

Lesson 02c Notes

Course Map

So, we've seen two ways in which we can use classes from within the Python standard library. Let's look at another example of how classes can make things easier.



Profanity Editor (Story)



Today was an embarrassing day. At 8 a.m., my boss sent out an email to everyone in the company, asking for volunteers to help him proofread some marketing documents. Since I had some time available, I replied by saying, I can take a shot at it. Or at least, that is what I wanted to say. I misspelled one key word in that sentence. And instead, wrote something really embarrassing. Why do the i and o keys have to be so close to each other on my keyboard? I was so embarrassed. I wish my spell checker caught curse words to alert clumsy writers like myself.

Let's write a program that detects curse words in a text. But before we do that, let me ask you, have you ever been in a situation like this? Have you ever sent or received a message that you felt was embarrassing? If so, we would love to hear about it. You can share your experience with us on the discussion forum. Check this box after you've submitted your stories.



Have you ever sent or received a message that was embarrassing?

- if so, we would love to hear about it
- share your experiences with us on the forum

Check this box after submitting your response on the forum
(To access the forum click the “Embarrassing Situations” discussion thread)

Profanity Editor (Output)

Thank you for sharing your stories. So, here is a document with some text that I want to check for profanity. Now the program that I've written, it reads the text from this document, and then scans it for curse words. And when I run my program, which by the way is hidden behind this graphic, it says that this document has no curse words. But if I go back to my document, and add a curse word, there it is. Let me save this document. And now if I run my program again, it detects the curse word. Let's build this program.

Planning Profanity Editor

Now, before we write this program, let me ask you. If you were writing the Check Profanity program, what steps would you take to get to the output? Now, don't worry about writing code right now. In simple English, just describe the steps you would need to take to get to the output. Once you have submitted your responses on the

forum,
make
to check
box
before
move

sure
this

you
on.



*If you were writing the “Check Profanity” program,
what steps would you take to get to the output?*

Check this box after submitting your response on the forum
(To access the forum click the “Planning Profanity Editor” discussion thread)

ts Reserved.

Way of Doing Things

1) Read Text

2) Check Text

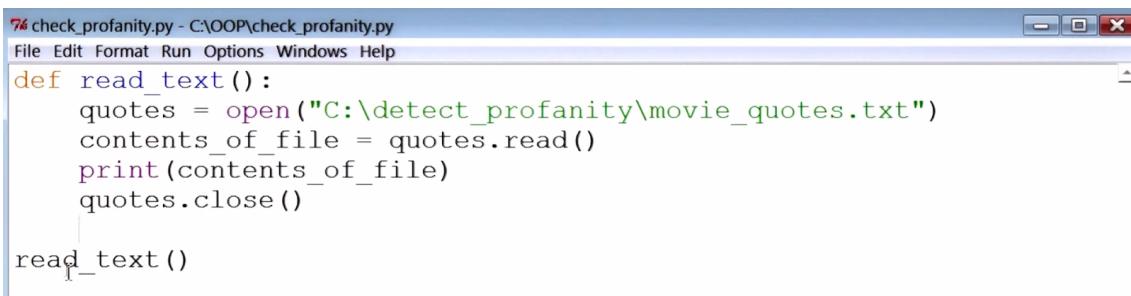
for Curse Wo

So in my mind there are two main steps in making this work. In the first step, we have to read text from a document. So let me write that down. Read text. And in the second step what we have to do is check this text for curse words. So let me write this down also. Check text for curse words. Lets begin by reading text from a document.

Reading From a File

So, here is our plan for the project. I will start with step one, which is reading text. Now, here is the file I want to check for curse words. For now, I have some famous movie quotes in here. By the way, this file could have contained some other text, like the draft of an email or an article you wrote. I have named this file movie_quotes.txt. Now, if you want to use this [exact file](#) in this program. It is available for you to download in the instructor notes. But you should feel free to use any other text file on your computer as well.

