Assignment 2, Part A: The Best, Then and Now

(21%, due 11:59pm Sunday, October 21st)

Overview

This is the first part of a two-part assignment. This part is worth 21% of your final grade for IFB104. Part B will be worth a further 4%. Part B is intended as a last-minute extension to the assignment, thereby testing the maintainability of your solution to Part A and your ability to work under time pressure. The instructions for completing Part B will not be released until Week 12. Whether or not you complete Part B you will submit only one solution, and receive only one mark, for the whole 25% assignment.

Motivation

People are always interested in lists of "the best" things in a wide range of categories. One of the features that makes such lists interesting is the way the entries change over time. Here you will develop a software application that allows its users to browse a number of online "top ten" lists, including the ability to compare old and new versions. Your application will have a Graphical User Interface that allows its user to preview the lists they are interested in. They will then be able to export a more detailed version of any chosen list, which can be examined in a standard web browser.

This "capstone" assignment is designed to incorporate all of the concepts taught in IFB104. To complete it you will need to: (a) use Tkinter to create an interactive Graphical User Interface; (b) download web documents using a Python script and use pattern matching to extract specific elements from them; and (c) generate an HTML document integrating the extracted elements, presented in an attractive, easy-to-read format.

Goal

Your aim in this assignment is to develop an interactive "app" which allows its users to preview and export top-ten lists downloaded from the web. There must be at least three distinct lists available, and both old and current versions of the lists must be made available. Most importantly, the online web documents from which you collect your lists must be ones that are updated on a regular basis, either daily or weekly, so that the old and new lists are different.

For the purposes of this assignment you have a free choice of which lists your application will display, provided they always contain at least ten items, are updated frequently, and include the name of each item listed and at least one distinctive "attribute" for each item. Your application must offer access to (at least) three entirely different lists. The lists could be:

- music charts,
- movie or television ratings,
- stock market listings,
- online gaming player rankings,
- book ratings,
- crowd-sourced popularity lists,

- customer ratings of products or services,
- web site statistics,
- etc.

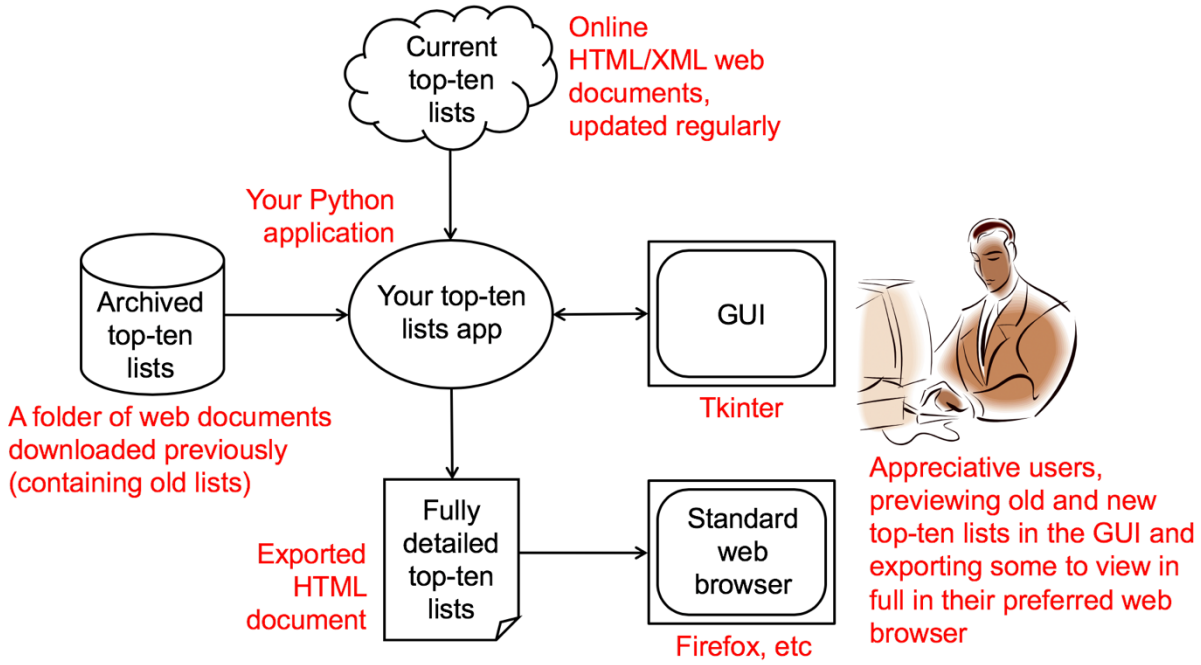
However, whatever lists you choose, you must confirm that the online web documents are updated frequently. For each item in each list the source web site must contain the item's name and some other distinguishing "attribute" of the item listed. Attributes could be:

- an image or photo,
- some additional property of the item other than its name, such as the author of a book or the lead actor in a movie,
- a detailed textual description of the item,
- some kind of numeric score, ideally one which justifies or explains the item's appearance in the list, such as a number of user votes or downloads,
- etc.

Appendix A below lists many web sites which *may* be suitable for this assignment, but you are encouraged to find your own of personal interest.

Note: An obvious source for such lists is sport. However, you cannot always rely on sporting lists being updated "out of season". Therefore, if you choose to use a sports-based list, you must confirm that the sport is being played during the period in which this assignment will be developed and assessed, i.e., mid-September to mid-November 2018!

Using the data in the online top-ten lists you are required to build an IT system with the following general architecture.



Your application will be a Python program with a Graphical User Interface. Under the user's control, it allows the contents of several top-ten lists to be previewed in the GUI. One collec-

tion of lists is static and is stored in an "archive", i.e., a folder of previously-downloaded HTML/XML documents. The other source of lists is the "live" Internet. Your application must offer both a previously-downloaded and a "current" list of three different kinds. Having previewed the lists in the GUI, the user can then choose to export them to an HTML file. This file will contain a more detailed version of the list than the one previewed in the GUI and can be studied by the user in any standard web browser at their leisure.

This is a large project, so its design allows it to be completed in distinct stages. You should aim to complete it incrementally, rather than trying to solve the whole problem at once. A suggested development sequence is:

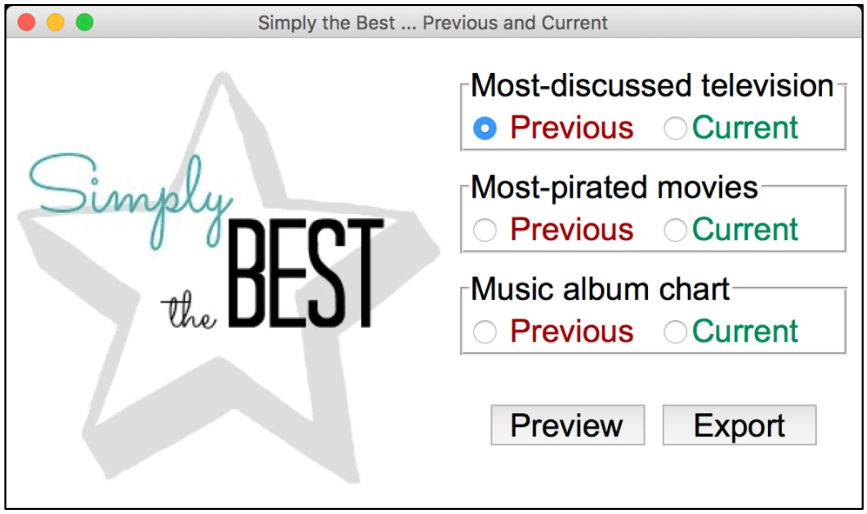
1. Develop code that allows the static, archived top-ten lists to be previewed in the GUI.
2. Extend your solution so that it allows the corresponding "live", online top-ten lists to be previewed.
3. Extend your solution further so that any of the lists previewed, archived or live, can be exported as HTML documents.

