Dictionaries in Python - Advanced Python 03 - Programming Tutorial

https://www.youtube.com/watch?v=LTXnQdrwyrw

Using dictionaries

DATA TYPES FOR DATA SCIENCE IN PYTHON

Jason Myers
Instructor

datacamp

Creating and looping through dictionaries

- Hold data in key/value pairs
- Nestable (use a dictionary as the value of a key within a dictionary)
- Iterable
- Created by dict() or {}

```
art_galleries = {}

for name, zip_code in galleries:
    art_galleries[name] = zip_code
```

2. Creating and looping through dictionaries
Dictionaries hold data in key/value pairs. While the key must be alphanumeric, the value can be any other data type. It's also possible to nest dictionaries so that you can work with grouped or hierarchical data. We can also iterate over the keys and values of a dictionary. We can also iterate over the items of a dictionary, which are tuples of the key value pairs)!

Printing in the loop

```
for name in art_galleries:
    print(name)
```

```
Zwirner David Gallery
Zwirner & Wirth
Zito Studio Gallery
Zetterquist Galleries
Zarre Andre Gallery
```

Safely finding by key

```
art_galleries['Louvre']
```

```
|---------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-1-4f51c265f287> in <module>()
--> 1 art_galleries['Louvre']

KeyError: 'Louvre'
```

- Getting a value from a dictionary is done using the key as an index
- If you ask for a key that does not exist that will stop your program from running in a KeyError

Safely finding by key (cont.)

- `.get()` method allows you to safely access a key without error or exception handling
- If a key is not in the dictionary, `.get()` returns `None` by default or you can supply a value to return

```
art_galleries.get('Louvre', 'Not Found')
```

```
'Not Found'
```

```
art_galleries.get('Zarre Andre Gallery')
```

```
'10011'
```

Working with nested dictionaries

```
art_galleries.keys()
```

```
dict_keys(['10021', '10013', '10001', '10009', '10011',
   ...: '10022', '10027', '10019', '11106', '10128'])
```

```
print(art_galleries['10027'])
```

```
{"Paige's Art Gallery": '(212) 531-1577',
'Triple Candie': '(212) 865-0783',
'Africart Motherland Inc': '(212) 368-6802',
'Inner City Art Gallery Inc': '(212) 368-4941'}
```

- The `.keys()` method shows the keys for a given dictionary

Accessing nested data

```
art_galleries['10027']['Inner City Art Gallery Inc']
```

```
'(212) 368-4941'
```

- Common way to deal with repeating data structures
- Can be accessed using multiple indices or the `.get()` method

7. Accessing nested data
and I can get the phone number of one of those galleries by using the gallery name which is the nested dictionary's key as a secondary index. Nesting dictionaries is a very common way to deal with repeating data structures such as yearly data, grouped, or hierarchical data such as organization reporting structures. You access nested values by providing multiple indices to the dictionary or using the get method on an index.

Let's practice!

DATA TYPES FOR DATA SCIENCE IN PYTHON

Altering dictionaries

DATA TYPES FOR DATA SCIENCE IN PYTHON

Jason Myers
Instructor

datacamp

Adding and extending dictionaries

- Assignment to add a new key/value to a dictionary
- `.update()` method to update a dictionary from another dictionary, tuples or keywords

```
print(galleries_10007)
```

```
{'Nyabinghi African Gift Shop': '(212) 566-3336'}
```

```
art_galleries['10007'] = galleries_10007
```

2. Adding and extending dictionaries
You can add data to a dictionary just by using a new key as an index and assigning it a value. It's also possible to supply a dictionary, list of tuples or a set of keywords arguments to the update() method to add values into a dictionary. I have a dictionary that contains the art galleries in the 10007 zip code and I want to add it to my art_galleries dictionary. I assign the zip code as the key and then the dictionary as the value.

Updating a dictionary

```
galleries_11234 = [
  ('A J ARTS LTD', '(718) 763-5473'),
  ('Doug Meyer Fine Art', '(718) 375-8006'),
  ('Portrait Gallery', '(718) 377-8762')]
art_galleries['11234'].update(galleries_11234)
print(art_galleries['11234'])
```

```
{'Portrait Gallery': '(718) 377-8762',
'A J ARTS LTD': '(718) 763-5473',
'Doug Meyer Fine Art': '(718) 375-8006'}
```

3. Updating a dictionary
I can also create a list of tuples, and supply them to the update() method. Here I'm supplying them to a dictionary index of the zip code since I want this data to be nested underneath it. Finally I'll print the zip code to be sure they were added. Notice how those are now a dictionary under that key.

