Introduction and lists

DATA TYPES FOR DATA SCIENCE IN PYTHON

Jason Myers
Instructor

datacamp

Data types

- Data type system sets the stage for the capabilities of the language
- Understanding data types empowers you as a data scientist

Container sequences

- Hold other types of data
- Used for aggregation, sorting, and more
- Can be mutable (list, set) or immutable (tuple)
- Iterable

Lists

- Hold data in order it was added
- Mutable
- Index

Accessing single items in list

```
cookies = ['chocolate chip', 'peanut butter', 'sugar']
```

```
cookies.append('Tirggel')
```

```
print(cookies)
```

```
['chocolate chip', 'peanut butter', 'sugar', 'Tirggel']
```

```
print(cookies[2])
```

```
sugar
```

5. Accessing single items in list
If I wanted to store a list of cookies I've eaten this week. I would begin by creating that list of cookies. When I eat another cookie, I'll want to add that to my list as well, which I can do with the append method. Then I can print the whole list of cookies. If I wanted to print the cookie I ate third; then I could use an index, which in this case would be 2 since indexes starts at zero, and print that element from the list. In addition to appending each cookie one at a time, we might also want to combine multiple lists.

Combining Lists

- Using operators, you can combine two lists into a new one

```
cakes = ['strawberry', 'vanilla']

desserts = cookies + cakes

print(desserts)
```

6. Combining Lists
Python empowers us to combine lists in a few ways. First, we can use operators like the plus sign to add two lists together. When we do this, we will get a new list object returned to us. For example, we can add a list of cookies and cakes to create a list of deserts. Additionally, we can use the extend method on the list to combine two lists effectively appending all the values from a second list.

```
['chocolate chip', 'peanut butter', 'sugar', 'Tirggel',
'strawberry', 'vanilla']
```

- `.extend()` method merges a list into another list at the end

Finding Elements in a List

- `.index()` method locates the position of a data element in a list

```
position = cookies.index('sugar')

print(position)
```

7. Finding Elements in a List
Earlier, we took advantage of the indexability of lists to return the item at a certain index. However, if we only know the value of an element, we can use it with the index method to get its index. Here I'm trying to remember when I had a sugar cookie.

```
3
```

```
cookies[3]
```

```
'sugar'
```

Removing Elements in a List

- `.pop()` method removes an item from a list and allows you to save it

```
name = cookies.pop(position)

print(name)
```

```
sugar
```

```
print(cookies)
```

```
['chocolate chip', 'peanut butter', 'Tirggel']
```

Iterating over lists

- `for` loops are the most common way of iterating over a list

```
for cookie in cookies:
    print(cookie)
```

```
chocolate chip
peanut butter
Tirggel
```

Sorting lists

- `sorted()` function sorts data in numerical or alphabetical order and returns a new list

```
print(cookies)
```

```
['chocolate chip', 'peanut butter', 'Tirggel']
```

```
sorted_cookies = sorted(cookies)

print(sorted_cookies)
```

```
['Tirggel', 'chocolate chip', 'peanut butter']
```

Let's practice!

DATA TYPES FOR DATA SCIENCE IN PYTHON

Meet the Tuples

DATA TYPES FOR DATA SCIENCE IN PYTHON

Jason Myers
Instructor

datacamp

Tuple, Tuple

- Hold data in order
- Index
- *Immutable*
- Pairing
- Unpackable

Zipping tuples

- Tuples are commonly created by zipping lists together with `zip()`
- Two lists: `us_cookies` , `in_cookies`

```
top_pairs = list(zip(us_cookies, in_cookies))
print(top_pairs)
```

```
[('Chocolate Chip', 'Punjabi'), ('Brownies', 'Fruit Cake Rusk'),
('Peanut Butter', 'Marble Cookies'), ('Oreos', 'Kaju Pista Cookies'),
('Oatmeal Raisin', 'Almond Cookies')]
```

Unpacking tuples

- Unpacking tuples is a very expressive way for working with data

```
us_num_1, in_num_1 = top_pairs[0]
print(us_num_1)
```

```
Chocolate Chip
```

```
print(in_num_1)
```

```
Punjabi
```

More unpacking in Loops

- Unpacking is especially powerful in loops

```
for us_cookie, in_cookie in top_pairs:
    print(in_cookie)
    print(us_cookie)
```

```
Punjabi
Chocolate Chip
Fruit Cake Rusk
Brownies
# ..etc..
```