Okay, let's start writing code. Now, I created a program called check_profanity.py. You could have called it something else as well. Inside it, I created a function called read_text, which is empty for now. Now, I know there is this function in Python called open, which allows you to work with files on your computer. This function takes in the address or the location of the file you want to open. So, on a Windows machine, I will browse to my file, which is movie_quotes.txt. Copy its location. Paste that. And then add the filename I want to open, which is movie_quotes.txt. Now, if you're on a Mac, there is a [helpful document](#) in the instructor notes for you. That document will tell you how to get the location of this file on a Mac. I will refer to this file as quotes.

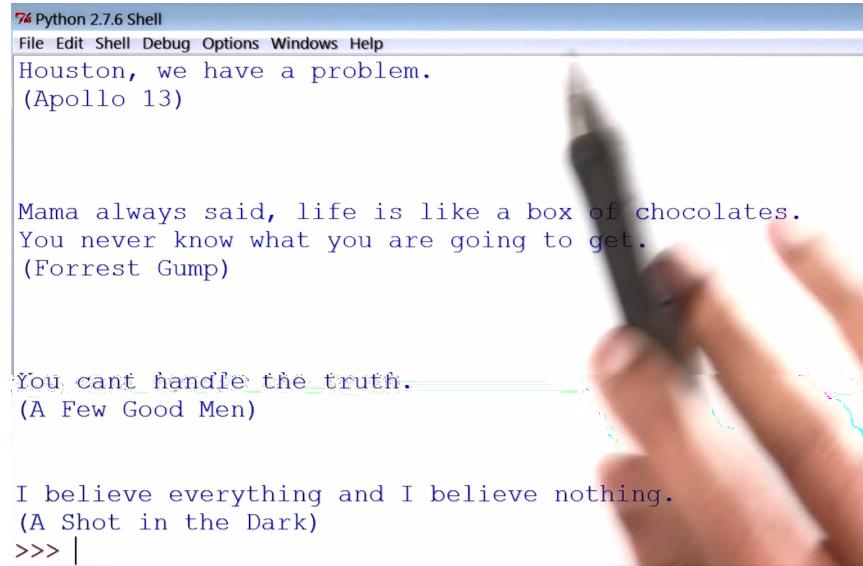


```
74 check_profanity.py - C:\OOP\check_profanity.py
File Edit Format Run Options Windows Help
def read_text():
    quotes = open("C:\detect_profanity\movie_quotes.txt")
    contents_of_file = quotes.read()
    print(contents_of_file)
    quotes.close()

read_text()
```

Now that I have a way of accessing movie_quotes.txt, I will use a function called read. So, I will say, quotes.read(). This will allow me to read the contents of movie_quotes. So, let me save that in a variable called contents_of_file. Let me print out those contents. And it's also a good convention to close out any file that we've opened

through the program. Okay. Now, I'm going to save and run this program and see where it gets us. And boom. Here is that output window with all of the movie quotes that we had read from the text file. Now, I don't know about you. But it just blows my mind that we can read from a text file with only a few lines of code.



The image shows a person's hand holding a pen, pointing towards a computer screen. The screen displays a Python 2.7.6 Shell window. The menu bar at the top includes File, Edit, Shell, Debug, Options, Windows, and Help. The main text area contains three movie quotes:

- Houston, we have a problem.
(Apollo 13)
- Mama always said, life is like a box of chocolates.
You never know what you are going to get.
(Forrest Gump)
- You can't handle the truth.
(A Few Good Men)

Below the quotes, there is some faint text and a cursor symbol.

Okay, so I am back at our program. Now, before I continue to execute any more of plan. Let's find out a bit more about this function called open.

Place Function Open

So I'm holding in my hand, open, the function that we just used in our code. Now, if you were to take a guess, where would you place open, in our model of the Python Standard Library, would it be inside OS, or would it be in a space all by itself, or would it be

Python
Library?
you place

Where in our model of the Python Standard Library should I place the function open?

outside the
Standard
Where would
open?

- Inside the module os
- In a space of its own inside the Python Standard Library
- Outside the Python Standard Library
- I don't know

Where Does Open Come From

Now the first thing to notice about open is that we did not import anything in our code to this function. In contrast, when we had to use a function like rename, we said, import os, but not with this function. This is because functions like open are so commonly used in Python that they are always available. Don't take my word for it. Let's look for it in the Python documentation.