Popping and deleting from dictionaries

- `del` instruction deletes a key/value
- `.pop()` method safely removes a key/value from a dictionary.

```
del art_galleries['11234']
galleries_10310 = art_galleries.pop('10310')
print(galleries_10310)
```

```
{'New Dorp Village Antiques Ltd': '(718) 815-2526'}
```

4. Popping and deleting from dictionaries
You can use the del python instruction on a dictionary key to remove data from a dictionary. However, it's important to remember that del will throw a KeyError if the key you are trying to delete does not exist. The pop() method provides a safe way to remove keys from a dictionary. Let's start by removing all the galleries from zipcode '11234', then I'm not sure if there are any galleries in zip code 10310, so I'm going to pop that zip code from the dictionary and save it. Finally, I'll print the popped value.

Let's practice!

DATA TYPES FOR DATA SCIENCE IN PYTHON

Pythonically using dictionaries

DATA TYPES FOR DATA SCIENCE IN PYTHON

Jason Myers
Instructor

1. Pythonically using dictionaries
So far, we've been working with dictionaries in a straight forward manner, but Python has more efficient ways to work with them. We refer to these manners of interacting as being Pythonic.

datacamp

Working with dictionaries more pythonically

- `.items()` method returns an object we can iterate over

```
for gallery, phone_num in art_galleries.items():
    print(gallery)
    print(phone_num)
```

```
'Miakey Art Gallery'
'(718) 686-0788'
'Morning Star Gallery Ltd'
'(212) 334-9330'}
'New York Art Expo Inc'
'(212) 363-8280'
```

2. Working with dictionaries more pythonically
Python provides an items() method which returns a dict_items object, but we can iterate over it as a list of key/value tuples. If I use the dot items() method on my original art_galleries dictionary it returns a tuple of the gallery name and the phone number. I can use tuple unpacking to go ahead and expand that into its two components in the loop. Finally, I print them so you can see the information.

Checking dictionaries for data

- `.get()` does a lot of work to check for a key
- `in` operator is much more efficient and clearer

```
'11234' in art_galleries
```

```
False
```

```
if '10010' in art_galleries:
    print('I found: %s' % art_galleries['10010'])
else:
    print('No galleries found.')
```

```
I found: {'Nyabinghi African Gift Shop': '(212) 566-3336'}
```

Let's practice!

DATA TYPES FOR DATA SCIENCE IN PYTHON

Working with CSV files

DATA TYPES FOR DATA SCIENCE IN PYTHON

Jason Myers
Instructor

datacamp

CSV Files

```
NAME,TEL,ADDRESS1,ADDRESS2,CITY,ZIP
O'reilly William & Co Ltd,(212) 396-1822,52 E 76th St,,New York,10021
```

Reading from a file using CSV reader

- Python `csv` module
- `open()` function provides a variable that represents a file, takes a path and a mode
- `csv.reader()` reads a file object and returns the lines from the file as tuples
- `.close()` method closes file objects

```
import csv
csvfile = open('ART_GALLERY.csv', 'r')
for row in csv.reader(csvfile):
    print(row)
```

Reading from a CSV - Results

```
['NAME', 'the_geom', 'TEL', 'URL', 'ADDRESS1',
'ADDRESS2', 'CITY', 'ZIP']
["O'reilly William & Co Ltd",
'POINT (-73.96273074561996 40.773800871637576)',
'(212) 396-1822', '52 E 76th St', '', 'New York',
'10021']
```

```
csvfile.close()
```

Creating a dictionary from a file

- Often we want to go from CSV file to dictionary
- DictReader does just that
- If data doesn't have a header row, you can pass in the column names

```
for row in csv.DictReader(csvfile):
    print(row)
```

csv.Reader() allows you to access CSV data using indexes and is ideal for simple CSV files. csv.DictReader() on the other hand is friendlier and easy to use, especially when working with large CSV files.

```
OrderedDict([('NAME', 'Odyssia Gallery'),
('the_geom', 'POINT (-73.96269813635554 40.7618747512849)'),
('TEL', '(212) 486-7338'),
('URL', 'http://www.livevillage.com/newyork/art/odyssia-gallery.html'),
('ADDRESS1', '305 E 61st St'), ...
```

Let's practice!

DATA TYPES FOR DATA SCIENCE IN PYTHON