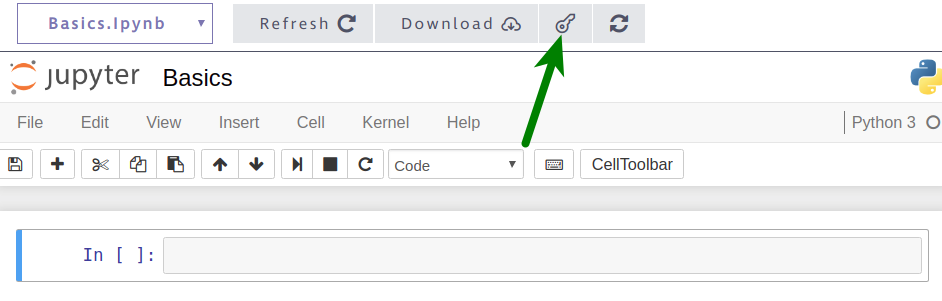
**Guided Project: Profitable App Profiles for the App Store and Google Play Markets**

**Requirements:  
Python (**[**Download**](https://www.python.org/downloads/)**)  
Jupyter Notebook ([path] -m pip install jupyterlab) (**[**instructions**](https://jupyter.org/install.html)**)**

You are a data analysts for a company that makes mobile apps. All the apps target an English-speaking audience, are free, and contain ads, how much money we make is mostly influenced by how many people use our app. You are supposed to gather data to show the devs what app they should make to make the most profit.

  
Sidenote:  
-You can use this on your portfolio  
-What you need to do is below the “instructions” heading, use the outline or the navigator to go through the document headings

# **STEP 1: Intro**

First, let’s tell your ~~readers~~ professor what your project is about. Use the first Markdown cell of the notebook to:

1. Add a title.
2. Explain in 1-2 paragraphs (no more):
   * What this is about
   * What your goal is

Don’t spend too much time here, this is a draft (an idea).

# **[Done]** STEP 2: **Finding, exploring, and picking data**

Now, we need to analyze data about the mobile apps available on Google Play and App Store.

### **Finding** data to use

There are a lot of apps and yeah it’s too costly to gather all of that data and analyze it, instead we are using a couple of samples (the ones from Moodle), way easier.

### Getting familiar with exploring data sets

The function below is used to explore the sets easierTo make the data sets easier for you to explore, we created a function named explore\_data() that you can repeatedly use to print rows in a readable way.

def explore\_data(dataset, start, end, rows\_and\_columns=False):

dataset\_slice = dataset[start:end]

for row in dataset\_slice:

print(row)

print('\n') # adds a new (empty) line after each row

​

if rows\_and\_columns:

print('Number of rows:', len(dataset))

print('Number of columns:', len(dataset[0]))

Disclaimer: dataset assumes there’s no header row, and there shouldn’t be for this project. If there was, skip the first row.

### Instructions

Now let's open the two data sets and explore them.

1. Open the two data sets we mentioned above, and save both as lists of lists.
   * If you run into an error named UnicodeDecodeError....  
     add encoding="utf8" to the open() function (for instance, use open('AppleStore.csv', encoding='utf8')).
2. Explore both data sets using the explore\_data() function above.
   * Print the first few rows of each data set.
   * Find the number of rows and columns of each data set.
3. Print the column names and **try to identify the columns that could help us with our analysis.**
   * If you don’t understand a column or if a column is not descriptive enough, add a link to the documentation so the readers can understand.

# [Done] STEP 3: Data Cleaning

Data Analysts spend most of their time during this process. You’ll be guided on how to do this.

## [Done] Part **I:** Looking at online reports of wrong data

Fix errors from these samples that had been reported by people online.

