Programming Foundations with Python:
Learn Object Oriented Programming

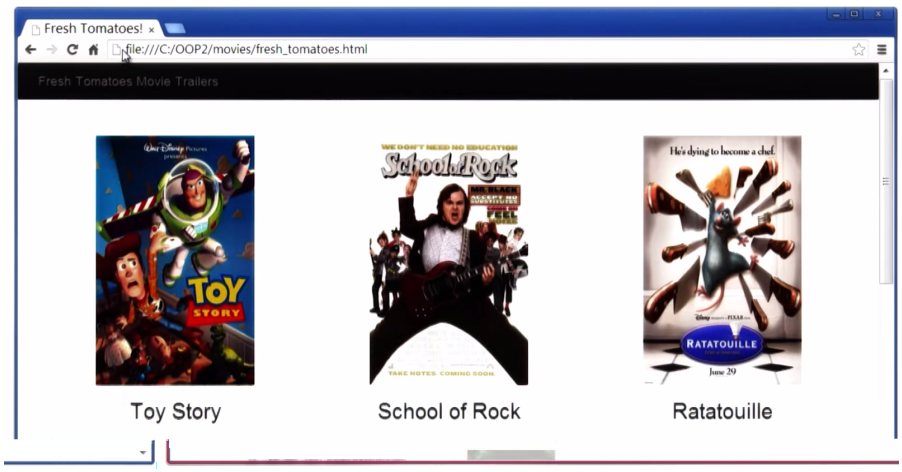
UDACITY

Lesson 03a Notes

Course Map

So we started out this course using functions. In lesson one, we built a couple of different projects. Then, we came across a scenario where using functions didn't quite work really well. This was when we were trying to build a movie website. This led us to lesson two, where we were introduced to classes and we built a bunch of different projects using classes. Here in lesson three we will write our very first class, and we will return to our movie website example. Let's begin.

Now the output of the program we are about to build will look a little bit like this. A



movie website with all of your favorite movies, and if you click on one of them its trailer will play. Let's build this.

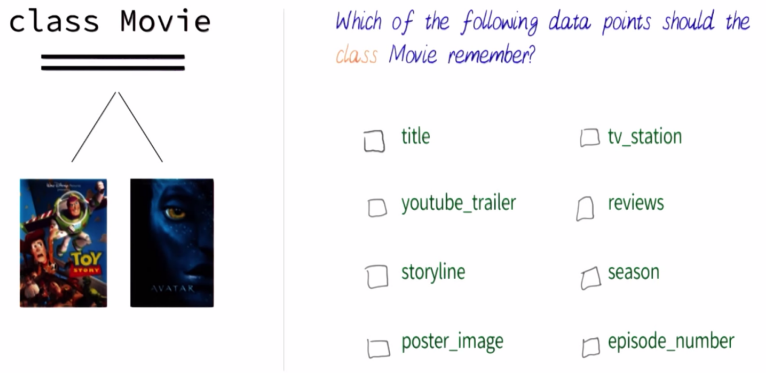
What Should Class Movie Remember

We will use classes to build this movie website. Let us begin, by recalling what we already know about classes.

Remember the blueprint analogy? Previously, we had compared a blueprint, to a class. We had said that a blueprint of a building contains information about the building. Like the number of rooms in the building. The height and the width of the walls. Now, we can use one blueprint to build several buildings. And these buildings we call instances or examples of that blueprint.

Classes are like a blueprint. Now we've seen previously that a class like Turtle contains information about what a Turtle can do. For instance, move forward, turn right. And we use that information to create multiple examples or instances of that class. We call them Brad and Angie that drew different shapes.

In the same way, we now want to create a class called Movie, which will allow us to create multiple instances of itself, instances like Toy Story and Avatar. So then the next big question in front of us is, what data should this class Movie contain?



Now I'm going to present a list of data points for you, that you can consider. Here they are. Which one of the following do you think the class movie, should remember for each movie? Check all of the boxes that you think the class Movie should remember.

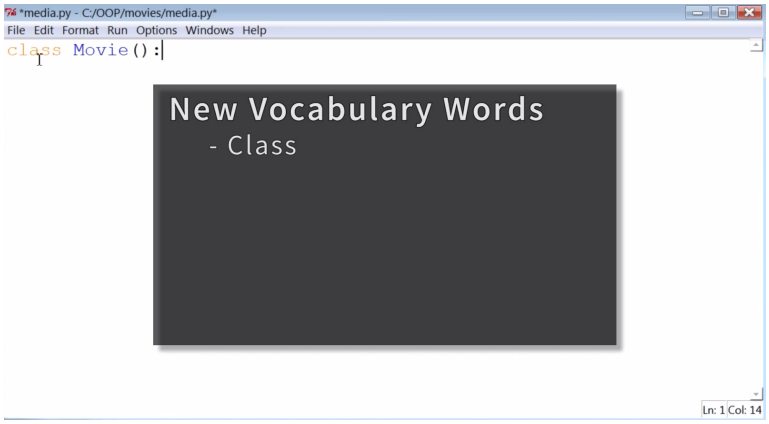
Our Take on Class Movie

So the truth is that the class Movie can contain whatever we want it to contain. And in this case, I will pick the movie's title, its youtube_trailer, its reviews, the storyline and its poster_image. Now that decision will mean, that if I said something like toy_story.storyline. The program will print this -- A story of a boy and his toys that come to life; and if I said, avatar.show_trailer(), the program would play avatar's YouTube trailer. So, in summation, when designing the class Movie, we probably want to remember data like the movie's title, its storyline, its poster image, and the link to its YouTube trailer. And one thing we may want to do with this data, is to have a show trailer function which opens the movie's YouTube trailer.

