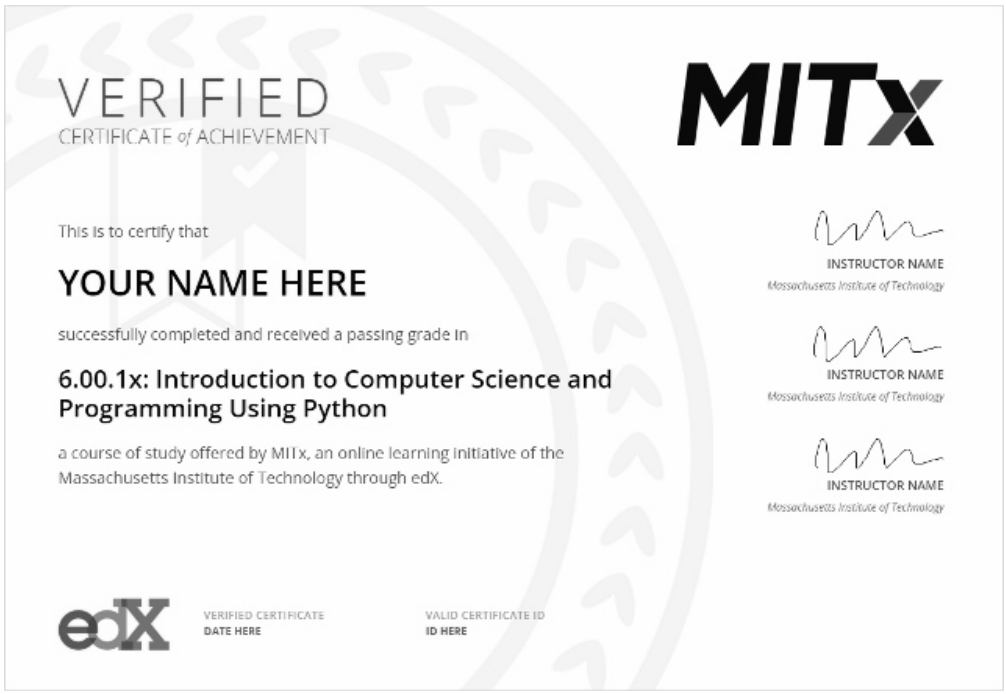
Verified Certificates



Interested in using an edX certificate to bolster a college application or to advance your career? With an edX verified certificate, you can demonstrate to colleges, employers, and colleagues that you successfully completed a challenging edX course, while helping to support the edX non-profit mission. You can register to earn a Verified Certificate directly here.

Video: Object Oriented Programming

Hide Discussion

Topic: Lecture 9 / Video: Object Oriented Programming

Add a Post

All Posts

[Tutorial] Object Oriented Programming: Classes and instances BEHIND THE SCENES!

discussion posted 4 months ago by **Kiara-Elizabeth** (Community TA)

Object Oriented Programming: Classes and Instances

Hi! Welcome to this section! First of all, **congratulations!** You're already halfway through the course. You should be very proud of yourself :-)

In this section we are going to start talking a little bit about classes, what they are and how they can represent real objects.

You will even learn that YOU are an example of Object Oriented Programming (OOP) and you didn't even know it!

So... Let's get started!

