Working with Known JSON Schemas

Introduction ¶

You've started taking a look at JSON files and you'll continue to explore how to navigate and traverse these files. One common use case of JSON files will be when you are connecting to various websites through their established APIs to retrieve data from them. With these, you are typically given a schema for how the data is structured and then will use this knowledge to retrieve pertinent information.

Objectives

You will be able to:

- Use the JSON module to load and parse JSON documents
- Extract data using predefined JSON schemas
- Convert JSON to a pandas dataframe

Reading a JSON Schema

In this lesson, you'll take a look at the response from the New York Times API. (We cover APIs in more depth in other lessons, but the general idea is that the New York Times makes some of its data available over the web, and it uses the JSON format to do so.)

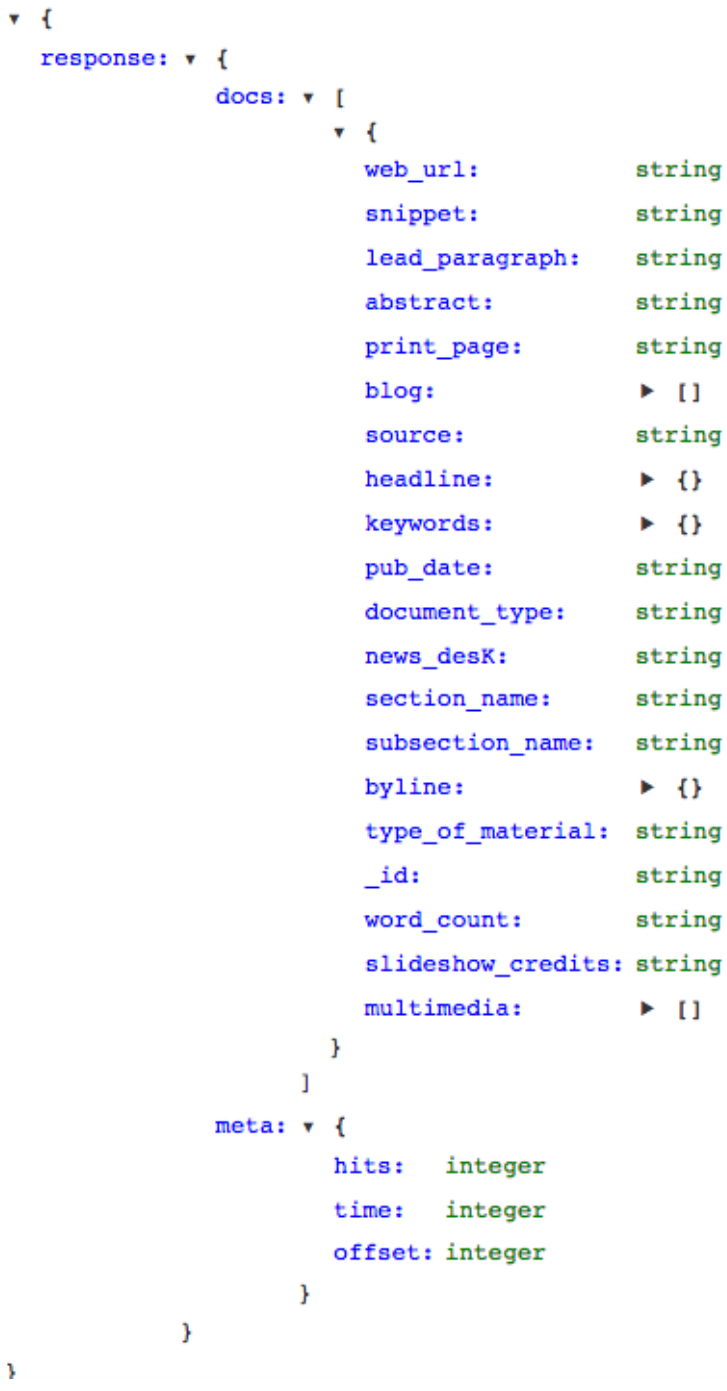
Here's the JSON schema provided for a section of the NY Times API:

