**2023/10/25 00:00|Лекция. Атрибуты и методы объекта. Указатель на свой объект в методах**

## Атрибуты и методы объекта. Указатель на свой объект в методах

На данном уроке продолжаем говорить про классы, про их атрибуты и методы.

**Атрибуты** — это переменные внутри класса. **Методы**— это функции внутри класса. По-другому можно сказать, что атрибуты — это характеристики, то есть какие-то уникальные черты наших объектов, а методы — это способности, то есть то, что умеет делать наш объект.

Допустим, у нас есть люди, возьмём тот же код, который делали на прошлом уроке(рис.1). Наши люди обладают уникальной характеристикой — у них есть имя. Про любого человека можно сказать, что у него есть имя, это его уникальная характеристика. Однако как так получается, что у каждого объекта своё имя?

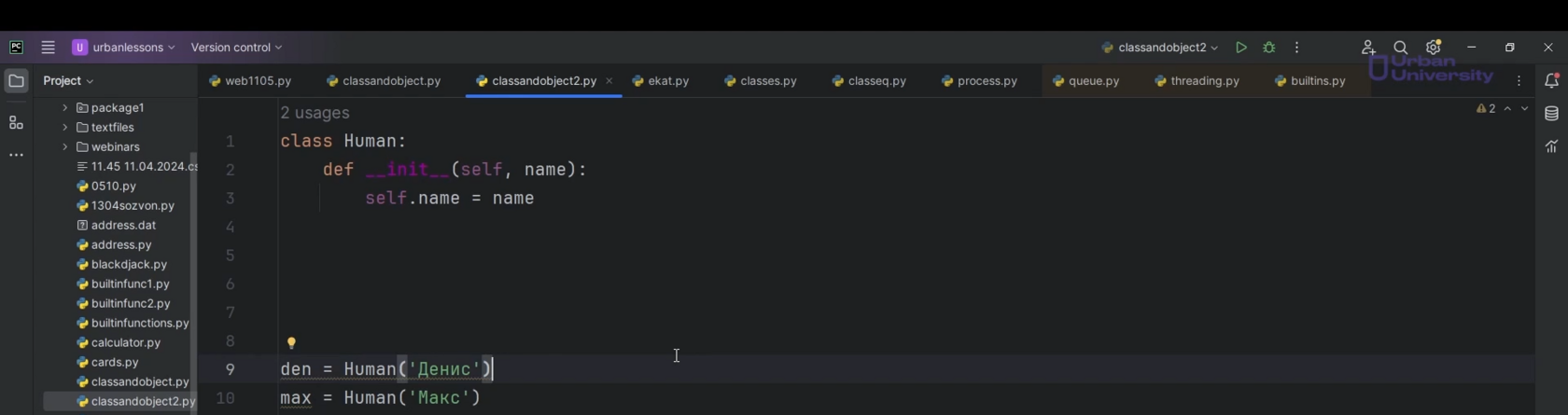


Рис.1

Видим здесь **self**, он является указателем на самого себя, то есть указателем на объект. Представьте, что вместо self(рис.2) тут будет находиться либо den, либо max. Когда что-то применяем по отношению к объекту, допустим, к объекту den, у нас вместо self будет находиться den. Когда применяем что-то по отношению к объекту max, вместо self, у нас будет находиться max. По сути, это указатель на самого себя.

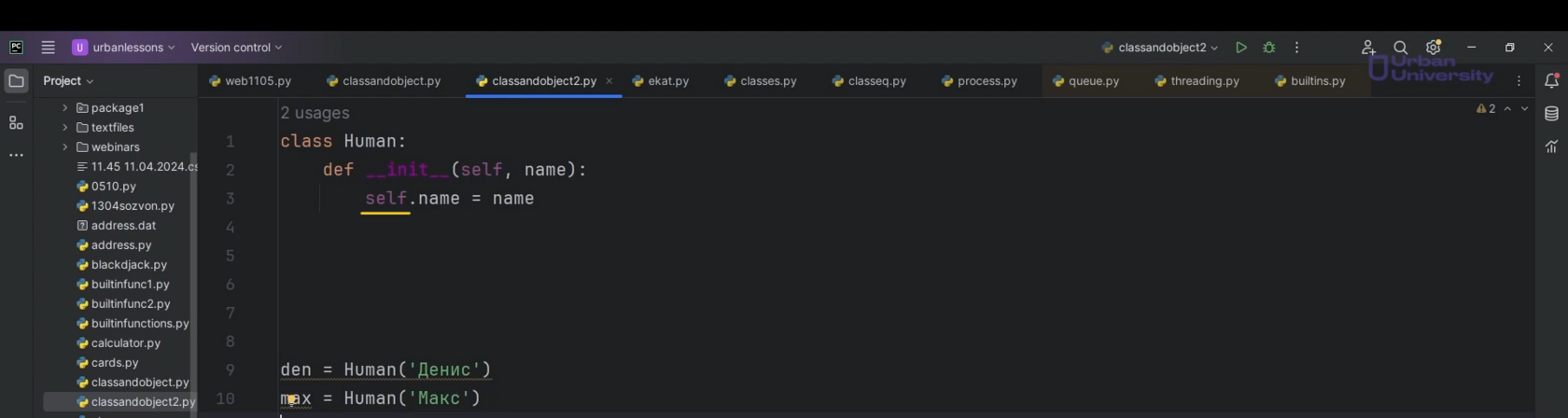


Рис.2

Каждый из нас является человеком и когда мы говорим свое имя, то имеем в виду себя. Иначе говоря, self указывает именно на нас. Однако возьмём другого человека. Представим, что есть ещё, допустим, Максим. Максим говорит: “Моё имя Максим”. Получается self в его случае указывал бы на него самого, так и здесь self является просто указателем на объект.