Concept of a Class

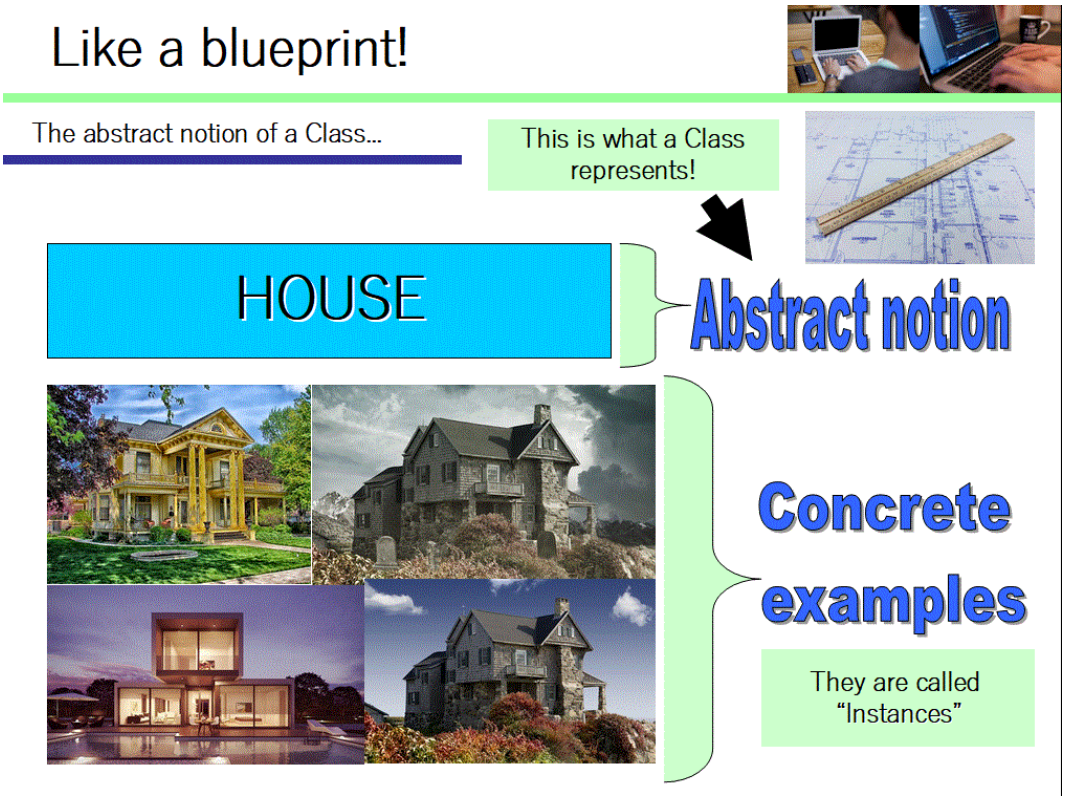
- Classes are like blueprints, we use them to represent an abstract notion or idea of how we describe real world objects according to certain criteria.

For example, House is the notion we have of a structure where people live that has bedrooms, bathrooms, floors, roofs, ceilings, and so on... they share some characteristics, but they don't necessarily have to be exactly the same for us to consider them houses.

This is what we would represent as a CLASS, the abstract notion of a house.

- Now, if we refer to a specific house, **concrete examples**, like my house, your neighbor's house, a house down the street, we are talking about specific or concrete examples of the abstract notion of what a house is. We call these examples **INSTANCES.**

In the tutorial we will go through several examples of these ideas, and at first they may seem too abstract, but I promise you as we go deeper into some examples, it will become much more concrete.