Enumerating positions

- Another useful tuple creation method is the `enumerate()` function
- Enumeration is used in loops to return the position and the data in that position while looping

```
for idx, item in enumerate(top_pairs):
    us_cookie, in_cookie = item
    print(idx, us_cookie, in_cookie)
```

```
(0, 'Chocolate Chip', 'Punjabi')
(1, 'Brownies', 'Fruit Cake Rusk')
# ..etc..
```

Be careful when making tuples

- Use `zip()`, `enumerate()`, or `()` to make tuples

```
item = ('vanilla', 'chocolate')
print(item)
```

```
('vanilla', 'chocolate')
```

- Beware of tailing commas!

```
item2 = 'butter',
print(item2)
```

```
('butter',)
```

Let's practice!

DATA TYPES FOR DATA SCIENCE IN PYTHON

Sets for unordered and unique data

DATA TYPES FOR DATA SCIENCE IN PYTHON

Jason Myers
Instructor

datacamp

Set

- Unique
- Unordered
- Mutable
- Python's implementation of Set Theory from Mathematics

Creating Sets

- Sets are created from a list

```
cookies_eaten_today = ['chocolate chip', 'peanut butter',
   ...: 'chocolate chip', 'oatmeal cream', 'chocolate chip']
types_of_cookies_eaten = set(cookies_eaten_today)
print(types_of_cookies_eaten)
```

```
set(['chocolate chip', 'oatmeal cream', 'peanut butter'])
```

Modifying Sets

- `.add()` adds single elements
- `.update()` merges in another set or list

```
types_of_cookies_eaten.add('biscotti')

types_of_cookies_eaten.add('chocolate chip')

print(types_of_cookies_eaten)
```

```
set(['chocolate chip', 'oatmeal cream', 'peanut butter', 'biscotti'])
```

4. Modifying Sets
When working with a set we will use the add method to add a new element to the set. It will only add the element if it is unique otherwise it just continues on. Also, we can add multiple items using the update method. The update method takes a list of items and adds each one to the set if it is not present.

Updating Sets

```
cookies_hugo_ate = ['chocolate chip', 'anzac']


types_of_cookies_eaten.update(cookies_hugo_ate)


print(types_of_cookies_eaten)
```

```
set(['chocolate chip', 'anzac', 'oatmeal cream',
'peanut butter', 'biscotti'])
```

Removing data from sets

- `.discard()` safely removes an element from the set by value
- `.pop()` removes and returns an arbitrary element from the set (KeyError when empty)

```
types_of_cookies_eaten.discard('biscotti')
print(types_of_cookies_eaten)
```

```
set(['chocolate chip', 'anzac', 'oatmeal cream', 'peanut butter'])
```

```
types_of_cookies_eaten.pop()
types_of_cookies_eaten.pop()
```

```
'chocolate chip'
'anzac'
```

6. Removing data from sets
When removing data from a set, we can use the discard method to safely remove an element from the set by its value. No error will be thrown if the value is not found. We can also use the pop method to remove and return an arbitrary element from the set.

Set Operations - Similarities

- `.union()` set method returns a set of all the names (`or`)
- `.intersection()` method identifies overlapping data (`and`)

```
cookies_jason_ate = set(['chocolate chip', 'oatmeal cream',
'peanut butter'])
cookies_hugo_ate = set(['chocolate chip', 'anzac'])
cookies_jason_ate.union(cookies_hugo_ate)
```

```
set(['chocolate chip', 'anzac', 'oatmeal cream', 'peanut butter'])
```

```
cookies_jason_ate.intersection(cookies_hugo_ate)
```

```
set(['chocolate chip'])
```

7. Set Operations - Similarities
The union method on a set accepts a set as an argument and returns all the unique elements from both sets as a new one. The intersection method also accepts a set and returns the overlapping elements found in both sets.

Set Operations - Differences

- `.difference()` method identifies data present in the set on which the method was used that is not in the arguments (`-`)
- Target is important!

```
cookies_jason_ate.difference(cookies_hugo_ate)
```

```
set(['oatmeal cream', 'peanut butter'])
```

```
cookies_hugo_ate.difference(cookies_jason_ate)
```

```
set(['anzac'])
```

8. Set Operations - Differences
We can use the difference method, which accepts a set, to find elements in one set that are not present in another set.

Let's practice!

DATA TYPES FOR DATA SCIENCE IN PYTHON