Okay, let's turn this design, into code.

Defining Class Movie

Okay, so the design of our class Movie, is here on the top right, and I'm going to begin with a blank programming file. I have named it, media.py, and it is saved inside a folder called, movies. Now, to write a new class I will use the Python keyword, class. And I will name it, Movie. You've probably noticed, that the first letter of the name of the class Movie, is in uppercase. Now, this letter doesn't necessarily need to be in uppercase, but, the Google style guide for Python suggests that it should be. Now, what is this thing called the Google style guide for Python you ask, Well, let's Google it. The first link here looks good to me. Now, this Style Guide, provides conventions to programmers all over the world on how to write their programs. If you scroll down a bit you will notice that there are conventions on everything - from how to name things, to how to write comments. By the way, the link to this webpage, is available in the instructor notes. I recommend that you read through this document. If you click on this link called Naming, you will notice that, when defining a ClassName, Google recommends that the first letter of that name be in capital letters. Okay, back to the code.



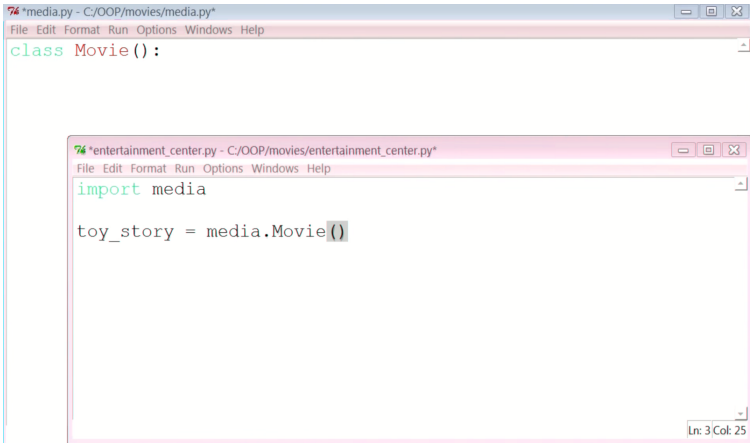
Now, thus far, we've encountered one new keyword, called class. In fact, we are going to come across several new words in this lesson, and I want to list them all, in one place for handy reference. The word class, is the first one on that list. We will review, all the words on this list before we end this lesson.

So, you may have noticed that our class, is empty thus far. Now to figure out what I want to do next with it, I will go back to my design. So, here I am, back with my class design. Now, I know that eventually, I would want to create multiple instances of the class movie. Instances like, Toy Story and Avatar. I also know that I have previously created instances of class Turtle. Instances like Brad, and to do that, I had to write a piece of code that read, turtle.Turtle(), so I guess I will have to write a piece of code like this, to create Toy Story, an instance of class Movie. Let's do that next.

What Happens When

So, I will return back to my code. And here, I will attempt to make an instance of class Movie, in much the same way we did for class Turtle. I will start by creating a new Python file. I will then save this file, in the same folder as my previous Python file. And I will call this new file entertainment_center.py.

So, here are my two Python files that I have created thus far. Here's the first one, where I began to define the class Movie. It was called media.py. And here is is the new file called entertainment_center.py. Now, here in this new file, I will say, import media. Now notice that this is my media file, which is where I was beginning to define my Movie class and therefore, with this line of code, I am telling my program, hey here, I want to use the contents of my previous Python file. Then I will go on to say toy_story = media.Movie(). So, you may notice here that media is name of my previous Python file and Movie is the name of the class that's defined inside that file.



One quick side note here is that I did not have to create a new file. In fact, I could have written this entire piece of code right here in my previous file. But it is good practice to keep the class definition in one file and to call your class, or to use your class from another file. Okay, let's continue.

So, there is one thing I want to highlight here. Which is that this line of code is pretty similar to how we created the turtle brad. In both cases, I am saying, some module or file name dot a class name. Now, the next thing we have to do is figure out exactly what happens when we run this line of code. One way to do that is to remember what happened behind the scenes when we created the turtle brad. So, let's investigate that together.

Do you remember what happens when we run the following line of code?

brad = turtle.Turtle()

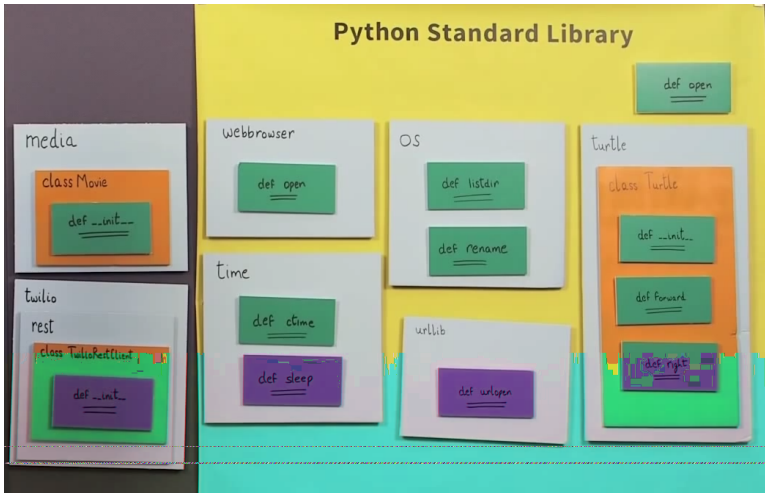
- Function named __init__ is called; it initializes or creates space in memory for new instance brad
- Function named __del__ is called; it deletes the instance brad
- Function named __cmp__ is called; it compares the instance brad to angie
- I don't know

Do you remember what happens when we run the following line of code? Here are some answer choices. A function named init is called; it initializes or creates space in memory for the new instance brad. Or, a function named D-E-L, or del it's called, it deletes the instance brad. A function named cmp, or compare is called. It compares the instance brad to angie, or, I don't know. Tell us what you think.