Reading open Documentation

So, here's the Python Standard Library. Now, if I scroll down a bit, I will come across a set of functions called the Built-In Functions. If I click on them, I'll find a bunch of functions which are always available in Python. And one of those functions is open. If I click on open, the documentation says that it opens a file. Which is exactly what we did in our program. And if I go on to read, it suggests that open returns an object of the file type. Now this word object is extremely curious. Where have we come across this word object before?

Lets investigate.

Connecting Turtle and open

So, the Python Standard Library suggests that the function open returns an object of the file type. Now this word object is really interesting because we have seen it before. Remember the time when we were drawing squares using the class Turtle? We were writing lines of code like this. This meant that Brad was an object or instance of the class Turtle.

Similarly, when we write lines like this, which is quotes equal to open file location, and the file, in our case, was a movie quotes file, we are saying that quotes is an object or an instance of file. Now let's think for a moment, what happens when we run this line of code. When we do run this line of code, a function called `__init__` is called from inside the class Turtle. And what that does is that it creates space in memory for the object Brad.

Now in the case of open, even though on surface it looks like we are merely calling a function, and that there is no class involved. If you look deep down inside the code

"open returns an object of the file type" (Python Standard Library)



`brad = turtle.Turtle()`
↳ an object of *Turtle*

turtle

```
class Turtle():  
    def __init__():
```



`quotes = open(file_location)`
↳ an object of *File*



for how open was actually implemented inside Python. We will find that open, in turn, is calling some init-like function to create an object of file. Alright, that's enough jargon for now. We will talk about instances and objects a little bit later in the course. For now, we can rejoice in the fact that part one of profanity checker is done.

Built in Python Functions

Now, before we write the second part of this program, I want you to experiment with built-in functions. Pick anyone of the built-in functions, read through their documentation. By the way, the [link to this page](#) is available in the instructor notes. Use your chosen built-in functions in a new program and then share your experiences on the forum.

You should answer questions like: Which built in function did you actually decide to use? What did your program do? And what problems did you run into? By the way,

being able to read documentation and experimenting with Python, is a big part of learning how to program. Once you've posted your responses on the forum, feel free to check this box before you continue.

1. Pick a built-in Python function. Read through its documentation.
(documentation link available in the instructor notes)
2. Use it in a new program and share your experience on the forum
(answer questions like which function did you use, what did the program do, what problems did you run into)

Check this box after submitting your response on the forum
(To access the forum click the "Exploring Built-In Functions" discussion thread)

Checking for Curse Words

The next step in our plan is to check the text that we've read thus far for profanity. Now, to do this, I searched the Internet and found this website called wdyl.com or what do you love. This website was developed by Google and one of the things it does is that it can tell us if a word is a curse word or not.

So, if I go to this website's [profanity link](#), which is profanity question mark q, by the way, this link is also available in the instructor notes and try out the word shot, S-H-O-T and see what happens. Ha, the website's response is false which means that this word, the word shot, is not a curse word. But if I change this word by just one letter and hit Enter. The response is a little bit different.

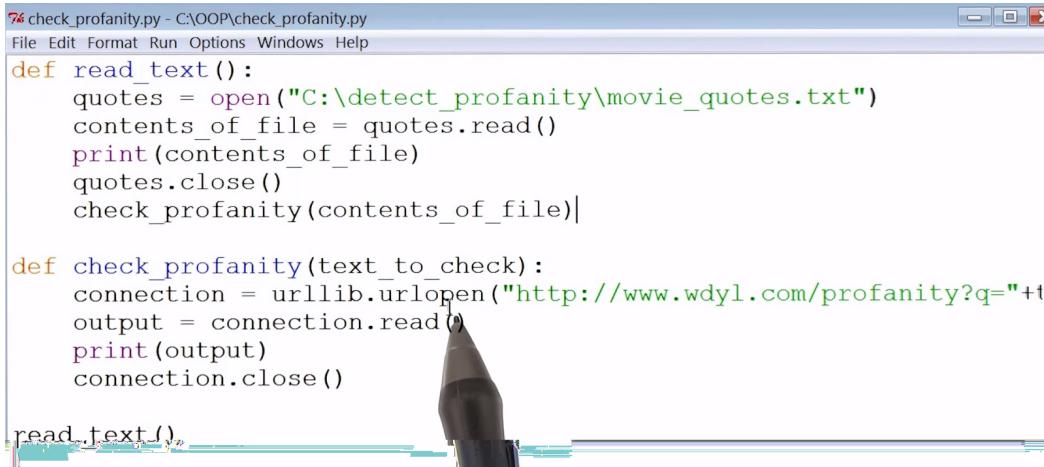
I can even give this website a sentence or a paragraph. So let me, in place of saying one word, say, I can take a shot at it and hit Enter, and the response is false, which means that this sentence is fine. But, if I change one word in that sentence, so I will say, I can, well, you know, and then hit Enter. Ooh, the response now is true, which means that there is a curse word in this sentence. All we have to do now is do what we've been doing, but using code.