Responses

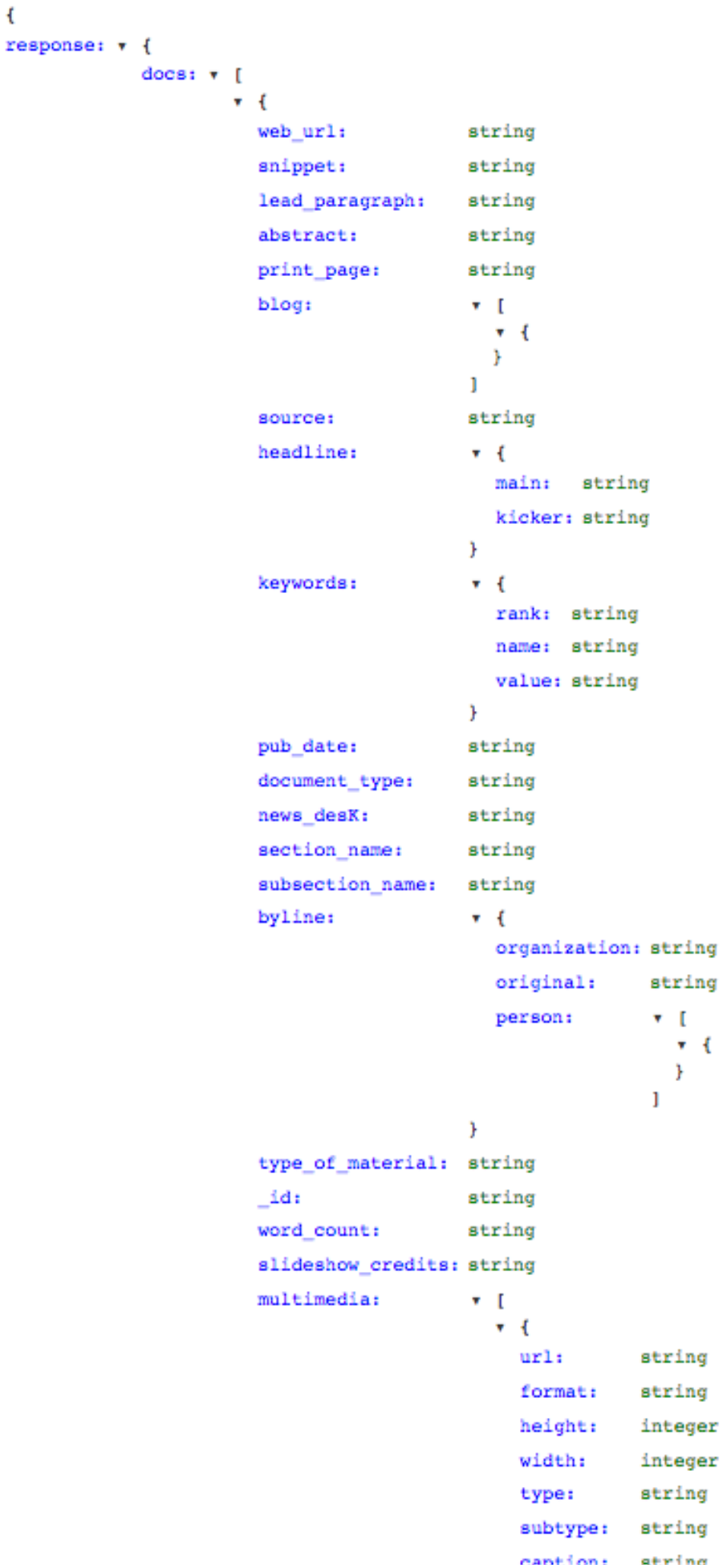
200

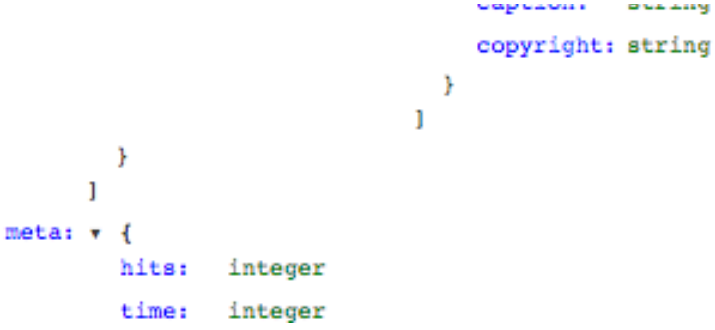
The docs requested by the article search.

Schema Example



or a more detailed view (truncated):


{
response: ▾ {
docs: ▾ [
▾ {
web_url: string
snippet: string
lead_paragraph: string
abstract: string
print_page: string
blog: ▾ [
▾ {
}
]
source: string
headline: ▾ {
main: string
kicker: string
}
keywords: ▾ {
rank: string
name: string
value: string
}
pub_date: string
document_type: string
news_desK: string
section_name: string
subsection_name: string
byline: ▾ {
organization: string
original: string
person: ▾ [
▾ {
}
]
}
type_of_material: string
_id: string
word_count: string
slideshow_credits: string
multimedia: ▾ [
▾ {
url: string
format: string
height: integer
width: integer
type: string
subtype: string



You can see that the master structure is a dictionary and has a key named `'response'`. The value associated with the `'response'` key is also a dictionary and has two keys: `'docs'` and `'meta'`. As you continue to examine the schema hierarchy, you'll notice the vast majority of the elements comprising this data structure, in this case, are dictionaries.

