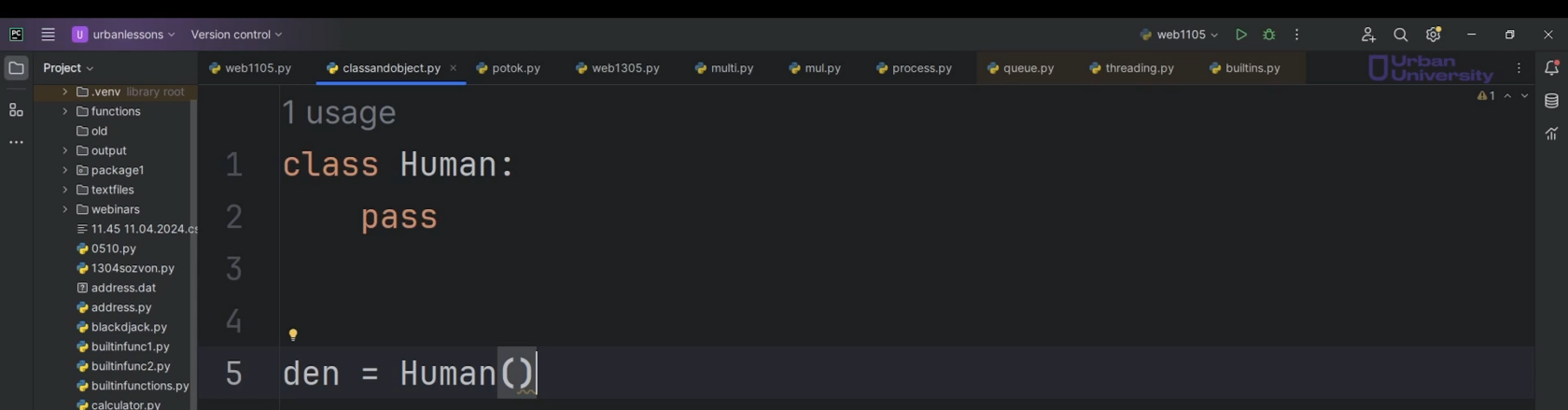


Рис.3

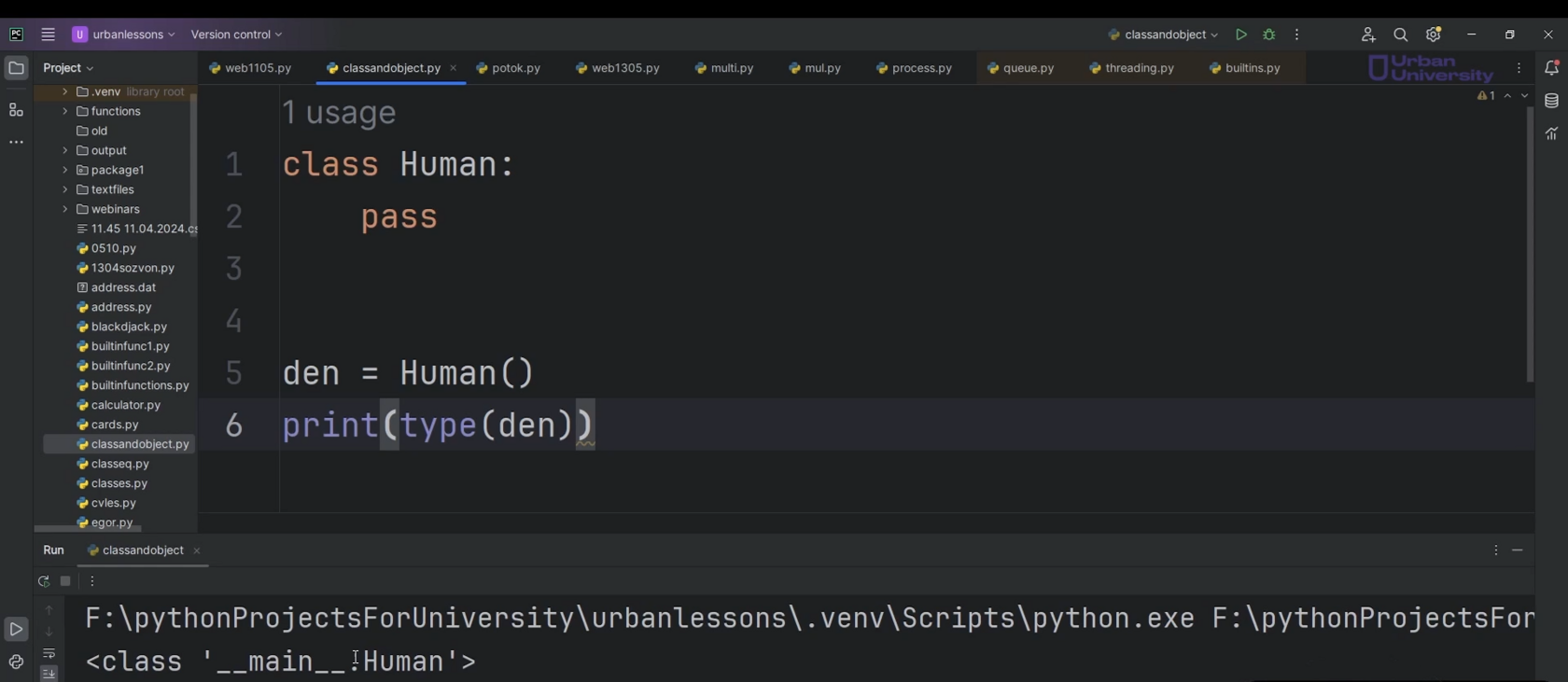


Рис.4

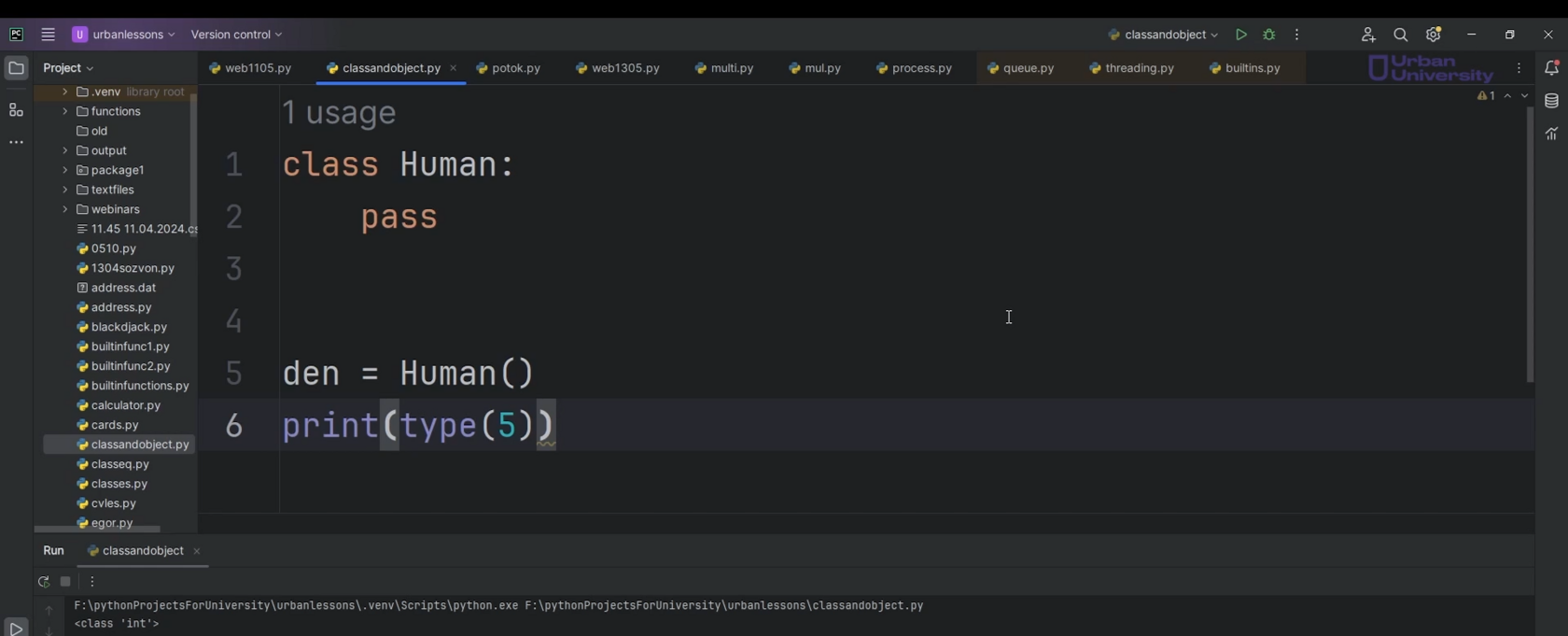


Рис.5

**Объекты, которые сейчас создали, они уникальны.**Если взять и сравнить Den и Max, то увидим, что это разные объекты(рис.6). При проверке с помощью оператора is, можно увидеть, что это все равно не одно и то же(рис.7). Получив id(den) и id(max), увидим абсолютно разные адреса в памяти(рис.8). Каждый созданный объект будет уникальным. По сути, на основе этой характеристики, на основе вот этой инструкции, создается уникальный объект, и нет никаких ограничений на количество объектов. Можно создавать их хоть 10, 20, 30, при этом каждый будет оставаться уникальным.