If you can't complete the whole assignment submit whatever parts you can get working. You will get **partial marks for incomplete solutions** (see the marking guide below).

Illustrative example

To demonstrate the idea, below is our own solution, which uses data extracted from three different web sites, one which lists the US TV shows most discussed in social media, one which lists the UK's most popular music albums, and one which lists movies currently being illegally downloaded online. All three of these lists are updated online at least weekly, so our program is designed to continue working even after the lists have changed.

The screenshot below shows our example solution's GUI when it first starts. We've called it "*Simply the Best*" after the Tina Turner song, but you should choose your own name and GUI design.



The GUI has a logo which identifies it, six radio buttons allowing the user to select lists of interest, and two push buttons allowing the user to preview or export the list selected. You do not need to copy this example and are encouraged to design your own GUI with equivalent functionality. For instance, menus could be used to allow the selection rather than radio

buttons. Our "*Simply the Best*" logo contains text which names the application, but if you choose an image with no text you must also add a textual label containing your application's name.

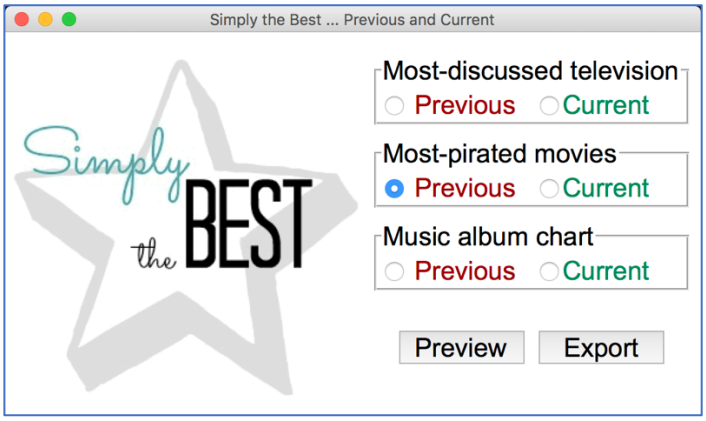
Previewing old lists

For each of the three types of list, our app allows the user to preview an old, previously-downloaded copy of the list or the current one online. Working with unchanging web documents is obviously easiest, so we recommend you start your solution with this requirement.

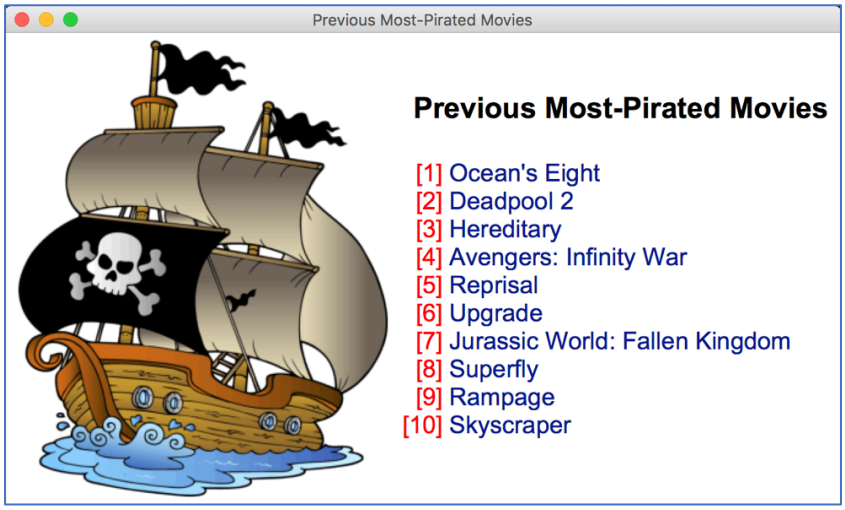
For each of our three lists we downloaded copies of the corresponding web pages (one on September 3rd and the other two on September 9th) and stored them in a folder to serve as our "archive" of old lists. The user can preview any of these lists by selecting the corresponding radio button and pressing the "Preview" button. For instance, pressing the button while the "previous most-discussed television" list is selected causes the following window to be displayed, showing the US Nielsen Survey ranking of TV shows most-discussed in social media on September 9th:

Our app pops up a new window to display this data, but again you are free to design your own GUI. A single window could be used for all functions instead. (The fact that some items are duplicated above is not an error. The original web site makes clear that different episodes of the same TV series were being discussed in social media at the same time.)

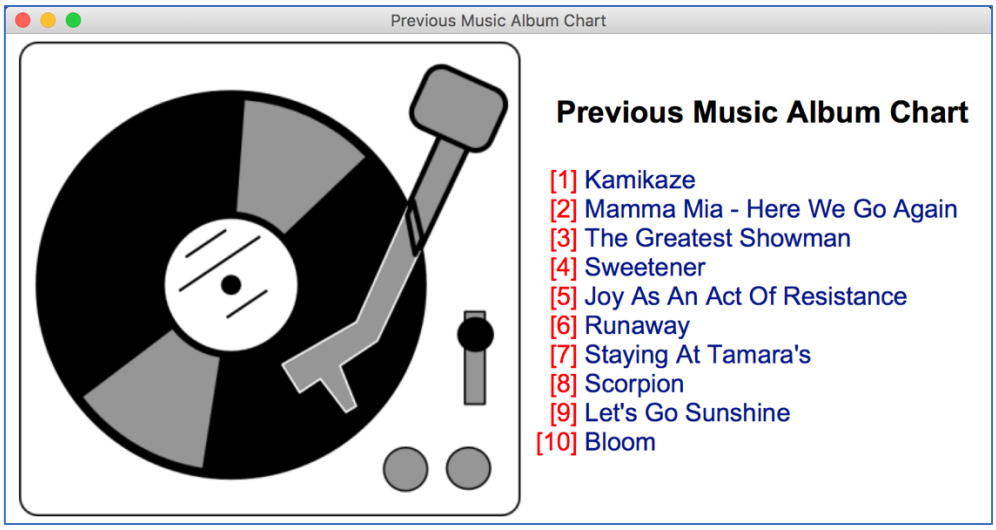
Of course, we can select a different old list for previewing via the GUI, as follows.



Pressing the "Preview" button in this case causes the following list to be displayed, based on a web document previously downloaded from the "TorrentFreak" web site.