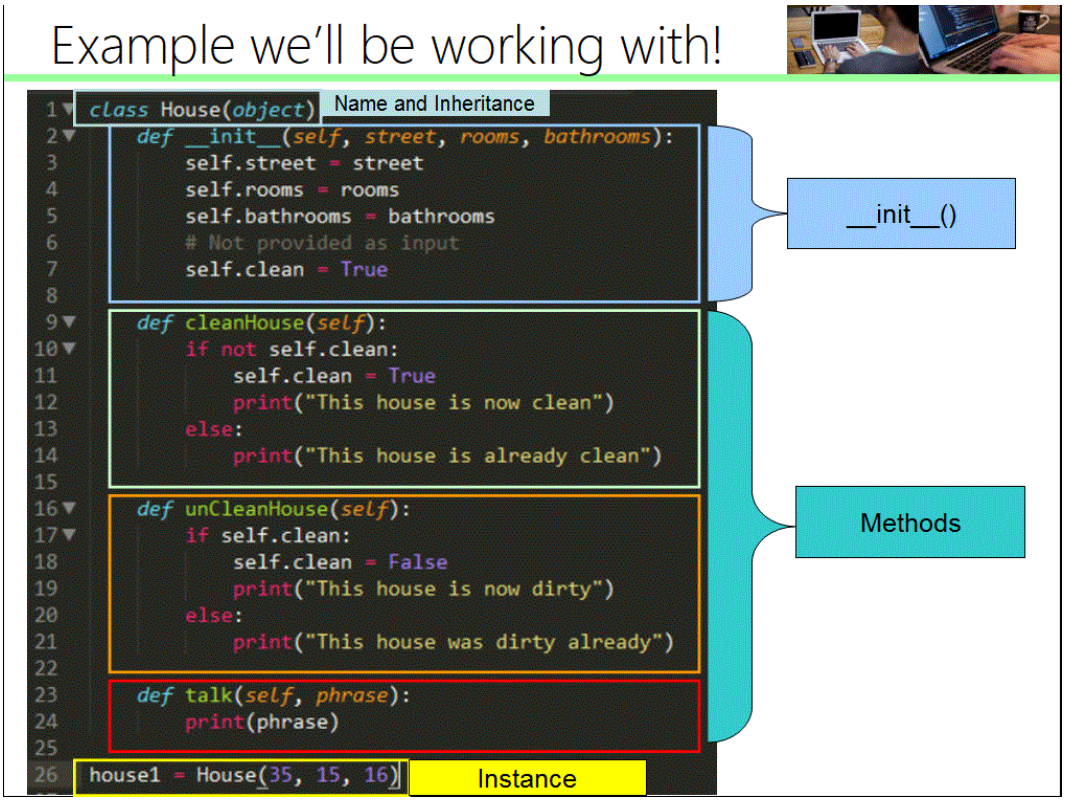
LEARNING CLASSES THROUGH AN EXAMPLE!

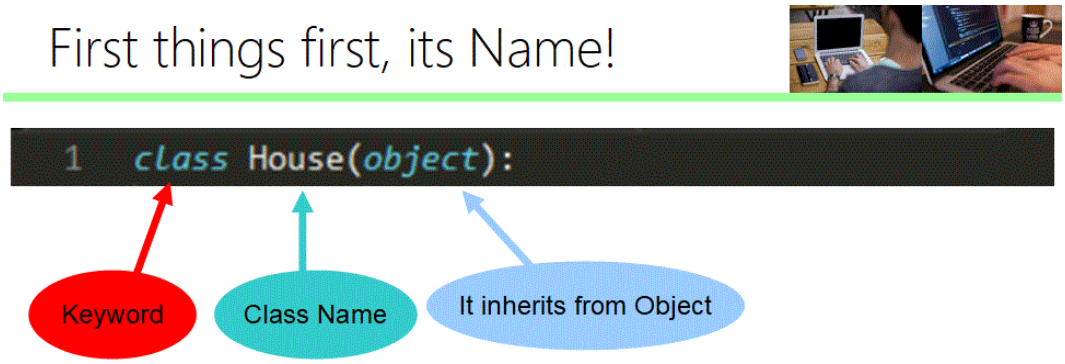
Here I present you the example we will be working with, the one I presented previously about Houses.

(Don't worry at the moment if you don't understand all the code in the example, we'll break it down by small components and discuss every aspect in detail).

- The first line in the example is where we declare that we want to create a class that is named House and that inherits from object (don't worry about this inheritance aspect right now, you will study it in lectures later on and we will discuss it in a later tutorial as well)