В нашем случае у нас есть объект den и объект max. Напомним, что **объект** — это, по сути, переменная, созданная на основе класса, а **класс**— это наша инструкция, наш план. В нем описано, какие характеристики и способности есть у наших объектов. Допустим, немного усовершенствуем наш код. У людей есть имена, возраст, значит можем добавить self.age, то есть в init (конструктор или инициализатор) добавили ещё возраст(рис.3). При запуске будем получать ошибку(рис.4), потому что здесь передали только имя.

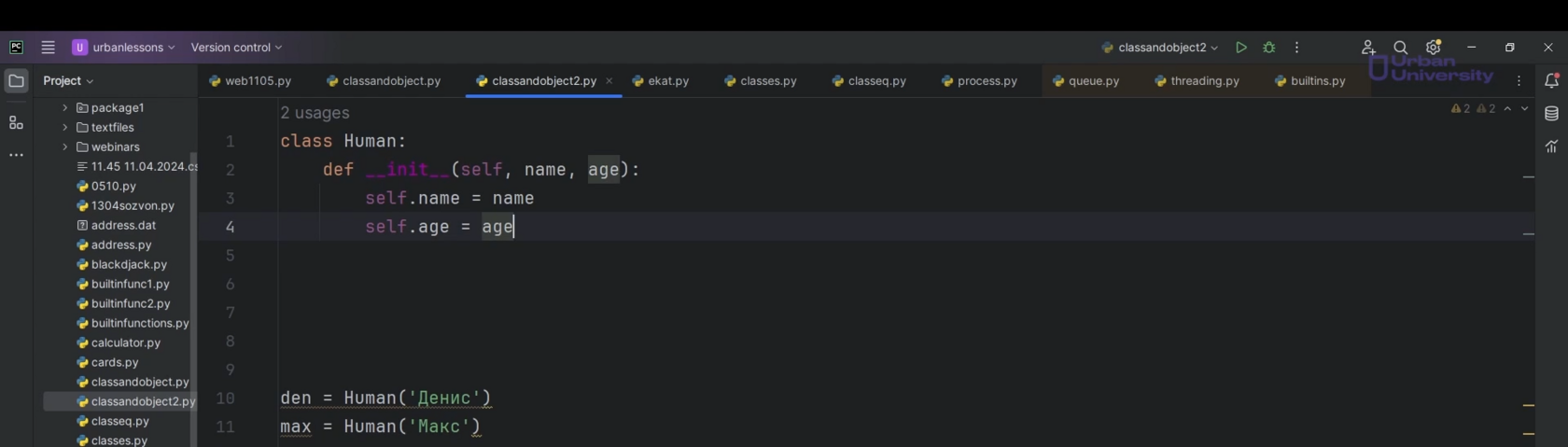


Рис.3

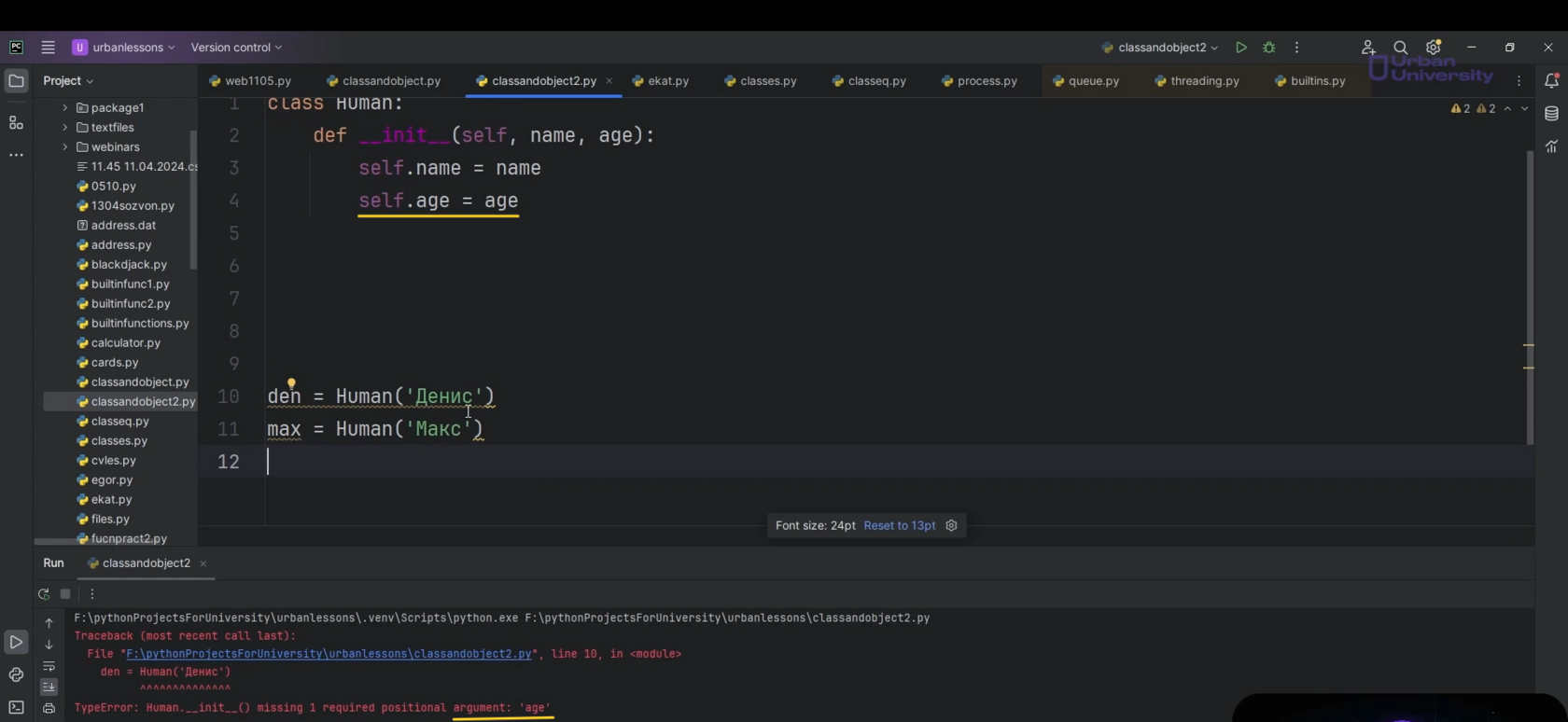


Рис.4

Давайте добавим возраст и в этом случае ошибка пропадает(рис.5). Теперь у нас den имеет 2 атрибута: name и age, max точно также имеет 2 атрибута. Даже несмотря на то, что