Let's summarize everything we learned in this course:

* How to work with strings.
* Object-oriented programming.
* Dates and times.

Next, we'll synthesize what we learned to complete a data analysis project.

Working on **guided projects** will give you hands-on experience with real world examples, so we encourage you to not only complete them, but to take the time to really understand the concepts.

These projects are *meant* to be challenging to better prepare you for the real world, so don't be discouraged if you have to refer back to previous missions. If you haven't worked with Jupyter Notebook before or need a refresher, we recommend completing our [Jupyter Notebook Guided Project](https://app.dataquest.io/m/349/project%3A-learn-and-install-jupyter-notebook" \t "_blank) before continuing.

In this project, we'll work with a data set of submissions to popular technology site [Hacker News](https://news.ycombinator.com/).



Hacker News is a site started by the startup incubator [Y Combinator](https://www.ycombinator.com/), where user-submitted stories (known as "posts") are voted and commented upon, similar to reddit. Hacker News is extremely popular in technology and startup circles, and posts that make it to the top of Hacker News' listings can get hundreds of thousands of visitors as a result.

You can find the data set [here](https://www.kaggle.com/hacker-news/hacker-news-posts), but note that it has been reduced from almost 300,000 rows to approximately 20,000 rows by removing all submissions that did not receive any comments, and then randomly sampling from the remaining submissions. Below are descriptions of the columns:

* id: The unique identifier from Hacker News for the post
* title: The title of the post
* url: The URL that the posts links to, if the post has a URL
* num\_points: The number of points the post acquired, calculated as the total number of upvotes minus the total number of downvotes
* num\_comments: The number of comments that were made on the post
* author: The username of the person who submitted the post
* created\_at: The date and time at which the post was submitted

Here are the first few rows of the data set:

| **id** | **title** | **url** | **num\_points** | **num\_comments** | **author** | **created\_at** |
| --- | --- | --- | --- | --- | --- | --- |
| 12224879 | Interactive Dynamic Video | http://www.interactivedynamicvideo.com/ | 386 | 52 | ne0phyte | 8/4/2016 11:52 |
| 10975351 | How to Use Open Source and Shut the F\*ck Up at the Same Time | http://hueniverse.com/2016/01/26/how-to-use-open-source-and-shut-the-fuck-up-at-the-same-time/ | 39 | 10 | josep2 | 1/26/2016 19:30 |
| 11964716 | Florida DJs May Face Felony for April Fools' Water Joke | http://www.thewire.com/entertainment/2013/04/florida-djs-april-fools-water-joke/63798/ | 2 | 1 | vezycash | 6/23/2016 22:20 |
| 11919867 | Technology ventures: From Idea to Enterprise | https://www.amazon.com/Technology-Ventures-Enterprise-Thomas-Byers/dp/0073523429 | 3 | 1 | hswarna | 6/17/2016 0:01 |
| 10301696 | Note by Note: The Making of Steinway L1037 (2007) | http://www.nytimes.com/2007/11/07/movies/07stein.html?\_r=0 | 8 | 2 | walterbell | 9/30/2015 4:12 |

We're specifically interested in posts whose titles begin with either Ask HN or Show HN. Users submit Ask HN posts to ask the Hacker News community a specific question. Below are a couple examples:

Ask HN: How to improve my personal website?

Ask HN: Am I the only one outraged by Twitter shutting down share counts?

Ask HN: Aby recent changes to CSS that broke mobile?

Likewise, users submit Show HN posts to show the Hacker News community a project, product, or just generally something interesting. Below are a couple of examples:

Show HN: Wio Link ESP8266 Based Web of Things Hardware Development Platform'

Show HN: Something pointless I made

Show HN: Shanhu.io, a programming playground powered by e8vm

We'll compare these two types of posts to determine the following:

* Do Ask HN or Show HN receive more comments on average?
* Do posts created at a certain time receive more comments on average?

Let's start by importing the libraries we need and reading the data set into a list of lists.

Instructions

1. Start by adding a title and writing a paragraph in a markdown cell introducing the project and the data set. The title and the introduction are tentative at this point, so don't spend too much time here — you can come back at the end of your work to refine them.
2. Read the hacker\_news.csv file in as a list of lists.
   * Assign the result to the variable hn.
3. Display the first five rows of hn.