`__init__()`



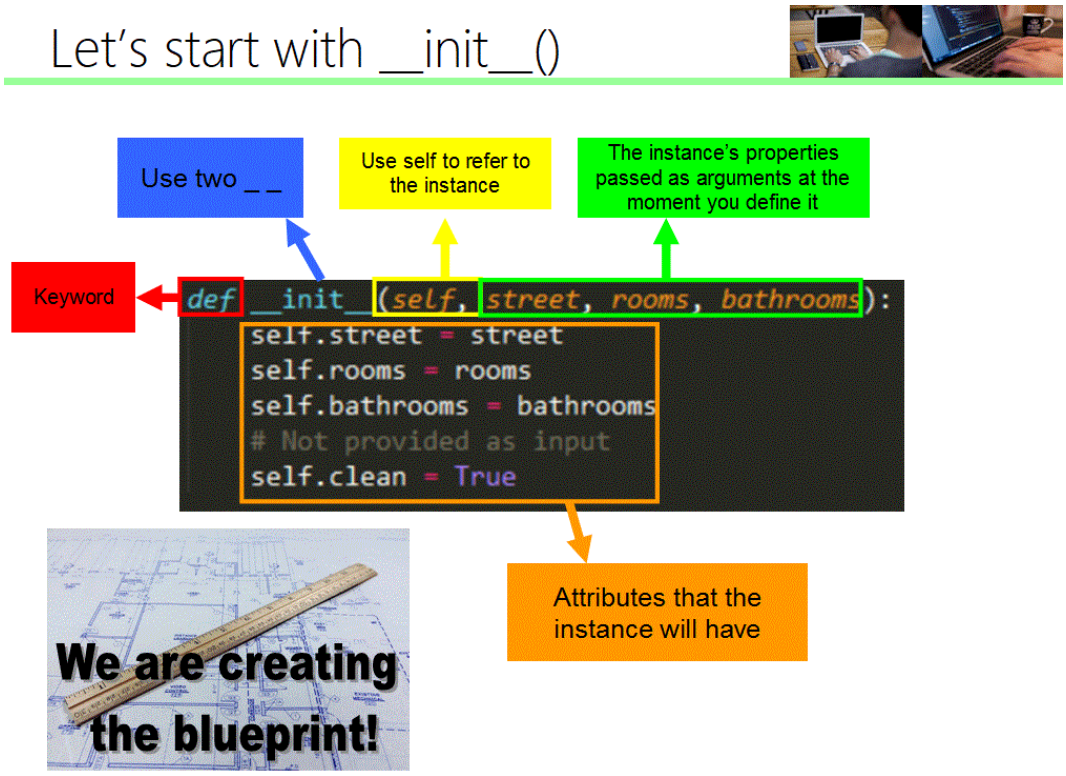


- Now, we first find a very peculiar function with a notation we haven't seen before. The keyword `def` is familiar to us, but using **double underscore** before and after the function's name is new to us. You must use the name **`init`** for this method if you want it to retain all the properties we will discuss.

- You can think of **`__init__( )`** as the function that creates a blueprint for creating instances of a class. This function runs when an instance is created and gives it the properties we have define in this "blueprint"

STEP BY STEP

Let's see what happens with the function `init ( )` when we create an instance.