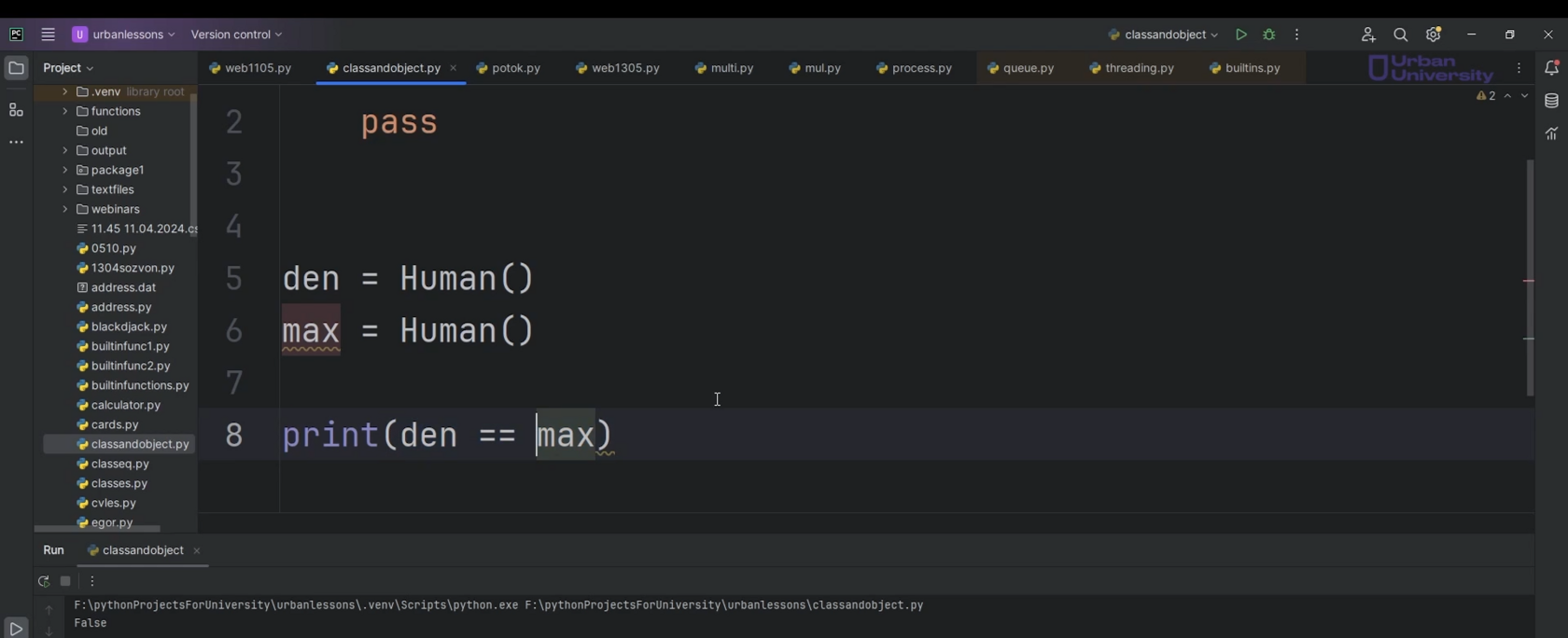


Рис.6

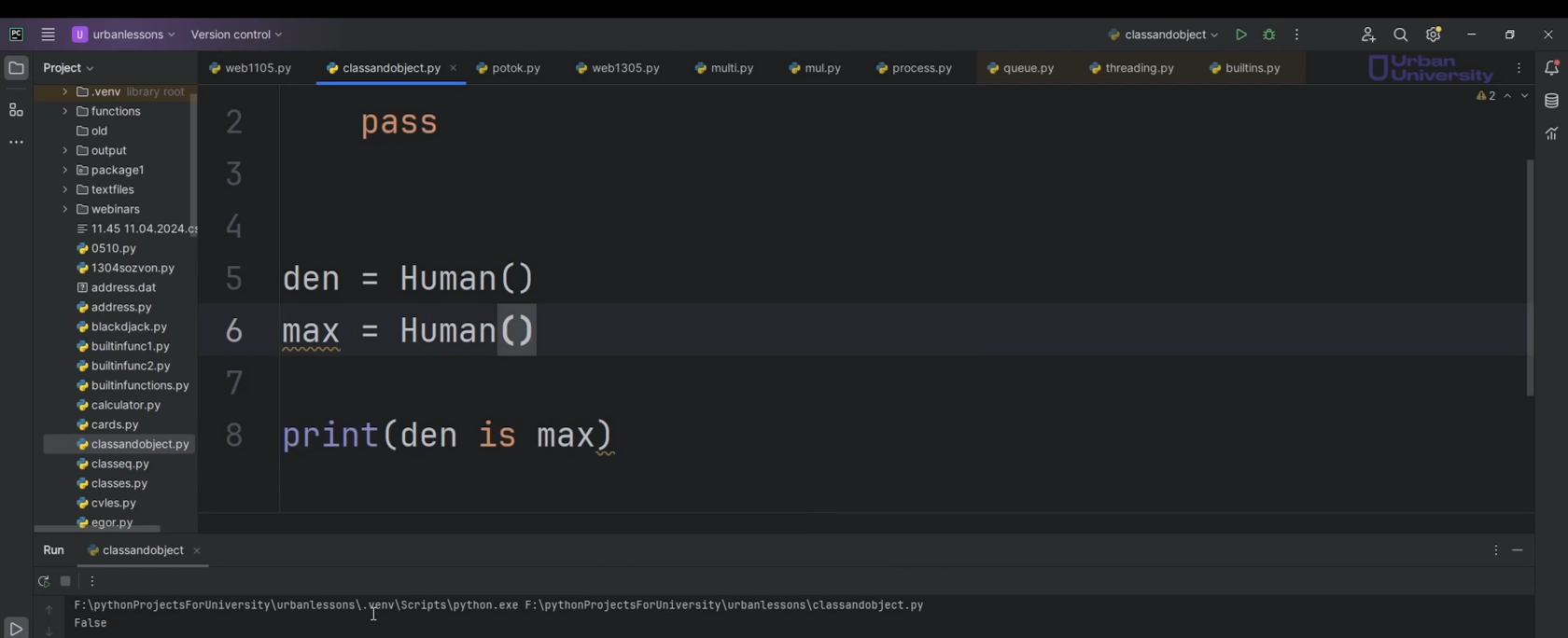


Рис.7

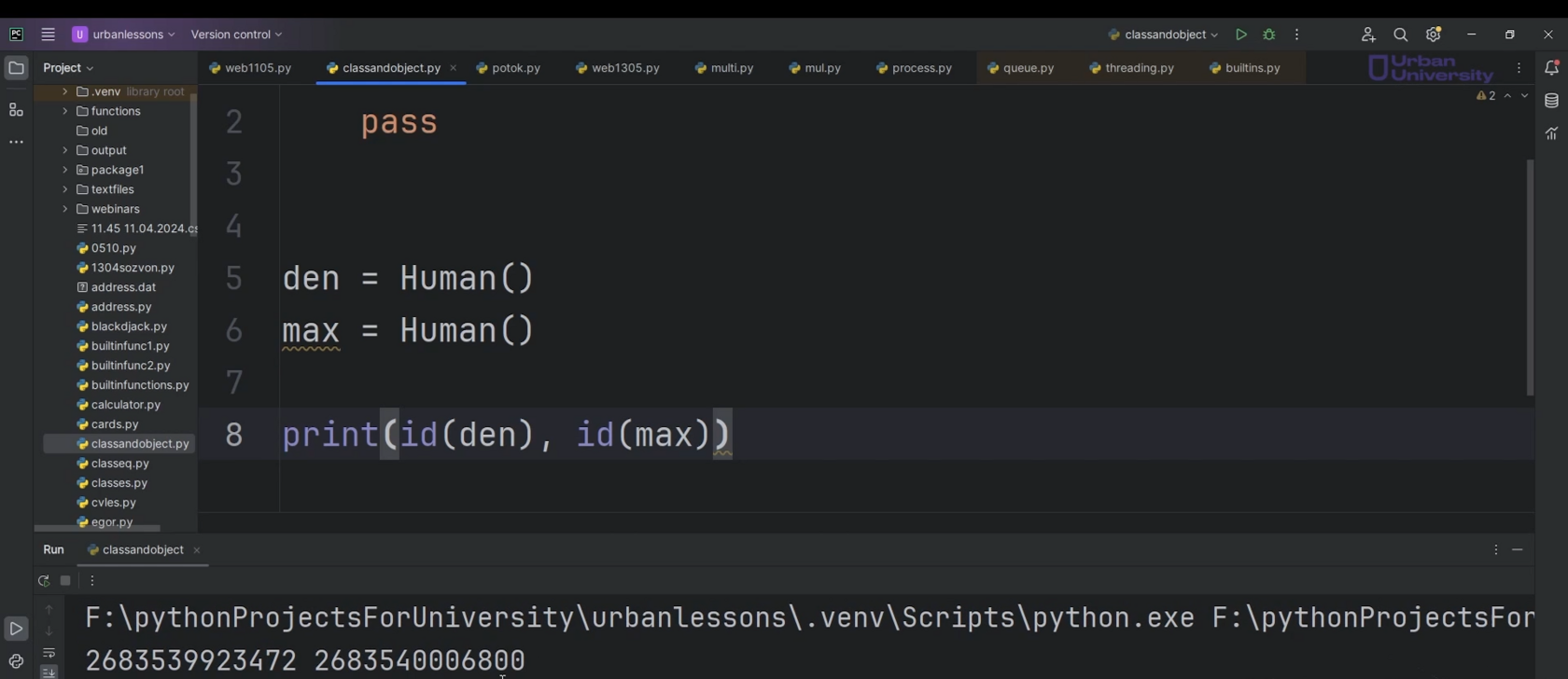


Рис.8

Чтобы добавить более какие-то уникальные вещи, при создании класса, по отношению к den и max, необходимо прописать конструкцию: def \_\_init\_\_ — это инициализатор(рис.9). По-другому можно сказать, что это конструктор класса. Сюда прописывается, что будет создаваться