Practical 3: Foundations (Part 2)

Getting to grips with Dictionaries, LOLs and DOLs

Table of contents

In this notebook we are exploring basic (in the sense of fundamental) data structures so that you understand both how to manage more complex types of data and are prepared for what we will encounter when we start using `pandas` to perform data analysis. To achieve that, you will need to be 'fluent' in nested lists and dictionaries; we will focus primarily on lists-of-lists and dictionaries-of-lists, but note that file formats like JSON can be understood as dictionaries-of-dictionaries-of-lists-of-... so this is just a *taster* of real-world data structures.

Tip

You should download this notebook and then save it to your own copy of the repository. Follow the process used *last* week (i.e. `git add ...`, `git commit -m "..."`, `git push`) right away and then do this again at the end of the class and you'll have a record of everything you did.

1 From Lists to Data (Little Steps)

We're going to start off using lists and dictionaries that *we* define right at the start of the 'program', but the *real* value of these data structures comes when we build a list or dictionary *from* data such as a file or a web page... and that's what we're going to do below!

First, here's a reminder of some useful methods (*i.e.* functions) that apply to lists which we covered in the lecture and practical in Week 2:

Method	Action
`list.count(x)`	Return the number of times x appears in the list
`list.insert(i, x)`	Insert value `x` at a given position `i`
`list.pop([i])`	Remove and return the value at position `i` (`i` is optional)
`list.remove(x)`	Remove the first element from the list whose value is `x`
`list.reverse()`	Reverse the elements of the list in place

Method	Action
`list.sort()`	Sort the items of the list in place
`list.index(x)`	Find the first occurence of `x` in the list
`list[x:y]`	Slice the list from index `x` to `y-1`

This should all be revision… because it's how we finished things up *last week*. But I want to go over it *briefly* again because we're going to build on it this week.

Hint

As before, `??` will highlight where one or more bit of code are missing and need to be filled in…

1.1 List Refresher

Difficulty: Low.

To complete these tasks, all of the methods that you need are listed above, so this is about testing yourself on your understanding *both* of how to read the help *and* how to index elements in a list.

The next line creates a list of (made up) Airbnb property names where each element is a string:

```
listings = ["Sunny 1-Bed", "Fantastic Dbl", "Home-Away-From-Home", "Sunny Single", "
```

1.1.1 List Arithmetic

Replace the `??` so that it prints Sunny Single.

Question

```
print(listings[?? + 1])
```

1.1.2 Negative List Arithmetic

Now use a **negative** index to print Whole House:

Question

```
print(listings[??])
```

1.1.3 Finding a Position in a List

Replace the `??` so that it prints the *index* for Fantastic Dbl in the list.

Question

```
print("The position of 'Fantastic Dbl' in the list is: " + str( ?? ))
```

1.2 Looking Across Lists

Connections

This section draws on the LOLs lecture and you will also find Code Camp's Loops session useful here.

Difficulty: Medium.

Notice that the list of `prices` is the same length as the list of `listings`, that's because these are (made-up) prices for each listing.

```
listings = ["Sunny 1-Bed", "Fantastic Dbl", "Home-Away-From-Home", "Sunny Single", "
prices = [37.50, 46.00, 125.00, 45.00, 299.99, 175.00]
```

1.2.1 Lateral Thinking

Given what you know about `listings` and `prices`, how do you print:

> "The nightly price for Home-Away-From-Home is £125.0."

But you have to do this *without* doing any of the following:

1. Using a list index directly (*i.e.* `listings[2]` and `prices[2]`) or
2. Hard-coding the name of the listing?

To put it another way, **neither** of these solutions is the answer:

```
print("The nightly price for Home\-Away\-From\-Home is £" + str(prices[2]) + ".")
# ...OR...
listing=2
print("The nightly price for Home\-Away\-From\-Home is £" + str(prices[listing[2]])
```

Tip

You will need to combine some of the ideas above and also think about the fact that the list index is that we need is the same in both lists... Also, remember that you'll need to wrap a `str(...)` around your temperature to make it into a string.

Question

```python
listing="Home-Away-From-Home" # Use this to get the solution...

# This way is perfectly fine
print("The nightly price of " + ?? + " is " + str(??))

# This way is more Python 3 and a bit easier to read
print(f"The nightly price of {??} is {??}")
```

1.2.2 Double-Checking Your Solution

```python
listing = 'Sunny Single'
```

You'll know that you got the 'right' answer to the question above if you can copy+paste your code and change only **one** thing in order to print out: "The nightly price of Sunny Single is £45.0"

Question

```python
listing = 'Sunny Single'
print(??)
```

1.2.3 Loops

Now use a `for` loop over the cities to print out the average temperature in each city:

Formatting Numbers

We often want to format numbers in a particular way to make the more readable. Commonly, in English we use commas for thousands separators and a full-stop for the decimal. Other countries follow other standards, but by default Python goes the English way. So:

```python
print(f"{1234567.25:.0f}")
print(f"{1234567.25:.1f}")
print(f"{1234567.25:.2f}")
print(f"{1234567:.2f}")
print(f"{1234567:,.2f}")
```

```
1234567
1234567.2
1234567.25
1234567.00
1,234,567.00
```

That should then help you with the output of the following block of code!