Accessing a Website with Code

So here we are, back with our code. Now, the only new stuff that I have added to the code thus far, is this function called, `check_profanity`; by the way, this function is empty for now. It takes in, one argument, or one piece of information, which is the

text we actually want to check for profanity.

Now, I know there is this module in Python called, urllib, which helps us get information from the internet. It, has a function called, urlopen, which takes in a link to a website. So here, I'm going to give it the full link, to the what do you love website. And add to that the text we actually want to check for profanity. Now,



```
76 check_profanity.py - C:\OOP\check_profanity.py
File Edit Format Run Options Windows Help
def read_text():
    quotes = open("C:\detect_profanity\movie_quotes.txt")
    contents_of_file = quotes.read()
    print(contents_of_file)
    quotes.close()
    check_profanity(contents_of_file)

def check_profanity(text_to_check):
    connection = urllib.urlopen("http://www.wdyl.com/profanity?q=" + t
    output = connection.read()
    print(output)
    connection.close()

read_text()
```

this function urlopen. Is going to help us make a connection, to this website. So, I'm going to call this, connection. Now I recognize that you may not be able to read the entire line of code that we've written here, so I'm going to temporarily, put in a return statement there, so you can read the whole line of code. And then I'm going to restore the code.

Okay, let's continue. Now, I want us to notice, that this function URL open, is quite similar to the other function that we have used in this program, which is open. Open, helps us read contents from a file on our computer. urlopen on the other hand, helps open a connection to a website on the internet. Then we can do things with that connection. Things like, read a response from that website. I'm going to call this response, output, and then print the output. After I've done that, I will close the connection.

So now, it's time for us to call this function check_profanity, and I will do that after I have successfully read the text from a file on my computer. So, let me add code to call the function, check_profanity, and to it, I will pass in the contents that we previous read from the file. I am going to save our program. Now if all goes well, the read_text function, will read the contents from the movie_quotes file, and then the check_profanity function, will check that text for curse words.

```
74 *check_profanity.py - C:\OOP\check_profanity.py*
File Edit Format Run Options Windows Help
import urllib

def read_text():
    quotes = open("C:\detect_profanity\movie_quotes.txt")
    contents_of_file = quotes.read()
    print(contents_of_file)
    quotes.close()
    check_profanity(contents_of_file)

def check_profanity(text_to_check):
    connection = urllib.urlopen("http://www.wdyl.com/profanity?q=" + t
    output = connection.read()
    print(output)
    connection.close()

read_text()
```

Let me run my program and oh, it seems like, I forgot to import urllib. So let me go back to the code and add that to the very top. There it is. Let me save my program, and run again.

And this time, the program worked and the programs response is false, which means that the movie quotes, have no curse words in them. Now, if I go back to my movie quotes file. And change just one word, let me save this file. And run my code one more time, now the response changes to true, which means that there was a curse word in our movie codes file. Alright, so it seems like we're getting very close to finishing this code. But before we do, I want us to figure out how Python knows about this think urllib and urlopen; lets find out.

Place urllib and urlopen

So, I'm holding in my hand urllib and urlopen. Now, if you were to take a guess and place these two in our model of the Python standard library. Where would you do it? Feel free to provide your answer in the box provided.

