**Setup**

Python 2.7 64bit for Windows with pip (should be installed by default) is required to run this project.

To run these Python commands as shown, these two directories should be included in your PATH environmental variables:

C:\Python27\  
C:\Python27\Scripts\

To install the required modules, type these commands into the command prompt:

pip install lxml==3.6.0  
pip install twisted[windows\_platform]==16.6  
pip install selenium  
pip install protobuf

To run the Selenium spider correctly, the version of Chrome browser should be supported by the version of ChromeDriver being used. Currently using ChromeDriver 2.29 with Chrome v58.

To update to the latest version of ChromeDriver, download chromedriver.exe from here, replace the file in the project folder, then update Chrome to the latest supported version.  
<https://sites.google.com/a/chromium.org/chromedriver/downloads>

If there are errors causing the googleplayapi or Scrapy spiders to not run after following the above steps, it may be due to missing Python modules that should be mentioned in the error message. Try to use ‘pip install <module>’ for any missing package exceptions that appear.

**Useful Commands**

Start a Scrapy spider and store data in the crawls folder if run from the root gplaycrawler directory

scrapy crawl <spidername> -o crawls\SampleName.json -s JOBDIR=crawls\SampleName

Run topfreeapps.py script and store data in SampleName.json (must be run from gplayapi directory)

topfreeapps.py SampleName.json 100

Run downloadapks.py script with a custom app file (running script without custom filename will search for and use default file pkgnames.txt) (must be run from gplayapi directory)

downloadapks.py SampleName.txt

Run python\_version.py to know which bit architecture the Python shell is running

python\_version.py

**File Breakdown**

**pipelines.py**

Used to process items returned by the Scrapy spider. This can be edited to change the default behavior of how data is stored, such as storing data in a database and applying rules to the data.

**items.py**

Contains item definitions for all the fields that will be processed in each spider. They should normally all be defined the same.

**spiders\ScrapySpider.py**

A spider that will only use the built-in Scrapy crawler to scrape data from the page. It will run faster than SeleniumSpider.py as it does not have to render anything on the page. However, it also cannot interact with elements on the page. Elements that require a JavaScript action, such as scrolling to view more items or clicking a link to open a popup modal, cannot be scraped.

**spiders\SeleniumSpider.py**

A spider that uses Selenium and the Chrome webdriver to render pages for crawling. It will run slower than ScrapySpider.py, but it can interact with elements on a page, allowing it to grab data loaded by JavaScript.

**gplayapi\googleplay.py**

The unofficial Google Play Python API. Read more at the original GitHub project for examples, forks, and a description <https://github.com/egirault/googleplay-api>

**gplayapi\config.py**

The file that handles authentication with the Google Play store. Attempting to use the gplayapi for the first time in a directory will run the authentication setup. It will create a new file named auth.json in the same directory the gplayapi is called that contains language, Google Service Framework ID (GSF), Google account email, Google account password. These will be stored as plain text. Multiple auth.json files may be created if the gplayapi is called from the command prompt in different directories.

The Google account will need to be connected to an Android device. The GSF for the device being used can be gotten from an app called Device ID by Evozi.

**gplayapi\topfreeapps.py**

Will attempt to download and grab some generic information from the top free apps from all the categories in the Google Play App store. Categories are manually listed in the file. The number of apps grabbed from each category is passed with the command to execute the script and is capped at 100 due to limits in the API.

The generic information grabbed is: Title, Package Name, Developer Name, Superdev status, Price, Offer Type, Version, Size, Rating, Number of Downloads, Description, Permissions.

**gplayapi\downloadapks.py**

Will attempt download all apps that are passed to it from a file named pkgnames.txt by default or given to it in the command to run the script. The file must contain the package names for all the apps that the script is to download. All package names must be written on a new line. Untested with large files, but the method used should handle memory management correctly. Any failed packages will be added to a new file named failed.txt. Receiving ‘list index out of range’ exception for packages may indicate that the APK cannot be downloaded. Number of exceptions thrown correlates with the number of APKs that failed to download.

**Credits**

Original projects used to create this one:

<https://github.com/manojps/google-play-apps-crawler-scrapy>

<https://github.com/egirault/googleplay-api>