Python Loops and Functions - Cumulative Lab

Introduction

You made it through another section — excellent work! This cumulative lab will return to the Amazon product review dataset and allow you to flex your new skills.

Objectives

You will be able to:

- Recall what you learned in the previous section
- Practice writing loops to pull multiple pieces of data from a dataset
- Practice writing functions for organization and avoiding repetition

Your Task: Dynamically Query Amazon Review Data

Once again, we are going to be working with data collected by Computer Science researchers at the University of California, San Diego. Their full paper citation is here:

> **Justifying recommendations using distantly-labeled reviews and fined-grained aspects** Jianmo Ni, Jiacheng Li, Julian McAuley Empirical Methods in Natural Language Processing (EMNLP), 2019 pdf

We are still using a cleaned-up, coffee-specific, sample version of their full dataset.

Photo by Philipp Cordts on Unsplash

In some cases, we will write the function signature for you, e.g.

```python
def review_sentiment(review):
    # Replace None with appropriate code
    None
```

Then you just need to fill in the relevant logic.

In other cases, you will need to write the function signature yourself, e.g.

```python
# Your code here
```

Requirements

1. Data Summary

While reusing some code from the previous cumulative lab, write code to loop over all of the records in the dataset to summarize its contents, specifically in terms of overall review sentiment and the years when the reviews were written.

2. Subset Sample

Provide a sample of records that meet particular criteria.

3. Individual Review Summary

Refactor the code from the previous cumulative lab so that it is contained in a function and prompts the user to select which review to summarize.

Data Summary

Once again, we've opened up the dataset and loaded it into a list of dictionaries called `reviews`.

```python
import json
with open("coffee_product_reviews.json") as f:
    reviews = json.load(f)
type(reviews)
```

```
list
```

Previously, we found the length of the collection, and looked into the data types of each record's keys and values

```python
num_reviews = len(reviews)
print("The coffee product review dataset contains {} reviews".format(num_reviews))

first_review = reviews[0]
first_review
```

```
The coffee product review dataset contains 86 reviews
```

```
{'rating': 5.0,
 'reviewer_name': 'Sns073194',
 'product_id': 'B00004RFRV',
 'review_title': 'Perfect cafsito every time',
 'review_time': '03 11, 2018',
 'images': ['https://images-na.ssl-images-
amazon.com/images/I/71d2cQEgJsL._SY88.jpg'],
 'styles': {'Size:': ' 6-Cup', 'Color:': ' Silver'}}
```

```
first_review.keys()
```

```
dict_keys(['rating', 'reviewer_name', 'product_id', 'review_title',
'review_time', 'images', 'styles'])
```

```
first_review.values()
```

```
dict_values([5.0, 'Sns073194', 'B00004RFRV', 'Perfect cafsito every time', '03
11, 2018', ['https://images-na.ssl-images-
amazon.com/images/I/71d2cQEgJsL._SY88.jpg'], {'Size:': ' 6-Cup', 'Color:': '
Silver'}])
```

This time, let's do something a bit more sophisticated. Specifically:

1. Count of positive, negative, and neutral reviews
2. List of years contained in the dataset

Count of Positive, Negative, and Neutral Reviews

Previously, we wrote something like this code to determine whether a specific review was positive, negative, or neutral:

```
selected_review = reviews[2]
selected_rating = selected_review["rating"]

if selected_rating >= 4:
    print("This is a positive review")
elif selected_rating <= 2:
    print("This is a negative review")
else:
    print("This is a neutral review")
```

```
This is a positive review
```

Now, rewrite that code as a function `review_sentiment`, which takes in a review dictionary as an argument, and returns the string `"positive"`, `"negative"`, or `"neutral"`

```
def review_sentiment(review):
    rating = review["rating"]

    if rating >= 4:
        return "positive"
    elif rating <= 2:
        return "negative"
    else:
        return "neutral"
```

```
review_sentiment(reviews[2])
```

```
'positive'
```

```
review_sentiment(reviews[4])
```

```
'negative'
```

```
review_sentiment(reviews[47])
```

```
'neutral'
```

Ok, this is already much cleaner than copying and pasting that `if` / `elif` / `else` sequence like we did before!