age одинаковый, что у den, что у max, это не является одним и тем же. Дело в том, что, когда создаются объекты класса, то есть эти переменные, появляется пространство имён. Оно принадлежит именно этому объекту, то есть у den своё имя, у max своё имя. То же самое можно сказать и про возраст.

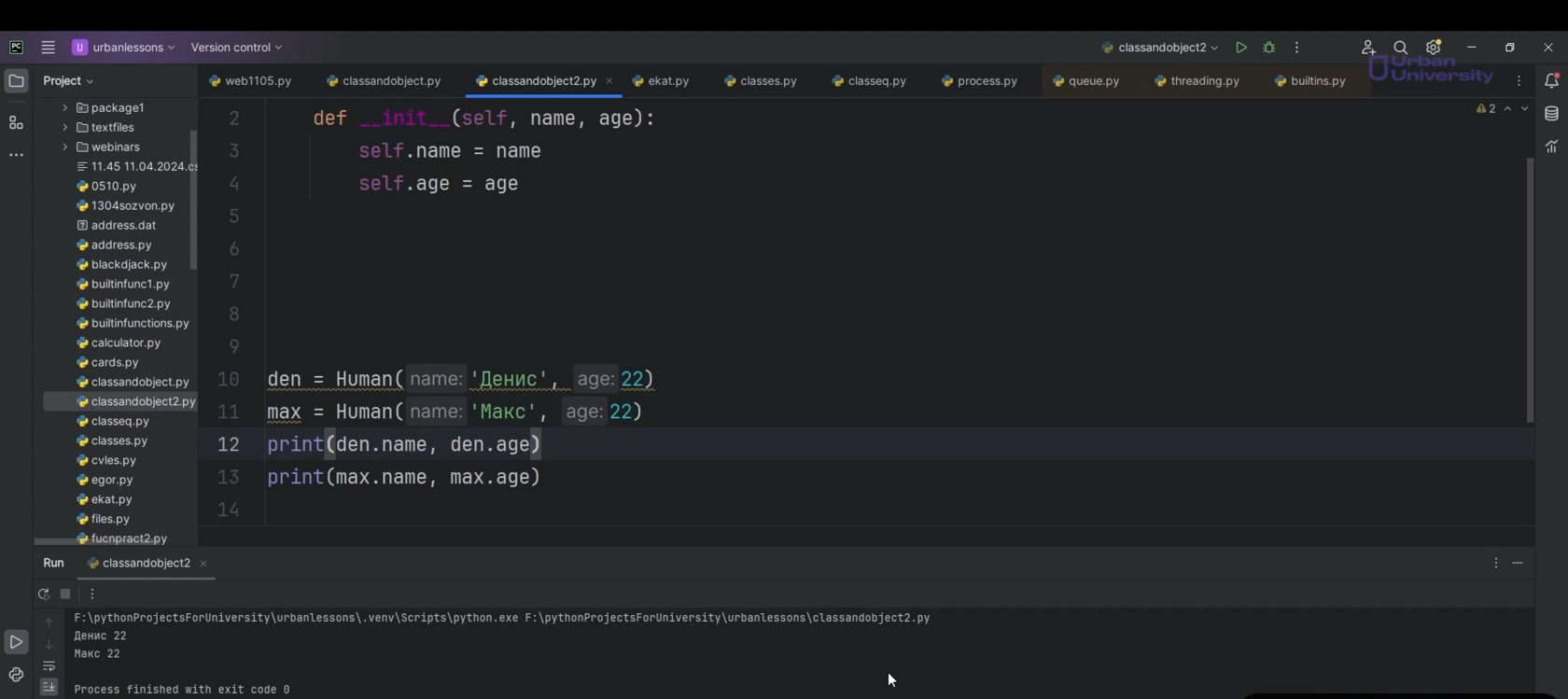


Рис.5

**Эти атрибуты или по-другому характеристики можно задавать и вручную.** Можно написать den.surname и вывести(рис.6). Получается добавили атрибут, который задаётся не при инициализации, то есть не при создании объекта, а просто в процессе работы программы у нас появился ещё 1 атрибут. Однако этот атрибут появился только у dеn. Если возьмём, например, max и захотим вывести у него атрибут surname, у нас ничего не получится(рис.7).

