DATA TYPES FOR DATA SCIENCE

Counting made easy

Jason Myers
Instructor

Collections Module

- Part of Standard Library
- Advanced data containers

Counter

- Special dictionary used for counting data, measuring frequency

```
In [1]: from collections import Counter

In [2]: nyc_eatery_count_by_types = Counter(nyc_eatery_types)

In [3]: print(nyc_eatery_count_by_type)
Counter({'Mobile Food Truck': 114, 'Food Cart': 74, 'Snack Bar': 24,
'Specialty Cart': 18, 'Restaurant': 15, 'Fruit & Vegetable Cart': 4})

In [4]: print(nyc_eatery_count_by_types['Restaurant'])
15
```

Counter to find the most common

- `.most_common()` method returns the counter values in descending order

```
In [1]: print(nyc_eatery_count_by_types.most_common(3))
[('Mobile Food Truck', 114), ('Food Cart', 74), ('Snack Bar', 24)]
```

DATA TYPES FOR DATA SCIENCE

Let's practice!

DATA TYPES FOR DATA SCIENCE

Dictionaries of unknown structure - defaultdict

Jason Myers
Instructor

Dictionary Handling

```
In [1]: for park_id, name in nyc_eateries_parks:
   ...:     if park_id not in eateries_by_park:
   ...:         eateries_by_park[park_id] = []
   ...:     eateries_by_park[park_id].append(name)

In [2]: print(eateries_by_park['M010'])
{'MOHAMMAD MATIN','PRODUCTS CORP.', 'Loeb Boathouse Restaurant',
'Nandita Inc.', 'SALIM AHAMED', 'THE NY PICNIC COMPANY',
'THE NEW YORK PICNIC COMPANY, INC.', 'NANDITA, INC.',
'JANANI FOOD SERVICE, INC.'}
```

Using defaultdict

- Pass it a default type that every key will have even if it doesn't currently exist
- Works exactly like a dictionary

```
In [1]: from collections import defaultdict

In [2]: eateries_by_park = defaultdict(list)

In [3]: for park_id, name in nyc_eateries_parks:
   ...:     eateries_by_park[park_id].append(name)

In [4]: print(eateries_by_park['M010'])
{'MOHAMMAD MATIN','PRODUCTS CORP.', 'Loeb Boathouse Restaurant',
'Nandita Inc.', 'SALIM AHAMED', 'THE NY PICNIC COMPANY',
'THE NEW YORK PICNIC COMPANY, INC.', 'NANDITA, INC.',
'JANANI FOOD SERVICE, INC.'}
```

defaultdict (cont.)

```
In [1]: from collections import defaultdict

In [2]: eatery_contact_types = defaultdict(int)

In [3]: for eatery in nyc_eateries:
   ...:     if eatery.get('phone'):
   ...:         eatery_contact_types['phones'] += 1
   ...:     if eatery.get('website'):
   ...:         eatery_contact_types['websites'] += 1

In [4]: print(eatery_contact_types)
defaultdict(<class 'int'>, {'phones': 28, 'websites': 31})
```

DATA TYPES FOR DATA SCIENCE

Let's practice!

DATA TYPES FOR DATA SCIENCE

Maintaining Dictionary Order with OrderedDict

Jason Myers
Instructor

Order in Python dictionaries

- Python version < 3.6 NOT ordered
- Python version > 3.6 ordered

Getting started with OrderedDict

```
In [1]: from collections import OrderedDict

In [2]: nyc_eatery_permits = OrderedDict()

In [3]: for eatery in nyc_eateries:
   ...:     nyc_eatery_permits[eatery['end_date']] = eatery

In [4]: print(list(nyc_eatery_permits.items())[:3]
('2029-04-28', {'name': 'Union Square Seasonal Cafe',
'location': 'Union Square Park', 'park_id': 'M089',
'start_date': '2014-04-29', 'end_date': '2029-04-28',
'description': None, 'permit_number': 'M89-SB-R', 'phone': '212-677-7818',
'website': 'http://www.thepavilionnyc.com/', 'type_name': 'Restaurant'})
```

OrderedDict power feature

- `.popitem()` method returns items in reverse insertion order

```
In [1]: print(nyc_eatery_permits.popitem())
('2029-04-28', {'name': 'Union Square Seasonal Cafe',
'location': 'Union Square Park', 'park_id': 'M089',
'start_date': '2014-04-29', 'end_date': '2029-04-28',
'description': None, 'permit_number': 'M89-SB-R', 'phone': '212-677-7818',
'website': 'http://www.thepavilionnyc.com/', 'type_name': 'Restaurant'})

In [2]: print(nyc_eatery_permits.popitem())
('2027-03-31', {'name': 'Dyckman Marina Restaurant',
'location': 'Dyckman Marina Restaurant', 'park_id': 'M028',
'start_date': '2012-04-01', 'end_date': '2027-03-31',
'description': None, 'permit_number': 'M28-R', 'phone': None,
'website': None, 'type_name': 'Restaurant'})
```

OrderedDict power feature (2)

- You can use the `last=False` keyword argument to return the items in insertion order

```
In [3]: print(nyc_eatery_permits.popitem(last=False))
('2012-12-07', {'name': 'Mapes Avenue Ballfields Mobile Food Truck',
'location': 'Prospect Avenue, E. 181st Street', 'park_id': 'X289',
'start_date': '2009-07-01', 'end_date': '2012-12-07',
'description': None, 'permit_number': 'X289-MT', 'phone': None,
'website': None, 'type_name': 'Mobile Food Truck'})
```

DATA TYPES FOR DATA SCIENCE

Let's practice!

DATA TYPES FOR DATA SCIENCE

namedtuple

Jason Myers
Instructor

What is a namedtuple?

- A tuple where each position (column) has a name
- Ensure each one has the same properties
- Alternative to a `pandas` DataFrame row

Creating a namedtuple

- Pass a name and a list of fields

```
In [1]: from collections import namedtuple

In [2]: Eatery = namedtuple('Eatery', ['name', 'location', 'park_id',
   ...: 'type_name'])

In [3]: eateries = []

In [4]: for eatery in nyc_eateries:
   ...:     details = Eatery(eatery['name'],
   ...:                      eatery['location'],
   ...:                      eatery['park_id'],
   ...:                      eatery['type_name'])
   ...:     eateries.append(details)

In [5]: print(eateries[0])
Eatery(name='Mapes Avenue Ballfields Mobile Food Truck',
location='Prospect Avenue, E. 181st Street',
park_id='X289', type_name='Mobile Food Truck')
```

Leveraging namedtuples

- Each field is available as an attribute of the namedtuple

```
In [1]: for eatery in eateries[:3]:
   ...:     print(eatery.name)
   ...:     print(eatery.park_id)
   ...:     print(eatery.location)

Mapes Avenue Ballfields Mobile Food Truck
X289
Prospect Avenue, E. 181st Street

Claremont Park Mobile Food Truck
X008
East 172 Street between Teller & Morris avenues

Slattery Playground Mobile Food Truck
X085
North corner of Valenti Avenue & East 183 Street
```

DATA TYPES FOR DATA SCIENCE

Let's practice!