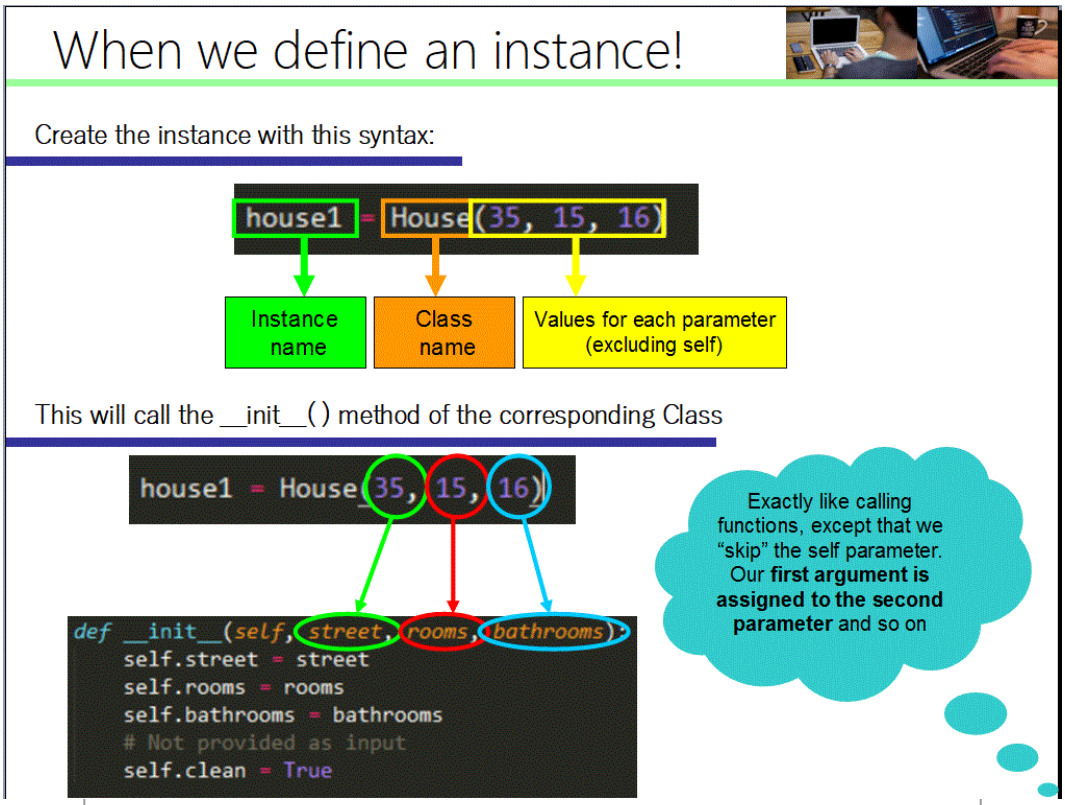
- First, we must create the instance itself by using the syntax depicted in the slide.

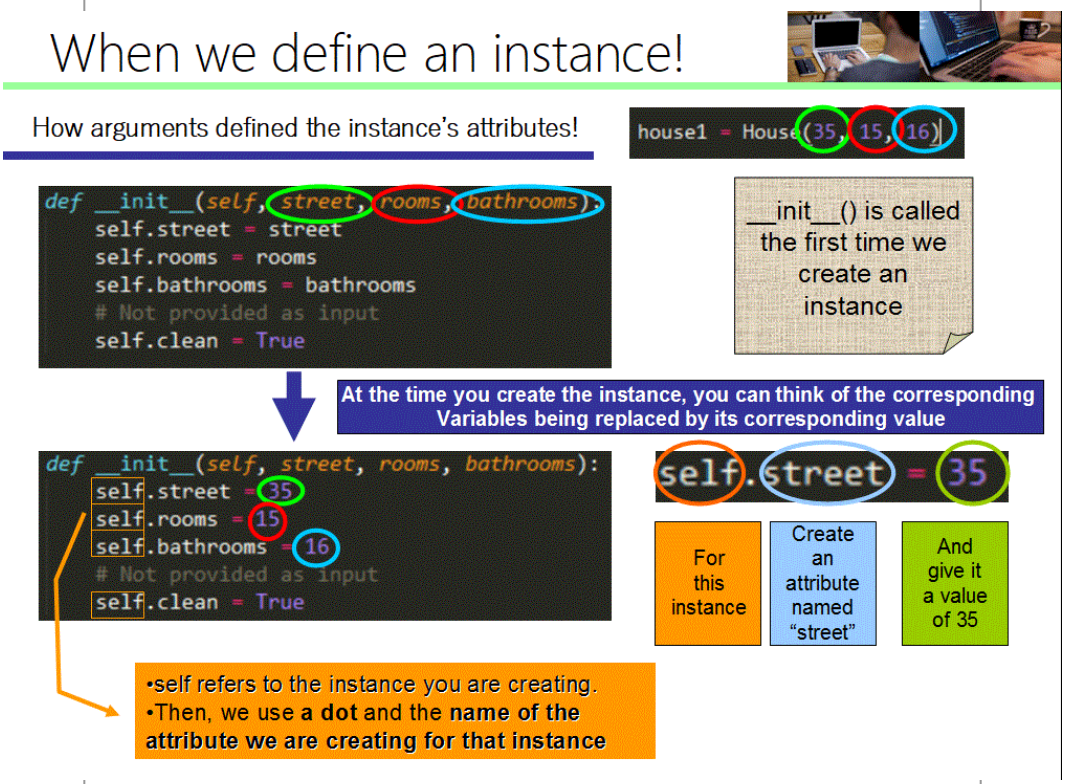
- Then, the function `__init__()` will be called and the arguments we passed in parenthesis will be added to the function scope just like we studied in previous functions.

