Learning Quiz 32: Python WWW API

Due Nov 27 at 1pm **Points** 5 **Questions** 5 **Available** until Dec 4 at 11:59pm **Time Limit** None
Allowed Attempts Unlimited

Instructions

Prior to completing this quiz, be sure to read:

- Section 11.2: Python WWW API (p. 379-386)

Please also go over Practice Problems 11.1-11.3 in the textbook (solutions at the end of the chapter) before attempting this quiz.

This quiz was created for learning purposes. You may attempt this quiz as many times as you would like. The highest score *prior to the deadline* will count towards the final course grade. No late submissions will be accepted.

Take the Quiz Again

Attempt History

	Attempt	Time	Score
KEPT	Attempt 2	3 minutes	5 out of 5
LATEST	Attempt 2	3 minutes	5 out of 5
	Attempt 1	7 minutes	4 out of 5

Score for this attempt: **5** out of 5
Submitted Nov 29 at 10:42am
This attempt took 3 minutes.

Question 1

1 / 1 pts

We can access webpages similar to how we read in text files in Chapter 4. We'll need to use the urlopen() function from the urllib.request module. While urlopen() is similar to the built-in open() function that we saw in Chapter 4, it differs in a couple of ways:

In the following lines of code, we import the urlopen function from urllib.request. We then open the Wikipedia page on Los Angeles, and read the html file. Notice that our read in url (myhtml) is of type "byte" which is not human-readable. However when we print myhtml, we are able to read the content. That is because Python does the converting for us.

```
from urllib.request import urlopen
response = urlopen('https://en.wikipedia.org/wiki/Los_Angeles')
myhtml = response.read()
print(type(myhtml))
print(myhtml)
```

Let's decode myhtml so that we can actually work with it. Notice that after using the decode() function, type(myhtml) tells us that it's a string. We can now work with the string.

```
myhtml = myhtml.decode()
print(type(myhtml))
print(myhtml)
```

Consider the following code:

```
from urllib.request import urlopen
response = urlopen('https://en.wikipedia.org/wiki/Python_(programming_language)')
read1 = response.read()
read2 = read1.decode()
```

What data type is read1? read2?

Correct!

read1 byte

Correct!

read2 str

Other Incorrect Match Options:

- float
- bool
- int

Question 2

1 / 1 pts

In general, HTML has a text pattern, and it helps to develop a class that can handle HTML components. A helpful class to use while working with HTML is the HTMLParser class in the html.parser module.

Fill in the blanks to import the HTMLParser class from the html.parser module.

from html.parser import HTMLParser

Answer 1:

Correct!

html.parser

Answer 2:

Correct!

HTMLParser

Question 3

1 / 1 pts

The HTMLParser class was written with functions that were intended to be overwritten. That means the overall class will handle the HTML for you, but what you want to do with the elements needs to be coded by you. Functions like handle_starttag(), handle_endtag(), and handle_data() can be though of as functions with 'pass' as the body. They do nothing.

In order to inherit the attributes of HTMLParser, but have functions behave differently, we'll need to use the concept of inheritance.

Fill in the blank below to create a class MyParser that prints 'anchor encountered' whenever it encounters an 'a' tag.

from [html.parser] import HTMLParser

[class] MyParser([HTMLParser]):

```
    def handle_starttag(self, tag, attrs):
        if tag == 'a':
            print('Anchor Encountered')
```

Answer 1:

Correct! html.parser

Answer 2:

Correct! class

Answer 3:

Correct! HTMLParser

Question 4

1 / 1 pts

If you enter the following lines of code in Python, you will find list of functions available to the HTMLParser class.

```
from html.parser import HTMLParser
help(HTMLParser)
```

Which of the following functions are intended to be overwritten?

Correct! ☑ handle_comment(self, data)

Correct! ☑ handle_startendtag(self, tag, attrs)

☐ parse_endtag(self, i)

- [ ] goahead(self, end)

Correct!

- [x] handle_charref(self, name)
- [ ] parse_pi(self, i)

Question 5

1 / 1 pts

Quiz 31 mentioned that we might encounter some absolute and some relative URLs. to handle relative URLs, we'll need to use the urljoin function from the urllib.parse module. This will help us determine the absolute URL of a web page if we are given the relative URL.

Fill in the blanks below so that we can use the urljoin function as written:

from urllib.parse import urljoin

url = 'http://www.w3.org/Consortium/mission.html'

relative = '/Consortium/contact.html' ##Hint, see urljoin() below

urljoin(url, relative)

Answer 1:

Correct!

urllib.parse

Answer 2:

Correct!

urljoin

Answer 3:

Correct!

relative

Quiz Score: **5** out of 5