Loading the Data File

As we have done in previous lessons, let's start by importing this data from the file. The code below uses the `json` module (documentation here (https://docs.python.org/3/library/json.html)) and built-in `open` function to load the data from a JSON file into a Python object called `data`.

In [16]:
```
import json
```

In [17]:
```
with open('ny_times_response.json', 'r') as f:
    data = json.load(f)
```

In [18]:
```
print(type(data))
print(data.keys())
print(data)
```

```
<class 'dict'>
dict_keys(['status', 'copyright', 'response'])
{'status': 'OK', 'copyright': 'Copyright (c) 2018 The New York Times Company. A
ll Rights Reserved.', 'response': {'docs': [{'web_url': 'https://query.nytimes.
com/gst/abstract.html?res=9C05E3D7113DE633A25754C1A9679D946597D6CF', 'snippet':
'Spent $22,200', 'abstract': 'Spent $22,200', 'print_page': '2', 'blog': {}, 's
ource': 'The New York Times', 'multimedia': [], 'headline': {'main': "HIGGINS,
SPENT $22,189.53.; Governor-Elect's Election Expenses -- Harrison $9,220.28.",
'kicker': None, 'content_kicker': None, 'print_headline': None, 'name': None,
'seo': None, 'sub': None}, 'keywords': [{'name': 'persons', 'value': 'HIGGINS,
LT. GOV.', 'rank': 0, 'major': None}], 'pub_date': '1904-11-17T00:00:00Z', 'doc
ument_type': 'article', 'type_of_material': 'Article', '_id': '4fc04eb745c1498b
0d23da00', 'word_count': 213, 'score': 1}, {'web_url': 'https://query.nytimes.c
om/gst/abstract.html?res=9E07E6DA1F3BE433A25750C2A9669D946593D6CF', 'snippet':
'', 'print_page': '15', 'blog': {}, 'source': 'The New York Times', 'multimedi
a': [], 'headline': {'main': 'GARDEN BOUTS CANCELED; Mauriello Says He Could No
t Be Ready on Nov. 3', 'kicker': '1', 'content_kicker': None, 'print_headline':
None, 'name': None, 'seo': None, 'sub': None}, 'keywords': [], 'pub_date': '194
4-10-23T00:00:00Z', 'document_type': 'article', 'type_of_material': 'Article',
'_id': '4fc21ebf45c1498b0d612b22', 'word_count': 149, 'score': 1}, {'web_url':
'https://query.nytimes.com/gst/abstract.html?res=9E07E1DB1330E531A15756C1A9639C
946492D6CF', 'snippet': 'Stock prices last week, on the lightest volume of the
year, sustained the largest losses in about two months. Some technicians were i
nclined to believe that if offerings became heavier on the decline, the average
s could challenge the lows of the f...', 'print_page': 'F1', 'blog': {}, 'sourc
e': 'The New York Times', 'multimedia': [], 'headline': {'main': 'Stock Drop Is
Biggest in 2 Months--Margin Rise Held Factor in Lightest Trading of 1955', 'kic
ker': '1', 'content_kicker': None, 'print_headline': None, 'name': None, 'seo':
None, 'sub': None}, 'keywords': [], 'pub_date': '1955-05-15T00:00:00Z', 'docume
nt_type': 'article', 'byline': {'original': 'By JOHN G. FORREST', 'person':
[{'firstname': 'John', 'middlename': 'G.', 'lastname': 'FORREST', 'qualifier':
None, 'title': None, 'role': 'reported', 'organization': '', 'rank': 1}], 'orga
nization': None}, 'type_of_material': 'Article', '_id': '4fc3b41d45c1498b0d7fd4
1e', 'word_count': 823, 'score': 1}, {'web_url': 'https://query.nytimes.com/gs
t/abstract.html?res=9504EEDE123BE733A25755C0A9679D946597D6CF', 'snippet': 'The
first public rehearsal and concert of the Philharmonic Society will be given at
Carnegie Hall on Friday afternoon and Saturday evening. Gustav F. Kogel, the fo
rmer director of the Museum Concert at Frankfort-on-the-Main, will be the condu
ctor....', 'abstract': 'Healy, Michael, will suit', 'print_page': '20', 'blog':
{}, 'source': 'The New York Times', 'multimedia': [], 'headline': {'main': 'MUS
IC OF THE WEEK', 'kicker': None, 'content_kicker': None, 'print_headline': Non
e, 'name': None, 'seo': None, 'sub': None}, 'keywords': [{'name': 'persons', 'v
alue': 'HEALY, MICHAEL', 'rank': 0, 'major': None}], 'pub_date': '1904-11-06T0
0:00:00Z', 'document_type': 'article', 'type_of_material': 'Article', '_id': '4
fc04eb745c1498b0d23da12', 'word_count': 2609, 'score': 1}, {'web_url': 'http
s://www.nytimes.com/1992/05/06/business/anacomp-inc-reports-earnings-for-qtr-to
-march-31.html', 'snippet': '', 'print_page': '20', 'blog': {}, 'source': 'The
New York Times', 'multimedia': [], 'headline': {'main': 'Anacomp Inc. reports e
arnings for Qtr to March 31', 'kicker': None, 'content_kicker': None, 'print_he
adline': None, 'name': None, 'seo': None, 'sub': None}, 'keywords': [{'name':
'subject', 'value': 'COMPANY EARNINGS', 'rank': 0, 'major': None}], 'pub_date':
'1992-05-06T00:00:00Z', 'document_type': 'article', 'news_desk': 'Financial Des
k', 'type_of_material': 'Statistics', '_id': '4fd1b3018eb7c8105d6d690a', 'word_
```

