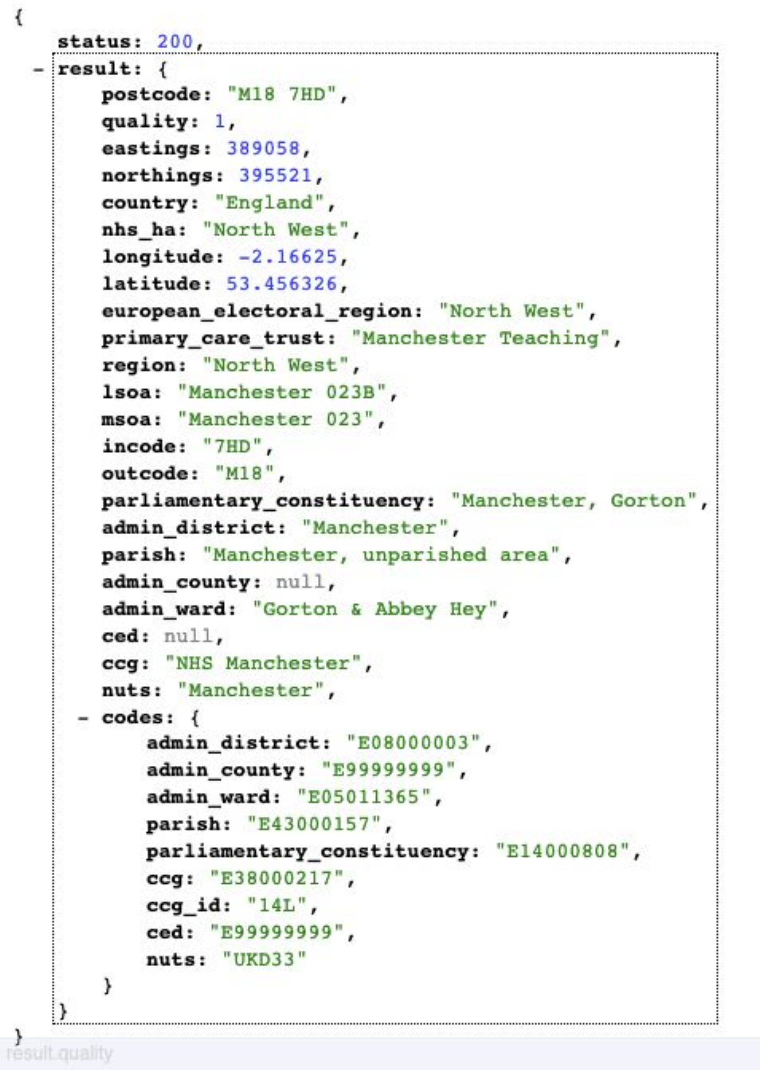
JSON

Paul Bradshaw

What is JSON?

- A language to describe data
- Same as a dictionary object: { key : value }
- Can be in a list, e.g. [{“name”:“Paul”}, {“name”:“Joy”}]
- Can have sub-branches

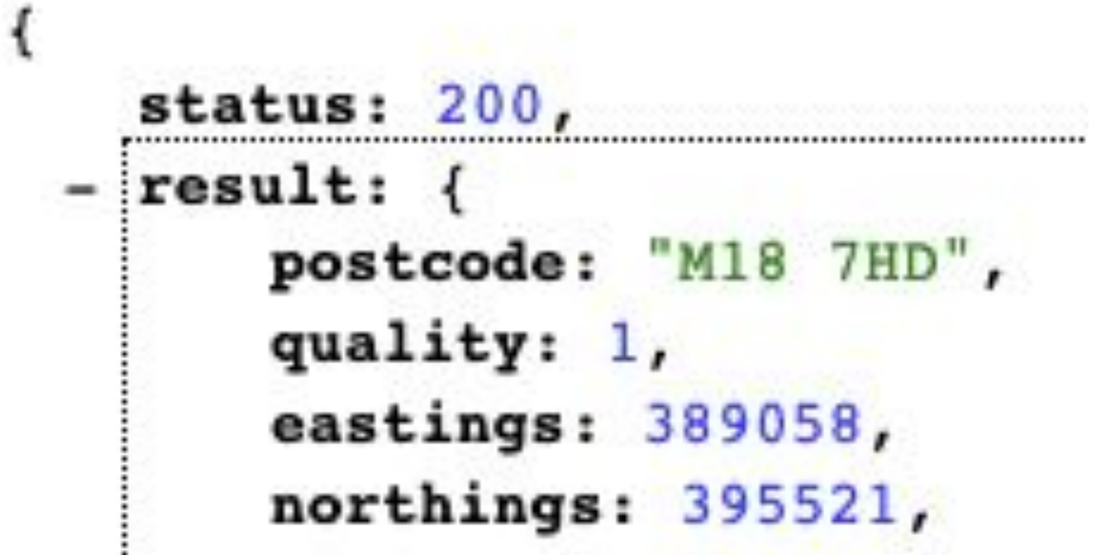


```
{
    status: 200,
  - result: {
        postcode: "M18 7HD",
        quality: 1,
        eastings: 389058,
        northings: 395521,
        country: "England",
        nhs_ha: "North West",
        longitude: -2.16625,
        latitude: 53.456326,
        european_electoral_region: "North West",
        primary_care_trust: "Manchester Teaching",
        region: "North West",
        lsoa: "Manchester 023B",
        msoa: "Manchester 023",
        incode: "7HD",
        outcode: "M18",
        parliamentary_constituency: "Manchester, Gorton",
        admin_district: "Manchester",
        parish: "Manchester, unparished area",
        admin_county: null,
        admin_ward: "Gorton & Abbey Hey",
        ced: null,
        ccg: "NHS Manchester",
        nuts: "Manchester",
      - codes: {
            admin_district: "E08000003",
            admin_county: "E99999999",
            admin_ward: "E05011365",
            parish: "E43000157",
            parliamentary_constituency: "E14000808",
            ccg: "E38000217",
            ccg_id: "14L",
            ced: "E99999999",
            nuts: "UKD33"
        }
    }
}
result.quality
```