Where Does Class Movie Fit

So to find out what happens when we run this line of code, we will go back in time and look at a video from a previous lesson when we were drawing shapes.

For now, you can think of a class as a neatly packaged box that puts things together really well. And it allows us to write code like turtle.turtle() with these strange-looking brackets as if we were calling a function. In fact, when we do run this piece of code, we are, in fact, calling a function, a function called init, which is defined inside the class Turtle. Init stands for initialize. And what it does is it creates space in memory for a new instance or a new object of the class Turtle. This instance, we called Brad.



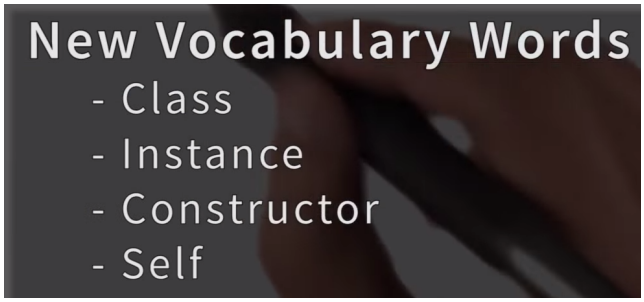
Okay, so we are back to the present time. Now, we started out by defining a file called media.py. Let me place it up on the board. Notice that I place this file outside the Python standard

library. Inside that file, we then defined a class called Movie. After that, we wrote a line of code that said media.Movie(). And what that did, was that it called the init function Define inside class Movie. And that created a new instance. That instance we called Toy Story. So to summarize, this function init initializes or creates space and memory for the new instance Toy Story. By the way, this new function init, is also called a constructor because it constructs space and memory for the new instance, which in this case happens to be Toy Story. So thus far, we've encountered two new words that we can add to our new word list. The first one is instance, which in this case, happens to be Toy Story. And the second one is a constructor, which in this case, is the function init. Now, we will revisit these words soon. But for now, let's define init.

Defining __init__

So, here I am, back at the code, and here I can see my class Movie, and within it I will begin to define, the function, __init__. Now, there is something very curious about this function already, which is that, there are two underscores on both sides of the word init. These underscores are a way for Python to tell us, the programmers, that hey, the name init, is essentially reserved in Python, and that, this is a special function or method. What's special about init? Is that, it initializes or creates space in memory to remember details like title, storyline, et cetera, for new movies that we are about to create.

Now init, takes a few pieces of information or arguments. The first of which, is the word self. Now, what is this thing called, self, you ask? Well, you can think of self as itself, or the instance being created. So, when we run this line of code - toy_story = media.Movie() - in our other Python file, by the way, just to remind you, here we are trying to create a new movie called, toy_story. And when we run this line of code, we know that the __init__ function inside the class Movie gets called. Self, in that case, points to toy_story.



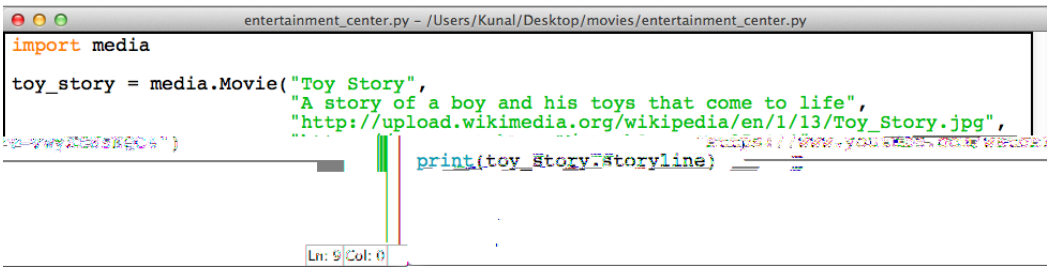
Now here, I have a confession to make. I remember when I was first learning object oriented programming years ago in college. This concept of self, was really difficult for me to understand. If that is the case with you, don't worry about it too much for now. This will get, really clear when we learn this entire piece of coding a few seconds. In any case, let's add the keyword self, to our list of new words.

Okay, I'm going to go back to the class Movie, and continue to define the function, init. Now, we want init, to initialize pieces of information like title, story line and others that we want to remember inside our class. Here's a way to do that. I will say, self.title, then self.storyline. The next one is poster_image_url, and finally, trailer_youtube_url. Now, we have to somehow initialize these variables, with information init is going to receive. And in particular, it's going to receive, four pieces of information. The first of which is a movie title, the next is the movie storyline, then the poster image of the movie, and finally, the trailer on YouTube. So,get this, once I've received these four pieces of information as arguments to the __init__ function, I can initialize my Movie variables. Here's how to do that. self.title, is the movie_title. self.storyline, is the movie_storyline. Next is the poster image. And finally, the trailer on youtube.

Now here, I have a confession to make, which is, that we've written a lot of code that looks new and strange. Well, the one thing I can say is that, we should all just take a pause and a deep breath together. For now, all I want you to do, is pause this video, and write down this code on your computer. All of this code will make sense as soon as we run this program.

Now, there is one key point I would like to highlight, before I execute this piece of code, which is, that if I go to my other Python file, which is entertainment_center.py. And run this piece of code. It will throw an error. The error will happen, because when we try to create the movie, Toy Story, the init function gets called, which is now looking for several arguments or pieces of information, although, we are not sending it any information whatsoever. So let me change that.

