learn-co-curriculum / **dsc-functions-with-arguments-lab**

No description, website, or topics provided.

5 commits **2** branches **0** releases **3** contributors View license

Branch: solution New pull request Create new file Upload files Find File Clone or download

This branch is 4 commits ahead, 3 commits behind master. Pull request Compare

mathymitchell updating median function solution to be accurate when lenght of list ... … Latest commit `8ca4e1f` on Apr 19

pytests	initial commit	4 months ago
.gitignore	initial commit	4 months ago
.learn	initial commit	4 months ago
CONTRIBUTING.md	initial commit	4 months ago
LICENSE.md	initial commit	4 months ago
README.md	updating median function solution to be accurate when lenght of list ...	2 months ago
index.ipynb	updating median function solution to be accurate when lenght of list ...	2 months ago

README.md

Functions With Arguments - Lab

Introduction

In this lesson, we have decided to visit one of our travel destinations! This time we have chosen to visit Albuquerque, but we aren't very familiar with this city and are quite hungry after our long flight. We will be working with information we pulled from the Yelp database to help us find a restaurant where we can satisfy our hunger. While Yelp is great for learning about what to do in Albuquerque, it gives us back a lot of information. We'll use what we know about functions and dictionaries to format and read our data more easily.

Objectives

You will be able to:

- Create and use custom functions with arguments
- Understand how function arguments can make functions more flexible and reusable

Exploring Two Restaurants in Albuquerque

Let's take a quick look at the information Yelp provides for a single restaurant:

```
fork_fig = {'categories': [{'alias': 'burgers', 'title': 'Burgers'},
  {'alias': 'sandwiches', 'title': 'Sandwiches'},
  {'alias': 'salad', 'title': 'Salad'}],
 'coordinates': {'latitude': 35.10871, 'longitude': -106.56739},
 'display_phone': '(505) 881-5293',
 'distance': 3571.724649307866,
 'id': 'fork-and-fig-albuquerque',
 'image_url': 'https://s3-media1.fl.yelpcdn.com/bphoto/_-DpXKfS3jv6DyA47g6Fxg/o.jpg',
 'is_closed': False,
 'location': {'address1': '6904 Menaul Blvd NE',
  'address2': 'Ste C',
  'address3': '',
  'city': 'Albuquerque',
  'country': 'US',
  'display_address': ['6904 Menaul Blvd NE', 'Ste C', 'Albuquerque, NM 87110'],
  'state': 'NM',
  'zip_code': '87110'},
 'name': 'Fork & Fig',
 'phone': '+15058815293',
 'price': '$$',
 'rating': 4.5,
 'review_count': 604,
 'transactions': [],
 'url': 'https://www.yelp.com/biz/fork-and-fig-albuquerque?adjust_creative=SYc8R4Gowqru5h4SBKZXsQ&utm_campaign=yelp_
```

Above is the information provided about `Fork & Fig`, but all restaurants are provided with this information. For example, here is the information provided by Yelp for another restaurant, `Frontier Restaurant`.

```
frontier_restaurant = {'categories': [{'alias': 'mexican', 'title': 'Mexican'},
  {'alias': 'diners', 'title': 'Diners'},
  {'alias': 'tradamerican', 'title': 'American (Traditional)'}],
 'coordinates': {'latitude': 35.0808088832532, 'longitude': -106.619402244687},
 'display_phone': '(505) 266-0550',
 'distance': 4033.6583235266075,
 'id': 'frontier-restaurant-albuquerque-2',
 'image_url': 'https://s3-media4.fl.yelpcdn.com/bphoto/M9L2z6-G0NobuDJ6YTh6VA/o.jpg',
 'is_closed': False,
 'location': {'address1': '2400 Central Ave SE',
  'address2': '',
  'address3': '',
  'city': 'Albuquerque',
  'country': 'US',
  'display_address': ['2400 Central Ave SE', 'Albuquerque, NM 87106'],
  'state': 'NM',
  'zip_code': '87106'},
 'name': 'Frontier Restaurant',
 'phone': '+15052660550',
 'price': '$',
 'rating': 4.0,
 'review_count': 1369,
 'transactions': [],
 'url': 'https://www.yelp.com/biz/frontier-restaurant-albuquerque-2?adjust_creative=SYc8R4Gowqru5h4SBKZXsQ&utm_campa
```

As we already know, one way to quickly view the attributes of a dictionary is to look at the keys of the dictionary.

```
fork_fig.keys()
```

```
dict_keys(['categories', 'coordinates', 'display_phone', 'distance', 'id', 'image_url', 'is_closed', 'location',
'name', 'phone', 'price', 'rating', 'review_count', 'transactions', 'url'])
```

```
frontier_restaurant.keys()
```

```
dict_keys(['categories', 'coordinates', 'display_phone', 'distance', 'id', 'image_url', 'is_closed', 'location',
'name', 'phone', 'price', 'rating', 'review_count', 'transactions', 'url'])
```

```
fork_fig.keys() == frontier_restaurant.keys()
```

```
True
```

As we can see from our above comparison, Yelp provides us with the same information for both restaurants.