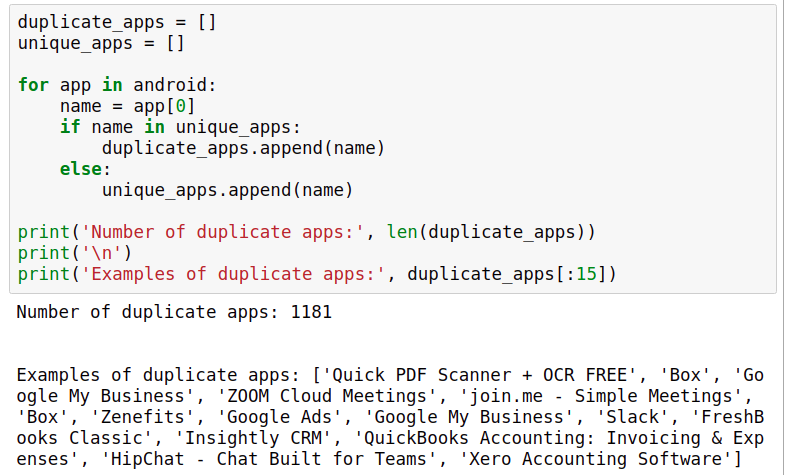
### Instructions

The Google Play data set has a dedicated [discussion section](https://www.kaggle.com/lava18/google-play-store-apps/discussion), and we can see that [one of the discussions](https://www.kaggle.com/lava18/google-play-store-apps/discussion/66015) describes an error for a certain row.

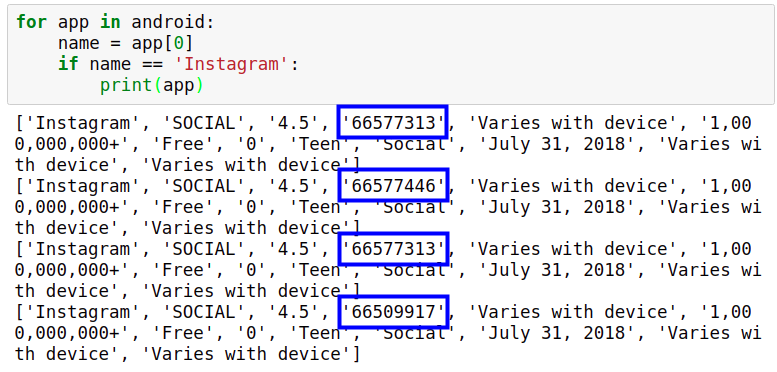
1. Read the discussion and find out what the index of the row is.
2. Print the row at that index to check whether it's indeed incorrect. Take into account the user reporting the error might or might have not removed the header row, so the index number might vary.
3. If the row has an error, remove the row using the [del](_blank) statement. For instance, to remove the row with the index 149 from a data set data that is stored as a list of list, you can use the code del data[149].
4. Read the [discussion section](https://www.kaggle.com/ramamet4/app-store-apple-data-set-10k-apps/discussion) for the App Store data set, and see whether you can find any reports of wrong data.

## [Done] Part II: Duplicated apps

There are 1181 duplicated apps on the Google Play Store data set. This is what we need to do to remove the duplicated apps from the data sets.



One thing we could do is remove the duplicate rows randomly, but we could probably find a better way.



Looking at this you can tell that the fourth row is the main difference here, the fourth row shows the number of reviews which means that the higher the number the more recent it is. We should try to keep only the most recent one and remove the rest.

We'll remove the rows on the next part. Now write some code.

**Instructions**

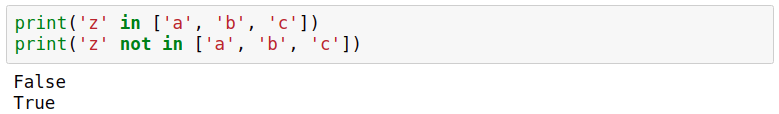
1. Explain to the professor that the Google Play data set has duplicate entries. Print a few duplicate rows to confirm.
2. Count the number of duplicates using the technique we learned above. (1181??)
3. Explain that you won't remove the duplicates randomly, and that you’ll keep the most recent one.

## [ConsultProf]Part III: Removing the duplicates

Note:Documentation left, need to consult professor, second step in instructions make no changes and I have no idea what it is for. If professor can’t help, analyze on my own or improvise and pretend like I know what it is for.

We looped through the Google Play data set and found that there are 1,181 duplicates. After we remove the duplicates, we should be left with 9,659 rows:

[W](_blank)e should use “not in” here



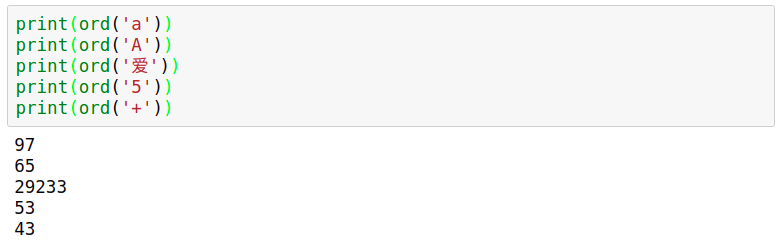
Now let's write the code to remove the duplicate entries.