Now, I know that first argument for the edit function is, self, which is added by default in Python. So I will skip it. The next one is movie title, which I know is, Toy Story, so I will add that. After that, is the movie storyline, so I will type that in next. A Story of a boy and his toys that come to life. The next one is, the poster image, so I will add that. I had previously gone on Wikipedia and found a link to, Toy Story's poster, so I added that. And finally, the Youtube trailer, which I will add also.

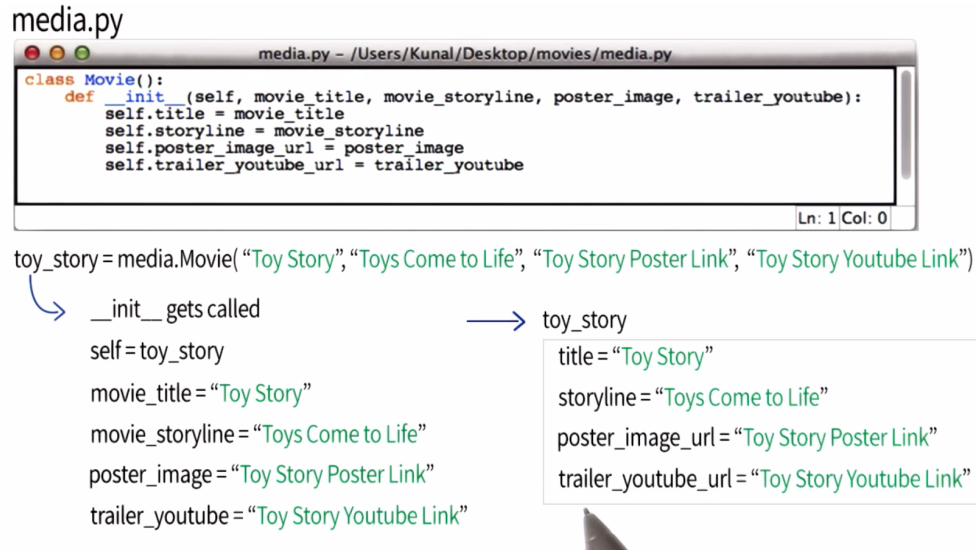


There it is. All right, the last thing I'm going to do here, is print Toy Story's story line. Let me save this file, and then go back to my class Movie file, and save that one also. And then come back again, to my entertainment_center.py file. Now here, if

everything goes correctly, my program should print out, toy_story.storyline. Let's run it. And there it is, Toy Story's storyline is getting printed. Okay, so, in the next video we are going to see what happens behind the scenes when we run this piece of code.

What is Going On Behind The Scene

Okay, so here is the code for the class Movie that we have written thus far. Now, bear in mind, that you may have to go into full-screen mode in order to see all of the details here properly. And I'm really curious to figure out what happens behind the scenes when I run this line of code. What happens when I create an instance called



toy_story by providing these four pieces of information or arguments. These pieces of information are the name of the movie, its storyline, the link to its poster, and the link to its YouTube URL. Now clearly, the last two of these are not really links or URLs just yet. They are English phrases and I did that because there wasn't enough space on the screen for me to add the full links or the URLs to the poster and the YouTube trailer.

So, the first thing that happens when we learn this line of code is that the init function gets called. And the init function, you will recall, is the function we defined inside class Movie. Self, in that case, is itself or the instance being created, which is toy_story. The next argument is movie_title whose value is Toy Story. Movie_storyline gets the value Toys come to life. The variable poster_image gets the right value. And finally, the trailer_youtube variable gets the correct link. Okay, so far so good.

Now, once init gets called and all of these arguments receive their correct values, all of the variables that are associated with the instance toy_story, they get initialized appropriately and these variables, you may notice, are title, storyline, poster_image_url and trailer_youtube_url. At this point, if I try to print out toy_story.storyline, it prints out the correct value.

All right, now that we know exactly what happens when we create this instance toy story. Let's go back to our design and find out what we have to do next.

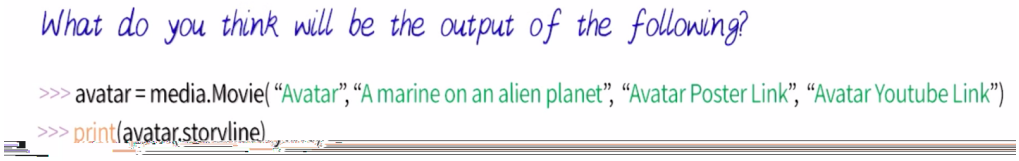
What Will Be the Output

So here we are. Back to our overarching design. And you'll notice that, thus far we've created just the one instance of our class Movie. And that instance is, Toy Story. The next thing for us to do, is create another instance of class movie. And this time we're going to create Avatar. So, to do that, I am back here at my code. Here's the code for the class movie. And, here is my entertainment_center.py file which is where I'm creating my first instance, Toy Story. Now, you may have noticed that I've reduced

the font size of my code, this is so I can squeeze all of the code, in one video frame so you can see it properly.

Next, I will create the instance called avatar, by calling media.Movie. The first piece of information I have to supply, is the name of the movie, which is Avatar. Next is its storyline, which is “a marine on an alien planet.” That is followed by Avatar's movie poster, which I will paste here. And the final piece of information is Avatar's YouTube trailer, which I will also copy in.