Where in our model of the Python Standard Library should we add urllib and urlopen ?

Where Does urllib Come From

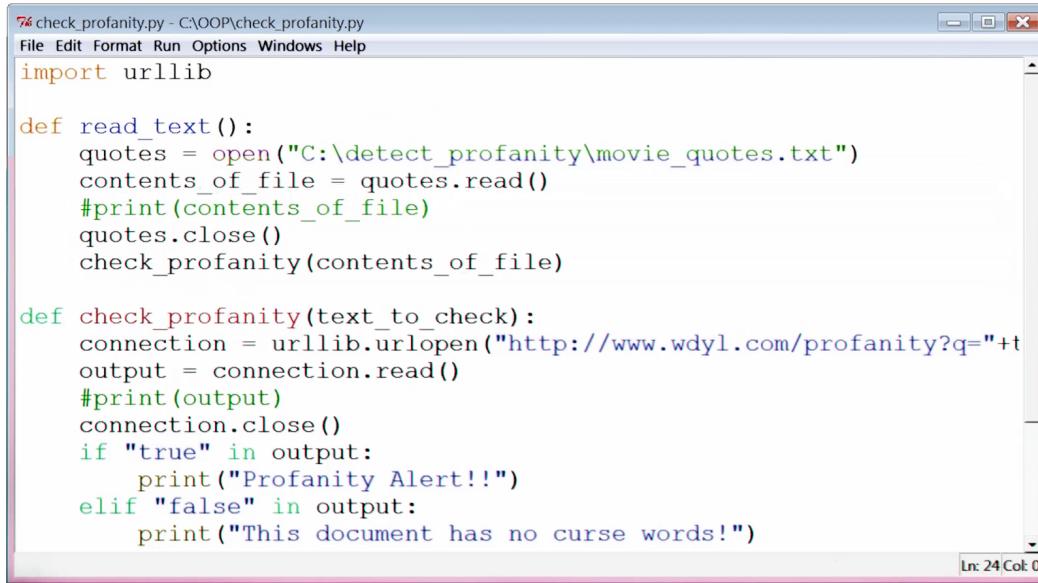


So, much like webbrowser, time and os, urllib is also a module or a file in Python. And urlopen is a function inside it. Don't take my word for it, let's look for these in the Python documentation.

Reading urllib Documentation

So, here is the Python Standard Library and if I scroll down here, I can find a module named `urllib`. Let me click on it and here it says, this module provides a high-level interface for fetching data across the World Wide Web, which is just a fancy way of saying, this module helps us get data from the Internet or websites. And if I scroll down on this page. I can find the function `urlopen`.

Printing a Better Output



The screenshot shows a Windows Notepad window titled "check_profanity.py - C:\OOP\check_profanity.py". The code is as follows:

```
File Edit Format Run Options Windows Help
import urllib

def read_text():
    quotes = open("C:\detect_profanity\movie_quotes.txt")
    contents_of_file = quotes.read()
    #print(contents_of_file)
    quotes.close()
    check_profanity(contents_of_file)

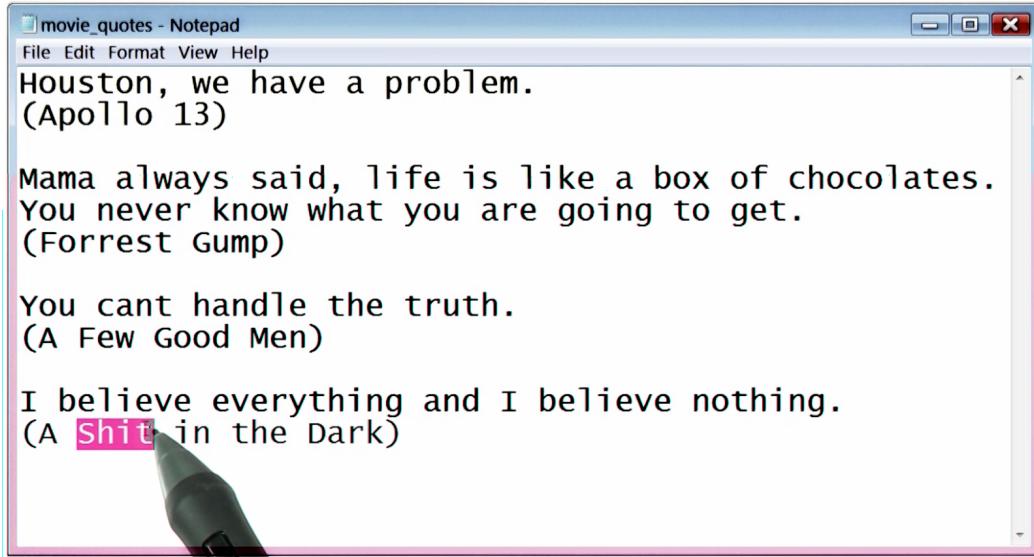
def check_profanity(text_to_check):
    connection = urllib.urlopen("http://www.wdyl.com/profanity?q=" + t
    output = connection.read()
    #print(output)
    connection.close()
    if "true" in output:
        print("Profanity Alert!!")
    elif "false" in output:
        print("This document has no curse words!")
    else:
        print("Could not scan the document properly.")

Ln: 24 Col: 0
```