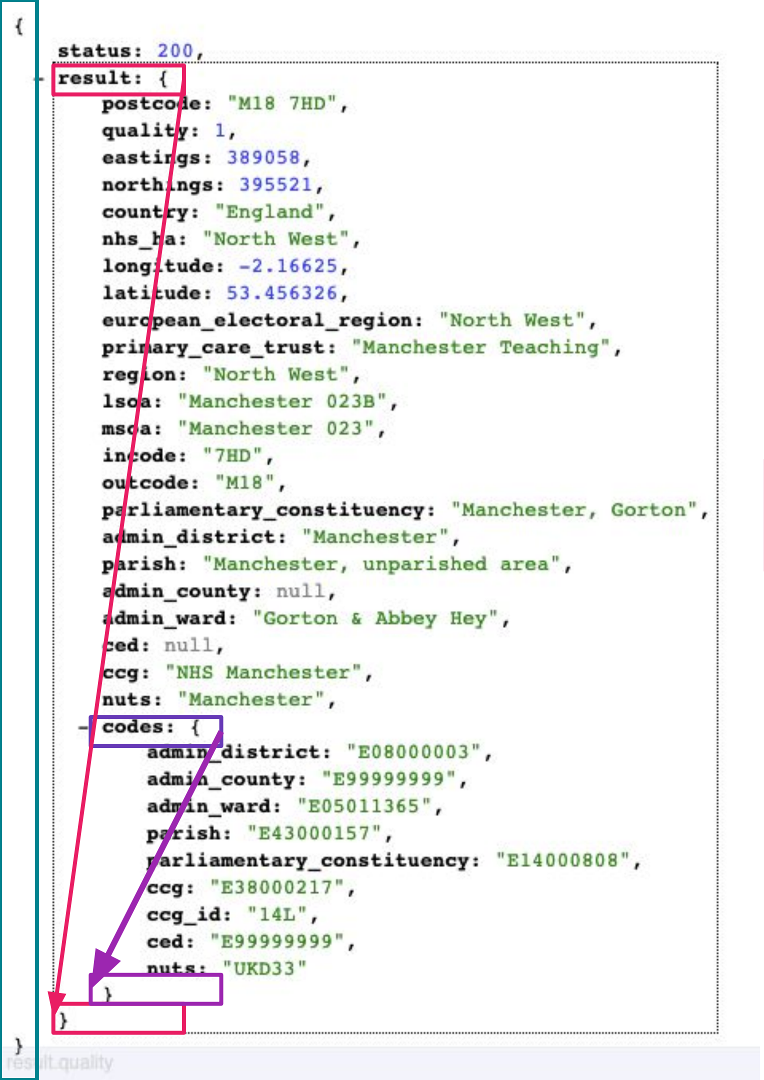
A ‘JSON object’

Spot the curly brackets.