Question

```
for l in listings:
    ??
```

The output should be:

```
The nightly price of Sunny 1-Bed is £37.50
The nightly price of Fantastic Dbl is £46.00
The nightly price of Home-Away-From-Home is £125.00
The nightly price of Sunny Single is £45.00
The nightly price of Whole House is £299.99
The nightly price of Trendy Terrace is £175.00
```

2 Dictionaries

Connections

This section draws on the Dictionaries lecture and Code Camp Dictionaries session.

Difficulty: Low.

Remember that dictionaries (a.k.a. dicts) are like lists in that they are data structures containing multiple elements. A key difference between dictionaries and lists is that while elements in lists are ordered, dicts (in most programming languages, though not Python) are unordered. This means that whereas for lists we use integers as indexes to access elements, in dictonaries we use 'keys' (which can multiple different types; strings, integers, etc.). Consequently, the important term here is key-value pairs.

2.1 Creating an Atlas

The code below creates an atlas using a dictionary. The dictionary `key` is a listing, and the `value` is the latitude, longitude, and price.

```
listings = {
    'Sunny 1-Bed': [37.77, -122.43, '£37.50'],
    'Fantastic Dbl': [51.51, -0.08, '£46.00'],
    'Home-Away-From-Home': [48.86, 2.29, '£125.00'],
    'Sunny Single': [39.92, 116.40 ,'£45.00'],
}
```

2.1.1 Adding to a Dict

Add a record to the dictionary for "Whole House" following the same format.

Question

```
??
```

2.1.2 Accessing a Dict

In *one* line of code, print out the price for 'Whole House':

Question

```
??
```

2.2 Dealing With Errors

Check you understand the difference between the following two blocks of code by running them.

```python
try:
    print(listings['Trendy Terrace'])
except KeyError as e:
    print("Error found")
    print(e)
```

```
Error found
'Trendy Terrace'
```

```python
try:
    print(listings.get('Trendy Terrace','Not Found'))
except KeyError as e:
    print("Error found")
    print(e)
```

```
Not Found
```

Notice that trying to access a non-existent element of a dict triggers a `KeyError`, while asking the dict to `get` the *same element* does not, it simply returns `None`. Can you think why, depending on the situtation, *either* of these might be the 'correct' answer?

2.3 Thinking Data

This section makes use of both the Dictionaries lecture and the DOLs to Data lecture.

Tip

In this section you'll need to look up (i.e. Google) and make use of a few new functions that apply to dictionaries: `<dictionary>.items()`, `<dictionary>.keys()`. *Remember*: if in doubt, add `print(...)` statements to see what is going on!

2.3.1 Iterating over a Dict

Adapting the code below, print out the city name and airport code for every city in our Atlas.

Question

```
for l in listings.keys():
    print(??)
```

The output should look something like this:

```
Sunny 1-Bed -> £37.50
Fantastic Dbl -> £46.00
Home-Away-From-Home -> £125.00
Sunny Single -> £45.00
Whole House -> £299.99
```

2.3.2 More Complex Dicts

How would your code need to change to produce the *same output* from this data structure:

```
listings = {
    'Sunny 1-Bed': {
        'lat': 37.77,
        'lon': -122.43,
        'price': '£37.50'},
    'Fantastic Dbl': {
        'lat': 51.51,
        'lon': -0.08,
        'price': '£46.00'},
    'Home-Away-From-Home': {
        'lat': 48.86,
        'lon': 2.29,
        'price': '£125.00'},
    'Sunny Single': {
        'lat': 39.92,
        'lon': 116.40,
        'price': '£45.00'},
}
```

Question

So to print out the below for each listing it's...

```
for l in listings.keys():
    print(??)
```

```
Sunny 1-Bed -> £37.50
Fantastic Dbl -> £46.00
Home-Away-From-Home -> £125.00
Sunny Single -> £45.00
```

2.3.3 More Dictionary Action!

And how would it need to change to print out the name and latitude of every listing?

Question

```
for l in listings.keys():
    print(??)
```

The output should be something like this:

```
Sunny 1-Bed is at latitude 37.77
Fantastic Dbl is at latitude 51.51
Home-Away-From-Home is at latitude 48.86
Sunny Single is at latitude 39.92
```

2.3.4 And Another Way to Use a Dict

Now produce the *same output* using this new data structure:

```
listings_alt = [
    {'name':     'Sunny 1-Bed',
     'position': [37.77, -122.43],
     'price':    '£37.50'},
    {'name':     'Fantastic Dbl',
     'position': [51.51, -0.08],
     'price':    '£46.00'},
    {'name':     'Home-Away-From-Home',
     'position': [48.86, 2.29],
     'price':    '£125.00'},
    {'name':     'Sunny Single',
     'position': [39.92, 116.40],
     'price':    '£45.00'},
    {'name':     'Whole House',
     'position': [13.08, 80.28],
    'price':    '£299.99'}
]
```

Question

```
for l in listings_alt:
    print(??)
```

The output should be something like this:

```
Sunny 1-Bed is at latitude 37.77
Fantastic Dbl is at latitude 51.51
Home-Away-From-Home is at latitude 48.86
Sunny Single is at latitude 39.92
Whole House is at latitude 13.08
```

2.3.5 Think Data!

What are some of the main differences that you can think of between `cities` and `cities_alt` *as* data? There is no right answer.

Tip

I just want you to think about these *as data*! If you were trying to use `cities` and `cities_alt` as data what differences would you find when accessing one or more 'records'?

- Point 1 here.
- Point 2 here.
- Point 3 here.

2.4 Add to Git/GitHub

Now follow the same process that you used last week to ensure that your edited notebook is updated in Git and then synchronised with GitHub.