All right. So, here is our new instance, Avatar. Let's print out it's storyline next. So I will say, print avatar.storyline. There it is. I'm also going to comment out my previous print statement, which was printing out toy_story’s story line. So, there it is, let me go ahead and save this file. Now before I go ahead and run this program, let me ask you a question. What do you think, will be the output of this line of code right here? Now I

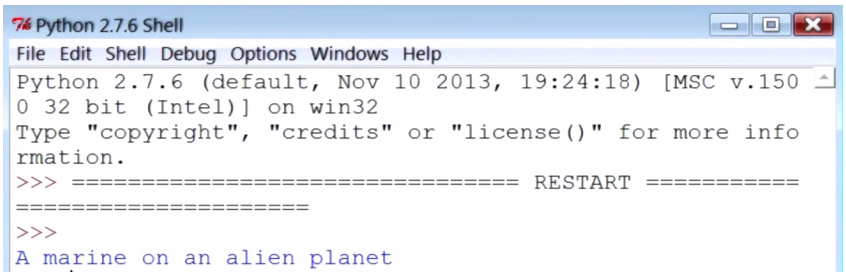


- A love story on a sinking ship
- A story of a boy and his toys that come to life
- You can’t handle the truth
- A marine on an alien planet
- Play Avatar’s trailer

know that the font of this code is really difficult to read. So what I'm going to do, is make it a little bit bigger. So here it is, the code in a bigger font size. Here is where we are creating the instance avatar, and here we're trying to print it story line. So, what do you think will be the output of the following code? Here are some answer choices.

Running the Code

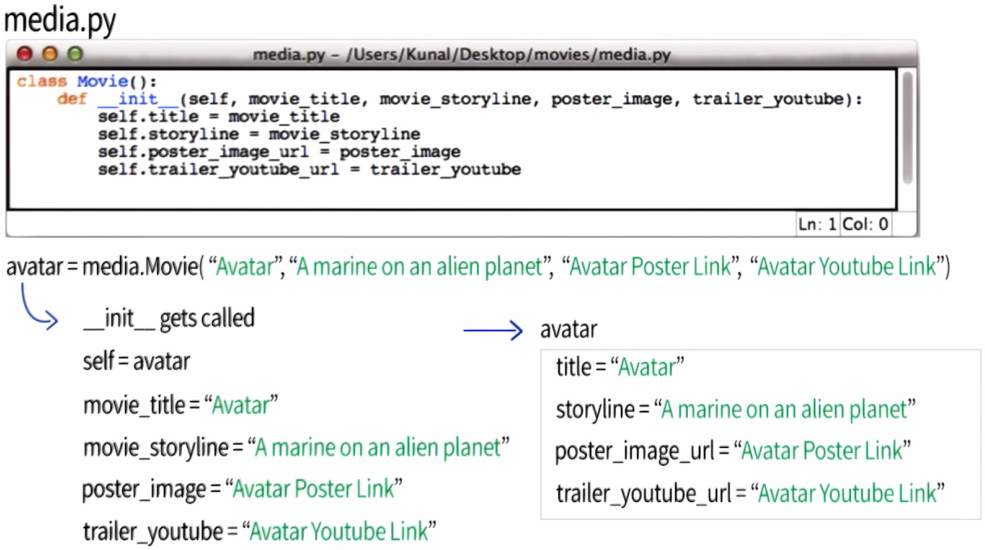
Alright. Let me go ahead and save this code, and then run it. Alright. There it is. The program is printing out the correct storyline, which is for the movie Avatar. Now next, what I want to do, is figure out exactly what's happening behind the scenes when I create the



instance Avatar.

Behind The Scene

Okay, so here is the code for our class Movie one more time, and now, I am really curious in figuring out what happens behind the scenes when I create the instance called avatar by providing these four pieces of information. These, by the way, are the movie's name, its storyline, the link to the movie's poster image, and the link to its YouTube URL. By the way, the last two of these two things are clearly not links or URLs. They sound more like English phrases, and I did that because there wasn't enough room on the screen for me to add the full URLs or links.



So, as soon as we run this line of code, the init function gets called. Self, in that case, is itself or the instance being created, which is avatar. The variable movie_title gets the correct value, which is Avatar. The movie_storyline variable gets the correct value. The poster_image gets the correct value also. And finally, the trailer_youtube_url gets the correct trailer YouTube link. Okay, so far so good. Now, once init gets called and all of these four arguments receive their appropriate values, all of the variables that are associated with the instance avatar, they get initialized appropriately. So, at this point, if I say avatar.storyline, the program will print out the correct value.

Okay, so now, I'm going to take a small step back and tie together a few pieces of information that we've been seeing. Here is our class Movie. And after defining the class Movie, I created two of its instances, toy_story and avatar. I could have created more instances, but for now, I've just created these two.

```
toy_story = media.Movie( "Toy Story", "Toys Come to Life", "Toy Story Poster Link", "Toy Story Youtube Link")
avatar = media.Movie( "Avatar", "A marine on an alien planet", "Avatar Poster Link", "Avatar Youtube Link")
```

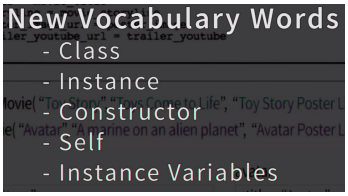
toy_story

```
title = "Toy Story"
storyline = "Toys Come to Life"
poster_image_url = "Toy Story Poster Link"
trailer_youtube_url = "Toy Story Youtube Link"
```

avatar

```
title = "Avatar"
storyline = "A marine on an alien planet"
poster_image_url = "Avatar Poster Link"
trailer_youtube_url = "Avatar Youtube Link"
```

Now, when I created these two instances, what I was really doing behind the scenes, is I was setting aside space for each instance. And within that space, each instance had their own copy of variables. These variables include title, storyline, poster_image_url and trailer_youtube_url. Here are toy_story's variables, and here are avatar's variables. Now, because these variables are unique to each instance of class movie, these variables are called instance variables. And that phrase, instance variables, is another new word that I will add to our list of words.