So here I am, back at the code again and if I scroll down, you will notice that I made one change to the code, which is, that I added an if statement to print out more easily readable output messages.

```
if "true" in output:
    print("Profanity Alert!!")
elif "false" in output:
    print("This document has no curse words!")
else:
    print("Could not scan the document properly.)
```

Let me run the code now. Aha. The program says that there is a profane word in the `movie_quotes` document. And if I go back to my `movie_quotes` document, I can find the culprit word.



So, let me go ahead and quickly change that. So, I will change that back and save this document and now. If I run my code, I get a message saying no curse words were found. Alright, the second step in our program is also done.

Profanity Editor: Mini Project

So now that we are finished designing our project, here is your assignment. We want you to show the program we've written together to a friend or a colleague. Now, you can continue to use the [Profanity Editor](#) website which is wdyl.com, or you can use some sort of a different internet service like [Pirate Speech](#). By the way, the link to this is available in the instructor notes. While you are doing that we want you to use your smartphone to record a video of the conversation with your friend or colleague. And then we want you to share your YouTube video with us on the discussion forum. Make sure you check this box after submitting your responses on the forum.



Profanity Editor - Mini Project

1. Show the program we wrote to a friend or a colleague
(Use Profanity Editor or a different service like Pirate Speech; link in the instructor notes)
2. Use your smart phone to record a video of that conversation
3. Share your YouTube video on the forum
(if you can't record a video, upload photos or text describing the conversation with your friend)

Check this box after submitting your response on the forum
(To access the forum click the "Profanity Editor - Mini Project" discussion thread)

Connecting Turtle, open and urllib

Now before we wrap this project up, I want to take a minute to connect some of the ideas that we have seen in this lesson thus far.



```
brad = turtle.Turtle()
```

↳ *an object of Turtle*

We started out by creating squares and we wrote a piece of code that said brad is equal to turtle.Turtle(). What we were really doing behind the scenes there is creating an object or an instance of the class Turtle. What we could then do with that instance is things like brad.forward().

We then wanted to do things like read contents from a file on our computer, so we wrote a piece of code that said quotes = open(a_file_location). What we were really doing behind the scenes there, is creating an object of the type file. We could then do things with that object like quotes.read().



```
quotes = open(file)
```

↳ *an object of File*

And when we wanted to access a website on the internet, we wrote a piece of code that said connection = urllib.urlopen(). Now

this piece of code also returned a file-like object or instance and we could then do things with that instance, like connection.read().



```
connection = urllib.urlopen()
```

↳ *a File like object*

In all three of these examples, we created instances or objects and then we used those objects. Now some of you may say that in the profanity editor example, all we used were functions. Functions like open and urlopen and that there was no class like we saw in the previous example with Turtle.

Well even though on surface it looks like we did not use any classes in the profanity editor example, if you look at how functions like open and urlopen are actually implemented in Python, we will find that some classes init like function is being called to return these objects. Okay, let's look at one more example of how to use classes. Things will get a lot clearer after that.