In the last screen, we read our data into a list of lists called hn. When you displayed the first five rows of hn, you should've seen the following data:

[

  ['id', 'title', 'url', 'num\_points', 'num\_comments', 'author', 'created\_at'],

  ['12224879', 'Interactive Dynamic Video', 'http://www.interactivedynamicvideo.com/', '386', '52', 'ne0phyte', '8/4/2016 11:52'],

  ['10975351', 'How to Use Open Source and Shut the Fuck Up at the Same Time', 'http://hueniverse.com/2016/01/26/how-to-use-open-source-and-shut-the-fuck-up-at-the-same-time/', '39', '10', 'josep2', '1/26/2016 19:30'],

  ['11964716', "Florida DJs May Face Felony for April Fools' Water Joke", 'http://www.thewire.com/entertainment/2013/04/florida-djs-april-fools-water-joke/63798/', '2', '1', 'vezycash', '6/23/2016 22:20'],

  ['11919867', 'Technology ventures: From Idea to Enterprise', 'https://www.amazon.com/Technology-Ventures-Enterprise-Thomas-Byers/dp/0073523429', '3', '1', 'hswarna', '6/17/2016 0:01']

]

Notice that the first list in the inner lists contains the column headers, and the lists after contain the data for one row. In order to analyze our data, we need to first remove the row containing the column headers. Let's remove that first row next.

Instructions

1. Extract the first row of data, and assign it to the variable headers.
2. Remove the first row from hn.
3. Display headers.
4. Display the first five rows of hn to verify that you removed the header row properly.

Now that we've removed the headers from hn, we're ready to filter our data. Since we're only concerned with post titles beginning with Ask HN or Show HN, we'll create new lists of lists containing just the data for those titles.

To find the posts that begin with either Ask HN or Show HN, we'll use the string method [startswith](https://docs.python.org/3/library/stdtypes.html" \l "str.startswith" \t "_blank). Given a string object, say, string1, we can check if starts with, say, dq by inspecting the output of the object string1.startswith('dq'). If string1 starts with dq, it will return True, otherwise it will return False.

print('dataquest'.startswith('Data'))

print('dataquest'.startswith('data'))

False

True

In the example above, the first print call gives us False because dataquest does not start with Data. The second print call prints True because dataquest does start with data. Capitalization matters.

If we wish to control for case, we can use the [lower](https://docs.python.org/3/library/stdtypes.html#str.lower) method which returns a lowercase version of the starting string. Here's an example:

print('DataQuest'.lower())

dataquest

Let's use these methods to separate posts beginning with Ask HN and Show HN (and case variations) into two different lists next.

Instructions

1. Create three empty lists called ask\_posts, show\_posts, and other\_posts.
2. Loop through each row in hn.
   * Assign the title in each row to a variable named title.
     + Because the title column is the second column, you'll need to get the element at index 1in each row.
3. Implement the following steps:
   * If the lowercase version of title starts with ask hn, append the row to ask\_posts.
   * Else if the lowercase version of title starts with show hn, append the row to show\_posts.
   * Else append to other\_posts.
4. Check the number of posts in ask\_posts, show\_posts, and other\_posts.

In the last screen, we separated the "ask posts" and the "show posts" into two list of lists named ask\_posts and show\_posts. Below are the first five rows in the ask\_posts list of lists:

[

  ['12296411', 'Ask HN: How to improve my personal website?', '', '2', '6', 'ahmedbaracat', '8/16/2016 9:55'],

  ['10610020', 'Ask HN: Am I the only one outraged by Twitter shutting down share counts?', '', '28', '29', 'tkfx', '11/22/2015 13:43'],

  ['11610310', 'Ask HN: Aby recent changes to CSS that broke mobile?', '', '1', '1', 'polskibus', '5/2/2016', 10:14'],

  ['12210105', 'Ask HN: Looking for Employee #3 How do I do it?', '', '1', '3', 'sph130', '8/2/2016 14:20'],

  ['10394168', 'Ask HN: Someone offered to buy my browser extension from me. What now?', '', '28', '17', 'roykolak', '10/15/2015 16:38']

]

Below are the first five rows in the show\_posts list of lists:

[

  ['10627194', 'Show HN: Wio Link ESP8266 Based Web of Things Hardware Development Platform', 'https://iot.seeed.cc', '26', '22', 'kfihihc', '11/25/2015 14:03'],

  ['10646440', 'Show HN: Something pointless I made', 'http://dn.ht/picklecat/', '747', '102', 'dhotson', '11/29/2015 22:46'],

  ['11590768', 'Show HN: Shanhu.io, a programming playground powered by e8vm', 'https://shanhu.io', '1', '1', 'h8liu', '4/28/2016 18:05'],

  ['12178806', 'Show HN: Webscope Easy way for web developers to communicate with Clients', 'http://webscopeapp.com', '3', '3', 'fastbrick', '7/28/2016 7:11'],

  ['10872799', 'Show HN: GeoScreenshot Easily test Geo-IP based web pages', 'https://www.geoscreenshot.com/', '1', '9', 'kpsychwave', '1/9/2016 20:45']

]

Next, let's determine if ask posts or show posts receive more comments on average.