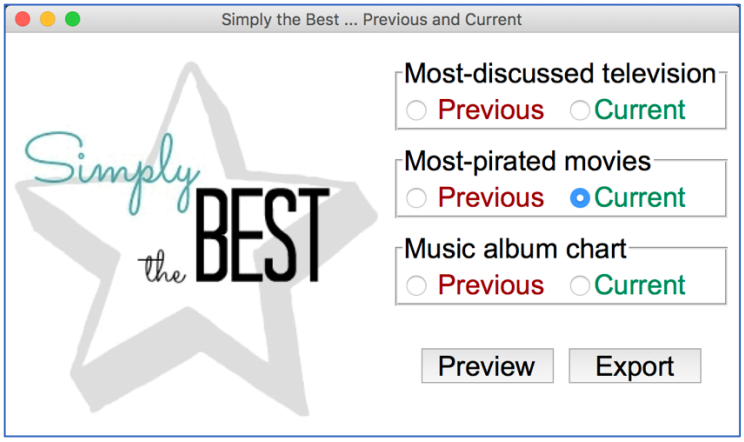
Similarly for the third type of list we offer, the UK's music album chart.



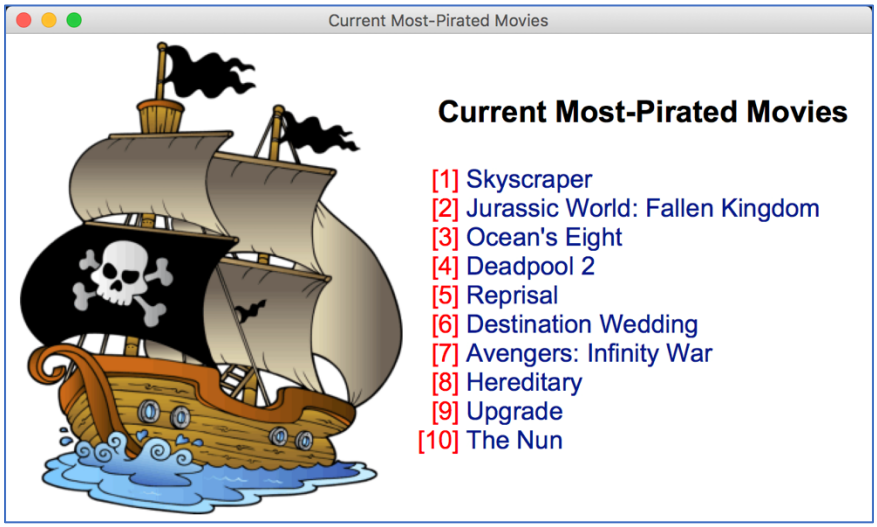
Our main GUI window and each of the three "preview" windows contains an evocative image. These GIF images are all stored in local files, in the same folder as our Python application.

Previewing current lists

As well as previewing old lists, the user can also preview the latest online data. For instance, having seen which movies were being pirated a while ago, the user can choose to see which movies are being pirated right now.

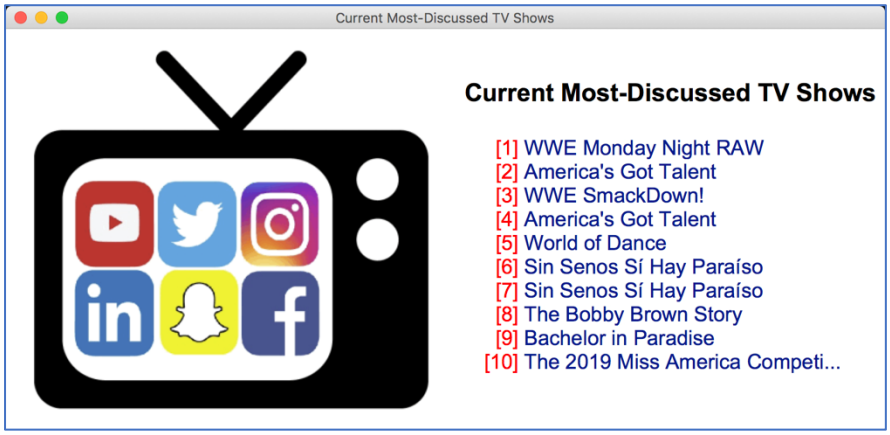


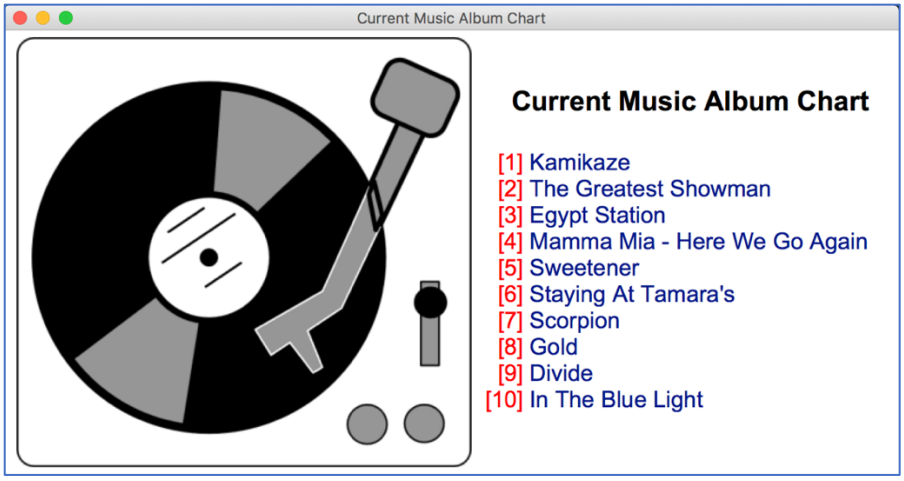
In this case our application downloads a "live" copy of the TorrentFreak web page from online and uses it to display the latest version of the list. The following list was produced when we ran our application on September 15th, 2018.



Notice that the list has been updated significantly since we downloaded the earlier version, with the order of the entries having changed and some new entries appearing in the top ten.

Similarly, when we viewed the current versions of the other two lists we offer on September 15th we found that both had changed since we downloaded our "archived" copies.

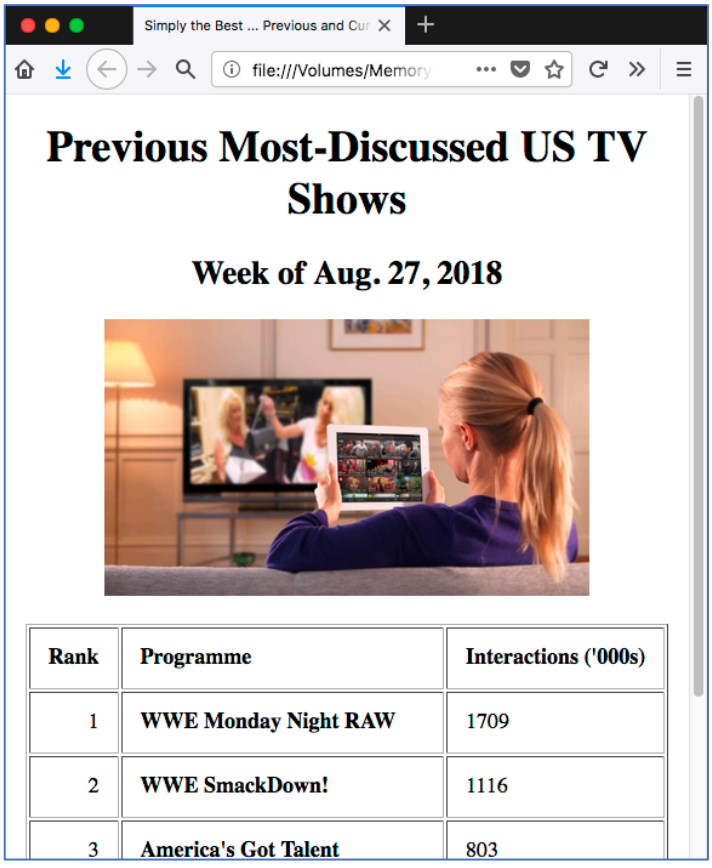




Exporting old and current lists

The list previews provided in the GUI are very simple, consisting of just an image, the list's name, and an ordered list of the items' names. To give the user further information about lists of interest, our app also allows lists to be exported as HTML documents that can be viewed in a web browser. The exported lists contain additional information, including the list's publication date, the file or web site from which the data was obtained, and some additional distinguishing attribute for each item in the list.