при инициализации объекта. Когда мы обращаемся к нашему классу, у нас происходит инициализация, срабатывает метод init. Он срабатывает 1 раз, когда создаётся объект класса и наш объект может получать какие-то уникальные характеристики. Все эти уникальные характеристики прописываются внутри этого метода init. Про self поговорим уже на следующем занятии.

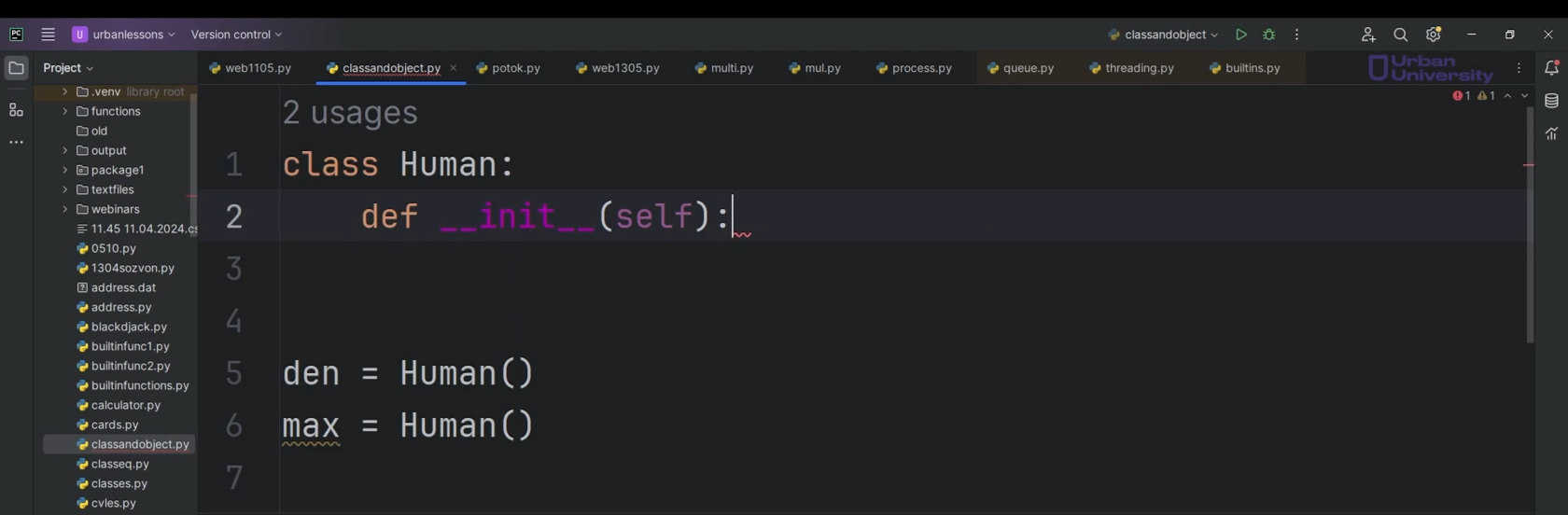


Рис.9

**Переменные, которые вы можете встретить внутри класса, их называют ещё характеристиками, либо же атрибутами. Функции внутри класса называют методами.**

У нас есть метод init. Внутри у него определена характеристика name, и у неё есть значение, den(рис.10). Теперь, при обращении к den, можно получить атрибут name. Если обращаться к max, также можно получить атрибут name, однако сейчас они все имеют одинаковые имена(рис.11).

