Counting made easy

DATA TYPES IN PYTHON

Jason Myers
Instructor

datacamp

Collections Module

- Part of Standard Library
- Advanced data containers

Counter

- Special dictionary used for counting data, measuring frequency

```
from collections import Counter
nyc_eatery_count_by_types = Counter(nyc_eatery_types)
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})
```

```
print(nyc_eatery_count_by_types['Restaurant'])
```

```
15
```

Counter to find the most common

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

```
print(nyc_eatery_count_by_types.most_common(3))
```

```
[('Mobile Food Truck', 114), ('Food Cart', 74), ('Snack Bar', 24)]
```

Let's practice!

DATA TYPES IN PYTHON

Dictionaries of unknown structure - defaultdict

DATA TYPES IN PYTHON

Jason Myers
Instructor

datacamp

Dictionary Handling

```
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)
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

```
from collections import defaultdict
eateries_by_park = defaultdict(list)
for park_id, name in nyc_eateries_parks:
    eateries_by_park[park_id].append(name)
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.', ...}
```

Using defaultdict

```
from collections import defaultdict
eatery_contact_types = defaultdict(int)
for eatery in nyc_eateries:
    if eatery.get('phone'):
        eatery_contact_types['phones'] += 1
    if eatery.get('website'):
        eatery_contact_types['websites'] += 1
print(eatery_contact_types)
```

```
defaultdict(<class 'int'>, {'phones': 28, 'websites': 31})
```

Let's practice!

DATA TYPES IN PYTHON

namedtuple

DATA TYPES IN PYTHON

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

```
from collections import namedtuple
Eatery = namedtuple('Eatery', ['name', 'location', 'park_id',
   ...: 'type_name'])
eateries = []
for eatery in nyc_eateries:
    details = Eatery(eatery['name'],
                     eatery['location'],
                     eatery['park_id'],
                     eatery['type_name'])
    eateries.append(details)
```

Print the first element

```
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

```
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 ...
```

Let's practice!

DATA TYPES IN PYTHON

Dataclasses

DATA TYPES IN PYTHON

Jason Myers
Instructor

datacamp

Why use dataclasses

- Support for default values
- Custom representations of the objects
- Easy tuple or a dictionary conversion
- Custom properties
- Frozen instances

Looking at our first dataclass

```
from dataclasses import dataclass
```

```
@dataclass
class Cookie:
    name: str
    quantity: int = 0
```

```
chocolate_chip = Cookie("chocolate chip", 13)
print(chocolate_chip.name)
print(chocolate_chip.quantity)
```

```
chocolate chip
13
```

Easy tuple or a dictionary conversion

```
from dataclasses import asdict, astuple


ginger_molasses = Cookie("ginger molasses", 8)
asdict(ginger_molasses)
```

```
{'name': 'ginger molasses', 'quantity': 8}
```

```
astuple(ginger_molasses)
```

```
('ginger molasses', 8)
```

Custom properties

```
from decimal import Decimal



@dataclass
class Cookie:
    name: str
    cost: Decimal
    quantity: int
```

```
    @property
    def value_of_goods(self):
        return int(self.quantity) * self.cost
```

Using custom properties

```
peanut = Cookie("peanut butter", Decimal("1.2"), 8)


peanut.value_of_goods
```

```
Decimal('9.6')
```

Frozen instances

```
@dataclass(frozen=True)
class Cookie:
    name: str
    quantity: int = 0

c = Cookie("chocolate chip", 10)
```

```
c.quantity = 15
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<string>", line 4, in __setattr__
dataclasses.FrozenInstanceError: cannot assign to field 'quantity'
```

Let's practice!

DATA TYPES IN PYTHON

Wrap-up

DATA TYPES IN PYTHON

Jason Myers
Instructor

Sequence data types

- Lists

```
['Chocolate Chip', 'Peanut Butter']
```

- Tuples

```
('Sugar', 'Eggs')
```

- Strings

```
'Cookies are wonderful'
```

Dictionaries

- Safely adding and removing

```
squirrels_by_park.pop("City Hall Park", {})
```

- Unpacking items

```
for field, value in squirrels_by_park.items():
```

- Handling nested data

```
for park in squirrels_by_park:
    print(squirrels_by_park[park].get('color', 'N/A'))
```

Numeric and logical types

- Integers `int(1)`
- Floats `float(1.333333334)`
- Decimals `Decimal(5.50)`
- Booleans `True or False`
- Sets `{'Anzac', 'Oatmeal Raisin'}`

Complex data types

- Counters

```
Counter(nyc_eatery_types)
```

- Defaultdicts

```
eateries_by_park = defaultdict(list)
for park_id, name in nyc_eateries_parks:
  eateries_by_park[park_id].append(name)
```

- Namedtuples

```
namedtuple('Worm', ['species', 'sex', 'mass'])
```

Complex data types

- Dataclasses

```
@dataclass
class WeightEntry:
    species: str
    flipper_length: int
    body_mass: int
    sex: str
```

Congratulations!

DATA TYPES IN PYTHON