### Instructions

1. Create the dictionary named reviews\_max
   * Loop through the Google Play data set (make sure you don't include the header row). For each iteration:
     + Assign the app name to a variable named name.
     + Convert the number of reviews to float. Assign it to a variable named n\_reviews.
     + If name already exists as a key in the reviews\_max dictionary **and** reviews\_max[name] < n\_reviews, update the number of reviews for that entry in the reviews\_max dictionary.
     + If name **is not in** the reviews\_max dictionary as a key, create a new entry in the dictionary where the key is the app name, and the value is the number of reviews. Make sure you don't use an else clause
   * Inspect the dictionary to make sure everything went as expected. Measure the length of the dictionary — remember that the expected length is 9,659 entries.
2. Use the dictionary you created above to remove the duplicate rows:
   * Start by creating two empty lists: android\_clean (which will store our new cleaned data set) and already\_added (which will just store app names).
   * Loop through the dataset again and do the same thing you did before for each iteration
   * If n\_reviews is the same as the number of maximum reviews of the app name (the number can be found in the reviews\_max dictionary) **and** name is not already in the list already\_added (read the solution notebook to find out why we need this supplementary condition):
     + Append the entire row to the android\_clean list (which will eventually be a list of list and store our cleaned data set).
     + Append the name of the app name to the already\_added list — this helps us to keep track of apps that we already added.
3. Explore the android\_clean data set to ensure everything went as expected. The data set should have 9,659 rows. The two steps above are a bit more involved, so make sure you use Markdown to explain the readers the steps you took.

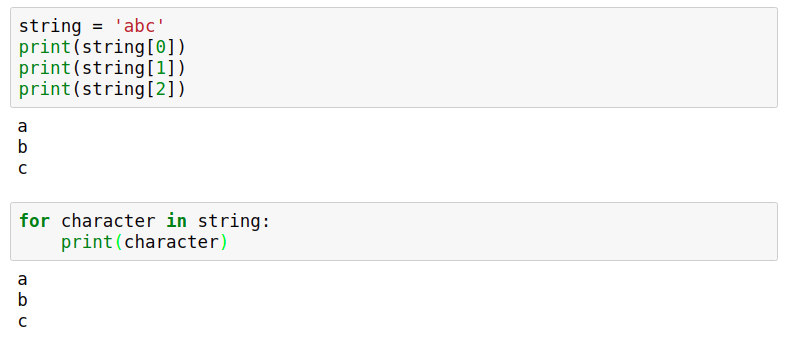
## **[Done] Part IV: Removing non-english speaking apps**

To do this, we are going to use the [ord()](_blank) function. Each character has a number associated with it, and ord() gets that character.



Characters with numbers that range 0 to 127 are english according to ASCII. Anything above is not an english character or it’s a symbol (like an emoji).

Short reminder on how to do stuff with each letter of a string



### Instructions

Disclaimer: Don’t use this with your dataset yet

1. Write a function that takes in a string and returns False if there's any character in the string that doesn't belong to the set of common English characters, otherwise it returns True.
   * Inside the function, iterate over the input string. Check if the value is higher than 127, if it’s higher then the function should immediately return False
   * If the loop finishes running without the return statement being executed, then it should return True.
2. Use these samples to see if it works properly:
   * 'Instagram'
   * '爱奇艺PPS -《欢乐颂2》电视剧热播'
   * 'Docs To Go™ Free Office Suite'
   * 'Instachat 😜'

### **This has false positives.**

Because emojis and symbols are a thing and they are not in the ASCII range.

1. If the input string has more than three characters outside the ASCII range (0 - 127), then it should return False (identify the string as non-English), otherwise it should return True.
2. Use the samples again
   * 'Docs To Go™ Free Office Suite'
   * 'Instachat 😜'
   * '爱奇艺PPS -《欢乐颂2》电视剧热播'
3. Now use this on your data set. If an app name is identified as English, append the whole row to a separate list.
4. See how many rows you have remaining for each data set.

## **[Done] Part V: Removing paid apps.**

1. Remove paid apps from the data sets. Make sure you identify the columns describing the app price correctly.
2. Check the length of each data set to see how many apps you have remaining.

# **STEP 4: Data Analysis**

## [Done] Part I: Profiling

Your aim is to determine the kinds of apps that are likely to attract more users because our revenue is highly influenced by the number of people using our apps.