- We then use the variables added to the function scope and set them as the values for the instance's attributes (right hand side of =)

- We **assign this value to an attribute** we've created for the instance by using:

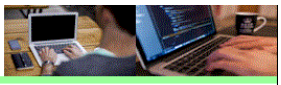
```
self.attributeName = variableName
```





Not all attributes need to rely on a variable passed as an argument! Some may already have a value predefined and all instances will initially have the same value for that attribute.

When we define an instance!



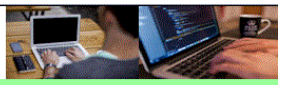
Not all attributes have to be passed as arguments!

```
def __init__(self, street, rooms, bathrooms):
    self.street = street
    self.rooms = rooms
    self.bathrooms = bathrooms
    # Not provided as input
    self.clean = True
```

This attribute already has a value, it doesn't depend on a variable passed as argument.

All instances will initially have the same value for this attribute

Check properties on Python's Shell



To access an instance's attribute

instanceName.attributeName

```
>>> class House(object):
        def __init__(self, street, rooms, bathrooms):
                self.street = 35
                self.rooms = 15
                self.bathrooms = 16
                # Not provided as input
                self.clean = True

        def cleanHouse(self):
                if not self.clean:
                        self.clean = True
                        print("This house is now clean")
                else:
                        print("This house is already clean")

        def unCleanHouse(self):
                if self.clean:
                        self.clean = False
                        print("This house is now dirty")
                else:
                        print("This house was dirty already")

        def talk(self, phrase):
                print(phrase)

>>> house1 = House(35, 15, 16)
>>> house1.street
35
>>> house1.rooms
15
>>> house1.bathrooms
16
>>> house1
<__main__.House object at 0x03FC7690>
>>> house1.clean
True
```

Here you can see that house1 is an instance of House

NOTE: In this case, the values of the attributes have been hard-coded, so instances will all have this value, the parameters are not affecting or changing these values. If you want to pass them as parameters and assign them, use the syntax explained before, assigning the parameter to `self.<attribute>`

SURPRISE! YOU ARE OBJECT ORIENTED PROGRAMMING!

And now we come to the big surprise I promised you at the beginning... YOU ARE AN EXAMPLE OF Object Oriented Programming AND YOU DIDN'T EVEN KNOW IT! :-)

- All humans share some criteria and categories of characteristics and can perform certain actions as individuals that can affect these initial attributes (these are called methods, we will discuss methods in the next tutorial).

We will use this example to show how in depth how instances work

INSTANCES IN DEPTH

We've discussed the class as a whole, so now let's talk about INSTANCES.

- **Instances are specific and concrete examples that belong to a class**.