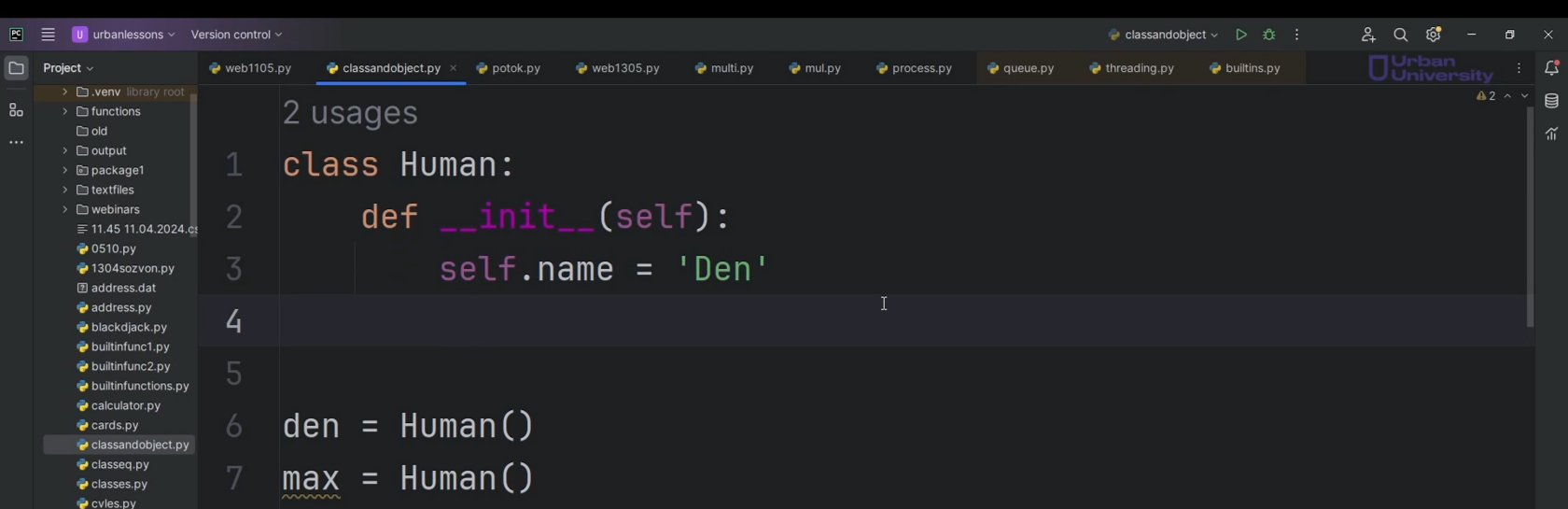


Рис.10

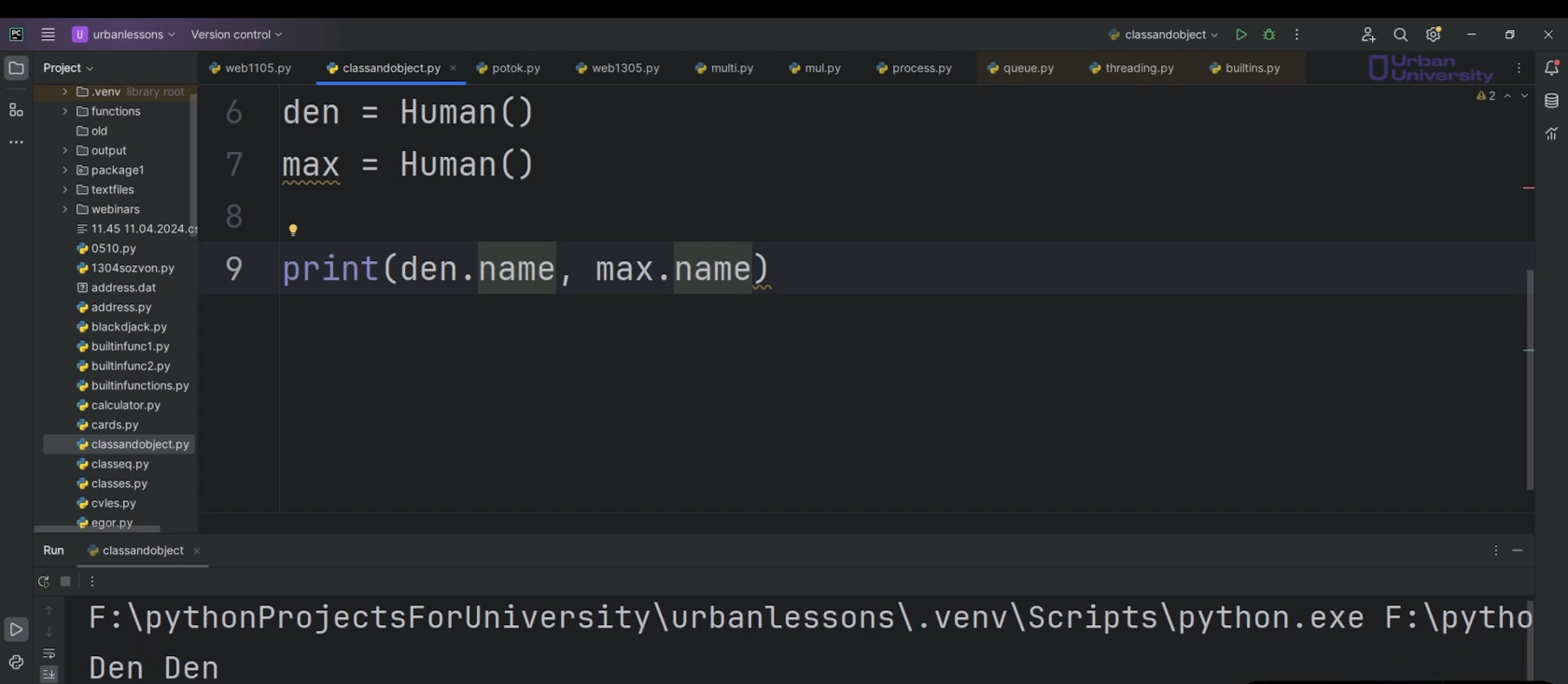


Рис.11

**Хотелось бы при создании объекта имя задавать вручную.** Для того чтобы это сделать достаточно после self прописать name и вместо den поставить name. Это как с обычной принимающей функцией. Теперь у нас вот эти скобочки den=Human() будут принимать значение name (self, name) и подставлять его под name (self.name = name)(рис.12). Теперь каждый объект класса, при создании будет обладать уникальным именем. Это происходит за счёт того, что имя, которое мы будем присваивать, точнее, передавать во время создания объекта, оно будет передаваться для каждого объекта своё. У нас может быть Денис, Максим и т.д(рис.13). В то же время таких характеристик может быть, на самом деле очень большое количество. В этом то вся и задумка классов. Мы создаём некую инструкцию, на основе которой создается большое количество объектов.