count': 129, 'score': 1}, {'web_url': 'https://query.nytimes.com/gst/abstract.html?res=9503EFDF153DE53ABC4C51DFB4678389669EDE', 'snippet': '', 'print_page': 'S9', 'blog': {}, 'source': 'The New York Times', 'multimedia': [], 'headline': {'main': 'Brooklyn Routs Yeshiva', 'kicker': '1', 'content_kicker': None, 'print_headline': None, 'name': None, 'seo': None, 'sub': None}, 'keywords': [], 'pub_date': '1972-12-24T00:00:00Z', 'document_type': 'article', 'type_of_material': 'Article', '_id': '4fc47bb045c1498b0da03363', 'word_count': 144, 'score': 1}, {'web_url': 'https://query.nytimes.com/gst/abstract.html?res=9F03E1DA1631E63BBC4D51DFB4678389669EDE', 'snippet': 'ALBUQUERQUE, N. M., Dec. 24 -- Holiday drinkers who have drunk too much can get free rides home from now until Jan. 2 through a test program being carried out by taxi companies, university students and a local alcohol safety organization. ', 'print_page': '11', 'blog': {}, 'source': 'The New York Times', 'multimedia': [], 'headline': {'main': 'Albuquerque Program Gives Drinkers a Lift', 'kicker': '1', 'content_kicker': None, 'print_headline': None, 'name': None, 'seo': None, 'sub': None}, 'keywords': [], 'pub_date': '1972-12-25T00:00:00Z', 'document_type': 'article', 'byline': {'original': 'Special to The New York Times', 'person': [{'firstname': None, 'middlename': None, 'lastname': None, 'qualifier': None, 'title': None, 'role': 'reported', 'organization': '', 'rank': 1}], 'organization': None}, 'type_of_material': 'Article', '_id': '4fc47bb045c1498b0da03367', 'word_count': 151, 'score': 1}, {'web_url': 'https://query.nytimes.com/gst/abstract.html?res=9905E6DD153EE03BBC4C51DFB667838F659EDE', 'snippet': '', 'print_page': '1', 'blog': {}, 'source': 'The New York Times', 'multimedia': [], 'headline': {'main': 'Front Page 7 -- No Title', 'kicker': '1', 'content_kicker': None, 'print_headline': None, 'name': None, 'seo': None, 'sub': None}, 'keywords': [], 'pub_date': '1944-10-24T00:00:00Z', 'document_type': 'article', 'type_of_material': 'Front Page', '_id': '4fc21ebf45c1498b0d612b3c', 'word_count': 29, 'score': 1}, {'web_url': 'https://query.nytimes.com/gst/abstract.html?res=9507EFDB1F3AE733A25755C0A96E9C946597D6CF', 'snippet': 'The employers and the unions have lined up in preparation for a long fight in the building war. An indication of the feeling that this will be a real fight is the appointment of a regular Press Committee by the employers. Members of their associat...', 'abstract': "housesmiths won't strike", 'print_page': '1', 'blog': {}, 'source': 'The New York Times', 'multimedia': [], 'headline': {'main': 'UNIONS AND BUILDERS READY FOR LONG FIGHT; None of the Strikers Back - Lock-Out Soon in Effect. 23,000 ALREADY INVOLVED Orders Sent to Every Building Employer Within Twenty-five Miles -- House-smiths Vote Not to Strike.', 'kicker': None, 'content_kicker': None, 'print_headline': None, 'name': None, 'seo': None, 'sub': None}, 'keywords': [{'name': 'glocations', 'value': 'NEW YORK CITY', 'rank': 0, 'major': None}, {'name': 'subject', 'value': 'STRIKES', 'rank': 0, 'major': None}, {'name': 'subject', 'value': 'PENN. MINERS THREATEN', 'rank': 0, 'major': None}, {'name': 'subject', 'value': 'BLDG. TRADES EMP. ASS. MAY LOCK OUT 30,000', 'rank': 0, 'major': None}, {'name': 'subject', 'value': '23,000 MEN INVOLVED', 'rank': 0, 'major': None}], 'pub_date': '1904-08-06T00:00:00Z', 'document_type': 'article', 'type_of_material': 'Front Page', '_id': '4fc04eb745c1498b0d23da17', 'word_count': 883, 'score': 1}], 'meta': {'hits': 15533655, 'offset': 0, 'time': 207}}}