To minimize risks and overhead, a validation strategy has been developed. You are gonna summarize this:

1. Build a minimal Android version of the app, and add it to Google Play.
2. If the app has a good response from users, we develop it further.
3. If the app is profitable after six months, we build an iOS version of the app and add it to the App Store.

Because our end goal is to add the app on both Google Play and the App Store, we need to find app profiles that are successful on both markets.

### Instructions

First let’s find the most common genres for each market. For this, we'll need to build frequency tables for a few columns in our data sets.

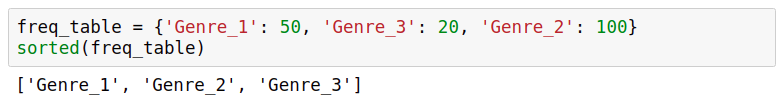
1. Explain to the professor why we want to find an app profile that fits both the App Store and Google Play. Explain in your own words our validation strategy for an app idea.
2. Identify the columns you could use to figure out what are the most common genres in each market.

## **[Done] Part II: Sorting genre**

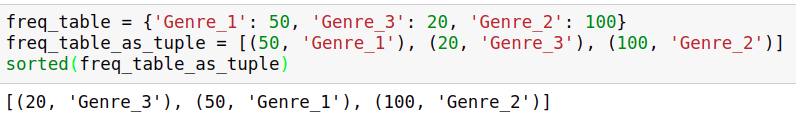
Two things you need to do

* Generate percentages from the tables.
* Display the percentages in descending order (in a second function).

We’ll use sorted() here. It doesn't work with dictionaries because it sees the dictionary keys.



However, it works on a list of tuples. To ensure the sorting works right, the dictionary value comes first, and the dictionary key comes second:



Using the workaround above, we wrote a helper function for you below named display\_table(), which you'll be able to combine with the function you're going to write in the next exercise. The display\_table() function you see below:

def display\_table(dataset, index):

table = freq\_table(dataset, index)

table\_display = []

for key in table:

key\_val\_as\_tuple = (table[key], key)

table\_display.append(key\_val\_as\_tuple)

​

table\_sorted = sorted(table\_display, reverse = True)

for entry in table\_sorted:

print(entry[1], ':', entry[0])

Let's now create a function for generating frequency tables, and use it in combination with the display\_table() function.

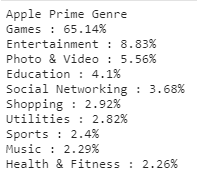
### Instructions

1. Create a function named freq\_table() that takes in two inputs: dataset and index.
   * The function should return the frequency table (as a dictionary) for any column we want. The frequencies should also be expressed as percentages.
2. Copy the display\_table() function we wrote above. Use it to display the frequency table of the columns prime\_genre, Genres, and Category. We'll analyze the resulting tables on the next part of step 4.

## **[Done] Part III: Studying analysis of genre**

### Instructions

1. Analyze the frequency table you generated for the prime\_genre column of the App Store data set.



* + What is the most common genre? What is the runner-up?

The most common is Games, after that it’s Entertainment

* + What other patterns do you see?

The least common are catalogs and medical

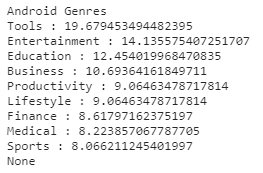
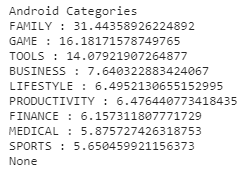
* + What is the general impression — are most of the apps designed for practical purposes (education, shopping, utilities, productivity, lifestyle) or more for entertainment (games, photo and video, social networking, sports, music)?

Most apps are designed for entertainment, with games being the most dominant in the atmosphere. Even if games were ignored, entertainment would still be the most dominant.

* + Can you recommend an app profile for the App Store market based on this frequency table alone? If there's a large number of apps for a particular genre, does that also imply that apps of that genre generally have a large number of users?

I can’t. The data only shows how many apps there are for an app. Nothing says that these apps are getting more appeal, it only shows how much competition there is.

1. Analyze the frequency table you generated for the Category and Genres column of the Google Play data set.



* + What are the most common genres?

Family / Game / Tools and Tools / Entertainment / Education

* + What other patterns do you see?

Medical, finance, and sports have the least amount of apps on both rows

* + Compare the patterns you see for the Google Play market with those you saw for the App Store market.