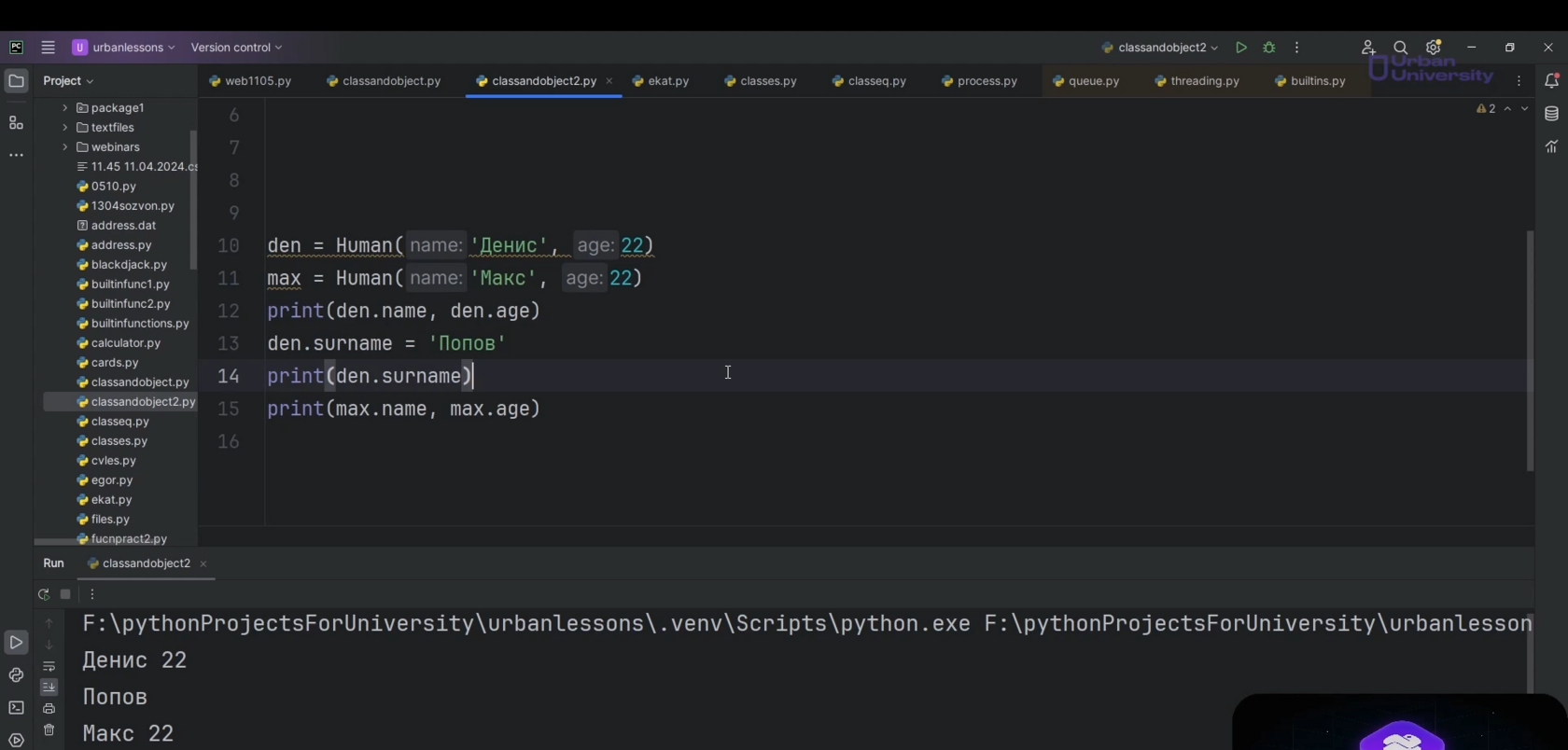


Рис.6

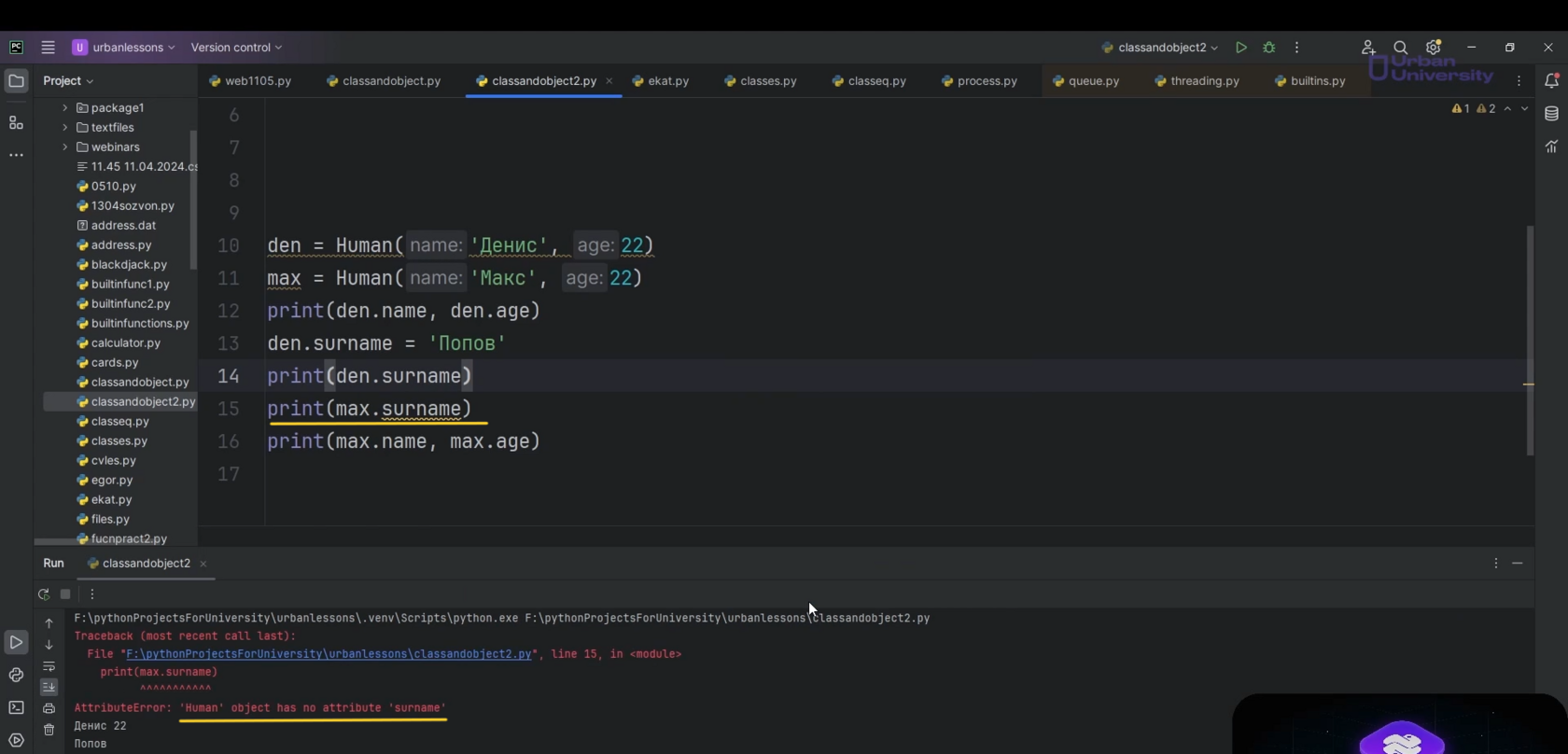


Рис.7

**Мы можем как создавать атрибуты, так и изменять уже существующие.** Если взять и написать, что den теперь 23, то при обращении к этому атрибуту получим уже другой результат — 23(рис.8).

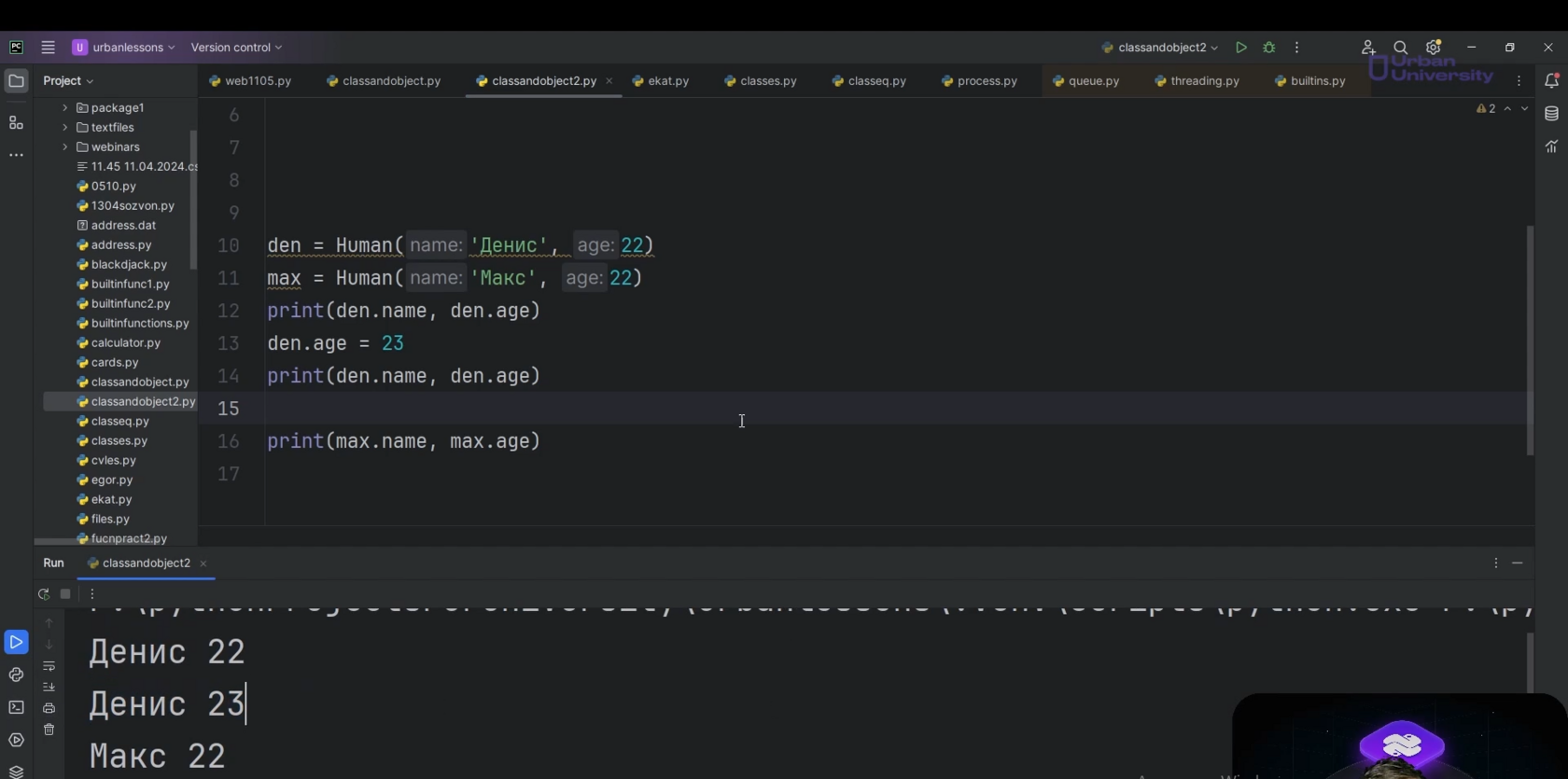


Рис.8

**Можем использовать методы.** Давайте перейдём к тому, что наделим наших людей способностью рассказать о себе. Для этого создадим ещё одну функцию. Назовём её say\_info(). Здесь опять же идёт указатель self, потому что нам нужно будет понимать, кто будет говорить информацию о себе, чтобы она была более индивидуальной. Добавим здесь f строку “Привет, меня зовут {self.name}, мне {self.age}”(рис.9). Теперь каждый объект человек сможет рассказать о себе, то есть у него есть способность say\_info().



Рис.9

**Как это работает?** Можно взять, допустим, den и вызвать на него метод say\_info(). Запустим программу, видим информацию о том что “Привет, меня зовут Денис, мне 22”(рис.10). То же самое можно применить для max(рис.11). Каждый объект класса Human будет обладать этим методом и точно также будет обладать этим набором характеристик. Этих методов, может быть сколько угодно. По сути, мы сейчас учимся создавать свою структуру данных, собственный тип, который будет обладать собственными характеристиками и способностями.

