Introduction and lists

DATA TYPES 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
```

Combining lists

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

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

desserts = cookies + cakes

print(desserts)
```

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

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

```
cookies.extend(cakes)
```

Finding elements in a list

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

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


print(position)
```

```
3
```

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

- **List comprehensions** are a common way of iterating over a list to perform some action on them

```
titlecase_cookies = [cookie.title() for cookie in cookies
print(titlecase_cookies)
```

```
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 IN PYTHON

Meet the tuples

DATA TYPES 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 IN PYTHON

Strings

DATA TYPES IN PYTHON

Jason Myers
Instructor

datacamp

Creating formatted strings

- f-strings (formatted string literals) - `f""`

```
cookie_name = "Anzac"
cookie_price = "$1.99"

print(f"Each { cookie_name } cookie costs { cookie_price }.")
```

```
"Each Anzac cookie costs $1.99."
```

Joining with strings

- `"".join()` uses the string it's called on to join an iterable

```
child_ages = ["3", "4", "7", "8"]

print(", ".join(child_ages))
```

```
"3, 4, 7, 8"
```

```
print(f"The children are ages {','.join(child_ages[0:3])}, and {child_ages[-1]}.")
```

```
"The children are ages 3, 4, 7, and 8."
```

Matching parts of a string

- `.startswith()` and `.endswith()` methods will tell you if a string starts or ends with another character or string

```
boy_names = ["Mohamed", "Youssef", "Ahmed"]
print([name for name in boy_names if name.startswith('A')])
```

```
["Ahmed"]
```

- Be careful as these and most string functions are case-sensitive.

Searching for things in strings

- The `in` operator searches for some value in some iterable type like a string.

```
"long" in "Life is a long lesson in humility."
```

```
True
```

```
"life" in "Life is a long lesson in humility."
```

```
False
```

An approach to being case insensitive

- `.lower()` method returns a lower case string

```
"life" in "Life is a long lesson in humility.".lower()
```

```
True
```

Let's practice!

DATA TYPES IN PYTHON