Is self Important

So thus far what we have done, is we have defined the class Movie and then we went ahead and defined a couple of instances of that class, namely Toy Story and Avatar. Next, what I want to do is I want to run a quick little part experiment which is, what do you think will happen if I remove the word self from in front of one of the instance variables. So let me just go ahead and do that right now. There.

Now I want to think about this question. Here is the code for the class Movie one more time, and this is the self I'm wanting to remove from in front of storyline. And I want you to think about this question especially in the context of running this code, which is print(avatar.storyline). Once you have a hypothesis, I want you to enter it in this box.

media.py

media.py – /Users/Kunal/Desktop/movies/media.py

```
class Movie():
    def __init__(self, movie_title, movie_storyline, poster_image, trailer_youtube):
        self.title = movie_title
        self.storyline = movie_storyline
        self.poster_image_url = poster_image
        self.trailer_youtube_url = trailer_youtube
```

Ln: 1 Col: 0

What do you think will happen if I remove the self keyword from in front of the instance variable storyline and then run the following code?

```
>>> print(avatar.storyline)
```

After you've submitted this response, I want you to make this change on your own computer and see if your hypothesis matches your result.

Removing self

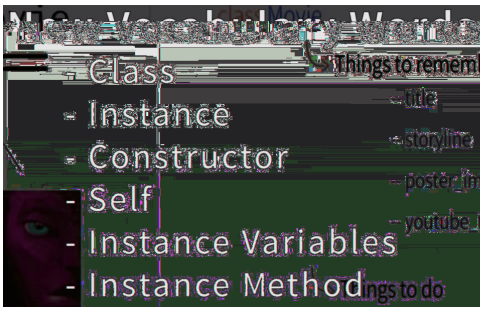
So when we remove the word, self, from in front of storyline, let me go ahead and do that one more time, what we are really doing is changing storyline from being an instance variable, accessible with instances like Toy Story and Avatar, to being a simple local variable inside the function init.

Now, I will still be able to access this variable, storyline, from anywhere inside this function init, but not with an instance like Avatar. And since we are attempting to do that in this print statement, my hypothesis is that Python will throw an error and say that this movie object Avatar has no story line.

So to check that hypothesis, I will save my media file, and then go back to my entertainment center file and run that file. There is the error. Python says that Movie instance, in this case, Avatar, has no attribute called storyline. So, to correct this, what I will do, is go back to the code and add self in front of storyline.

Next Up show trailer

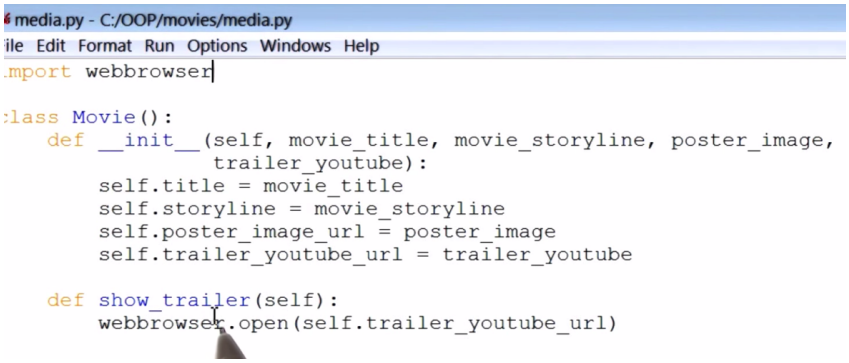
Okay so here I am, at the original design of the class Movie, to figure out the next thing we have to do with our code. So, thus far what we have done is defined the class Movie, and taken care of all of the instance variables we want to remember. We also defined a couple of instances of the class Movie, namely Toy Story and Avatar. The one thing we still haven't done, is define this function show_trailer. Now this function, as it's name suggests, will open a browser and play the trailer of a movie. So essentially what we want to do is run a line of code like this avatar.show_trailer(). And when that runs we want it to play the trailer of the movie Avatar. By the way, a function that is defined inside a class and is associated with an instance is called an instance method. That is another new word for our list. More on these new vocabulary words soon. First, let's define show_trailer.



Playing Movie Trailer

So, here is my code for the class Movie, and behind it, is hidden the other programming file, where we have defined multiple instances of the class movie, namely Toy Story and Avatar. So, what I will do, is go back to my code for the class Movie, and define a new function or method there called,

show_trailer. Now, each instance method, whether it be init or show_trailer takes the first argument as, self. So I will add that. All, show_trailer has to do, is open the web browser, with the correct URL. And the link or the URL, is stored in the instance variable, trailer_youtube_url. The way to access this instance variable, is through self. One last thing I need to make this code work, is to import webbrowser. There. Let me go ahead and save this file.



Now that we have defined the method, show_trailer, the next thing for us to do, is to call this method, using our other programming file. So I will go there, and here, I already have the instance or object called, Avatar. So I will use Avatar to call the show_trailer method. I'm going to go ahead and save this file. And then I will run it. [Avatar's Trailer Plays] Wow, that was outstanding.

Play Your Favorite Trailer

So now that we've played Avatar's movie trailer, here's your challenge. I want you to comment out this line, the line that shows the trailer for Avatar. Then, I want you to create a new object, but this time of your favorite movie, and I want you to play its trailer instead. Here are the instructions one more time. Much like Toy Story and Avatar, we want you to create a new instance or object of the class Movie. Once you've done that, we want you to play a trailer of that movie using the new instance or object. And finally, we want you to share a screenshot of your program's output on the discussion forum. Now, in case you have a way to video record your computer screen, perhaps using a smartphone, you can also share a Youtube video of your output with us. Make sure to check this box after submitting your response on the

forum.