For instance, if the user exports the archived list of most-discussed TV shows, our app creates an HTML file which looks like the following when opened in a web browser.



4	**World of Dance**	623
5	**America's Got Talent**	528
6	**Jersey Shore: Family Vacation**	390
7	**Sin Senos Sí Hay Paraíso**	322
8	**Sin Senos Sí Hay Paraíso**	283
9	**Bachelor in Paradise**	253
10	**Love & Hip Hop: Hollywood**	459

This data sourced from Archive/TV-Top-10s-20180909.html

When viewed in this form we are told the name of the previously-downloaded HTML file from which our Python program extracts the information. We also learn the publication date for the list. Note that this date is extracted from the source web document; it is *not* the date the list was downloaded.

Most importantly, the exported list contains an extra attribute for each list item. In this case the attribute chosen is the number of social interactions that led to the item being included on the list. Also notice that a different image is used to identify the list. So that the generated HTML file can be viewed on any computer, the image(s) contained in the exported document must all be links to online images, not local files.

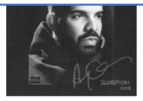
Of course, current lists can also be exported. When the current music chart list was exported (on September 15th, 2018) our program created the following web document.

Current UK Music Album Chart

14 September 2018 - 20 September 2018

Rank	Album Title	Cover Art
1	**Kamikaze**	

	2	**The Greatest Showman**	
	3	**Egypt Station**	
	4	**Mamma Mia - Here We Go Again**	
	5	**Sweetener**	
	6	**Staying At Tamara's**	

7	**Scorpion**	
8	**Gold**	
9	**Divide**	
10	**In The Blue Light**	

This data sourced from http://www.officialcharts.com/charts/albums-chart/

All the details, including the publication date up the top, were extracted from the online web document, which is identified down the bottom. In this case the extra "attribute" is the album's cover art, which was linked to our document using appropriate HTML "`img`" tags.

Similarly for the most-pirated movies. Part of the "current" list on September 15th is shown below. Here the additional attribute added is the movies' IMDb ratings.

Current Most-Pirated Movies

September 10, 2018

Rank	Movie	IMDb Rating
1	**Skyscraper**	6.1
2	**Jurassic World: Fallen Kingdom**	6.5

You are *not* required to follow the details of our demonstration GUI or exported documents. You are strongly encouraged to use your own skills and initiative to design your own solution, provided it has all the functionality shown above.

Extracting the HTML elements

To produce the lists for previewing and exporting, our application used regular expressions to extract elements from the relevant web documents, whether they were stored in the static archive or downloaded when the program is run.

To do so, we first downloaded copies of the web pages and then studied them to identify text patterns that would allow us to find the specific parts we wanted. For instance, we found by inspecting the HTML code from the UK music album charts that the album's names always appeared in a hyperlink of the following form.

`<a href="/search/albums/`*rest_of_the_URL*`/">`*album_name*`</a>`

This knowledge was enough to allow us to create a regular expression which returned all the text patterns of this form, and thereby extract the album names using Python's `findall` function. The URLs of the cover art images were extracted similarly.

Sometimes it's easier to use other Python features as well as, or instead of, regular expressions. For instance, we found the US Nielsen Survey web page hard to use for this assignment because the web site contains multiple lists embedded in the same HTML document. In this case we used Python's string `find` method to discover the starting point of the "social media" list we wanted, and then used string indexing to extract just the part of the HTML code containing the list we needed. Cutting the document down in this way made it easier to use `findall` to extract the relevant elements.

Sometimes it's also necessary to download more than one web page to get all the data we want. The TorrentFreak "pirated movies" list was an example of this. The web site cited above as the source of this data is actually an RSS Feed which has links to multiple copies of the list. To display the "current" list it was therefore necessary in this case to first extract the URL of the latest such list and use that information to separately download the web page containing the list of interest.

In all cases, care was also taken to ensure that no HTML/XML tags or other HTML entities appeared in the extracted text when displayed in either the GUI or the exported HTML documents. In some cases it was necessary to delete or replace such mark-ups in the text after it was extracted from the original web document. The lists seen by the user must not contain any extraneous tags or unusual characters that would interfere with the list's appearance.

Exporting the HTML document

A small part of the HTML code generated by our Python program is shown below, in this case for the current most-discussed TV shows. Although not intended for human consumption, the generated HTML code is nonetheless laid out neatly, and with comments indicating the purpose of each part.

```
1
2  <!DOCTYPE html>
3  <html>
4
5    <head>
6      <!-- An automatically generated HTML documen
7      <meta charset="UTF-8">
8      <title>Simply the Best ... Previous and Curr
9      <!-- Fix the width of all elements and centr
10     <style>
11         h1 {margin-left: auto; margin-right: aut
12         h2 {margin-left: auto; margin-right: aut
13         table {margin-left: auto; margin-right:
14         td {padding: 15px}
15         th {padding: 15px}
16     </style>
17   </head>
18
19   <body>
20
21     <!-- Title, date and indicative image -->
22     <h1 align="center">Current Most-Discussed TV
23     <h2 align="center">Week of Sept. 3, 2018</h2
24     <p align="center"><img src="http://agentv.es
25
26     <!-- Top ten items (rank, name, attribute) -
27     <table border=1>
28     <tr>
29         <th align = 'left'>Rank</th>
30         <th align = 'left'>Programme</th>
31         <th align = 'left'>Interactions ('000s)<
32     </tr>
33
```