You should see that there are two additional keys `'status'` and `'copyright'` which were not shown in the schema documentation. As with most forms of documentation, it's important to be aware that published schemas may differ somewhat from the actual data, and your code should be able to handle these unexpected differences, within reason.

Loading Specific Data

Looking at the schema, you might be interested in retrieving a specific piece of data, such as the articles' headlines. Notice that this is a key under `'docs'`, which is under `'response'`. So the schema is roughly: data --> 'response' --> 'docs' --> 'headline', something like `data['response']['docs']['headline']`.

Let's see what happens if we try that:

In [19]:
```
data['response']['docs']['headline']
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_180/1495757031.py in <module>
----> 1 data['response']['docs']['headline']

TypeError: list indices must be integers or slices, not str
```

Ok, this error message is saying that somewhere along the way, **we treated something like a dictionary when it was actually a list**. Let's break down that chain of commands to figure out what went wrong.

We are pretty sure that `data['response']` will not cause an error, since we already checked that `data` is type `dict`, and that `'response'` is one of the keys. But what is the type of `data['response']`?

In [ ]:
```
type(data['response'])
```

Ok, that's a dictionary, too. How about `data['response']['docs']`?

In [ ]:
```
type(data['response']['docs'])
```

So, that is the source of the error. We tried to treat this as a dictionary (accessing the value associated with the key `'headline'`) but it's a list!

If you scroll back up to the schema pictured above, this makes sense. The value associated with the `'docs'` key is shown surrounded by `[` and `]`, right before the `{` and `}`, indicating that this is a *list* of dictionaries, not just a dictionary.

You'll run into this kind of distinction repeatedly when working with JSON data. Sometimes values will be nested in unexpected ways, or you'll miss a key detail when you're skimming the schema. What's most important is that you're able to keep going and figure out what went wrong, not that you get it right on the first try!

Now that we know that this is a list, let's extract it and print out some more information about it:

In [21]:
```
docs = data['response']['docs']

print("`docs` is a data structure of type", type(docs))
print("It contains", len(docs), "elements")
print("The first element is type", type(docs[0]))
```

```
`docs` is a data structure of type <class 'list'>
It contains 9 elements
The first element is type <class 'dict'>
```

This confirms what we expected. Now we can loop over that list of dictionaries and print the values associated with the `'headline'` keys:

In [22]:
```
for doc in docs:
    print(doc['headline'])
```

```
{'main': "HIGGINS, SPENT $22,189.53.; Governor-Elect's Election Expenses -- Har
rison $9,220.28.", 'kicker': None, 'content_kicker': None, 'print_headline': No
ne, 'name': None, 'seo': None, 'sub': None}
{'main': 'GARDEN BOUTS CANCELED; Mauriello Says He Could Not Be Ready on Nov.
3', 'kicker': '1', 'content_kicker': None, 'print_headline': None, 'name': Non
e, 'seo': None, 'sub': None}
{'main': 'Stock Drop Is Biggest in 2 Months--Margin Rise Held Factor in Lightes
t Trading of 1955', 'kicker': '1', 'content_kicker': None, 'print_headline': No
ne, 'name': None, 'seo': None, 'sub': None}
{'main': 'MUSIC OF THE WEEK', 'kicker': None, 'content_kicker': None, 'print_he
adline': None, 'name': None, 'seo': None, 'sub': None}
{'main': 'Anacomp Inc. reports earnings for Qtr to March 31', 'kicker': None,
'content_kicker': None, 'print_headline': None, 'name': None, 'seo': None, 'su
b': None}
{'main': 'Brooklyn Routs Yeshiva', 'kicker': '1', 'content_kicker': None, 'prin
t_headline': None, 'name': None, 'seo': None, 'sub': None}
{'main': 'Albuquerque Program Gives Drinkers a Lift', 'kicker': '1', 'content_k
icker': None, 'print_headline': None, 'name': None, 'seo': None, 'sub': None}
{'main': 'Front Page 7 -- No Title', 'kicker': '1', 'content_kicker': None, 'pr
int_headline': None, 'name': None, 'seo': None, 'sub': None}
{'main': 'UNIONS AND BUILDERS READY FOR LONG FIGHT; None of the Strikers Back -
Lock-Out Soon in Effect. 23,000 ALREADY INVOLVED Orders Sent to Every Building
Employer Within Twenty-five Miles -- House-smiths Vote Not to Strike.', 'kicke
r': None, 'content_kicker': None, 'print_headline': None, 'name': None, 'seo':
None, 'sub': None}
```