The Google Play Store has a lot more apps designed for utility compared to the App Store. This is most likely because of the freedom android devices provide to developers, and also because the code is open source.

* + Can you recommend an app profile based on what you found so far? Do the frequency tables you generated reveal the most frequent app genres or what genres have the most users?

I can’t suggest an app profile. It only shows the most frequent app genres.

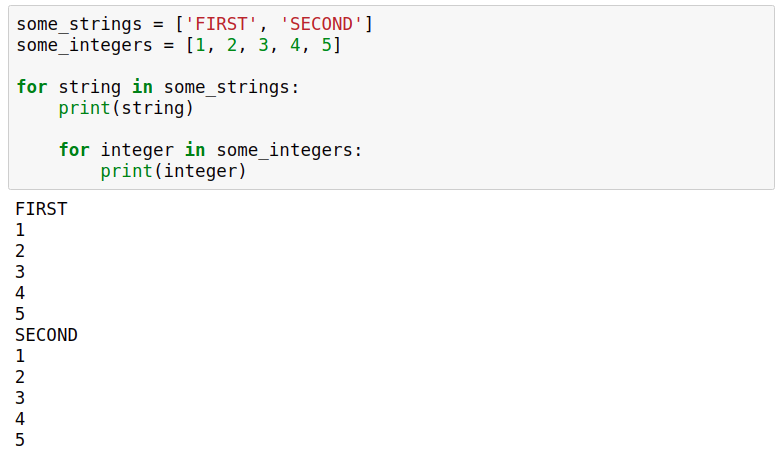
## **Part IV: Sorting by popularity (App Store)**

We found out from the tables we analyzed that the App Store is dominated by apps made for fun, while Google Play is dominated by fun but also practical apps. Now we want to know how many people install these apps, this is shown on Google Play, but not on the App Store. We can workaround the app store by using the number of user ratings instead (found on the rating\_count\_tot column)

Let's start with calculating the average number of user ratings per app genre on the App Store. To do that, we'll need to:

* Isolate the apps of each genre.
* Sum up the user ratings for the apps of that genre.
* Divide the sum by the number of apps belonging to that genre (not by the total number of apps).

To calculate the average number of user ratings for each genre, we'll use a for loop inside of another for loop, this is called a **nested loop**.



### Instructions

1. Start by generating a frequency table for the prime\_genre column to get the unique app genres (below, we'll need to loop over the unique genres). You can use the freq\_table() function you wrote previously on part II
2. Loop over the unique genres of the App Store data set. For each iteration (below, we'll assume that the iteration variable is named genre):
   * do total = 0. This variable will store the sum of the number of user ratings specific to each genre.
   * Do len\_genre = 0. This variable will store the number of apps specific to each genre.
   * Loop over the App Store data set, and for each iteration:
     + Save the app genre to a variable named genre\_app.
     + If genre\_app is the same as genre (the iteration variable of the main loop), then:
       - Save the number of user ratings of the app as a float.
       - Add up the number of user ratings to the total variable.
       - Increment the len\_genre variable by 1.
   * Compute the average number of user ratings by dividing total by len\_genre. This should be done outside the nested loop.
   * Print the app genre and the average number of user ratings. This should also be done outside the nested loop.
3. Analyze the results and try to come up with at least one app profile recommendation for the App Store. (Basically, tell the devs what kind of app would be more profitable)

## **Part V: Sorting by popularity (Play Store)**

The install numbers are not precise, it has symbols and commas, but that’s fine. The only problem is that we need to convert each install number from string to float. This means we need to remove the commas and the plus characters, otherwise the conversion will fail.

Remove characters with [str.replace(old, new)](_blank) method.

Now let's calculate the average number of installs per app genre for the Google Play data set. We'll need to use a nested loop, just like in the previous screen.

### Instructions

1. Do the exact same shit as before but for google play, literally just copy paste and change variable names and change the index.
2. Come up with an app profile recommendation for Google Play. Remember, your aim is to recommend an app genre that shows potential for being profitable on both the App Store and Google Play.

# CONCLUSION

That’s it, you did it, wooo, celebration, that’s all, submit this to your professor and good job. Remember that you need to share and explain what app do you suggest the devs to make.

When in doubt of your project, remember to visit the solution notebook (and also, that your profile recommendations from Step 4 Part IV and V are completely up to you, doesn’t have to be the same as the solution notebook.)

Revise your work, **submit as proj03.ipynb**, remember to have your name and the project name on the first cell, and that’s all.