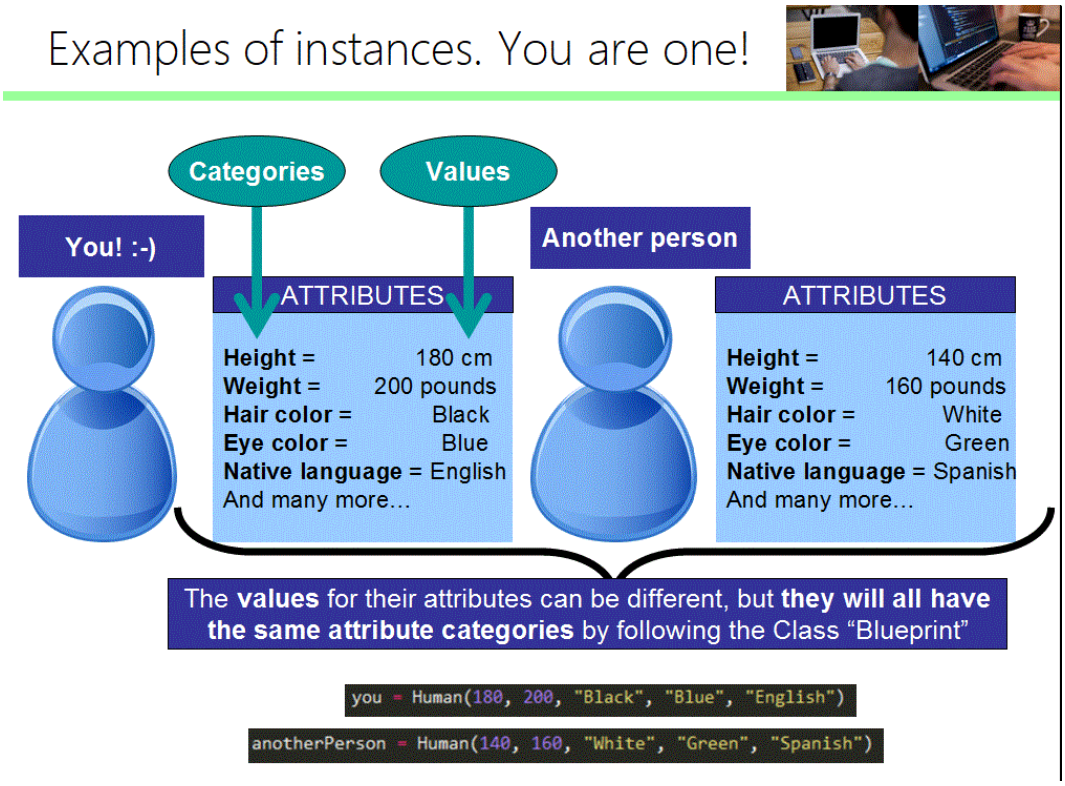
- In the case of the class Human, you are an instance of this class because you share all the attribute categories for the class, you follow the "blueprint" defined by the class and you can perform all the actions (methods) from the class.

BUT... and this is a very serious BUT that you will have to remember to fully understand Object Oriented Programming....

- Instances have their own individual values for each one of these attribute categories defined in the class.

You may have a specific height, weight, hair color, and so on for every attribute defined in the Human class, but another person will not have the same values for these attributes as you.

- This is the point of Object Oriented Programming, **it provides the blueprint to create individual instances that share the same structure but have their own set of characteristics**. They can perform operations on their attributes, change them, and other instances won't be affected.