Or if you want to just print the main headlines themselves:

In [23]:
```
for doc in docs:
    print(doc['headline']['main'])
```

```
HIGGINS, SPENT $22,189.53.; Governor-Elect's Election Expenses -- Harrison $9,2
20.28.
GARDEN BOUTS CANCELED; Mauriello Says He Could Not Be Ready on Nov. 3
Stock Drop Is Biggest in 2 Months--Margin Rise Held Factor in Lightest Trading
of 1955
MUSIC OF THE WEEK
Anacomp Inc. reports earnings for Qtr to March 31
Brooklyn Routs Yeshiva
Albuquerque Program Gives Drinkers a Lift
Front Page 7 -- No Title
UNIONS AND BUILDERS READY FOR LONG FIGHT; None of the Strikers Back - Lock-Out
Soon in Effect. 23,000 ALREADY INVOLVED Orders Sent to Every Building Employer
Within Twenty-five Miles -- House-smiths Vote Not to Strike.
```

Flattening Data (i.e. Breaking Out Nested Data)

Let's say we want to create a list of dictionaries containing information about the documents contained in this JSON. It should contain the publication date (value associated with `pub_date` key), word count (value associated with `word_count` key), and both the `'main'` and `'kicker'` associated with the `headline` key. This list should be called `doc_info_list` and should look something like this:

```
[
    {
        'headline_main': "HIGGINS, SPENT $22,189.53.; Governor-Elect's E
lection Expenses -- Harrison $9,220.28.",
        'headline_kicker': None,
        'pub_date': '1904-11-17T00:00:00Z',
        'word_count': 213
    },
    {
        'headline_main': 'GARDEN BOUTS CANCELED; Mauriello Says He Could
Not Be Ready on Nov. 3',
        'headline_kicker': '1',
        'pub_date': '1944-10-23T00:00:00Z',
        'word_count': 149
    },
    ...
]
```

The tricky part is, each dictionary needs to be "flat", meaning that each key is associated with a single string or number value, not a deeper data structure. So we need to flatten the nested `headline` dictionary.

It's also conventional when flattening data to make a compound name for the newly-created keys. So, let's call the new keys `headline_main` and `headline_kicker`.

Recall the structure of a `headline` dictionary:

In [24]:
```
docs[2]['headline']
```

Out[24]:
```
{'main': 'Stock Drop Is Biggest in 2 Months--Margin Rise Held Factor in Lightes
t Trading of 1955',
 'kicker': '1',
 'content_kicker': None,
 'print_headline': None,
 'name': None,
 'seo': None,
 'sub': None}
```

So, first let's write a function that takes in that complete dictionary, and returns a copy with only the `'main'` and `'kicker'` keys and values, now labeled `'headline_main'` and `'headline_kicker'`:

In [25]:
```
def extract_headline_info(headline_dict):
    result = {}
    result['headline_main'] = headline_dict['main']
    result['headline_kicker'] = headline_dict['kicker']
    return result
```

Then we test it out:

In [26]:
```
extract_headline_info(docs[2]['headline'])
```

Out[26]:
```
{'headline_main': 'Stock Drop Is Biggest in 2 Months--Margin Rise Held Factor i
n Lightest Trading of 1955',
 'headline_kicker': '1'}
```

In [27]:
```
extract_headline_info(docs[0]['headline'])
```

Out[27]:
```
{'headline_main': "HIGGINS, SPENT $22,189.53.; Governor-Elect's Election Expens
es -- Harrison $9,220.28.",
 'headline_kicker': None}
```