Robustness

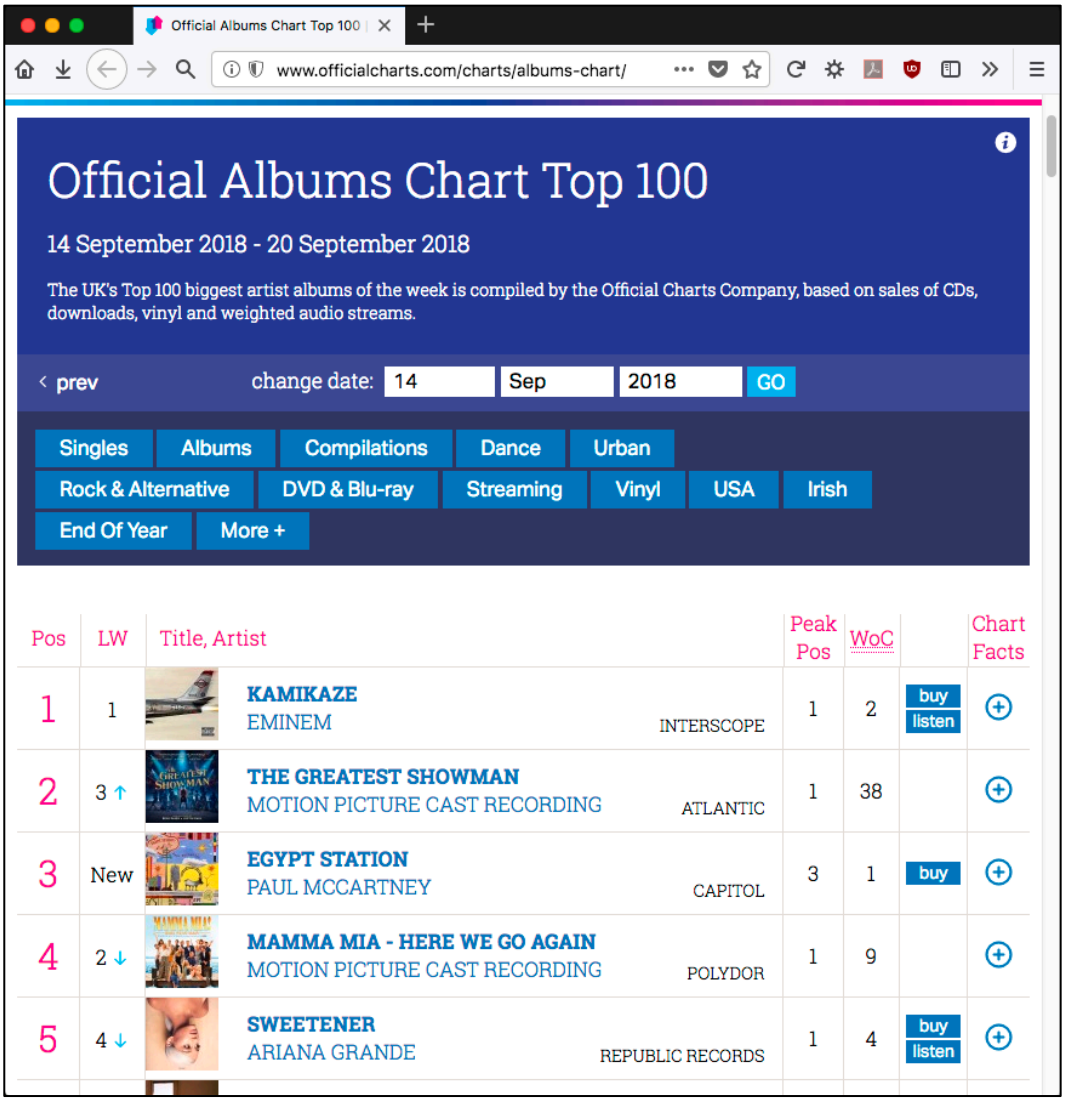
Another important aspect of the system is that it must be resilient to user error. This depends, of course, on your choice of GUI widgets and how they interact. Whatever the design of your GUI, you must ensure that the user cannot cause it to "crash".

For instance, in our demonstration solution the user can press the buttons in any order they like. It is not necessary to preview a list before exporting it. However, if this was the case then the "Export" button should be disabled until the relevant list is previewed, or some sort of clear error message should be displayed if the user pushes the buttons in the wrong order.

Where the data comes from

A significant challenge for this assignment is that web servers deliver different HTML/XML documents to different web browsers, news readers and other software clients. This means the web document you see in a browser may be different from the web page downloaded by your Python application. For this reason, to create your "old" lists you should download the web documents using our `web_doc_downloader` program, or a similar application. This will ensure that the "old" and "current" pages have the same format, thus making your pattern matching task easier.

For instance, the music chart site we used appears as follows when we view it in a web browser.



Notice that all the elements we needed for our application appear in the document, including the publication date, the names of the albums and (links to) the album covers. The challenge is extracting just these elements from the HTML source code.

Most importantly, however, this is not how your Python program "sees" this document. It will receive it as a single, very long character string. For instance, the source code for the above page is actually as shown overleaf when downloaded by a Python script. It is this form of the data that your Python program must work with. From it you will need to use some kind of pattern matching to find the textual elements needed to construct the lists to display in your GUI and your exported document. Most importantly, you must do so in a general way that will still work when the contents of each source web page is updated.

Obviously working with such complex code is challenging. You should begin with your static, "archived" documents to get some practice at pattern matching before trying the dynamically changeable web documents downloaded from online.

```
<!doctype html>
<!--[if lt IE 7]><html class="no-js ie6 oldie" lang="en"><![endif]-->
<!--[if IE 7]><html class="no-js ie7 oldie" lang="en"><![endif]-->
<!--[if IE 8]><html class="no-js ie8 oldie" lang="en"><![endif]-->
<!--[if gt IE 8]><!-->
<html class="no-js" lang="en">
<!--<![endif]-->

<head>


<meta charset="utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />

<title>Official Albums Chart Top 100 | Official Charts Company</title>
<meta name="description" content="The UK&#39;s Top 100 biggest artist albu
<meta name="keywords" content="Top 40, UK Top 40, Charts, Top 40 UK, UK Ch
<meta name="viewport" content="width=device-width,initial-scale=1" />


    <meta property="og:image" content="http://www.officialcharts.com"/>

    <link rel="shortcut icon" href="/favicon.ico"/>

    <link href="http://fonts.googleapis.com/css?family=Roboto+Slab:400,300
    <link rel="stylesheet" href="/Css/style.css" />


    <!-- Ad integration -->


<script async="async" src="https://www.googletagservices.com/tag/js/gpt.js

<script>
    var googletag = googletag || {};
    googletag.cmd = googletag.cmd || [];

    var fn_pageskin = "false";
    if (screen.width >= 412) {
        fn_pageskin = "true";
    }
```

Specific requirements and marking guide

To complete this task you are required to produce an application in Python 3 similar to that above, using the provided `the_best.py` template file as your starting point. In addition you must provide a folder containing (at least) three previously-downloaded web documents to serve as your archive of old lists and all the GIF images needed to support your GUI. (All of the images in the exported HTML file must be online images and must *not* be included in your submission.)

Your complete solution must support *at least* the following features.

- **An intuitive Graphical User Interface (4%).** Your application must provide an attractive, easy-to-use GUI. You have a free choice of which Tkinter widgets to do the job, as long as they are effective and clear for the user. This interface must have the following features:
    - An image which acts as a "logo" to identify your application. Note that, in general, Python Tkinter implementations only support GIF images. The image file should be included in the same folder as your Python application.
    - Your GUI must name your application in both the Tkinter window's title and as a large heading in the displayed interface. The name may appear as part of the logo (as it does in our demonstration solution), but if the logo does not contain any text you must separately add a textual label to the GUI to give your application a name.

  - A widget or widgets that allow the user to select three different top-ten lists, in both old and current forms. It must be possible to easily distinguish the archived "old" lists from the current "live" ones.
  - A widget or widgets that allows the user to choose to preview or export any of the six (or more) lists on offer. Note that this capability could be combined with the one above, depending on which kind of GUI widgets you choose to use.

- **Previewing old "archived" top-ten lists in the GUI (4%).** Your GUI must be capable of displaying (at least) three distinct archived top-ten lists. For each list you must show
  - a heading in the GUI (not just the window title) identifying the list,
  - an image/logo that characterises the list, and
  - a numbered list with (at least) the top-ten items extracted from the archived web document.