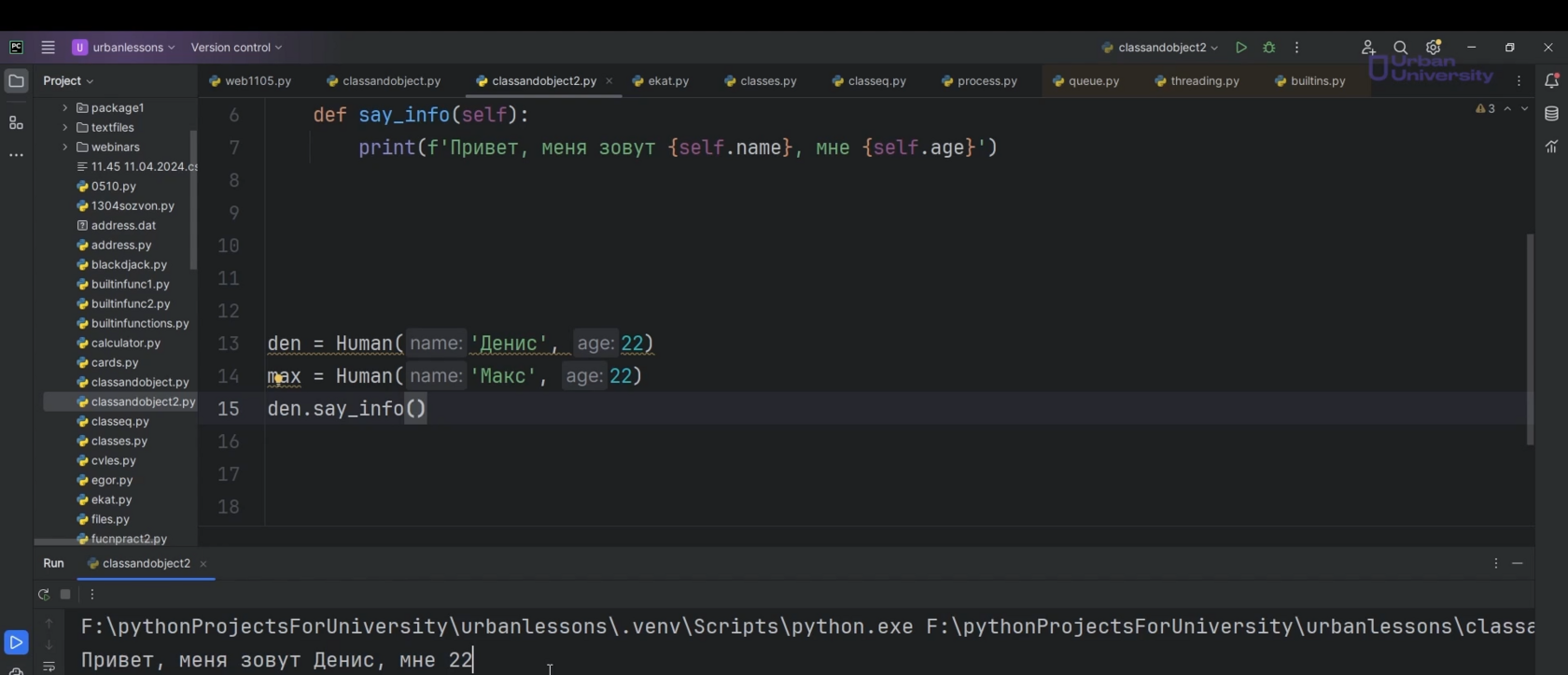


Рис.10

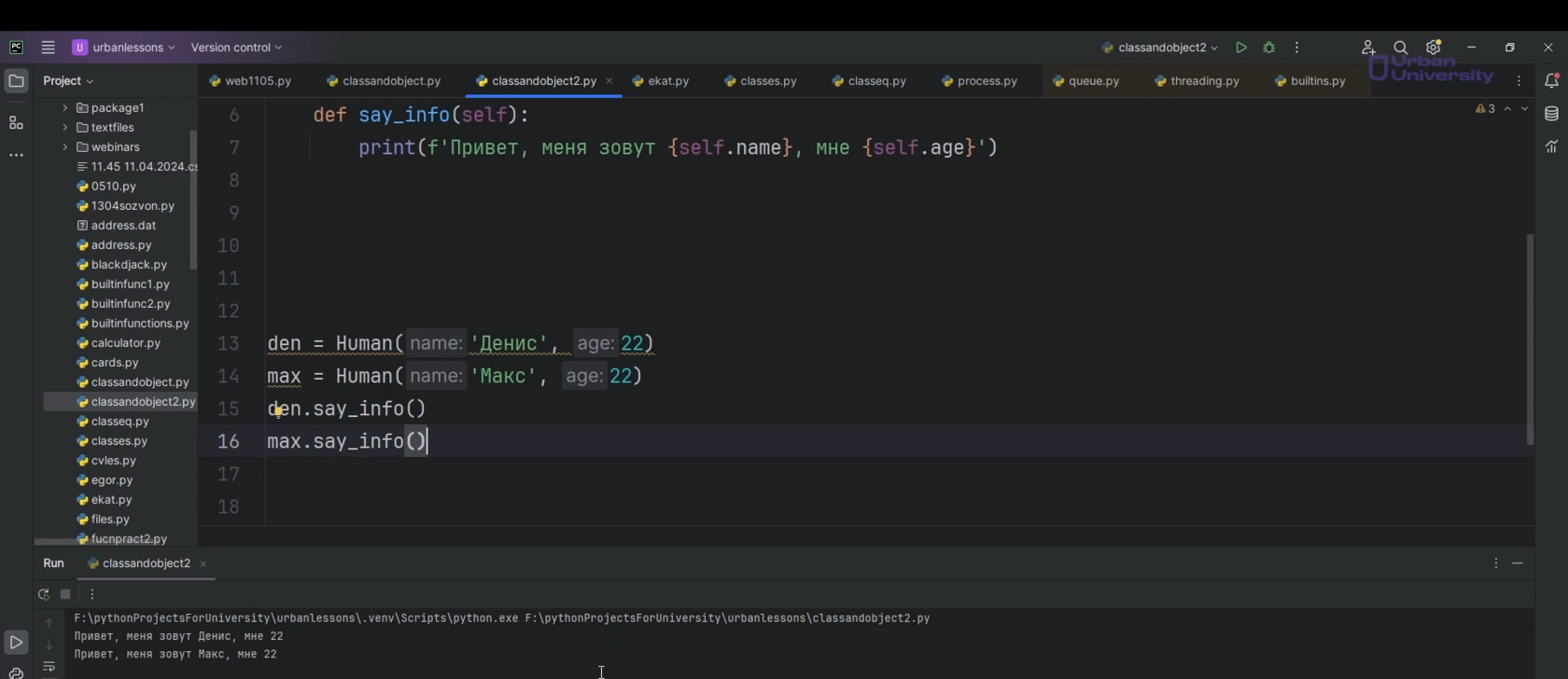


Рис.11

Ранее нам доводилось работать с другими классами, например: целые числа, списки, словари, кортежи и т.д. Про каждый этот тип можно сказать, что, например, у строки есть такая характеристика длина (кол-во символов) и методы, которые позволяют нам получить строку в верхнем регистре, строку в нижнем регистре, а в нашем случае у нас свой тип. Он называется Human и из характеристик это: имя, возраст у каждого объекта и способность рассказать о себе. Мы учимся создавать собственные структуры данных. Можно сделать вызов say\_info внутри init (self.say\_info)(рис.12). В случае dеn, этот метод будет приниматься от его лица. В случае max, этот метод будет приниматься от лица max. Напоминаем, что init срабатывает 1 раз, когда создаётся объект класса. Скобки (name: ..., age: ...) вызывают срабатывание метода init.