Writing our functions

Ok, now let's write our functions. Write a function called `restaurant_name` that, provided a dictionary representing a restaurant like you saw above, returns that restaurant's name.

```
def restaurant_name(restaurant):
    return restaurant['name']
```

```
restaurant_name(frontier_restaurant) # 'Frontier Restaurant'
```

```
'Frontier Restaurant'
```

```
restaurant_name(fork_fig) # 'Fork & Fig'
```

```
'Fork & Fig'
```

Now write a function called `restaurant_rating` that returns the rating of the provided restaurant.

```
def restaurant_rating(restaurant):
    return restaurant['rating']
```

```
restaurant_rating(frontier_restaurant) # 4.0
```

```
4.0
```

```
restaurant_rating(fork_fig) # 4.5
```

```
4.5
```

Comparing restaurants

Now let's write a function called `is_better` that returns `True` if a restaurant has a higher rating than an alternative restaurant. The first argument should be called `restaurant` and the second argument should be called `alternative`. The function returns `False` if the two ratings are equal.

```
def is_better(restaurant, alternative):
    if restaurant['rating'] > alternative['rating']:
        return True
    return False
```

```
is_better(frontier_restaurant, fork_fig) # False
```

```
False
```

```
is_better(fork_fig, frontier_restaurant) # True
```

```
True
```

```
is_better(fork_fig, fork_fig) # False
```

```
False
```

Now let's write a function called `is_cheaper` that returns `True` if a restaurant has a lower price, that is the restaurant has fewer `'$'` signs, than an alternative restaurant. The first argument should be called `restaurant` and the second argument should be called `alternative` . The function returns `False` if the two prices are equal.

> **Hint:** *Strings in Python respond to then `len` function.*

```
def is_cheaper(restaurant, alternative):
    if len(restaurant['price']) < len(alternative['price']):
        return True
    return False
```

```
is_cheaper(fork_fig, frontier_restaurant) # False
```

```
False
```

```
is_cheaper(frontier_restaurant, fork_fig) # True
```

```
True
```

```
is_cheaper(fork_fig, fork_fig) # False
```

```
False
```

Now write a function called `high_rating` that takes a `restaurant` as a first argument and a rating (in the form of a number) as the second argument and returns `True` if the given restaurant's rating is greater than or equal to the provided rating and returns `False` otherwise.

```
def high_rating(restaurant, rating):
    if restaurant['rating'] >= rating:
        return True
    return False
```

```
high_rating(fork_fig, 4) # True
```

```
True
```

```
high_rating(fork_fig, 5) # False
```

```
False
```

```
high_rating(frontier_restaurant, 4) # True
```

```
True
```

Awesome! We have built out some pretty cool functions so far. Let's now think about a case where we have more than just two data points to operate on. We have added some more "restaurants" below and are going to add them to our list of restaurants. Don't worry that they have a slightly different amount of data.

We are going to need a function `mean_review_count` to give us an idea what the ideal range for review_count is. This function should take in a list of restaurant dictionaries and return the mean of the review counts for the collection of restaurant dictionaries.

```
dennys = {'categories': [{'alias': 'breakfast', 'title': 'Breakfast'},
  {'alias': 'diners', 'title': 'Diners'},
  {'alias': 'tradamerican', 'title': 'American (Traditional)'}],
 'is_closed': False,
 'name': "Denny's",
 'price': '$',
 'rating': 3.0,
 'review_count': 1200}

ihop = {'categories': [{'alias': 'breakfast', 'title': 'Breakfast'},
  {'alias': 'diners', 'title': 'Diners'},
  {'alias': 'tradamerican', 'title': 'American (Traditional)'}],
 'is_closed': False,
 'name': "IHOP: International House of Pancakes",
 'price': '$',
 'rating': 3.45,
 'review_count': 1588}

mcdonalds = {'categories': [{'alias': 'breakfast', 'title': 'Breakfast'},
  {'alias': 'burgers', 'title': 'Burgers'},
  {'alias': 'fast food', 'title': 'Good Food Fast'}],
 'is_closed': False,
 'name': "McDonalds",
 'price': '$',
 'rating': 3.45,
 'review_count': 2455}

pearl_street_oyster_bar = {'categories': [{'alias': 'seafood', 'title': 'Seafood'},
  {'alias': 'gourmet', 'title': 'Gourmet'},
  {'alias': 'Shellfish', 'title': 'Shellfish'}],
 'is_closed': False,
 'name': "Pear Street Oyster Bar",
 'price': '$$$',
 'rating': 4.75,
 'review_count': 350}
```

```
restaurant_list = [pearl_street_oyster_bar, mcdonalds, ihop, dennys, fork_fig, frontier_restaurant]
```

```
# code goes here
def mean_review_count(list_of_restaurants):
    reviews = []
    for rest in list_of_restaurants:
        reviews.append(rest['review_count'])
    mean = sum(reviews)/len(reviews)
    return mean
```

```
mean_review_count(restaurant_list)
```

```
1261.0
```

Next, let's maybe look at the median review, since, we want to make sure that there isn't any outliers in our data. Ideally the median and mean will be somewhat close, but obviously this would me more accurate given a larger sample size. Define a function `median_review_count` that again takes in a list of restaurant dictionaries and returns the median count of reviews. Remember that if a data set is even, to get the median we average the two middle data points.

```
# code goes here
def median_review_count(list_of_restaurants):
    length = len(list_of_restaurants)
    n_reviews = sorted([r['review_count'] for r in list_of_restaurants])
    if ((length % 2) == 0):
        half = int(length/2)
        median = (n_reviews[half] + n_reviews[half-1])/2
        return median
    else:
        half = int((length - 1)/2)
        median = n_reviews[half]
        return median
```

```
median_review_count(restaurant_list)
```

```
1284.5
```

Summary

Great! In this lab we saw how to pass both single and multiple arguments to functions.