Now, write a function to loop over all of the reviews in the list, and count how many are positive, negative, and neutral.

The function should be called `get_sentiment_counts`, take one argument (the list of reviews), and return a dictionary containing the counts. A counter dictionary has been initialized for you with `"positive"`, `"negative"`, and `"neutral"` as the keys and values starting at 0.

```
def get_sentiment_counts(review_list):

    sentiment_counts = {
        "positive": 0,
        "negative": 0,
        "neutral": 0
    }

    for review in review_list:
        sentiment = review_sentiment(review)
        sentiment_counts[sentiment] += 1

    return sentiment_counts

get_sentiment_counts(reviews)
```

```
{'positive': 67, 'negative': 15, 'neutral': 4}
```

This spread of sentiments seems reasonable. There is a well-known skew towards positive reviews in general, similar to "grade inflation", and people with neutral opinions are less likely to write reviews in the first place.

List of Years Contained in the Dataset

Previously, we wrote something like this code to extract the year of a review from the review dictionary:

```
selected_review = reviews[2]
selected_review_time = selected_review["review_time"]
selected_review_year = int(selected_review_time[-4:])
selected_review_year
```

```
2017
```

Now, rewrite that code as a function `review_year`, which takes in a review dictionary as an argument, and returns the year as an integer:

```python
def review_year(review):
    review_time = review["review_time"]
    return int(review_time[-4:])
```

```python
review_year(reviews[2])
```

```
2017
```

```python
review_year(reviews[4])
```

```
2017
```

```python
review_year(reviews[47])
```

```
2015
```

Now, write a function called `get_years` to loop over all of the reviews in the review list and create a list of the years you find. Each year should only appear once, in ascending order. The function should accept one argument (`review_list`) and should return a list of integers representing the years.

Hints:

- Remember that you can use the `set()` function to keep only the unique elements in a list. Just make sure you use `list()` afterwards to convert it back to a list data type. This is not the only solution, however!
- There is a list method named `.sort()` (look it up in the python list documentation here) that will automatically order the years; you don't need to write sorting logic "by hand"

```python
def get_years(review_list):
    # Option without set()
    years = []
    for review in review_list:
        year = review_year(review)
        if year not in years:
            years.append(year)
    years.sort()
    return years

print(get_years(reviews))
print(type(get_years(reviews)))

def get_years(review_list):
    # Option with set()
    years = []
    for review in review_list:
        years.append(review_year(review))
    years = list(set(years))
    years.sort()
    return years

print(get_years(reviews))
print(type(get_years(reviews)))
```

```
[2007, 2008, 2009, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018]
<class 'list'>
[2007, 2008, 2009, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018]
<class 'list'>
```

Now we know that we have data spanning 2007-2018, with no data from 2010. In some contexts that absence might be worth investigating — is it a random artifact of our sample, or are we missing 2010 data for a reason that matters? For now we'll just keep moving on to the next section, now that we have a clearer sense of the kinds of reviews in our dataset and the years they were written.

Subset Sampling

Once you have an overall sense of a dataset, it's a good idea to ask *what are some examples of records in each category?* For example, what are some examples of a negative review?

Although 86 records are few enough that you could technically read through all of them and just mentally note what we see, let's use an approach that will scale better to larger datasets with more categories: **filtering** to a subset of records then **sampling** to achieve a digestible amount of information.

Filtering

Here we are going to make use of another built-in Python function: `filter()` (docs here). To use this function, we first need to write a helper function that returns `True` or `False` based on the value passed in.