Instructions

1. Find the total number of comments in ask posts and assign it to total\_ask\_comments.
   * Set total\_ask\_comments to 0.
2. Use a for loop to iterate over the ask posts.
   * Because the num\_comments column is the fifth column in ask\_posts, you'll need to get the element at index 4 in each row.
     + You'll also need to convert the value to an integer so that we can calculate the sum of all the comments.
     + Add this value to total\_ask\_comments.
3. Compute the average number of comments on ask posts and assign it to avg\_ask\_comments.
4. Print avg\_ask\_comments.
5. Find the total number of comments in show posts and assign it to total\_show\_comments.
   * Set total\_show\_comments to 0.
6. Use a for loop to iterate over the show posts.
   * Because the num\_comments column is the fifth column in show\_posts, you'll need to get the element at index 4 in each row.
     + You'll also need to convert the value to an integer so that we can calculate the sum of all the comments.
     + Add this value to total\_show\_comments.
7. Compute the average number of comments on show posts and assign it to avg\_show\_comments.
8. Print avg\_show\_comments.
9. Do show posts or ask posts receive more comments on average? Write a markdown cell explaining your findings.

In the last screen, you should've determined that, on average, ask posts receive more comments than show posts. Since ask posts are more likely to receive comments, we'll focus our remaining analysis just on these posts.

Next, we'll determine if ask posts created at a certain *time* are more likely to attract comments. We'll use the following steps to perform this analysis:

1. Calculate the amount of ask posts created in each hour of the day, along with the number of comments received.
2. Calculate the average number of comments ask posts receive by hour created.

In this screen, we'll tackle the first step — calculating the amount of ask posts and comments by hour created. We'll use the [datetime module](https://docs.python.org/3/library/datetime.html) to work with the data in the created\_at column.

Recall that we can use the datetime.strptime() constructor to parse dates stored as strings and return datetime objects:

date\_1\_str = "December 24, 1984"

date\_1\_dt = dt.datetime.strptime(date\_1\_str, "%B %d, %Y")

Note in the example above that %B represents the full name of the month, %d represents a two digit day, and %Y represents the four digit year. If you need to parse a string in a different format, you'll have to modify these codes.

Let's use this technique to calculate the amount of ask posts created per hour, along with the total amount of comments.

Instructions

1. Import the [datetime module](https://docs.python.org/3/library/datetime.html) as dt.
2. Create an empty list and assign it to result\_list. This will be a list of lists.
3. Iterate over ask\_posts and append to result\_list a list with two elements:
   * The first element shall be the column created\_at.
     + Because the created\_at column is the seventh column in ask\_posts, you'll need to get the element at index 6 in each row.
   * The second element shall be the number of comments of the post.
     + You'll also need to convert the value to an integer.
4. Create two empty dictionaries called counts\_by\_hour and comments\_by\_hour.
5. Loop through each row of result\_list.
6. Extract the hour from the date, which is the first element of the row.
7. Use the datetime.strptime() method to parse the date and create a datetime object.
8. Use the string we want to parse as the first argument and a string that specifies the format as the second argument.
   * Use the datetime.strftime() method to select just the hour from the datetime object.
   * **If the hour isn't** a key in counts\_by\_hour:
     + Create the key in counts\_by\_hour and set it equal to 1.
     + Create the key in comments\_by\_hour and set it equal to the comment number.
   * **If the hour is** already a key in counts\_by\_hour:
     + Increment the value in counts\_by\_hour by 1.
     + Increment the value in comments\_by\_hour by the comment number.

In the last screen, we created two dictionaries:

* counts\_by\_hour: contains the number of ask posts created during each hour of the day.
* comments\_by\_hour: contains the corresponding number of comments ask posts created at each hour received.

Next, we'll use these two dictionaries to calculate the average number of comments for posts created during each hour of the day.

To illustrate the technique, let's work with the following dictionary:

sample\_dict = {

               'apple': 2,

               'banana': 4,

               'orange': 6

              }

Suppose we wanted to multiply each of the values by ten and return the results as a list of lists. We can use the following code:

fruits = []

​

for fruit in sample\_dict:

   fruits.append([fruit, 10\*sample\_dict[fruit]])

Below are the results:

[['apple', 20], ['banana', 40], ['orange', 60]]

In the example above, we:

* Initialized an empty list (of lists) and assigned it to fruits.
* Iterated over the keys of sample\_dict and appended to fruits a list whose:
  + First element is the key from sample\_dict.
  + Second element is the value corresponding to that key multiplied by ten.

Let's use this format to create a list of lists containing the hours during which posts were created and the average number of comments those posts received.