1. Much like Toy Story and Avatar, create a new instance or object of class Movie. (make sure you choose your favorite movie here)

2. Play a trailer of that movie using the new instance or object

3. Share a screenshot of your program's output on the forum
(if you have a way to video record your computer screen, you can also share a Youtube video of your output)

☐ Check this box after submitting your response on the forum
(To access the forum click the "Play Movie Trailer" discussion thread)

Recap Vocab

Thank you for sharing your responses on the forum. Now, in writing all of this code, we've seen a lot of new words. Let's review all of them.

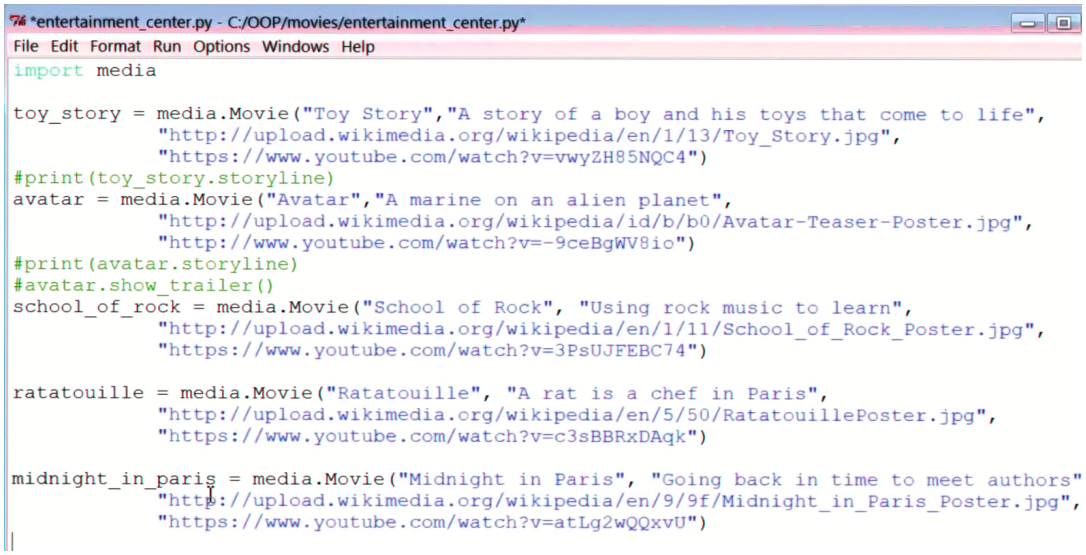
So here I have both of my programming files on the screen at the same time. Here's my first programming file where I'm making the class Movie. And the second one, when I'm defining instances of that class, instances like Toy Story and Avatar. Now, you may have to go into Full Screen mode to be able to read all of the details on the screen.

Okay, let's start reviewing all of the new words. So, we started out by defining a class. The class keyword allows us to make classes. You can think of a class as a blueprint. And a class can have both data and methods. Now, we can create multiple instances of a class. In this case, we created instances like Toy Story and Avatar. We could have created many more. Now, when we create an instance of a class, instance like Toy Story, the classes constructor gets called. This is essentially the init method inside the class. It is here that we initialize all of the data associated with the instance. The constructor uses the word, self. Now, you can think of self as itself or the instance in question. So, when the instance Toy Story is getting created, self is Toy Story. All the variables associated with a specific instance are called instance variables. Now, these are unique to an object and can be accessed using self inside the class and the instance name outside the class. And, finally, all of the functions inside the class that are associated with an instance and have the first argument as self are called instance methods. So there you have it. All of the new words that we have learned thus far, right there on one screen.

Designing the Movie Website

So now that we've learned, a bunch of new words related with classes and objects, we will move on to creating a movie website. And here is the final output that we are after. All of our favorite movies, on a web page. And as you have seen previously, if you click on anyone of these movies, we see its trailer. Let's build this.

So here I am, back with my code. Now thus far, we've made only two movie objects. Toy Story and Avatar. I'm going to make a few more. In fact, four more movie objects. But before I do that, I want to make this window, a little bit bigger, so we can see all of this code really easily. Okay. So here are the two instances we have created thus far. Toy Story and Avatar. Now before I proceed, I will go ahead and comment out the previous output statements that I created. This print statement and the show trailer



method. Now, I'm going to add a new movie object, for a movie called, School of Rock. There it is. You will notice, that I have changed the name of the movie, its story-line, its poster image, and its trailer values appropriately. Next, I will add another movie object. This time, for an animated movie called Ratatouille. Next, I will add a movie object called, Midnight in Paris. Now this, happens to be one of my favorite movies. And the final movie object I will add, is for a movie called, the Hunger Games. Okay, so thus far, we have created six movie objects. So far, so good.

Now, to turn this into a movie website. You will need a piece of code that we wrote. Now we call this code, fresh_tomatoes.py. By the way, this file is available for you to download in the instructor notes. Now before we use it, let's see what this file actually does.