So, create a function `is_negative` that takes in a review dictionary as an argument and returns `True` if the review is negative, `False` otherwise:

```
def is_negative(review):
    sentiment = review_sentiment(review)
    return sentiment == "negative"

print(is_negative(reviews[2]))  # False (postive review)
print(is_negative(reviews[4]))  # True
print(is_negative(reviews[47])) # False (neutral review)
```

```
False
True
False
```

Now we can use the `filter()` function to create a list of negative reviews:

```
list(filter(is_negative, reviews))
```

```
[{'rating': 1.0,
  'reviewer_name': 'EJ',
  'product_id': 'B00004RFRV',
  'review_title': 'Rusted spots everywhere fresh out the box...nasty',
  'review_time': '06 4, 2017',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/71Dbr6X0bYL._SY88.jpg'],
  'styles': {'Size:': ' 9-Cup', 'Color:': ' Silver'}},
 {'rating': 1.0,
  'reviewer_name': 'mathman54',
  'product_id': 'B00004RFRV',
  'review_title': "The bottom looks like it has rusted and I don't know how to
...",
  'review_time': '02 15, 2016',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/71qt4Hnra8L._SY88.jpg',
   'https://images-na.ssl-images-amazon.com/images/I/71Wkg8MesdL._SY88.jpg'],
  'styles': {'Size:': ' 12-Cup', 'Color:': ' Silver'}},
 {'rating': 1.0,
  'reviewer_name': 'Maria Fernandez',
  'product_id': 'B00004RFRV',
  'review_title': 'Is it a Bialetti? Came dirty with coffee grounds.',
  'review_time': '03 22, 2015',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/71aPg3ZuCzL._SY88.jpg',
   'https://images-na.ssl-images-amazon.com/images/I/71-2keZacOL._SY88.jpg',
   'https://images-na.ssl-images-amazon.com/images/I/71Z4C4G-0dL._SY88.jpg'],
  'styles': {'Size:': ' 6-Cup', 'Color:': ' Silver'}},
 {'rating': 1.0,
  'reviewer_name': 'Toni Bautista',
  'product_id': 'B00005IBX9',
  'review_title': 'Great but NOT PERFECT -missing Side water panel',
  'review_time': '02 24, 2017',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/71X5NEjZG5L._SY88.jpg'],
  'styles': {'Color:': ' Brushed Chrome'}},
 {'rating': 1.0,
  'reviewer_name': 'SAinVA',
  'product_id': 'B00005LM0T',
  'review_title': 'Too much waste',
  'review_time': '09 7, 2017',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/71shZggLRHL._SY88.jpg'],
  'styles': {'Size:': ' 34 oz.', 'Package Type:': ' Standard Packaging'}},
 {'rating': 1.0,
  'reviewer_name': 'Silicon Valley',
  'product_id': 'B00005LM0T',
  'review_title': 'FRAME RUSTS WITHIN A FEW WEEKS',
  'review_time': '03 10, 2016',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/71FREAeBLIL._SY88.jpg',
   'https://images-na.ssl-images-amazon.com/images/I/71qkhDjseuL._SY88.jpg',
   'https://images-na.ssl-images-amazon.com/images/I/7143mVMMoCL._SY88.jpg'],
  'styles': {'Size:': ' 34 oz.', 'Package Type:': ' Standard Packaging'}},
 {'rating': 1.0,
  'reviewer_name': 'Andrew Furlong',
  'product_id': 'B00005MF9C',
  'review_title': 'but instead ran along the underside of the part of the maker