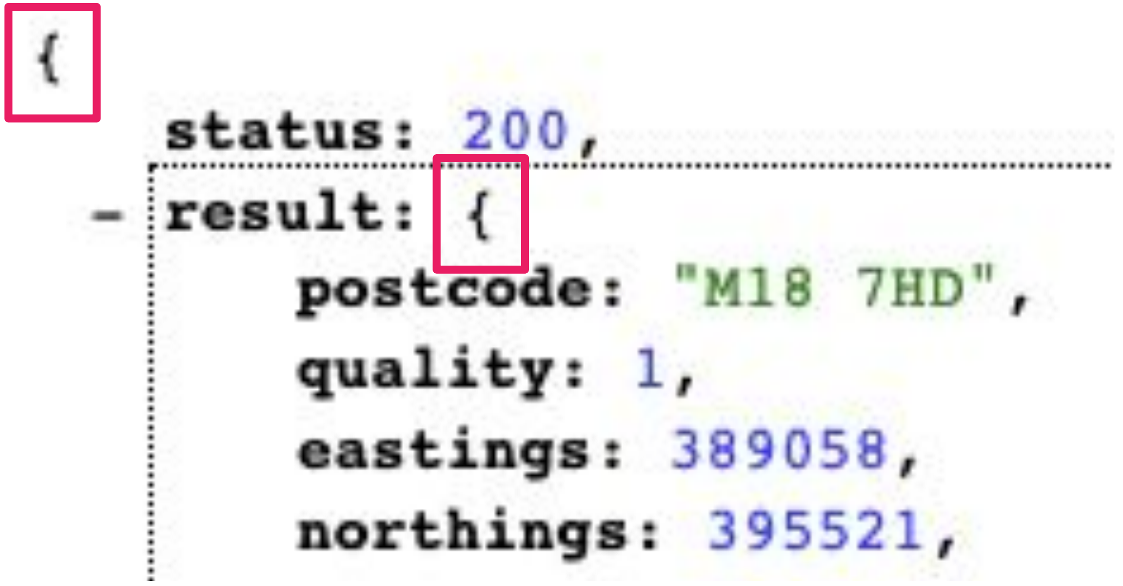
Spot the colons, commas

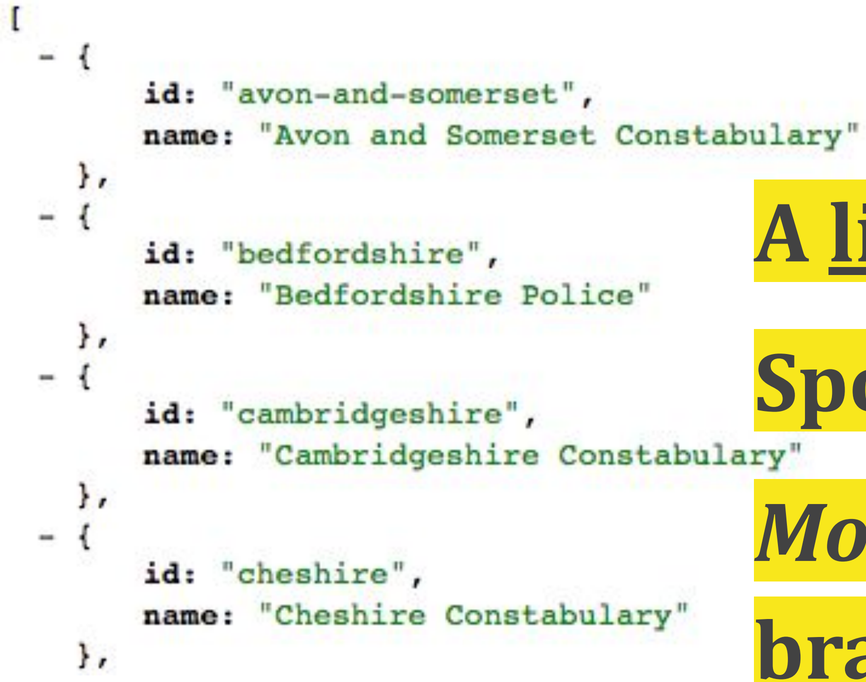


```
{
    status: 200,
  - result: {
        postcode: "M18 7HD",
        quality: 1,
        eastings: 389058,
        northings: 395521,
```


```
{
  status: 200,
- result: {
      postcode: "M18 7HD",
      quality: 1,
      eastings: 389058,
      northings: 395521,
      country: "England",
      nhs_ha: "North West",
      longitude: -2.16625,
      latitude: 53.456326,
      european_electoral_region: "North West",
      primary_care_trust: "Manchester Teaching",
      region: "North West",
      lsoa: "Manchester 023B",
      msoa: "Manchester 023",
      incode: "7HD",
      outcode: "M18",
      parliamentary_constituency: "Manchester, Gorton",
      admin_district: "Manchester",
      parish: "Manchester, unparished area",
      admin_county: null,
      admin_ward: "Gorton & Abbey Hey",
      ced: null,
      ccg: "NHS Manchester",
      nuts: "Manchester",
    - codes: {
          admin_district: "E08000003",
          admin_county: "E99999999",
          admin_ward: "E05011365",
          parish: "E43000157",
          parliamentary_constituency: "E14000808",
          ccg: "E38000217",
          ccg_id: "14L",
          ced: "E99999999",
          nuts: "UKD33"
      }
  }
}
result.quality
```

Spot the nested curly brackets (brackets within brackets)


```
{
  status: 200,
- result: {
      postcode: "M18 7HD",
      quality: 1,
      eastings: 389058,
      northings: 395521,
```



```
[
  - {
        id: "avon-and-somerset",
        name: "Avon and Somerset Constabulary"
    },
  - {
        id: "bedfordshire",
        name: "Bedfordshire Police"
    },
  - {
        id: "cambridgeshire",
        name: "Cambridgeshire Constabulary"
    },
  - {
        id: "cheshire",
        name: "Cheshire Constabulary"
    },
```

A list of JSON objects:

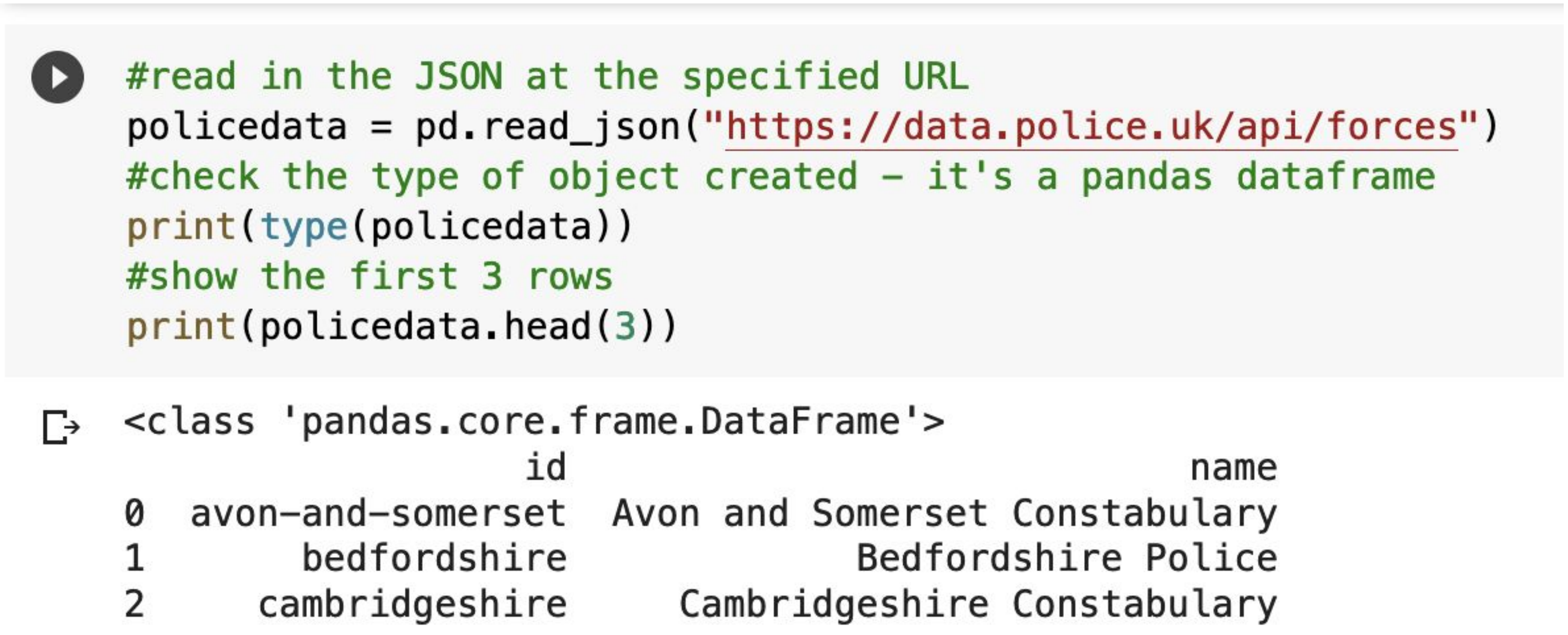
Spot the square bracket

***More than one set* of curly brackets**

Spot the commas *between* each set

Use pandas to import it into Python

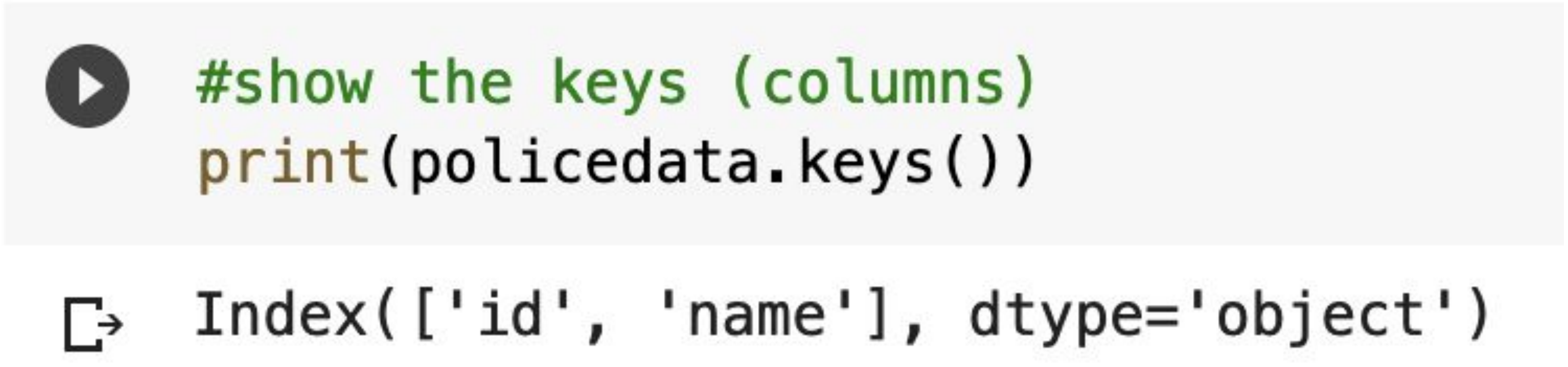
- Use **pd.read_json( )** to read the JSON URL into a pandas object
- Use **.keys( )** to show the keys of an object



```python
#read in the JSON at the specified URL
policedata = pd.read_json("https://data.police.uk/api/forces")
#check the type of object created - it's a pandas dataframe
print(type(policedata))
#show the first 3 rows
print(policedata.head(3))
```

```
<class 'pandas.core.frame.DataFrame'>
                  id                            name
0  avon-and-somerset  Avon and Somerset Constabulary
1        bedfordshire             Bedfordshire Police
2      cambridgeshire     Cambridgeshire Constabulary
```

https://github.com/paulbradshaw/pythonin12parts/blob/main/part9/12pandas_api.ipynb