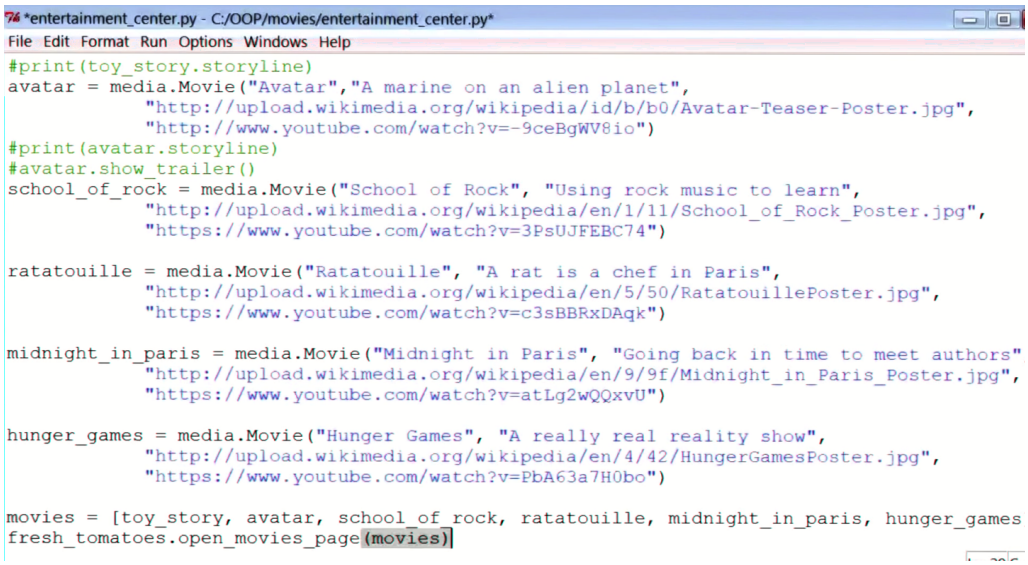
This file, fresh_tomatoes.py, has a function inside it called, open_movies_page. Let's look at this function a bit more closely. Now, what this function does, is that it takes in, a list of movies as input, and as output it creates and opens an HTML page or a website, that shows the movies you gave it in the first place. So what I'm going to do next, is use this file, fresh_tomatoes.py, and it's function, open_movies_page, inside

our code.

Coding the Movie Website

So, the first thing you want to do, is download this file, fresh_tomatoes.py, from the instructor notes. Once you've downloaded it, make sure you save this file, in the exact same folder, as the python file where we created all of these movie instances.

Okay, so I'm going to begin to use the fresh_tomatoes file. And then call the function, open_movies_page. Now, I know that this function open_movies_page takes in a list or an array of movies. I do have a bunch of movies that I have created over here, but I do need to put them all in a list or in a array. Let me do that next. So I will define an empty array or a list called movies. There it is, and to this list, I will add all of the movies I had created previously. So the first movie to add is, Toy_story, next one is Avatar, next is school_of_rock, the next one is Ratatouille. It's spelling

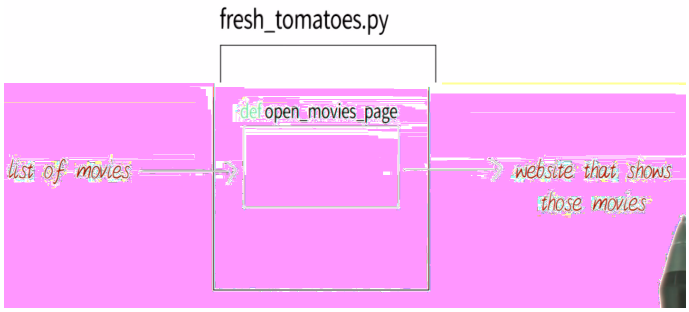


is a little difficult. I think I got it. Next in line is, midnight_in_paris. And the final one is hunger_games. So, there is our movie list. By the way, in case you need more information about how lists work, you can find a helpful link in the instructor notes.

All right. Now that I have a list of movies, I can give this list as input to the open_movies_page function. Let me do that next. So I will provide, movies as the argument to, open_movies_page function. There it is. Now, before I save and run this entire program, I know that I still haven't imported the file, fresh_tomatoes in my program. So, I will scroll all the way up, and import fresh_tomatoes. Notice that, this line of code will work only if the fresh_tomatoes file is in the exact same folder as our current Python file. Okay, time to save and run this program. And there is our movies website. If I click on any one of the links, its trailer will play. [Music Plays]

Movie Website Mini Project

So let me start this video by saying that in case this code did not work for you, ask us questions on the discussion forum. We are here to help. Now, you may be curious about this file called fresh_tomatoes.py, so let's talk about it next. Now, I want you to recall that this file, fresh_tomatoes.py, had a function inside it called, open_movies_page. Now this function, receives a list of movies, and then, creates and opens an HTML file or a web page that showcases those movies. By the way, it stores this HTML file, in the exact same folder, as all of your other movie program files. So let me go to that folder now.



And here is my movies folder, with all of my Python movie files right here. Notice that this is the HTML file that we just created by running our program. This is the file that actually showed us all of our movies and played the trailer. And here is the fresh_tomatoes Python file that you downloaded from our website. Let me open the HTML file. Alright. Here is the code for our Python file, fresh_tomatoes.py. Now, while I won't go through this entire code in detail, I encourage you to read through this program. I want you to understand how it takes in a list of movies and creates a website to show those movies. Now, we've added comments to this file that will make this code readable.

The next thing we want to do is we want you to share the movie website that you've created thus far. So here's your task, we want you to create your very own movie website with your favorite movies. Once you've done that we want you to share any one of the following two things with us on the forum. You can take a screenshot or an image of your movie website and share that with us on the forum, or if you have access to a smartphone we want you to record and share a YouTube video of you showing your movie website to a friend. Once you've submitted your responses on the forum, make sure to check this box.

Comfort Level

Thank you for sharing your responses with us. I look forward to going through all of them on the discussion forum. Now that we've finished building our movie website, I want to ask you this question one more time. How comfortable do you feel in your current ability to write computer program? Here are some answer choices.