that holds the grounds and finally dropping at an angle at which a',
  'review_time': '02 22, 2017',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/71kh1XatynL._SY88.jpg'],
  'styles': {'Color:': ' Black/Stainless Steel'}},
 {'rating': 2.0,
  'reviewer_name': 'fred o.',
  'product_id': 'B00005MF9C',
  'review_title': 'Good purchase',
  'review_time': '09 4, 2016',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/71tRehtvN+L._SY88.jpg',
   'https://images-na.ssl-images-amazon.com/images/I/719K20U+MaL._SY88.jpg',
   'https://images-na.ssl-images-amazon.com/images/I/71jd6yc1GdL._SY88.jpg'],
  'styles': {'Color:': ' Black/White'}},
 {'rating': 1.0,
  'reviewer_name': 'fifrox',
  'product_id': 'B00005MF9C',
  'review_title': 'Works great for a week then fails!',
  'review_time': '03 10, 2016',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/81QVkUT4hdL._SY88.jpg'],
  'styles': {'Color:': ' Black/Stainless Steel'}},
 {'rating': 1.0,
  'reviewer_name': 'cas',
  'product_id': 'B00005NCWQ',
  'review_title': 'Garbage!!!',
  'review_time': '03 26, 2018',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/71Bxydi3DQL._SY88.jpg'],
  'styles': {'Size:': ' 8-Cup'}},
 {'rating': 1.0,
  'reviewer_name': 'GoClick',
  'product_id': 'B00005OTXM',
  'review_title': 'Returned - produced revolting burnt plastic flavored coffee,
and seemed flawed.',
  'review_time': '04 16, 2015',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/71o+7Zru+iL._SY88.jpg',
   'https://images-na.ssl-images-amazon.com/images/I/81sbfP4tc8L._SY88.jpg',
   'https://images-na.ssl-images-amazon.com/images/I/71vhOebyGaL._SY88.jpg'],
  'styles': {'Style Name:': ' COFFEE MAKER ONLY'}},
 {'rating': 2.0,
  'reviewer_name': 'Picks n Pans',
  'product_id': 'B00006F2LW',
  'review_title': 'Cracked lid out of the box',
  'review_time': '02 27, 2008',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/317fvUiuNDL._SY88.jpg'],
  'styles': {'Color:': ' Black'}},
 {'rating': 2.0,
  'reviewer_name': 'LMM',
  'product_id': 'B00008ELEA',
  'review_title': 'smelled like the cord was burning',
  'review_time': '06 6, 2016',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/51xXENQhIGL._SY88.jpg'],
  'styles': {'Size:': ' 4-Cup'}},
 {'rating': 1.0,
  'reviewer_name': 'Marcia S.',
  'product_id': 'B0000A1ZMS',
  'review_title': 'Look what happened to this once great Cuisinart coffeemaker.',
  'review_time': '11 12, 2016',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/51B8eue4nHL._SY88.jpg',
   'https://images-na.ssl-images-amazon.com/images/I/519NziREFIL._SY88.jpg',
   'https://images-na.ssl-images-amazon.com/images/I/51v+A1zWqTL._SY88.jpg'],
  'styles': {'Color:': ' Black', 'Style Name:': ' Coffeemaker'}},
 {'rating': 2.0,
  'reviewer_name': 'Jimmie',
  'product_id': 'B0000A1ZMS',
  'review_title': 'Leaks and more',
  'review_time': '01 13, 2011',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/51U4K+PzETL._SY88.jpg'],
  'styles': {'Color:': ' Black', 'Style Name:': ' Coffeemaker'}}]
```

