Data Structures

Slides created by Cameron Mohne

What Are Some Data Structures (Python)

Ordered Storage, By Index — **[Lists]**

Organized Storage, By Keys — **{ Dicts }**

Ordered Storage, By Index, with Limited Space — **(Tuples)**

Oh, also, you can't change the data stored inside once you make it.

That’s cool, that’s a lot of stuff!
There can’t possibly be more,
right?

Well...

Here's some more...

Top Element Only!

You can only remove the most recently added element!

Stacks

Like A Line

Remove the oldest element, add elements to the back!

Queues

Order Doesn't Matter, Elements Do.

No duplicate elements (usually). We don't have an order, but we can tell you if something is or isn't in the set!

Sets

Linked Lists

Each Element Is A Structure

Each "link" is an element, and gives you the location of the next element in the list.

Hashmaps Structures

Separated Storage Pt. 2

Hashmaps are used for organizing data in many separate buckets instead of just one.

Custom Structures

Whatever you need!

You can make your own data structures with your own functions!

It’s okay. We won’t be worrying about those for now.

Stay tuned for nested structures, though.

First: The Basics

Let’s take a deeper look at lists, dicts, and tuples.

Lists

What are they good for?
Lists are good for having indexed data points where you don't necessarily need to know what values are in each index. Generally, lists are used for alike data that are a singular data type. In Python, though, you can mix whatever types you desire in lists. Just be careful with how you use those lists.

How should I use them?
However you want! Lists are applicable in many places. Whether you are storing names, daily customers, or anything else, lists can be super handy! As mentioned previously, we generally use lists for things that related closely or as a storage for things that are all a part of a certain group.

Anything else?
Yeah. Later, we're going to look at nesting lists in other structures, and nesting other structures in lists.

How do I make/use them!?
Some list functionality:

```
list_name = [] # make an empty list
list_name.append(elem) # add elem to the list
list_name.remove(elem) # remove the first instance of elem
list_name[index] # get the elem at the given index in the list

list_name = [elem1, elem2, ...] # make a list with elems in it
                                  already (uniform initialization)
```

Dictionaries (Dicts)

What are they good for?
Dicts are great at organizing data in a way where you can look up exactly what you want! If you want the days in the month, maybe you'll use a dictionary that maps each month to the corresponding amount of days. (Well, roughly. February is kinda weird.)

How should I use them?
Dicts are great when you want to organize your data and be able to look it up based off of specific keys. You can use dictionaries to group related things like lists, however, you may want to also store some specific properties for each element in the dictionary.

Anything else?
Yeah. Later, we're going to look at nesting lists in other structures, and nesting other structures in lists.

How do I make/use them!?
Some dict functionality:

dict_name = {} # create an empty dict
dict_name = {key: value, key2: value2, ...} # Uniform initialization
dict_name[key] # get the value for the given key

If you have a dict already, you can add/change a key-value pair with:
dict_name[key] = value

To remove a key, you can use:
del dict_name[key]

But the key must exist. You can add a guard using an if-statement:
if key in dict_name: del my_dict[key]

Optionally, you could opt for this instead:
dict_name.pop(key, None)

Functionally they are the same, but if you know a key will exist, del dict_name[key] is faster as pop returns if they key existed in the dictionary, or None if it didn't.

Tuples

What are they good for?
Tuples are good for when you have a set structure of data you want to use, maybe something like (name, birthday, id_num). You can use them like lists to obtain data, but we cannot manipulate the data inside once made.

How do I make/use them!?
Some tuple functionality:

```
tuple_name = (elem1, ...) # Tuples can only be initialized, not declared
tuple_name[index] # get the data at the specified index
```

How should I use them?
As described above, whenever you have an unchanging structure that you want to use, you may consider using a tuple. These things may be related, or maybe unrelated! More commonly than not, though, they are related by some overarching property.

Anything else?
Yeah. You can technically return multiple variables from functions, which are stored as, you guessed it, tuples!