  The list items must be extracted from HTML/XML files previously downloaded from online and stored in your "archive" folder. The documents must be stored in exactly the form they were downloaded from the web server; they cannot be edited or modified in any way. Pattern matching must be used to extract the relevant elements from the documents so that the code would still work if the archived documents were replaced with others in the same format. To keep the size of the archive folder manageable only single HTML/XML source files can be stored. No image files may be stored in the archive.

- **Previewing "live" online top-ten lists in the GUI (4%).** Your GUI must be capable of displaying (at least) three distinct "live" top-ten lists, as currently available online at the time the program is run. For each list you must show
  - a heading in the GUI (not just the window title) identifying the list,
  - an image/logo that characterises the list, and
  - a numbered list with (at least) the top-ten items extracted from the online web document.

  The list data must be extracted from HTML/XML files downloaded from online when the program is run. Pattern matching must be used to extract the relevant elements from the documents so that the code still works even after the online documents are updated. The chosen source web sites must be ones that are updated on a regular basis, at least daily or weekly.

- **Exporting HTML documents containing (old or current) top-ten lists (5%).** Your program must be able to generate an HTML document containing any top-ten list selected by the user, from both the "archived" lists and the online ones. The data exported must be written as an HTML document in the same local folder as your Python program and must be easy to identify through an appropriate choice of file name. Each generated file must contain HTML markups that make its contents easily readable in any standard web browser, and it must be self-contained (i.e., not reliant on any other local files). When viewed in a browser, the generated document must be neat and well-presented and must contain (at least) the following features:

- A heading identifying the list.
- The name of the local HTML file or the URL address of the online web page from which the top-ten data was extracted. This must be sufficient to allow the original source documents to be found easily (so that the markers can compare the original web pages with your displayed lists).
- The publication date for the list, extracted from the source document (not just the date when the file was downloaded because they may not be the same).
- An image characterising the list, downloaded from online when the generated HTML document is viewed (i.e., not from a local file on the host computer).
- A numbered list of (at least) the top-ten items. For each item (at least) the following data must be displayed:
  - The item's position in the top ten.
  - The item's name.
  - Some other distinguishing attribute of the item. The attribute may be textual or may be an image.

  All of this data must be extracted via pattern matching from web documents downloaded from online. Most importantly, each of these sets of items *must all belong together*, e.g., you can't have the name of one item paired with an attribute of another. Each of the elements must be extracted from the original document separately.

When viewed in a browser the exported document must be neatly laid out and appear well-presented regardless of the browser window's dimensions. The textual parts extracted from the original documents must not contain any visible HTML/XML tags or entities or any other spurious characters. The images must all be links to images found online, not in local files, should be of a size compatible with the rest of the document, and their aspect ratio should be preserved (i.e., they should not be stretched in one direction).

- **Good Python and HTML code quality and presentation (4%).** Both your Python program code and the generated HTML code must be *presented in a professional manner*. See the coding guidelines in the *IFB104 Code Presentation Guide* (on Blackboard under *Assessment*) for suggestions on how to achieve this for Python. In particular, each significant Python or HTML code segment must be *clearly commented* to say what it does, e.g., "Extract the link to the photo", "Show the item's number", etc.
- ***Extra feature (4%).*** *Part B of this assignment will require you to make a 'last-minute extension' to your solution. The instructions for Part B will not be released until just before the final deadline for Assignment 2.*

You can add other features if you wish, as long as you meet these basic requirements. You must complete the task using only basic Python 3 features and the modules already imported into the provided template. **You may not use any Python modules that need to be downloaded and installed separately, such as "Beautiful Soup" or "Pillow". Only modules that are part of a standard Python 3 installation may be used.**

However, your solution is *not* required to follow precisely our example shown above. Instead you are strongly encouraged to *be creative* in your choices of web sites to access, the design of your Graphical User Interface, and the design of your generated HTML document.

Support tools

To get started on this task you need to download various web pages of your choice and work out how to extract the necessary elements for displaying data in the GUI and generating the HTML output file. You also need to allow for the fact that the contents of the web documents from which you get your data will change regularly, so you cannot hardwire the locations of the elements into your program. Instead you must use Python's string `find` method and/or regular expression `findall` function to extract the necessary elements, no matter where they appear in the HTML/XML source code.

To help you develop your solution, we have included two small Python programs with these instructions.

1. `web_doc_downloader` is a Python program containing a function called `download` that downloads and saves the source code of a web document as a Unicode file, as well as returning the document's contents to the caller as a character string. A copy of this function also appears in the provided program template. You can use it both to save copies of your chosen web documents for storage in your archive, as well as to download "live" web documents in your Python application at run time. Although recommended, you are not required to use this function in your solution, if you prefer to write your own "downloading" code to do the job.
2. `regex_tester` is an interactive program introduced in the lectures and workshops which makes it easy to experiment with different regular expressions on small text segments. You can use this together with downloaded text from the web to help perfect your regular expressions. (There are also many online tools that do the same job you could use instead.)

Internet ethics: Responsible scraping

The process of automatically extracting data from web documents is sometimes called "scraping". However, in order to protect their intellectual property, and their computational resources, owners of some web sites may not want their data exploited in this way. They will therefore deny access to their web documents by anything other than recognised web browsers such as Firefox, Internet Explorer, etc. Typically in this situation the web server will return a short "access denied" document to your Python script instead of the expected web document (Appendix B).

In this situation it's possible to trick the web server into delivering you the desired document by having your Python script impersonate a standard web browser. To do this you need to change the "user agent" identity enclosed in the request sent to the web server. Instructions for doing so can be found online. We leave it to your own conscience whether or not you wish to do this, but note that this assignment can be completed successfully without resorting to such subterfuge.

Security warning and plagiarism notice

This is an individual assessment item. All files submitted will be subjected to software plagiarism analysis using the MoSS system (http://theory.stanford.edu/~aiken/moss/). Serious violations of the university's policies regarding plagiarism will be forwarded to the Science and Engineering Faculty's Academic Misconduct Committee for formal prosecution.

As per QUT rules, you are not permitted to copy *or share* solutions to individual assessment items. In serious plagiarism cases SEF's Academic Misconduct Committee prosecutes both the copier and the original author equally. It is your responsibility to keep your solution secure. In particular, **you must not make your solution visible online via cloud-based code development platforms such as GitHub**. Note that free accounts for such platforms are usually public. If you wish to use such a resource, do so only if you have a *private* repository that cannot be seen by anyone else. For instance, students can apply for a *free* private repository in GitHub to keep their work secure (https://education.github.com/pack). However, we recommend that the best way to avoid being prosecuted for plagiarism is to keep your work well away from the Internet!

Deliverables

You should develop your solution by completing and submitting the provided Python template file `the_best.py`. Submit this in a "zip" archive containing all the files needed to support your application as follows:

1. Your `the_best.py` solution. Make sure you have completed the "statement" at the beginning of the Python file to confirm that this is your own individual work by inserting your name and student number in the places indicated. **Submissions without a completed statement will be assumed <u>not</u> to be your own work.**
2. Several *small* GIF files needed to support your GUI interface, but **no other image files**.
3. A folder containing the previously-downloaded web documents used for your static "archive" of old lists. Again, this folder may contain HTML/XML source code files only. **It must not contain any image files**. All images needed for your exported HTML document must be sourced from online when it is viewed in a web browser.