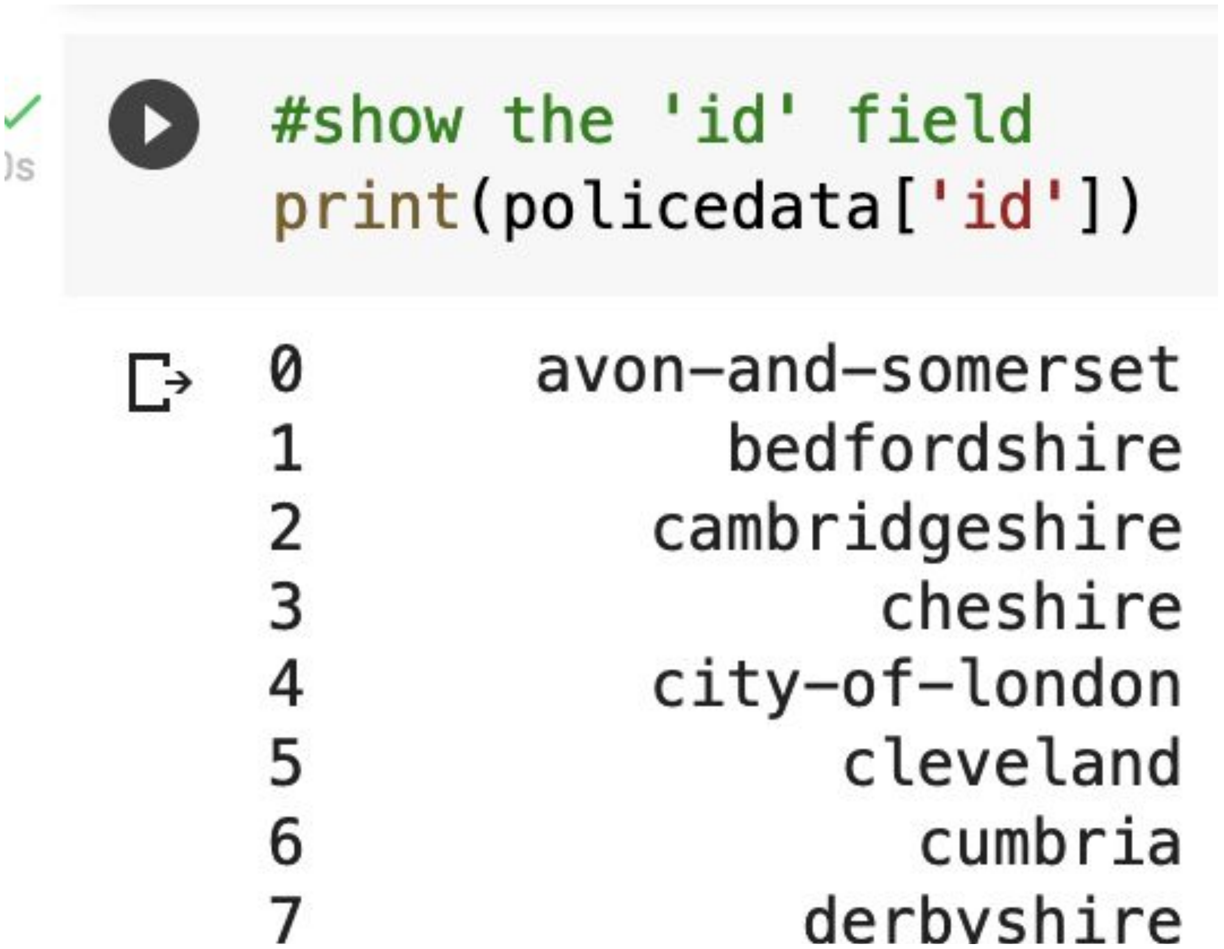
```
#show the keys (columns)
print(policedata.keys())
```

```
Index(['id', 'name'], dtype='object')
```

https://github.com/paulbradshaw/pythonin12parts/blob/main/part9/12pandas_api.ipynb

Drilling down into JSON

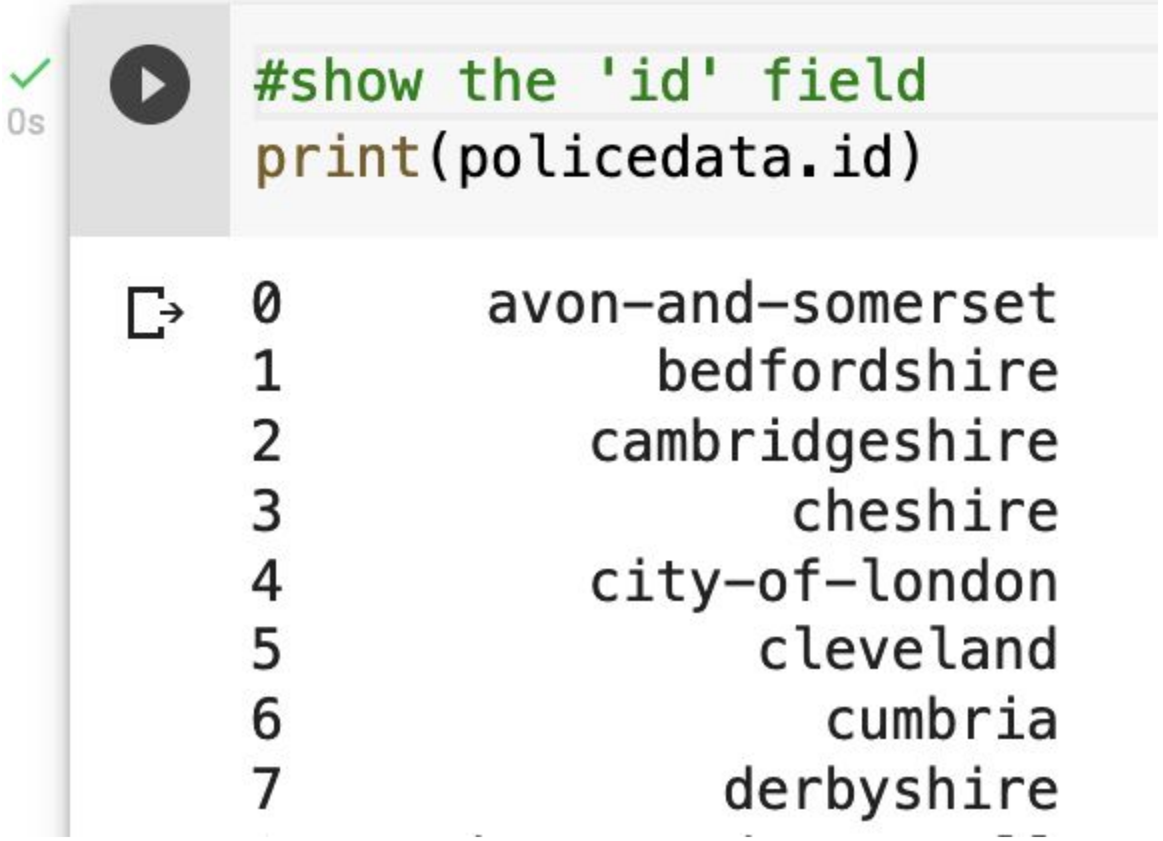
- Put the name of the key in square brackets to extract the value, e.g. **policedata[‘id’]**
- Dot-notation can also be used if there’s no space, e.g. **policedata.id**



```
#show the 'id' field
print(policedata['id'])
```

```
0        avon-and-somerset
1              bedfordshire
2            cambridgeshire
3                  cheshire
4            city-of-london
5                 cleveland
6                   cumbria
7                derbyshire
```

https://github.com/paulbradshaw/pythonin12parts/blob/main/part9/12pandas_api.ipynb