Write a function called `get_negative_reviews` that returns a list of these reviews. It should take the list of all reviews as an argument.

(This can be a one-line function.)

```python
def get_negative_reviews(review_list):
    return list(filter(is_negative, review_list))

len(get_negative_reviews(reviews)) # 15
```

```
15
```

Sampling

Again, since we have a relatively small dataset, we could just look at all 15 reviews. But let's take a more scalable approach instead, and take a random sample of negative reviews.

Recall the `random` module, which must be imported:

```python
import random
```

We'll use the `random.sample()` function, which takes in a collection and a number, and returns that number of elements from the collection.

So, for example, if we want 3 negative reviews:

```
# We did not set a random seed, so your results may differ
random.sample(get_negative_reviews(reviews), 3)
```

```
[{'rating': 1.0,
  'reviewer_name': 'Maria Fernandez',
  'product_id': 'B00004RFRV',
  'review_title': 'Is it a Bialetti? Came dirty with coffee grounds.',
  'review_time': '03 22, 2015',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/71aPg3ZuCzL._SY88.jpg',
   'https://images-na.ssl-images-amazon.com/images/I/71-2keZacOL._SY88.jpg',
   'https://images-na.ssl-images-amazon.com/images/I/71Z4C4G-0dL._SY88.jpg'],
  'styles': {'Size:': ' 6-Cup', 'Color:': ' Silver'}},
 {'rating': 1.0,
  'reviewer_name': 'Marcia S.',
  'product_id': 'B0000A1ZMS',
  'review_title': 'Look what happened to this once great Cuisinart coffeemaker.',
  'review_time': '11 12, 2016',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/51B8eue4nHL._SY88.jpg',
   'https://images-na.ssl-images-amazon.com/images/I/519NziREFIL._SY88.jpg',
   'https://images-na.ssl-images-amazon.com/images/I/51v+A1zWqTL._SY88.jpg'],
  'styles': {'Color:': ' Black', 'Style Name:': ' Coffeemaker'}},
 {'rating': 1.0,
  'reviewer_name': 'SAinVA',
  'product_id': 'B00005LM0T',
  'review_title': 'Too much waste',
  'review_time': '09 7, 2017',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/71shZggLRHL._SY88.jpg'],
  'styles': {'Size:': ' 34 oz.', 'Package Type:': ' Standard Packaging'}}]
```

Now, put that code into a function `get_negative_review_sample`. This function should take a list of reviews and the number of samples to select, and should return a sample of negative reviews.

(You can assume that `num_samples` is a valid number. The number of samples must be less than or equal to the number of elements in the collection.)

```
def get_negative_review_sample(review_list, num_samples):
    return random.sample(get_negative_reviews(review_list), num_samples)

get_negative_review_sample(reviews, 4)
```

```
[{'rating': 1.0,
  'reviewer_name': 'Toni Bautista',
  'product_id': 'B00005IBX9',
  'review_title': 'Great but NOT PERFECT -missing Side water panel',
  'review_time': '02 24, 2017',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/71X5NEjZG5L._SY88.jpg'],
  'styles': {'Color:': ' Brushed Chrome'}},
 {'rating': 1.0,
  'reviewer_name': 'SAinVA',
  'product_id': 'B00005LM0T',
  'review_title': 'Too much waste',
  'review_time': '09 7, 2017',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/71shZggLRHL._SY88.jpg'],
  'styles': {'Size:': ' 34 oz.', 'Package Type:': ' Standard Packaging'}},
 {'rating': 1.0,
  'reviewer_name': 'fifrox',
  'product_id': 'B00005MF9C',
  'review_title': 'Works great for a week then fails!',
  'review_time': '03 10, 2016',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/81QVkUT4hdL._SY88.jpg'],
  'styles': {'Color:': ' Black/Stainless Steel'}},
 {'rating': 1.0,
  'reviewer_name': 'EJ',
  'product_id': 'B00004RFRV',
  'review_title': 'Rusted spots everywhere fresh out the box...nasty',
  'review_time': '06 4, 2017',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/71Dbr6X0bYL._SY88.jpg'],
  'styles': {'Size:': ' 9-Cup', 'Color:': ' Silver'}}]
```

Repeat the same process for positive reviews. That means we need:

1. A helper function `is_positive` (this can't just be `not is_negative` since neutral reviews are neither)
2. A function `get_positive_reviews` which returns a list of all positive reviews
3. A function `get_positive_review_sample` which returns a sample of positive reviews with the specified length

```python
def is_positive(review):
    sentiment = review_sentiment(review)
    return sentiment == "positive"

def get_positive_reviews(review_list):
    return list(filter(is_positive, review_list))

def get_positive_review_sample(review_list, num_samples):
    return random.sample(get_positive_reviews(review_list), num_samples)

get_positive_review_sample(reviews, 4)
```

```
[{'rating': 5.0,
  'reviewer_name': 'Feles (muy Mala)',
  'product_id': 'B00004RFRV',
  'review_title': 'Awesome portion control for one person!',
  'review_time': '08 5, 2017',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/71BcwbkGyfL._SY88.jpg'],
  'styles': {'Size:': ' 6-Cup', 'Color:': ' Purple'}},
 {'rating': 5.0,
  'reviewer_name': 'Cherie E.',
  'product_id': 'B000063SRL',
  'review_title': 'More than just a cutting board!',
  'review_time': '08 10, 2016',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/61oleNoBZ7L._SY88.jpg',
   'https://images-na.ssl-images-amazon.com/images/I/71FumLGotfL._SY88.jpg',
   'https://images-na.ssl-images-amazon.com/images/I/71MHMHDel+L._SY88.jpg',
   'https://images-na.ssl-images-amazon.com/images/I/61gZsMGlRlL._SY88.jpg',
   'https://images-na.ssl-images-amazon.com/images/I/61QVdUMKB2L._SY88.jpg'],
  'styles': {'Size:': ' 14 by 17 inches', 'Color:': ' Granite Color'}},
 {'rating': 5.0,
  'reviewer_name': 'A. Kronberg',
  'product_id': 'B00005OTXY',
  'review_title': 'this pot did a great job of boiling water in 20 minutes',
  'review_time': '09 4, 2017',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/71jzwfSoRiL._SY88.jpg'],
  'styles': {'Size:': ' One Size', 'Color:': ' Black'}},
 {'rating': 5.0,
  'reviewer_name': 'A. Sullivan',
  'product_id': 'B000063SSI',
  'review_title': 'Hand wash only',
  'review_time': '06 10, 2015',
  'images': ['https://images-na.ssl-images-
amazon.com/images/I/71j9qBm514L._SY88.jpg'],
  'styles': {'Package Quantity:': ' 1'}}]
```

Individual Review Summary

In addition to summarizing the dataset overall and sampling based on criteria, we want the user to be able to query any given record in order to view a summary. Before, we created a variable called `review_index` that the user could modify. Now, let's write some reusable code that doesn't require the user to write any Python at all!

Recall that before, our final code looked something like this:

```
review_index = 2

# Extract review from list of reviews
selected_review = reviews[review_index]

# Extract title
selected_review_title = selected_review["review_title"]

# Extract rating and format as positive, negative, or neutral
selected_rating = selected_review["rating"]
if selected_rating >= 4:
    selected_sentiment = "positive"
elif selected_rating <= 2:
    selected_sentiment = "negative"
else:
    selected_sentiment = "neutral"

# Extract author
selected_author = selected_review["reviewer_name"]

# Extract year (doesn't need to be int for this use case)
selected_year = selected_review["review_time"][-4:]

print(f'"{selected_review_title}": This was a {selected_sentiment} review written by
```

```
"Bialetti is the Best!": This was a positive review written by Karen in 2017.
```

Rewrite that code as a function called `get_review_summary`, which takes a review dictionary as an argument, and returns a string that resembles the previous summary string, e.g.

```
"Bialetti is the Best!": This was a positive review written by Karen in 2017.
```

Hint: look back at the functions you have previously written to see which ones might be useful to call within this function!

```
def get_review_summary(review):

    title = review["review_title"]
    sentiment = review_sentiment(review)
    author = review["reviewer_name"]
    year = review_year(review)

    return f'"{title}": This was a {sentiment} review written by {author} in {year}.

print(get_review_summary(reviews[2]))
```

```
"Bialetti is the Best!": This was a positive review written by Karen in 2017.
```

Now, instead of copying and pasting that every time, we can just call it repeatedly!

Write a function that prompts the user to enter a review index, then prints the relevant review summary. The function should be called `review_summary_prompt` , it should take a list of reviews as an argument, and should print information but not return anything.

