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

View license

0 stars 122 forks

Star Watch

Code Issues Pull requests Actions Projects Security Insights

solution

This branch is 14 commits ahead, 13 commits behind master. Contribute

hoffm386 fix objectives spacing ... on Aug 1 15

View code

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

README.md

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:

- Declare and use a function with arguments

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=SYc8R4Gow
```

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=
```

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 Functions to Extract Information

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'
```

```
restaurant_name(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
```

```
restaurant_rating(fork_fig)
```

```
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.

This function should *call* (AKA *invoke*) your existing `restaurant_rating` function.

```
def is_better(restaurant, alternative):
    return restaurant_rating(restaurant) > restaurant_rating(alternative)
```

```
is_better(frontier_restaurant, fork_fig)
```

```
False
```

```
is_better(fork_fig, frontier_restaurant)
```

```
True
```

```
is_better(fork_fig, fork_fig)
```

```
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 restaurant_price_len(restaurant):
    return len(restaurant['price'])

def is_cheaper(restaurant, alternative):
    return restaurant_price_len(restaurant) < restaurant_price_len(alternative)
```

```
is_cheaper(fork_fig, frontier_restaurant)
```

```
False
```

```python
is_cheaper(frontier_restaurant, fork_fig)
```

```
True
```

```python
is_cheaper(fork_fig, fork_fig)
```

```
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.

```python
def high_rating(restaurant, rating):
    return restaurant_rating(restaurant) >= rating
```

```python
high_rating(fork_fig, 4)
```

```
True
```

```python
high_rating(fork_fig, 5)
```

```
False
```

```python
high_rating(frontier_restaurant, 4)
```

```
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 restaurant dictionaries 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 typical value 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, front
```

```
def mean_review_count(list_of_restaurants):
    review_counts = []
    for restaurant in list_of_restaurants:
        review_counts.append(restaurant['review_count'])
    mean = sum(review_counts)/len(review_counts)
    return mean
```

```
mean_review_count(restaurant_list)
```

```
1261.0
```

Now we have an idea of how many reviews a typical restaurant has, but none of these restaurants have exactly that number of reviews.

Which restaurants have a `review_count` within 150 of the average? ("within 150" meaning exactly average, <= 150 fewer reviews than the average, or <= 150 more reviews than the average)

Return a list of the restaurant names.

```
def near_average_review_count(list_of_restaurants):
    mean = mean_review_count(list_of_restaurants)

    results = []
    for restaurant in list_of_restaurants:
        review_count_difference = abs(restaurant['review_count'] - mean)
        if review_count_difference <= 150:
            results.append(restaurant_name(restaurant))

    return results
```

```
near_average_review_count(restaurant_list)
```

```
["Denny's", 'Frontier Restaurant']
```

Summary

Great! In this lab we saw how to pass both single and multiple arguments to functions. Function arguments can make functions more flexible and reusable!

Releases

No releases published

Packages

No packages published

Contributors 6

Languages

● **Jupyter Notebook** 85.2% ● **Python** 14.8%