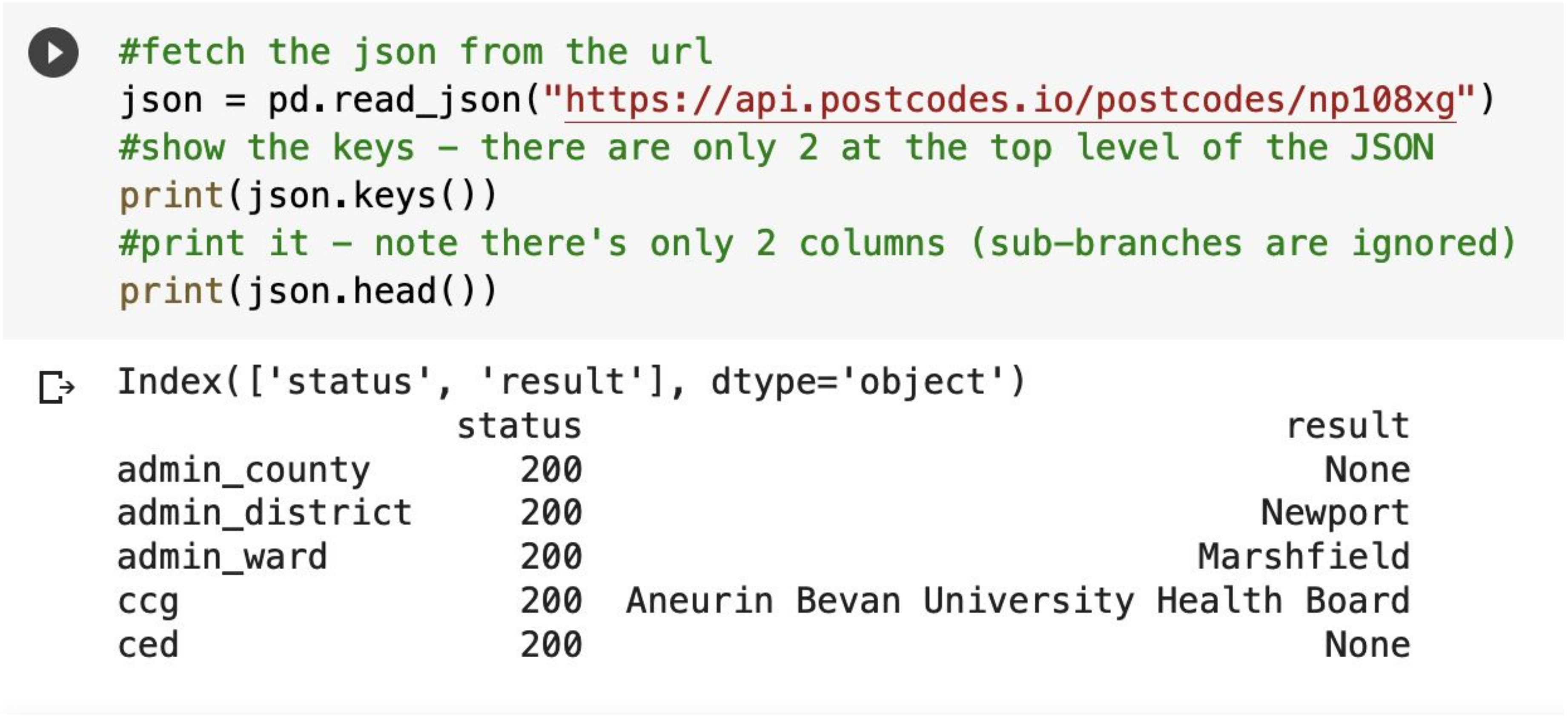
```
#show the 'id' field
print(policedata.id)
```

```
0       avon-and-somerset
1             bedfordshire
2           cambridgeshire
3                 cheshire
4           city-of-london
5                cleveland
6                  cumbria
7               derbyshire
```

https://github.com/paulbradshaw/pythonin12parts/blob/main/part9/12pandas_api.ipynb

