P1: Movie Trailer Website

Submitted by Rachel Scherer on 12/20

This is the first of several projects submitted for review and consideration in the pursuit of Udacity’s Full Stack Nanodegree. This project meets requirements outlined in an Udacity-supplied project guide last accessed on 12/20/15 [google docs]. The primary parameters for completion are outlined in the guide, and include elements related to Functionality, Code Quality, Comments, and Documentation.

The specific aim of the project is to build Python code which specifies a list of movie titles and related attributes (poster art, movie trailer, and tagline) to be displayed on a webpage for user review. I have extended this project by adding opportunities for user input, input validation and error handling, and front-end modifications to clarify the intent of the project to the end user, which I will call out below.

Quick start

- Unzip the provided file. The following .py files are required for appropriate execution:
    - final-cut.py
    - media.py
    - fresh_tomatoes.py
    - green_tomatoes.py
- Ensure that you have access to Python 2.7 and that urllib is installed

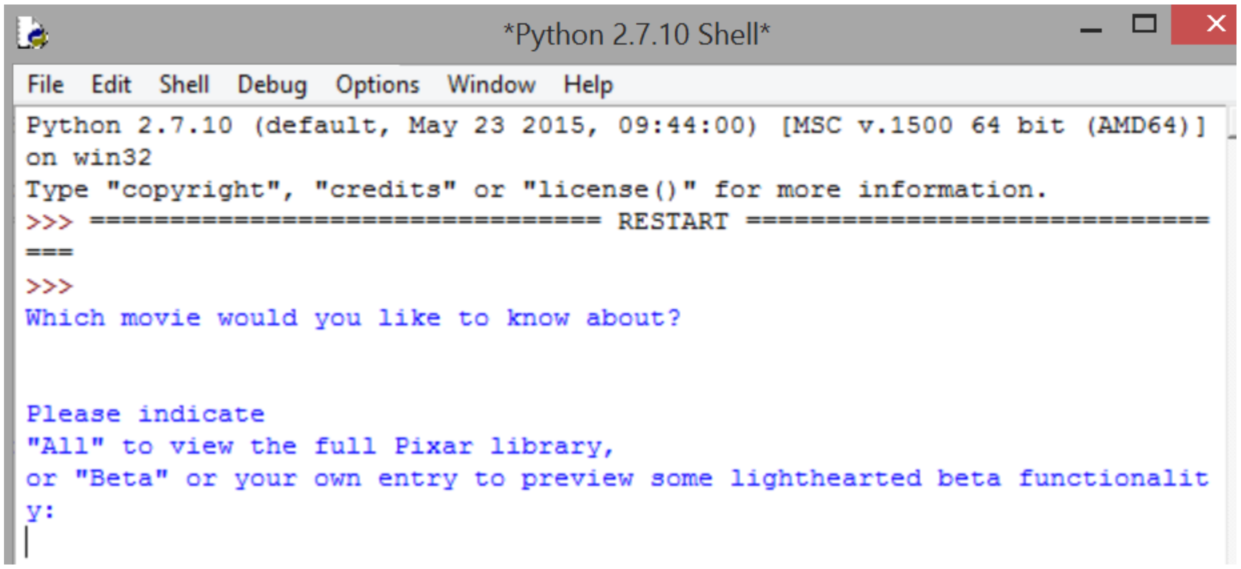
What's included

Within the download you'll find the following directories and files. You will be opening and running final_cut.py for most normal execution purposes.

```
.zip/
├── final_cut.py
├── media.py
├── fresh_tomatoes.py
├── green_tomatoes.py
```

Running the Project & Description of Extensions

Ensure that all required libraries are installed. Access final_cut.py and run (F5). Code will execute as follows:



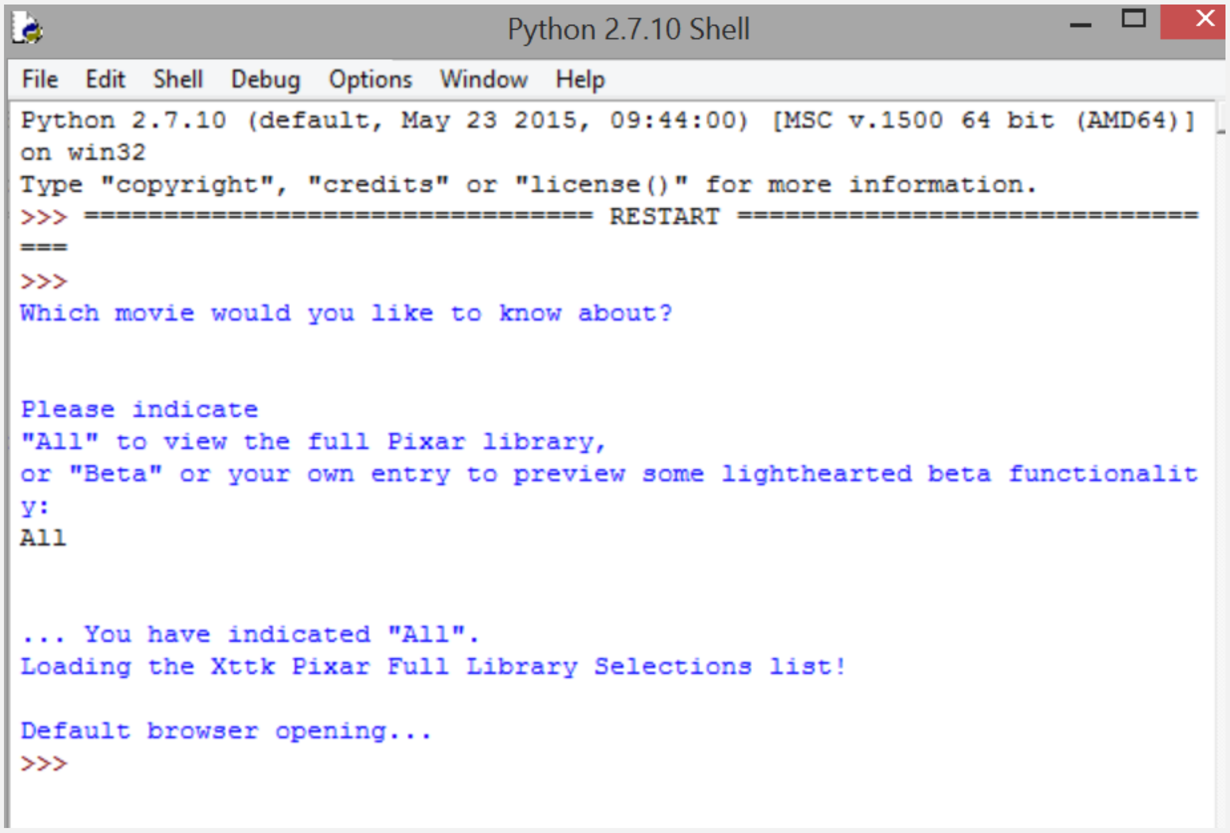
```
Python 2.7.10 (default, May 23 2015, 09:44:00) [MSC v.1500 64 bit (AMD64)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ==============================
===
>>>
Which movie would you like to know about?


Please indicate
"All" to view the full Pixar library,
or "Beta" or your own entry to preview some lighthearted beta functionalit
y:
```

There were two large guiding values for this project. Firstly and primarily, my intent was to meet the project requirements. Secondly, my intent was to have fun and use this as a case for applying what I've learned in the course and beyond, to provide a little bit of a more fun and dynamic user experience. Users are prompted to go one of a couple directions at this point with the help of a new raw_input function variable:

1. Type "All" – User is taken to a pre-configured catalog of Pixar movies, built of custom classes linking to a parent in media.py and launching a web experience generated via fresh_tomatoes.py [2]. Web experience has been customized to