Display the message `"Please enter a review index: "` when prompting for input. You can assume that the user will enter a valid index between 0 and 85.

Hints:

- Use the built-in `input()` function (check the documentation here to see how to use it!)
- Remember that this function always returns a string, so you will have to convert the user-supplied index into an integer, otherwise you'll get the error `TypeError: list indices must be integers or slices, not str`
- If you're wondering about the type of a given variable, you can use the built-in `type()` function

```
def review_summary_prompt(list_of_reviews):
    user_input = input("Please enter a review index: ")
    index = int(user_input)
    review = list_of_reviews[index]
    print(get_review_summary(review))
```

Run this cell, and try entering 2, 4, 52 (examples of positive, negative, neutral reviews)

You can also try any index you want, between 0 and 85!

```python
review_summary_prompt(reviews) # entered 2
```

```
Please enter a review index: 2
"Bialetti is the Best!": This was a positive review written by Karen in 2017.
```

```python
review_summary_prompt(reviews) # entered 4
```

```
Please enter a review index: 4
"Rusted spots everywhere fresh out the box...nasty": This was a negative review
written by EJ in 2017.
```

```python
review_summary_prompt(reviews) # entered 52
```

```
Please enter a review index: 52
"Simple, 4 cup coffee maker, looks nice, difficult basket hinge": This was a
neutral review written by Donald E. Fulton in 2012.
```

Putting It All Together

In this section, we are just calling several of the previously-created functions to double-check that they are working as expected. You do not need to write any more code, although if you notice something wrong with one of your functions you can go back and fix it! Just make sure that you re-run the cell declaring the function if you want the behavior of calling the function to change.

Data Summary

```python
print(f"The coffee product review dataset contains {len(reviews)} reviews")
print()
print("Review sentiment:")
for key, value in get_sentiment_counts(reviews).items():
    print(f"{value} {key} reviews")
print()
print("Review years:")
print(get_years(reviews))
```

```
The coffee product review dataset contains 86 reviews

Review sentiment:
67 positive reviews
15 negative reviews
4 neutral reviews

Review years:
[2007, 2008, 2009, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018]
```

Subset Samples

```python
print("Examples of positive reviews:")
positive_samples = get_positive_review_sample(reviews, 5)
for review in positive_samples:
    print(get_review_summary(review))
print()
print("Examples of negative reviews:")
negative_samples = get_negative_review_sample(reviews, 5)
for review in negative_samples:
    print(get_review_summary(review))
```

```
Examples of positive reviews:
"Best piece of furniture my daughter has!": This was a positive review written by
Melissa Coy in 2017.
"One great French Press coffee maker": This was a positive review written by
Michael Z (QA Engineer) in 2016.
"Awesome portion control for one person!": This was a positive review written by
Feles (muy Mala) in 2017.
"Love my Moka pots!": This was a positive review written by B. Laska in 2015.
"Allow the auroma of perculated coffee to fill your home......": This was a
positive review written by Sierra&#039;s Mom in 2015.

Examples of negative reviews:
"Look what happened to this once great Cuisinart coffeemaker.": This was a
negative review written by Marcia S. in 2016.
"The bottom looks like it has rusted and I don't know how to ...": This was a
negative review written by mathman54 in 2016.
"Leaks and more": This was a negative review written by Jimmie in 2011.
"Great but NOT PERFECT -missing Side water panel": This was a negative review
written by Toni Bautista in 2017.
"Works great for a week then fails!": This was a negative review written by
fifrox in 2016.
```

Summary Prompt

```
review_summary_prompt(reviews)
```

```
Please enter a review index: 7
"The bottom looks like it has rusted and I don't know how to ...": This was a
negative review written by mathman54 in 2016.
```

Conclusion

Congratulations, you made it to the end of another cumulative lab! In this lab you practiced refactoring previously-written code to use functions, and using loops to avoid repetition and perform analyses of the whole dataset as well as certain subsets.