Drilling further into JSON

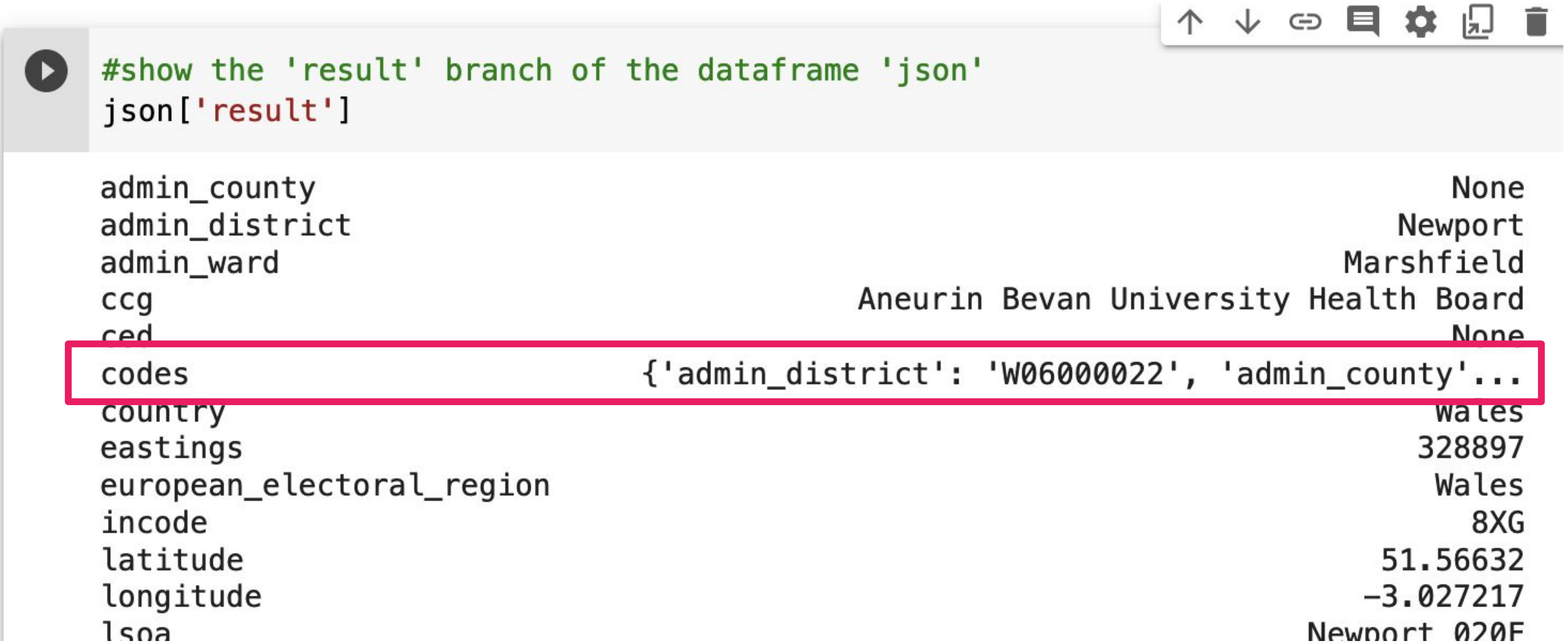
- When JSON has sub-branches, *add* sub-branch keys to the end to drill down further, e.g. **postcodejson[‘result’][‘ccg’]**
- You can also add **.keys( )** to see the keys in *that* branch



```
#fetch the json from the url
json = pd.read_json("https://api.postcodes.io/postcodes/np108xg")
#show the keys - there are only 2 at the top level of the JSON
print(json.keys())
#print it - note there's only 2 columns (sub-branches are ignored)
print(json.head())
```

```
Index(['status', 'result'], dtype='object')
                status                                  result
admin_county       200                                    None
admin_district     200                                 Newport
admin_ward         200                              Marshfield
ccg                200  Aneurin Bevan University Health Board
ced                200                                    None
```

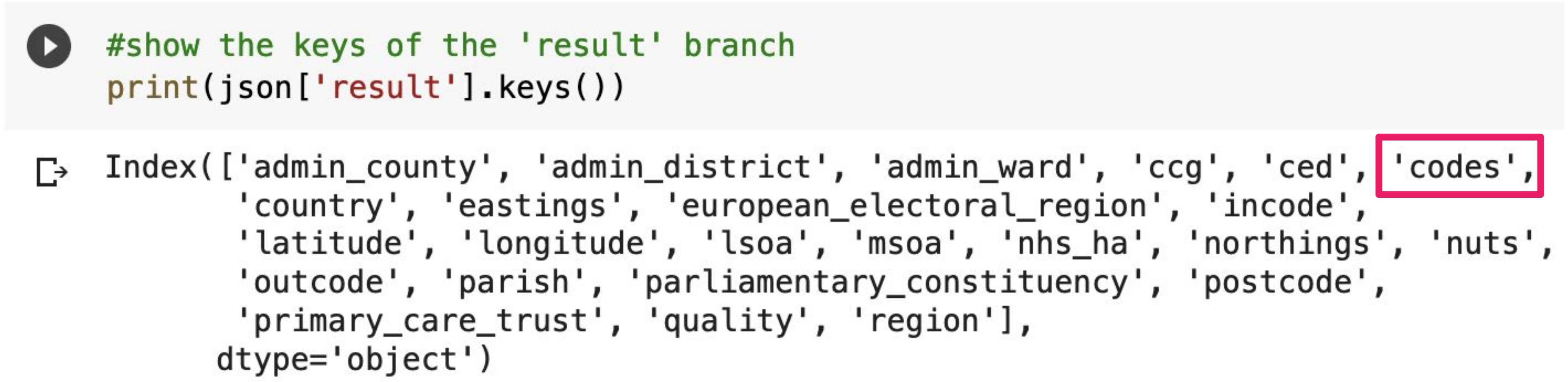
https://github.com/paulbradshaw/pythonin12parts/blob/main/part9/12pandas_api.ipynb



```
#show the 'result' branch of the dataframe 'json'
json['result']
```

```
admin_county                                                None
admin_district                                           Newport
admin_ward                                           Marshfield
ccg                           Aneurin Bevan University Health Board
ced                                                         None
codes                {'admin_district': 'W06000022', 'admin_county'...
country                                                    Wales
eastings                                                  328897
european_electoral_region                                  Wales
incode                                                       8XG
latitude                                                51.56632
longitude                                              -3.027217
lsoa                                                Newport 020E
```