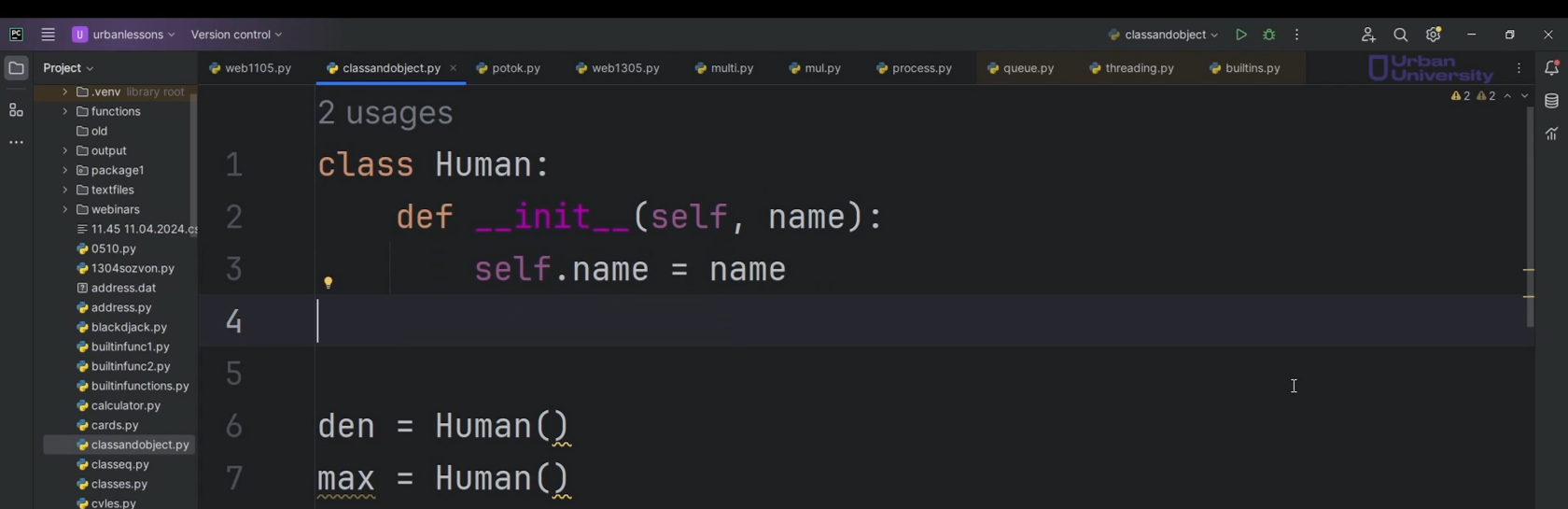


Рис.12

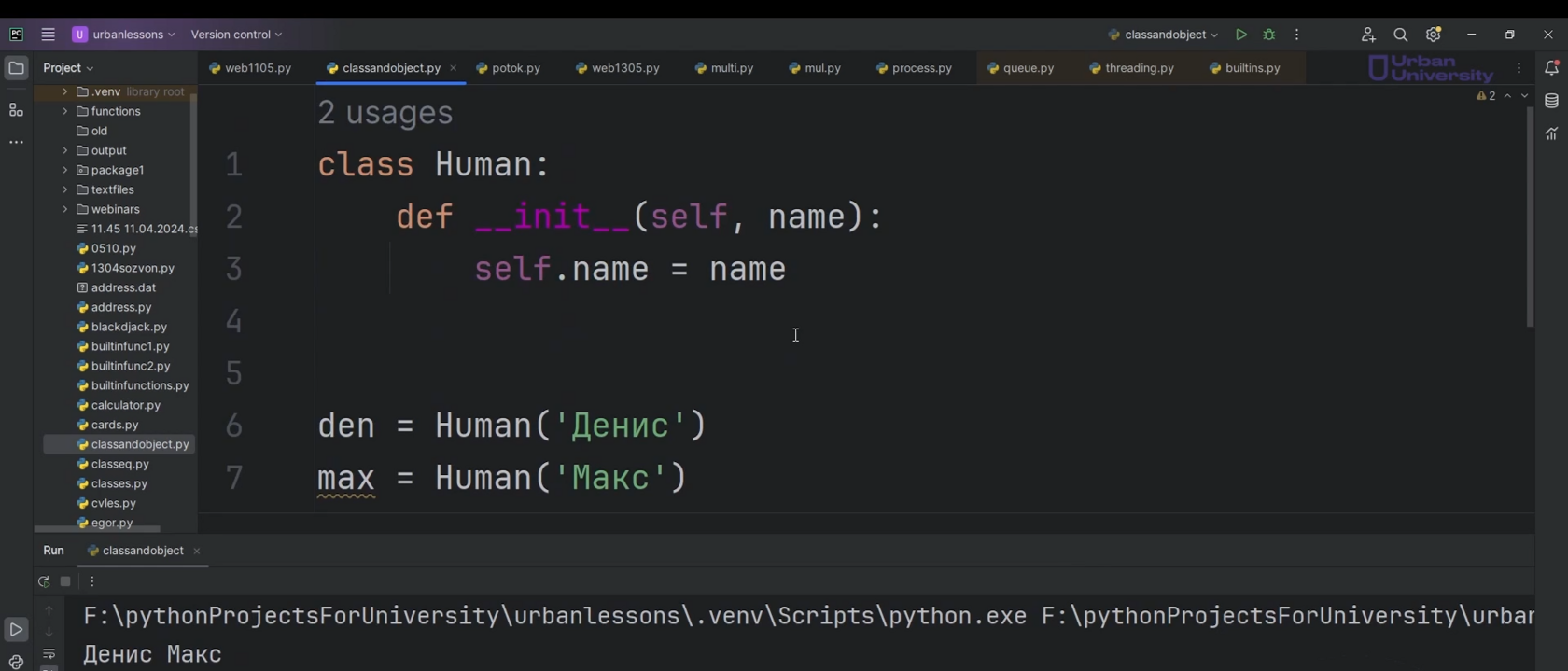


Рис.13