Once you have completed your solution and have zipped up these items submit them to Blackboard as a single file. **Submit your solution compressed as a "zip" archive. Do not use other compression formats such as "rar" or "7z".**

Apart from working correctly your Python and HTML code must be well-presented and easy to understand, thanks to (sparse) commenting that explains the *purpose* of significant elements and *helpful* choices of variable and function names. **Professional presentation** of your code will be taken into account when marking this assignment.

If you are unable to solve the whole problem, submit whatever parts you can get working. You will receive **partial marks for incomplete solutions**.

How to submit your solution

A link is available on Blackboard under Assessment for uploading your solution before the deadline (11:59pm Sunday, October 21st). Note that you can submit as many drafts of your

solution as you like. You are strongly encouraged to *submit draft solutions* before the deadline as insurance against computer and network failures. If you are unsure whether or not you have successfully uploaded your file, upload it again!

Students who encounter problems uploading their files to Blackboard should contact the IT Helpdesk (ithelpdesk@qut.edu.au; 3138 4000) for assistance and advice. Teaching staff will not answer email queries on the weekend the assignment is due, so ensure that you have successfully uploaded at least one solution by close-of-business on Friday, October 19th.

Appendix A: Some web sites that may prove helpful

For this assignment you need to find three (or more) regularly-updated online lists, containing at least ten items at all times. This appendix suggests some web sites which *may* be suitable for this assignment, but we don't guarantee that they all are. In particular, we have *not* checked them all to see if

- the lists change at least weekly, preferably daily, and
- they can be successfully downloaded by a Python program.

You should check both of these things before deciding to use any of the web sites below or any others you find yourself.

The following sites *appear* to be updated frequently, so *may* be suitable for this assignment.

- Top 100 stocks, https://www.barchart.com/stocks/top-100-stocks, plus several other promising pages at the same site.
- Surprisingly bullish stocks (ranked by standard deviations), https://www.barchart.com/stocks/price-surprises.
- Rotten Tomatoes has lots of lists that look useful, such as top box office, https://www.rottentomatoes.com/browse/in-theaters, most popular TV, https://www.rottentomatoes.com/browse/tv-list-2 and top DVD and streaming, https://www.rottentomatoes.com/browse/top-dvd-streaming.
- Another top ten box office movies list, https://www.fandango.com/rss/top10boxoffice.rss.
- The most-played tracks on Soundcloud, https://soundcloud.com/charts/top, which can also be listed by genres.
- Spotify charts Australia, https://spotifycharts.com/regional/au/daily/latest, plus other countries and dates.
- Various Apple RSS feeds, including songs, albums, movies, tv, and apps can be found at https://www.apple.com/rss/. For instance, iTunes store top songs, http://ax.itunes.apple.com/WebObjects/MZStoreServices.woa/ws/RSS/topsongs/xml, iTunes top albums http://ax.itunes.apple.com/WebObjects/MZStoreServices.woa/ws/RSS/topalbums/xml , iTunes top paid apps, http://ax.itunes.apple.com/WebObjects/MZStoreServices.woa/ws/RSS/toppaidapplications/xml, etc.
- The following pages contain the same Apple data, but in a more human-readable format, e.g., iTunes song charts, http://www.apple.com/au/itunes/charts/songs/, iTunes album charts, https://www.apple.com/au/itunes/charts/albums/, iTunes TV charts, https://www.apple.com/au/itunes/charts/tv-shows/, iTunes book charts, https://www.apple.com/au/itunes/charts/paid-books/, and so on for free/paid apps, movies, music videos.
- Google trends has many sites such as Google Australia daily search trends, https://trends.google.com/trends/trendingsearches/daily?geo=AU, plus the same data

as an RSS feed, https://trends.google.com/trends/trendingsearches/daily/rss?geo=AU, and Google real-time search trends, https://trends.google.com/trends/trendingsearches/realtime?geo=AU&category=all.

- Google Play lists such as top-selling free apps, https://play.google.com/store/apps/category/GAME/collection/topselling_free, top selling apps, https://play.google.com/store/apps/collection/topselling_paid, and top grossing apps, https://play.google.com/store/apps/category/GAME/collection/topgrossing.
- A large number of sales charts in UK and Irish "home entertainment" (games, music, video, etc.), https://www.chart-track.co.uk/index.jsp, although most seem to need a subscription.
- Another list of daily box office receipts, https://www.boxofficemojo.com/daily/chart/.
- Lists of the week's top-ten pirated movies, https://torrentfreak.com/category/dvdrip/feed/, *as used in our demonstration solution*.
- Players' rankings for Steam, http://steamcharts.com/top, seemingly updated continuously?
- Ultimate Guitar's top songs ranked by clicks in the last 24 hours, https://www.ultimate-guitar.com/top/tabs, which presumably updates frequently, but this has not been confirmed.
- US Nielsen rankings for television, music, video games, social interactions, https://www.nielsen.com/us/en/top10s.html, *as used in our demonstration solution*. (Update rates vary and all lists are embedded in one web document.)
- New Zealand music charts, singles, https://www.nztop40.co.nz/chart/singles, and albums, https://www.nztop40.co.nz/chart/albums, updated weekly.
- Several UK music charts, *as used in our demonstration solution*, http://www.officialcharts.com/charts/, updated weekly, e.g., singles, http://www.officialcharts.com/charts/singles-chart-update/, vinyl albums, http://www.officialcharts.com/charts/vinyl-albums-chart/, etc.
- Australian TV ratings, https://decidertv.com/ratings/, apparently updated daily.
- US TV Guide's trending TV shows, https://www.tvguide.com/trending-tonight/.
- American Top 40 music list, https://www.at40.com/charts/top-40-238/latest/, updated weekly.

Here are some other sites that may be suitable, but it's not always clear how often they are updated, so you need to check this. It's not enough to take the publisher's word. Just because a site says "updated daily" doesn't mean it will *change* daily!

- Top ten baby names, https://www.babynameguide.com/toptenDaily.asp, updated daily but how often does it change?
- Rotten Tomatoes top 100 movies, including "today's" top rated movies, http://www.rottentomatoes.com/top/bestofrt/.