Instructions

1. Use the example above to calculate the average number of comments per post for posts created during each hour of the day.
2. The result should be a list of lists in which the first element is the hour and the second element is the average number of comments per post. Assign the result to a variable named avg\_by\_hour. Display the results.

In the last screen, we calculated the average number of comments for posts created during each hour of the day, and stored the results in a list of lists named avg\_by\_hour:

[

['09', 5.5777777777777775],

['13', 14.741176470588234],

['10', 13.440677966101696],

['14', 13.233644859813085],

['16', 16.796296296296298],

['23', 7.985294117647059],

['12', 9.41095890410959],

['17', 11.46],

['15', 38.5948275862069],

['21', 16.009174311926607],

['20', 21.525],

['02', 23.810344827586206],

['18', 13.20183486238532],

['03', 7.796296296296297],

['05', 10.08695652173913],

['19', 10.8],

['01', 11.383333333333333],

['22', 6.746478873239437],

['08', 10.25],

['04', 7.170212765957447],

['00', 8.127272727272727],

['06', 9.022727272727273],

['07', 7.852941176470588],

['11', 11.051724137931034]

]

Although we now have the results we need, this format makes it hard to identify the hours with the highest values. Let's finish by sorting the list of lists and printing the five highest values in a format that's easier to read.

Instructions

1. Create a list that equals avg\_by\_hour with swapped columns.
   * Create an empty list and assign it to swap\_avg\_by\_hour.
   * Iterate over the rows of avg\_by\_hour and append to swap\_avg\_by\_hour a list whose first element is the second element of the row, and whose second element is the first element of the row.
2. Print swap\_avg\_by\_hour.
3. Use the [sorted() function](https://docs.python.org/3/library/functions.html#sorted) to sort swap\_avg\_by\_hour in descending order. Since the first column of this list is the average number of comments, sorting the list will sort by the average number of comments.
   * Set the reverse argument to True, so that the highest value in the first column appears first in the list.
   * Assign the result to sorted\_swap.
4. Print the string "Top 5 Hours for Ask Posts Comments".
5. Loop through each average and each hour (in this order) in the first five lists of sorted\_swap.
6. Use the [str.format() method](https://docs.python.org/3/library/stdtypes.html" \l "str.format" \t "_blank) to print the hour and average in the following format: 15:00: 38.59 average comments per post.
   * To format the hours, use the datetime.strptime() constructor to return a datetime object and then use the strftime() method to specify the format of the time.
   * To format the average, you can use {:.2f} to indicate that just two decimal places should be used.
7. Which hours should you create a post during to have a higher chance of receiving comments? Refer back to the [documentation for the data set](https://www.kaggle.com/hacker-news/hacker-news-posts) to convert the times to the time zone you live in. Write a markdown cell explaining your findings.

That's it for the guided steps! Here's a quick summary of what we accomplished in this guided project:

* We set a goal for the project.
* We collected and sorted the data.
* We reformatted and cleaned the data to prepare it for analysis.
* We analyzed the data.

Curious to see what other students have done on this project? [Head over to our Community to check them out](https://community.dataquest.io/tags/c/social/share/49/356). While you are there, please remember to show some love and give your own feedback!

And of course, we welcome you to share your own project and show off your hard work. Head over to our Community to [share your finished Guided Project](https://community.dataquest.io/tags/c/social/share/49/356)!

Guided projects can be used to build a portfolio to showcase to potential employers, so we encourage you to keep working on this. Here are some next steps for you to consider:

* Determine if show or ask posts receive more points on average.
* Determine if posts created at a certain time are more likely to receive more points.
* Compare your results to the average number of comments and points other posts receive.
* Use Dataquest's [data science project style guide](https://www.dataquest.io/blog/data-science-project-style-guide/) to format your project.

You're welcome to keep working on the project here, but we recommend downloading it to your computer using the download icon above the notebook and working on it locally.

If you choose to work on the next steps independently, you'll inevitably not know how to perform certain tasks or hit errors that you won't know how to resolve. Don't get discouraged! This is part of the learning process. Although referring back to previous missions is a great way to refresh your memory on certain topics, there are also a couple tools you should get familiar with because you'll need to use them in a real world job setting.

The best thing to do if you hit an error you can't resolve or don't know how to perform a task is search for the answer on Google. When you search, make sure to include the word "python" — otherwise, you'll get results from other programming languages. For example, instead of searching for "how to find the first element in a list," search for "python how to find the first element in a list."

As you search, you'll see one site constantly appear at the top of the results — Stack Overflow. Stack Overflow is an online community where people ask and answer programming questions. In most situations, you'll find that someone has asked the same question as you or a similar question that can help you. The community is very active, so the answers are almost always accurate.

Congratulations, this is the end of the course!