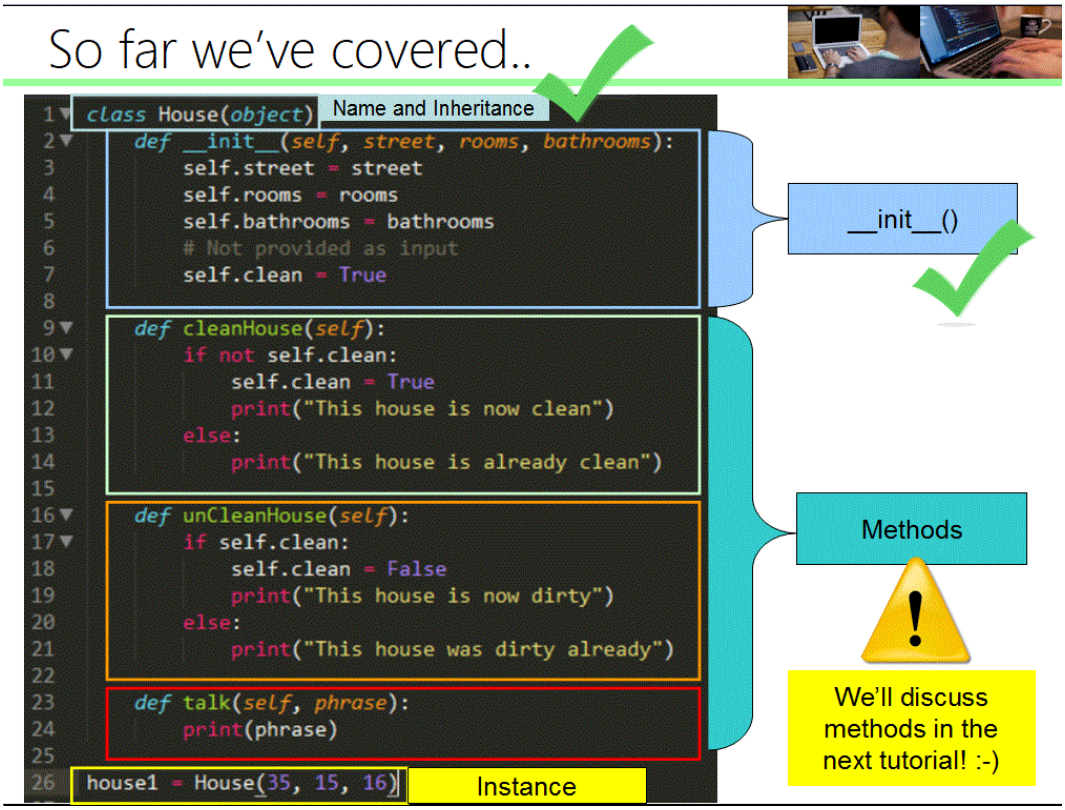
SO FAR… :)

So far we've covered naming a class, the `__init__()` method and the concept of an instance and how to create them. In the next tutorial we will discuss methods and how they are related to classes and instances. STAY TUNED! :-)

NOTE: Check out this link to learn more on the difference between an object and an instance :)

https://stackoverflow.com/questions/3333330/difference-between-object-and-instance

---



Quote from the link:

> A blueprint for a house design is like a class description. All the houses built from that blueprint are objects of that class. A given house is an instance

---

SECOND PART OF THE TUTORIAL!

- Object Oriented Programming: Method BEHIND THE SCENES!

Hope this helps!

If you have any questions, please post them on the forums, community TAs and your classmates will always be there to help you!

Estefania.

P.S = images of characters and houses used in the diagrams were taken from pixabay.com under public domain.

This post is visible to everyone.

Add a Response

3 responses

Old_Youngman
4 months ago

thanks, this really helps!

I'm very glad it helped you 😃

Estefania.

posted 4 months ago by **Kiara-Elizabeth** (Community TA)

Add a comment

qilin_w
3 months ago

```
For some reason that my codes come out as a error:

house1 = House(35,15,16)
Traceback (most recent call last):

  File "<ipython-input-14-4fea420bf090>", line 1, in <module>
    house1 = House(35,15,16)

TypeError: object() takes no parameters

below is the code I copied

class House(object):
    def _init_(self, street, rooms, bathrooms):
        self.street= street
        self.rooms = rooms
        self.bathrooms = bathrooms
```

Can anyone spot any problem which could cause the error message pls?

'init' method has double underscore.

posted 3 months ago by **Sauravsihag**

Thanks very much!

posted 3 months ago by **qilin_w**

Add a comment

LarryW69
3 months ago

There's a graphic that says "`__init__()` is called the first time we create an instance". Isn't it called every time we create an instance? `"the first time" = "whenever"`?

Yeah. Is that not what other languages call a construct method?

posted 3 months ago by **siafanathan**

That's right, this method is called when an instance is created. 👍

- You can find more information on the **init**() method in this article.

Estefania

posted 3 months ago by **Kiara-Elizabeth** (Community TA)

Add a comment

Showing all responses