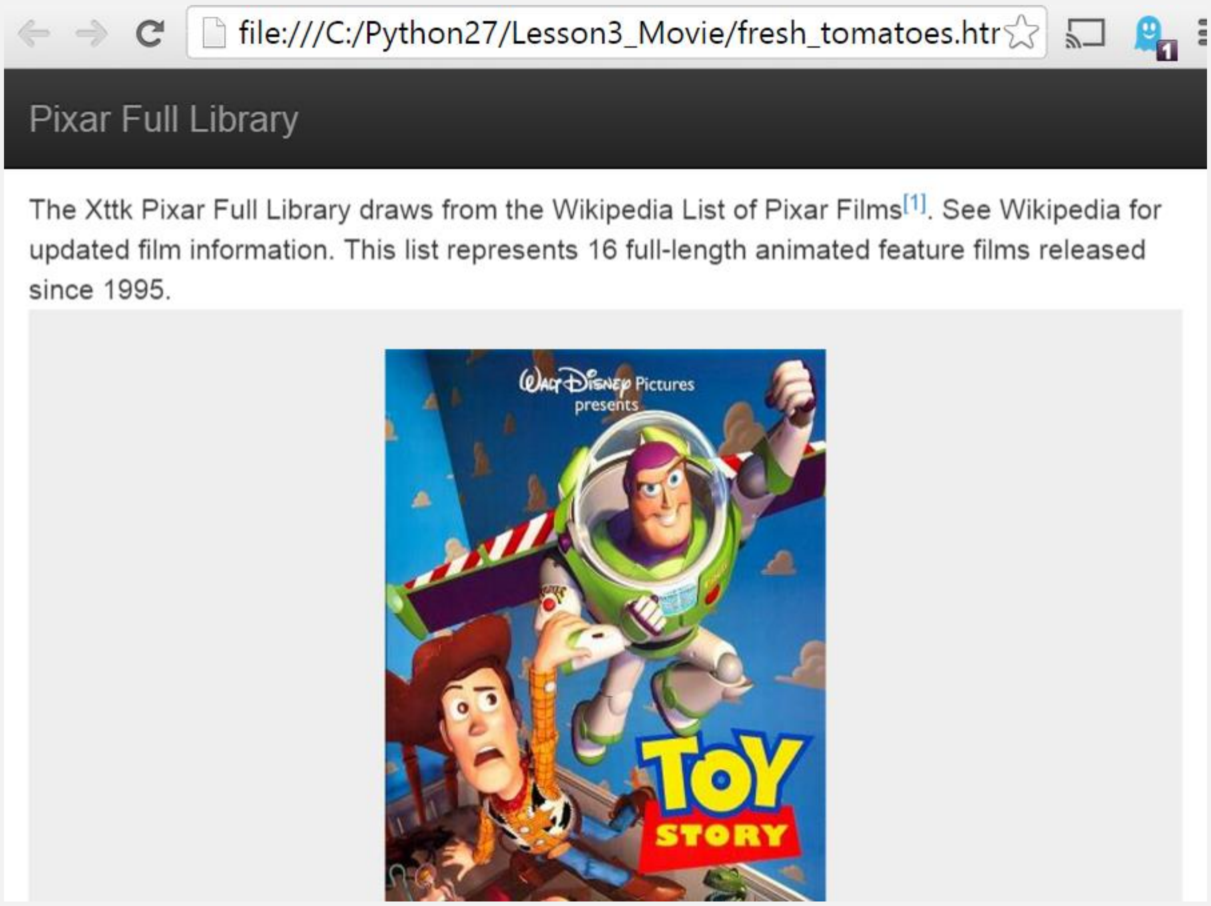
provide additional information about the content presented [3] with the addition of a meaningful page title and catalog-entry-specific taglines.

[2] User responds “All” to initial prompt



```
Python 2.7.10 (default, May 23 2015, 09:44:00) [MSC v.1500 64 bit (AMD64)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ==============================
===
>>> 
Which movie would you like to know about?


Please indicate
"All" to view the full Pixar library,
or "Beta" or your own entry to preview some lighthearted beta functionalit
y:
All


... You have indicated "All".
Loading the Xttk Pixar Full Library Selections list!

Default browser opening...
>>> 
```

[3] Pixar-Customized Website



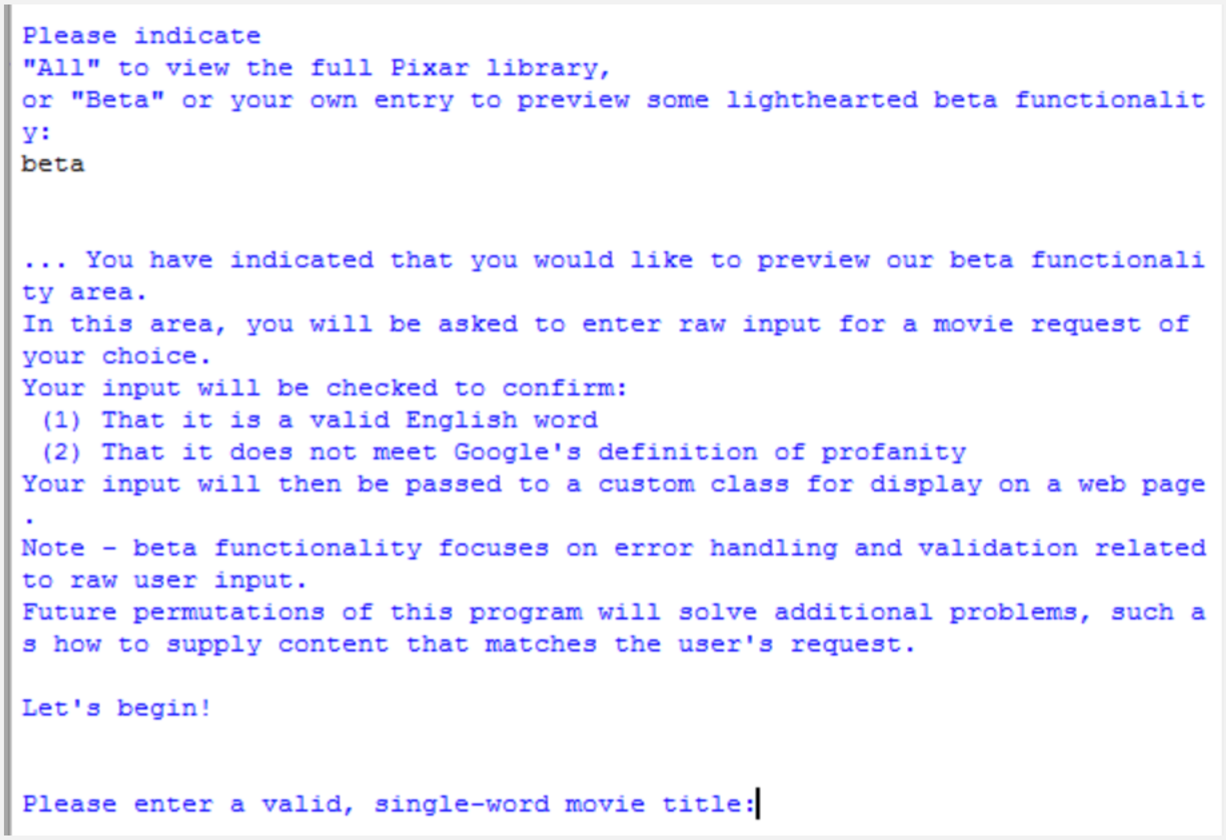
file:///C:/Python27/Lesson3_Movie/fresh_tomatoes.htr