Now let's write another function that calls that function, then adds the `pub_date` and `word_count` keys and values:

In [28]:
```
def extract_doc_info(doc):
    info = extract_headline_info(doc['headline'])
    info['pub_date'] = doc['pub_date']
    info['word_count'] = doc['word_count']
    return info
```

Again, testing it out on a couple examples:

In [29]:
```
extract_doc_info(docs[2])
```

Out[29]:
```
{'headline_main': 'Stock Drop Is Biggest in 2 Months--Margin Rise Held Factor i
n Lightest Trading of 1955',
 'headline_kicker': '1',
 'pub_date': '1955-05-15T00:00:00Z',
 'word_count': 823}
```

In [30]:
```
extract_doc_info(docs[0])
```

Out[30]:
```
{'headline_main': "HIGGINS, SPENT $22,189.53.; Governor-Elect's Election Expens
es -- Harrison $9,220.28.",
 'headline_kicker': None,
 'pub_date': '1904-11-17T00:00:00Z',
 'word_count': 213}
```

Now we can loop over the full list and create `doc_info_list` :

In [32]:
```
doc_info_list = [extract_doc_info(doc) for doc in docs]
doc_info_list
```

Out[32]:
```
[{'headline_main': "HIGGINS, SPENT $22,189.53.; Governor-Elect's Election Expen
ses -- Harrison $9,220.28.",
  'headline_kicker': None,
  'pub_date': '1904-11-17T00:00:00Z',
  'word_count': 213},
 {'headline_main': 'GARDEN BOUTS CANCELED; Mauriello Says He Could Not Be Ready
on Nov. 3',
  'headline_kicker': '1',
  'pub_date': '1944-10-23T00:00:00Z',
  'word_count': 149},
 {'headline_main': 'Stock Drop Is Biggest in 2 Months--Margin Rise Held Factor
in Lightest Trading of 1955',
  'headline_kicker': '1',
  'pub_date': '1955-05-15T00:00:00Z',
  'word_count': 823},
 {'headline_main': 'MUSIC OF THE WEEK',
  'headline_kicker': None,
  'pub_date': '1904-11-06T00:00:00Z',
  'word_count': 2609},
 {'headline_main': 'Anacomp Inc. reports earnings for Qtr to March 31',
  'headline_kicker': None,
  'pub_date': '1992-05-06T00:00:00Z',
  'word_count': 129},
 {'headline_main': 'Brooklyn Routs Yeshiva',
  'headline_kicker': '1',
  'pub_date': '1972-12-24T00:00:00Z',
  'word_count': 144},
 {'headline_main': 'Albuquerque Program Gives Drinkers a Lift',
  'headline_kicker': '1',
  'pub_date': '1972-12-25T00:00:00Z',
  'word_count': 151},
 {'headline_main': 'Front Page 7 -- No Title',
  'headline_kicker': '1',
  'pub_date': '1944-10-24T00:00:00Z',
  'word_count': 29},
 {'headline_main': 'UNIONS AND BUILDERS READY FOR LONG FIGHT; None of the Strik
ers Back - Lock-Out Soon in Effect. 23,000 ALREADY INVOLVED Orders Sent to Ever
y Building Employer Within Twenty-five Miles -- House-smiths Vote Not to Strik
e.',
  'headline_kicker': None,
  'pub_date': '1904-08-06T00:00:00Z',
  'word_count': 883}]
```

Thus we have successfully flattened the required data!

Transforming JSON to Alternative Formats

Viewing the Raw Dataset in Pandas

You've also previously started to take a look at how to transform JSON to DataFrames. Investigating the schema, a good option for this could again be the value associated with the `'docs'` key. While this still has nested data itself, it's often easier to load the entire contents as a DataFrame for viewing and then use additional functions to break apart the internally nested data from there.

So, first we will display the full information associated with the `'docs'` key:

In [33]:
```
import pandas as pd
pd.DataFrame(data['response']['docs'])
```

Out[33]:

	web_url	snippet	abstract	print_page	blog	s
0	https://query.nytimes.com/gst/abstract.html?re...	Spent $22,200	Spent $22,200	2	{}	
1	https://query.nytimes.com/gst/abstract.html?re...		NaN	15	{}	
2	https://query.nytimes.com/gst/abstract.html?re...	Stock prices last week, on the lightest volume...	NaN	F1	{}	
3	https://query.nytimes.com/gst/abstract.html?re...	The first public rehearsal and concert of the ...	Healy, Michael, will suit	20	{}	
4	https://www.nytimes.com/1992/05/06/business/an...		NaN	20	{}	
5	https://query.nytimes.com/gst/abstract.html?re...		NaN	S9	{}	
6	https://query.nytimes.com/gst/abstract.html?re...	ALBUQUERQUE, N. M., Dec. 24 -- Holiday drinker...	NaN	11	{}	
7	https://query.nytimes.com/gst/abstract.html?re...		NaN	1	{}	
8	https://query.nytimes.com/gst/abstract.html?re...	The employers and the unions have lined up in ...	housesmiths won't strike	1	{}	

Note that because the value associated with the `'headline'` key is a dictionary, it is displayed in this crowded, messy way within the DataFrame, including `{` and `'` characters.