- Statistics for Twitter, http://twittercounter.com/pages/100, radio station followers, https://twittercounter.com/pages/100/radio-station, TV show followers, https://twittercounter.com/pages/100/tv-show, artist followers, https://twittercounter.com/pages/100/artist, and many more.
- Video game sales, with lots of lists but probably not updated often, http://vgsales.wikia.com/wiki/Video_Game_Sales_Wiki, plus many others, e.g., http://vgsales.wikia.com/wiki/Best_selling_game_franchises.
- MYX music chart top ten, https://myx.abs-cbn.com/charts/myx-daily-top-10-pinoy/.
- Australian music charts, https://australian-charts.com/.
- Chess rankings, https://2700chess.com/, but may not be updated often.
- Lots of lists of uploaded images, links to videos, etc., https://funnyjunk.com, but very complicated web page, and probably too difficult to use for this assignment.
- GuildWars statistics, https://leaderboards.guildwars2.com/en/na/achievements, apparently updated daily?
- World of Wargraphs player statistics, https://www.worldofwargraphs.com/pvp-stats/best-players.
- Video Game Charts, http://www.vgchartz.com/, but very complex document structure.
- UK Top 40 singles chart, http://www.officialcharts.com/charts/uk-top-40-singles-chart/.
- BBC Top 40 singles chart, http://www.bbc.co.uk/radio1/chart/singles, and album chart, http://www.bbc.co.uk/radio1/chart/albums.
- Country music Top 40 chart, http://www.ct40.com/, but very little info for each song.
- Vodafone Big Top 40 music chart, https://www.bigtop40.com/.
- Movie Times Top 10 movies, http://www.the-movie-times.com/thrsdir/TopTen.cgi (very complex page).
- What seems to be a list of trending YouTube videos, https://www.youtube.com/feed/trending.
- Official World Golf Ranking, http://www.owgr.com/ranking (although we generally suggest avoiding sports rankings, because they may be out of season, this one seems to change a lot).
- YouTube channels, most subscribed, https://www.statsheep.com/p/Top-Subscribers, and most watched, https://www.statsheep.com/p/Top-Video-Views (plus several other categories, but it's not clear how often they're updated).
- Character rankings for Street Fighter 5, https://www.eventhubs.com/tiers/sf5/, and Super Smash Bros 4, https://www.eventhubs.com/tiers/ssb4/, and similarly for lots and lots of other games.
- Australian top character rankings for Super Smash Bros, https://qldsmash.com/Elo/SSBU, and also some other games.

- Publisher's Weekly Top 10 books, http://www.publishersweekly.com/pw/nielsen/top100.html, Top 10 science-fiction books, https://www.publishersweekly.com/pw/nielsen/xscifi.html, plus dozens of other categories.
- Most popular Anime series, http://myanimelist.net/topanime.php?type=bypopularity (claims to update twice daily), plus top Anime TV series, https://myanimelist.net/topanime.php?type=tv and top currently airing Anime, https://myanimelist.net/topanime.php?type=airing, and several other such lists.
- Aria albums chart, http://www.ariacharts.com.au/charts/albums-chart, singles chart, https://www.ariacharts.com.au/charts/singles-chart, music DVDs, https://www.ariacharts.com.au/charts/music-dvds-chart, plus many more music categories.
- IMDb movie box office receipts, https://www.imdb.com/chart/boxoffice, plus simpler mobile version, https://m.imdb.com/chart/boxoffice/.
- The Ranker site has hundereds of crowd-sourced popularity lists, but it's not clear how often each is updated, e.g., fantasy books, https://www.ranker.com/list/best-fantasy-book-series/ranker-books, best animated TV series, https://www.ranker.com/crowdranked-list/the-greatest-animated-series-ever-made, plus thousands of lists on virtually any topic!
- Billboard Top 100 lists, https://www.billboard.com/charts, probably updated weekly, e.g., the Hot 100, https://www.billboard.com/charts/hot-100.
- Ranker's RSS feed of its most popular lists, https://www.ranker.com/feed/mostpopularlists.rss.
- Card deck ratings for an online game, http://hearthpwn.com/decks?sort-=rating&sort-=rating, updated regularly, but very complicated page structure.
- World's worst spammers, https://www.spamhaus.org/statistics/spammers/, updated daily, but does it change often?
- List of trending books, https://www.goodreads.com/shelf/show/trending, but no indication of update frequency.
- League of Legends statistics, https://www.leaguespy.net/league-of-legends/champions, possibly updated weekly?
- Battlefield 4 statistics, http://bf4stats.com/leaderboards/pc_player_score.
- Games rankings for Dota 2, https://www.gosugamers.net/dota2/rankings, Counter-Strike, http://www.gosugamers.net/counterstrike/rankings, etc.
- Various music charts, https://www.beatport.com/genre/trance/7, but site has a very complex structure.
- IMDb top-rated movies, https://www.imdb.com/chart/top, plus many other lists, e.g., most popular comedy titles, https://www.imdb.com/search/title?genres=comedy, top-rated TV shows, https://www.imdb.com/chart/toptv, most popular movies, http://www.imdb.com/chart/moviemeter, most popular celebrities, https://m.imdb.com/chart/starmeter, but may not change often?

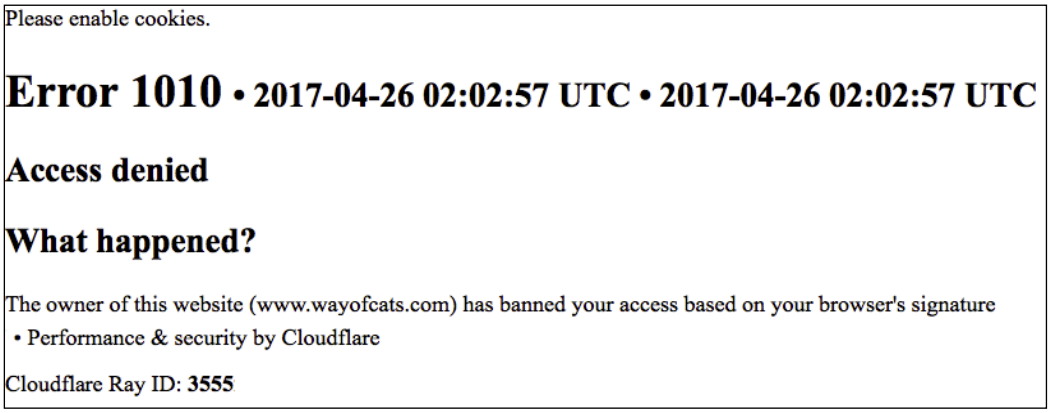
The following sites look promising at first, but may not update frequently. If you want to use one of these lists you must first confirm that it changes frequently enough.

- A huge number of lists, https://www.statista.com/, but most are probably not updated often.
- A huge number of lists, http://www.listchallenges.com, but complex pages and unclear update frequency.
- Lots of lists, https://www.therichest.com, but complex pages and unclear update frequency.
- Wonderslist has a huge number of lists, e.g., https://www.wonderslist.com/10-cartoon-characters-with-psychological-disorders/, but they don't look like they are updated regularly.

Appendix B: Web sites that block access to Python scripts

As noted above, some web servers will block access to web documents by Python programs in the belief that they may be malware or in order to protect the owner's computing resources and data assets. In this situation they usually return a short HTML document containing an "access denied" message instead of the document requested. This can be very confusing because you can usually view the document without any problems using a standard web browser even though your Python program is delivered something entirely different.

If you suspect that your Python program isn't being allowed to access your chosen web page, use the `web_doc_downloader` program to check whether or not Python programs are being sent an access denied message. When viewed in a web browser, such messages typically look something like the following example. In this case blog `www.wayofcats.com` has used anti-malware application *Cloudflare* to block access to the blog's contents by our Python program.


Please enable cookies.

Error 1010 • 2017-04-26 02:02:57 UTC • 2017-04-26 02:02:57 UTC

Access denied

What happened?

The owner of this website (www.wayofcats.com) has banned your access based on your browser's signature

• Performance & security by Cloudflare

Cloudflare Ray ID: **3555**

In this situation you are encouraged to choose another source of data. Although it's possible to trick some web sites into delivering blocked pages to a Python script by changing the "user agent" signature sent to the server in the request we *don't* recommend doing so, partly because this solution is not reliable and partly because it could be considered unethical to deliberately override the website owner's wishes.