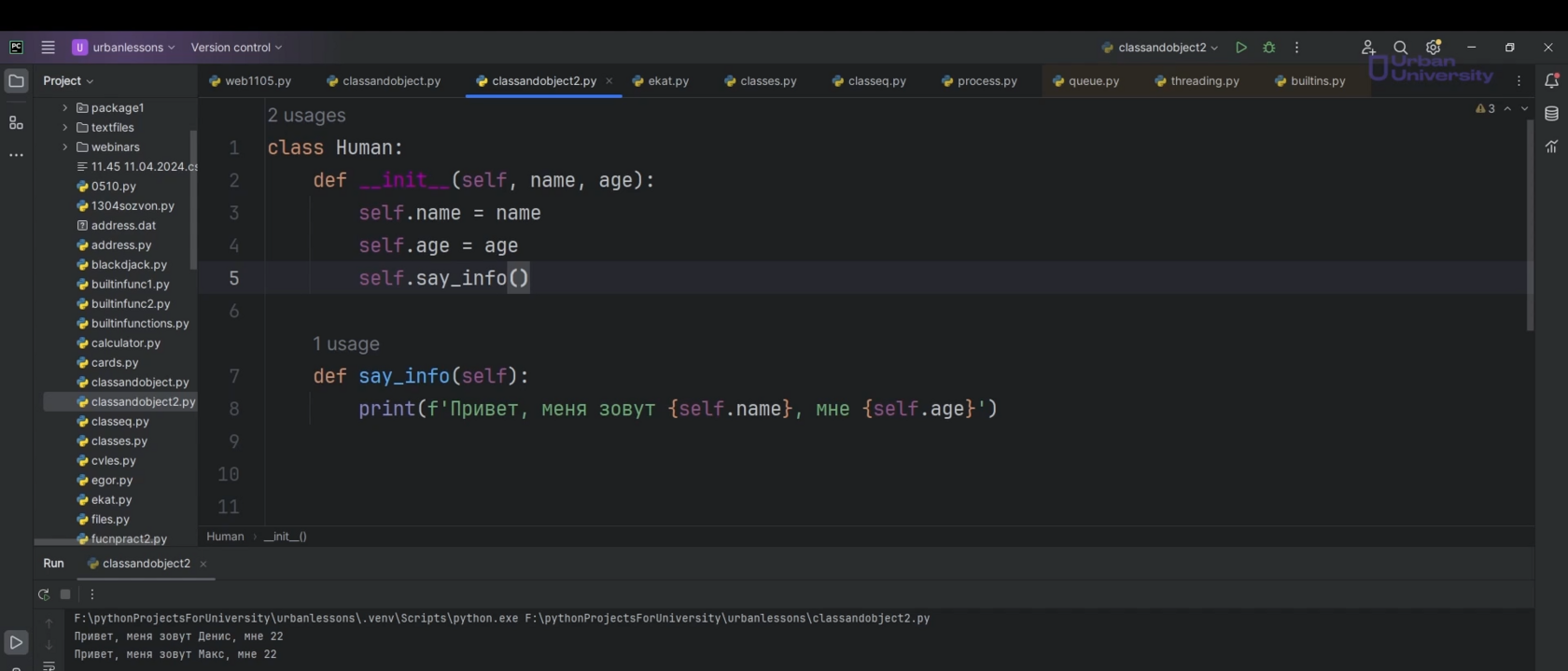


Рис.12

При вызове видим “Привет, меня зовут Денис, мне 22 привет”, “Меня зовут Макс, мне 22”, но нигде не вызывали этот метод. Он вызывается по умолчанию в конструкторе нашего класса. Это значит, что можно в одном методе вызывать другой, а также в одном методе можем изменять их характеристики. Например, можем сделать метод birthday() (день рождения). Он будет увеличивать возраст того по отношению к возрасту того, кто вызвал этот метод: будет браться его возраст, увеличиваться на 1 и выводиться какая-нибудь f строка “У меня день рождения, мне теперь {self.age}”(рис.13). Теперь мы можем, например, взять max, вызвать для него метод birthday(), запустить программу(рис.14).