And Now: The Scary Stuff

Nested Data Structures

What if I take this list...

```
hi_im_a_list = [“and”, “here”, “are”, “my”, “elements”]
```

And I take this dict...

```
hi_im_a_dict = { }
```

And say, “wow, would you look at that!”

```
hi_im_a_dict = { hi_im_a_list: [“and”, “here”, “are”, “my”, “elements”] }
```

This is what we call a nested data structure: when one data structure contains another data structure as an element.

In our case, our outer structure is a dict and the inner structure is a list.

Why are nested data structures relevant?

Well let’s take a look!

Let me ask you to do the following without nested data structures:

Keep track of a whole bunch of countries and their daily COVID data for over a year.

Make an inverse search index for hundreds (in reality, millions) of pages. Meaning, map individual words to every website they appear on

Keep track of all of the items in a store (by aisle), how much they cost, how much we can stock at max, and their barcode number.

Starting to see some of the usefulness?

Hopefully! But now, you may be asking:

> “Well, how DO we do those things?”

Let’s break it down with what we now know!

Nested Data Structures

Examples

Keep track of a whole bunch of countries and their daily COVID data for over a year.

For this, we have a live code demo.

So as you saw...

By using a dictionary of countries as keys, we can keep track of all of the countries we have data on.

We used a list as our values as all of our data started ranged from Jan 22nd to May 12th, meaning we didn't really need to store the specific dates.

However, if we did want to keep track of specific days, what could we do to make our data structure support such a requirement?

Make an inverse search index for hundreds (in reality, millions) of pages. Meaning, map individual words to every website they appear on.

This is how search engines ACTUALLY work.

Search Engines: It’s a tough question!

Well let’s consider what we are being asked to do.
It’s a tactic I hopefully drilled into your heads by now!
(Stanford students worship the assignment specs.)

Search Engines

We are being asked to map words to all of the websites you can find the word in. Don’t worry about how we know what websites have what word, just pretend that we happen to know from file data.

What outer structure would you use?

Search Engines

We are being asked to map words to all of the websites you can find the word in. Don’t worry about how we know what websites have what word, just pretend that we happen to know from file data.

Well it sounds like our outer structure is asking for a dictionary!

Now what would our keys be?

Search Engines

We are being asked to map words to all of the websites you can find the word in. Don’t worry about how we know what websites have what word, just pretend that we happen to know from file data.

Well it sounds like our outer structure is asking for a dictionary! Okay, we can just directly use the words we are mapping as our keys.

What about values now?

Search Engines

We are being asked to map words to all of the websites you can find the word in. Don't worry about how we know what websites have what word, just pretend that we happen to know from file data.

Well it sounds like our outer structure is asking for a dictionary! Okay, we can just directly use the words we are mapping as our keys. Exactly! We can just make a list of all of the websites the word is in. This way, we can find every word in our dictionary when we search, and generate all of the sites with that word!

Keep track of all of the items in a store **(by aisle)**, how much they cost, how much we can stock at max, and their barcode number.

Stores have to keep track of all of these things!

Store Storage

We are being asked to keep track of all of the items in the store, but by aisle. That means we need to separate our data in a way that reflects this! There are several ways of approaching this problem.

Store Storage

We are being asked to keep track of all of the items in the store, but by aisle. That means we need to separate our data in a way that reflects this! There are several ways of approaching this problem.

What are some ideas?

Hopefully you see one handy structure pattern

We are being asked to keep track of all of the items in the store, but by aisle. That means we need to separate our data in a way that reflects this! There are several ways of approaching this problem.

Here's one: we create a dictionary that keeps track of all of the aisles in the store, and each dictionary stores it's own dictionary! This dictionary will keep track of all of the items in the aisle, and map the items to a tuple storing the item's information! (As we don't change the item information, we could also use a dict here.)

That’s a lot!

Hopefully you see the usefulness of nested data structures and feel more comfortable with potentially trying to implement these concepts into your own work in the future!

Thank You!

I will miss all of you, I had a great first year as a Code in Place Section Leader because of you all!

I am proud of how far all of you have come, I hope you learned a lot!