Pixar Full Library

The Xttk Pixar Full Library draws from the Wikipedia List of Pixar Films[1]. See Wikipedia for updated film information. This list represents 16 full-length animated feature films released since 1995.

2. Type “Beta” – User is advised of the Beta program and instructed as to how to participate [4].

[4] Beta Opt-In



```
Please indicate
"All" to view the full Pixar library,
or "Beta" or your own entry to preview some lighthearted beta functionalit
y:
beta


... You have indicated that you would like to preview our beta functionali
ty area.
In this area, you will be asked to enter raw input for a movie request of
your choice.
Your input will be checked to confirm:
 (1) That it is a valid English word
 (2) That it does not meet Google's definition of profanity
Your input will then be passed to a custom class for display on a web page
.
Note - beta functionality focuses on error handling and validation related
to raw user input.
Future permutations of this program will solve additional problems, such a
s how to supply content that matches the user's request.

Let's begin!


Please enter a valid, single-word movie title:
```

For the beta/extended functionality area, I became interested in the problem of handling novel and not-previously-defined user input.

Because of the option to input something novel/undefined, we now have to handle anything that a human comes up with. Some potential problem areas:

(1) Keyboard-Mashing (Typos and Invalid Words)

In final_cut.py, we call check_word using the raw input as the argument to protect against typos and cat-on-the-keyboard entries. check_word leverages urllib and the geturl function to connect to http://dictionary.reference.com/browse/ and determine whether redirected url matches requested query. Note on the concept here: dictionary.reference.com valid word urls match a specific convention. If a word is invalid we are redirected to a suggested entry page, which would not match our expectations and flags for invalid entry. If check fails, users are given an opportunity to try another entry. This problem area is addressed by my project.

(2) Profanity

After we get the raw input, we call check_profanity using the input as an argument. check_profanity leverages urllib to connect to wdyl.com/profanity?q= and determine whether user input string matches a google-maintained database of not-so-nice words. Note: per Udacity lesson, wdyl.com returns a string value of true or false matching against a list that google maintains. It is likely that users could get creative and foil this check. If check fails, users are given an opportunity to try another entry. This problem area is addressed by my project.

(3) Legitimate Movie Queries

Once we've passed our validation points, for the sake of the thought experiment we take a perhaps generous assumption that we are dealing with a legitimate movie query. We have some specific goals in mind with regard to how we want to present this output. In this under-development area, we may not have met all of these goals yet:

(a) Trailer, Title, Poster, and Tagline populate a new, custom class of media.py defined by raw user input;

(b) Trailer, Title, Poster, and Tagline are presented in format from modified fresh_tomatoes.py file provided in Udacity course (green_tomatoes.py);

(c) All elements in the custom class are relevant to each other, that is, pertain to the same movie.

And finally:

(d) No errors on the presentation layer.

To accommodate the under-development functionality, a modified fresh_tomatoes.py file was created to generate a different user experience in the beta area,

Bugs and feature requests

The project has been pretty well vetted, however I look forward to receiving feedback with regard to bugs and feature requests in my review. I have already noted several areas for improvement on my initial design, including building upon the user input validation/error handling functionality to accommodate multiple-word user entries and varied capitalization; adding calls to return dynamically-generated-or-retrieved content to the custom request page; and consolidating to one python file the display of content currently differentiated across the standard, project-requirements output html page (generated by fresh_tomatoes.py) and the page meant to display custom user input (generated by green_tomatoes.py).

Documentation

WYSIWYG. At this time, the body of documentation for this project is presented along with the project itself, as part of the uploaded .zip package. The intent is to move the entire project to GitHub following submission.

Creators

Rachel Scherer

- https://www.linkedin.com/in/rachelpscherer