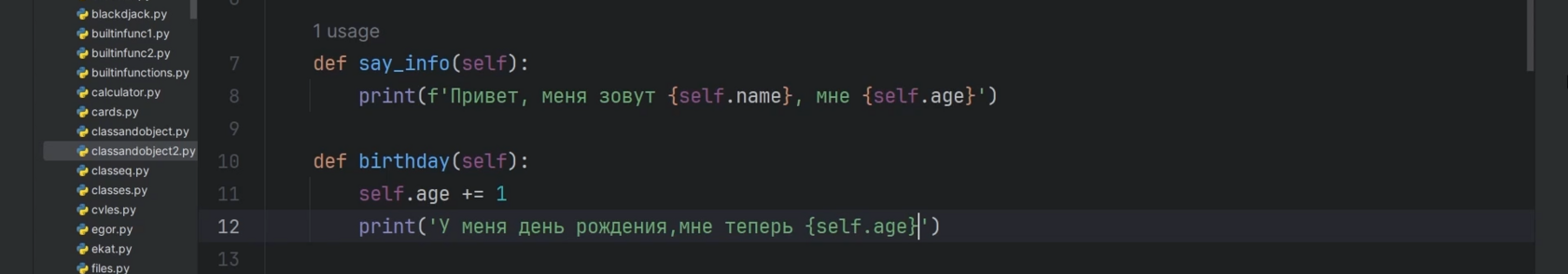


Рис.13

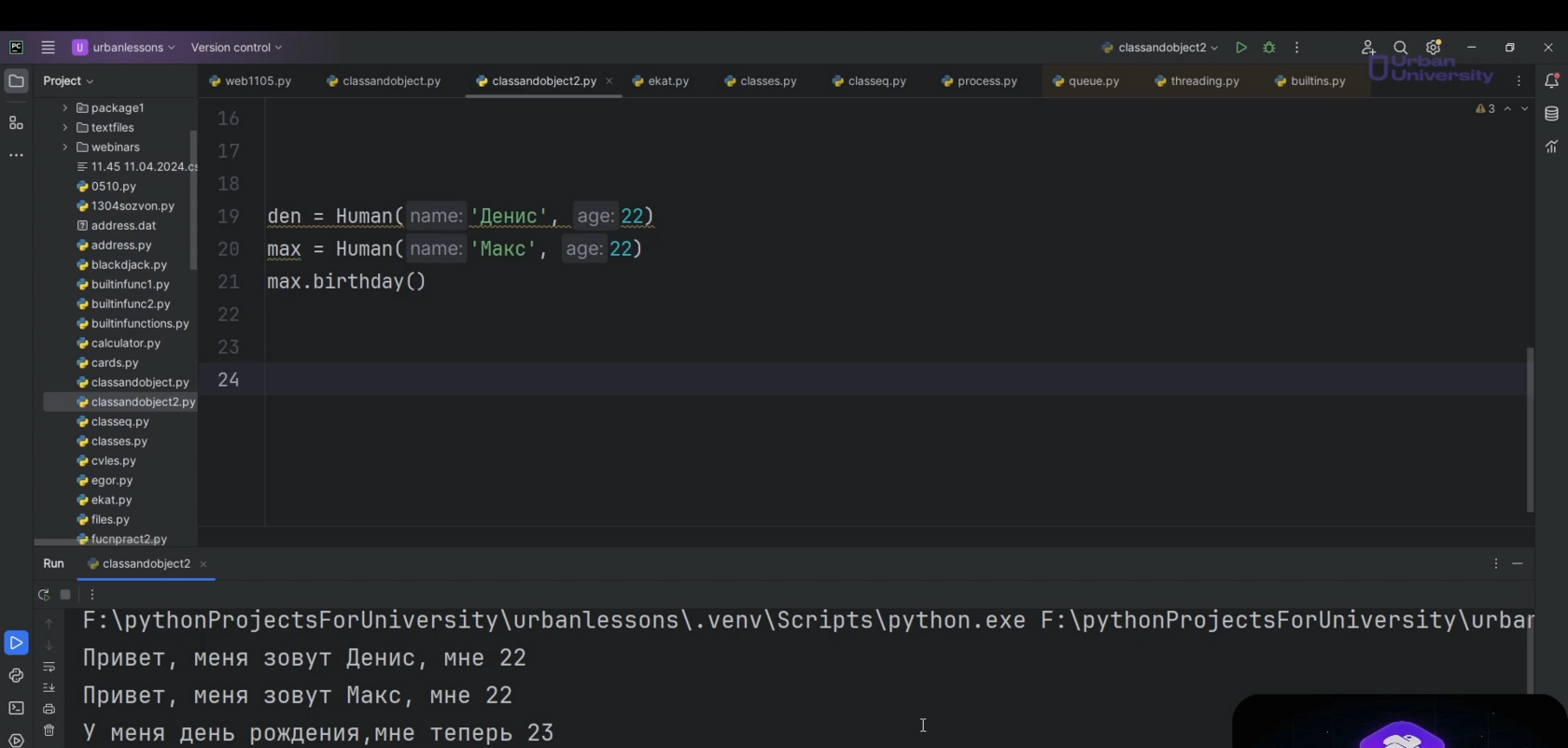


Рис.14

Видим, что, когда он создавался, ему было 22, но после вызова метода birthday() ему стало 23. У нас есть возможность изменять эти атрибуты. Точно также какой-то метод может создавать новый атрибут. Например, birthday() может проверяться возраст. Если возраст достигает определённого значения, будет добавляться какая-нибудь характеристика, наподобие “водительские права”. В принципе, нас ограничивает только лишь наша фантазия.