Viewing the Flattened Info List

Because `doc_info_list` is already flattened so the value associated with each key is just a number or string, it looks much neater when loaded into pandas:

In [34]:
```
pd.DataFrame(doc_info_list)
```

Out[34]:

	headline_main	headline_kicker	pub_date	word_count
0	HIGGINS, SPENT $22,189.53.; Governor-Elect's E...	None	1904-11-17T00:00:00Z	213
1	GARDEN BOUTS CANCELED; Mauriello Says He Could...	1	1944-10-23T00:00:00Z	149
2	Stock Drop Is Biggest in 2 Months--Margin Rise...	1	1955-05-15T00:00:00Z	823
3	MUSIC OF THE WEEK	None	1904-11-06T00:00:00Z	2609
4	Anacomp Inc. reports earnings for Qtr to March 31	None	1992-05-06T00:00:00Z	129
5	Brooklyn Routs Yeshiva	1	1972-12-24T00:00:00Z	144
6	Albuquerque Program Gives Drinkers a Lift	1	1972-12-25T00:00:00Z	151
7	Front Page 7 -- No Title	1	1944-10-24T00:00:00Z	29
8	UNIONS AND BUILDERS READY FOR LONG FIGHT; None...	None	1904-08-06T00:00:00Z	883

We could also re-create this from the raw data using pandas rather than base Python:

In [35]:
```
# Create dataframe of raw docs info
df = pd.DataFrame(data['response']['docs'])

# Make new headline_main and headline_kicker columns
df['headline_main'] = df['headline'].apply(lambda headline_dict: headline_dict['m
df['headline_kicker'] = df['headline'].apply(lambda headline_dict: headline_dict[

# Subset to only the relevant columns
df = df[['headline_main', 'headline_kicker', 'pub_date', 'word_count']]
df
```

Out[35]:

	headline_main	headline_kicker	pub_date	word_count
0	HIGGINS, SPENT $22,189.53.; Governor-Elect's E...	None	1904-11-17T00:00:00Z	213
1	GARDEN BOUTS CANCELED; Mauriello Says He Could...	1	1944-10-23T00:00:00Z	149
2	Stock Drop Is Biggest in 2 Months--Margin Rise...	1	1955-05-15T00:00:00Z	823
3	MUSIC OF THE WEEK	None	1904-11-06T00:00:00Z	2609
4	Anacomp Inc. reports earnings for Qtr to March 31	None	1992-05-06T00:00:00Z	129
5	Brooklyn Routs Yeshiva	1	1972-12-24T00:00:00Z	144
6	Albuquerque Program Gives Drinkers a Lift	1	1972-12-25T00:00:00Z	151
7	Front Page 7 -- No Title	1	1944-10-24T00:00:00Z	29
8	UNIONS AND BUILDERS READY FOR LONG FIGHT; None...	None	1904-08-06T00:00:00Z	883

Wahoo! This is a good general strategy for transforming nested JSON: create a DataFrame and then break out nested features into their own column features.

Outputting to JSON

Finally, take a look at how you can write data back to JSON. Like loading, you first open a file (this time in write mode) and use the json package to interact with that file object. Only instead of `json.load` to load the contents of the file into a Python object, you call `json.dump` to write the contents of the Python object into the file.

In [36]:
```
with open('doc_info_list.json', 'w') as f:
    json.dump(doc_info_list, f)
```

Then if we want to load that cleaned dataset for future use, we can open that new file:

In [37]:
```
with open('doc_info_list.json') as f:
    doc_info_list_from_disk = json.load(f)
```

The new file should contain identical information to the original Python variable:

In [38]:
```
doc_info_list_from_disk == doc_info_list
```

Out[38]:
```
True
```

Summary

There you have it! In this, you practiced using JSON some more, this time interpreting an example schema diagram in order to retrieve information. You also looked at a general procedure for transforming nested data to pandas DataFrames (create a DataFrame, and then break apart nested data using lambda functions to create additional columns). Finally, you also took a brief look at saving data to JSON files.