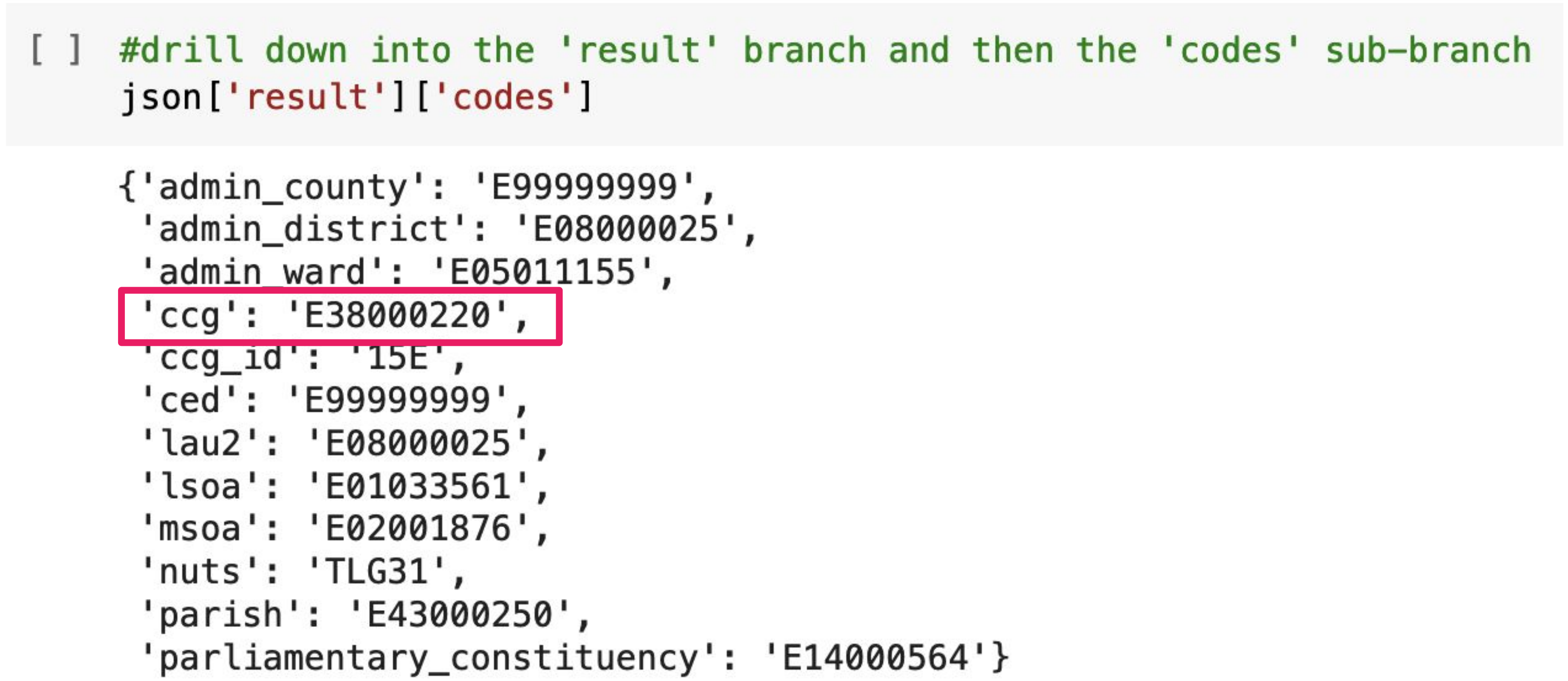
https://github.com/paulbradshaw/pythonin12parts/blob/main/part9/12pandas_api.ipynb



```
#show the keys of the 'result' branch
print(json['result'].keys())
```

```
Index(['admin_county', 'admin_district', 'admin_ward', 'ccg', 'ced', 'codes',
       'country', 'eastings', 'european_electoral_region', 'incode',
       'latitude', 'longitude', 'lsoa', 'msoa', 'nhs_ha', 'northings', 'nuts',
       'outcode', 'parish', 'parliamentary_constituency', 'postcode',
       'primary_care_trust', 'quality', 'region'],
      dtype='object')
```

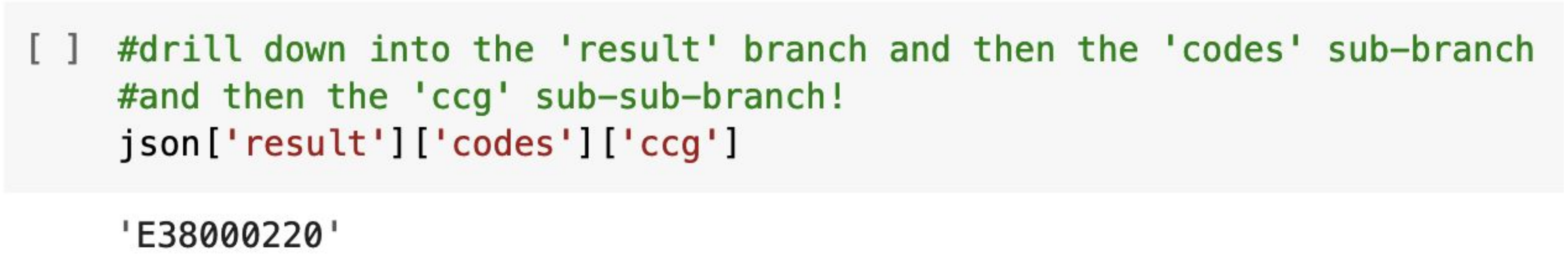
https://github.com/paulbradshaw/pythonin12parts/blob/main/part9/12pandas_api.ipynb



```
[ ]  #drill down into the 'result' branch and then the 'codes' sub-branch
     json['result']['codes']
```

```
{'admin_county': 'E99999999',
 'admin_district': 'E08000025',
 'admin_ward': 'E05011155',
 'ccg': 'E38000220',
 'ccg_id': '15E',
 'ced': 'E99999999',
 'lau2': 'E08000025',
 'lsoa': 'E01033561',
 'msoa': 'E02001876',
 'nuts': 'TLG31',
 'parish': 'E43000250',
 'parliamentary_constituency': 'E14000564'}
```

https://github.com/paulbradshaw/pythonin12parts/blob/main/part9/12pandas_api.ipynb



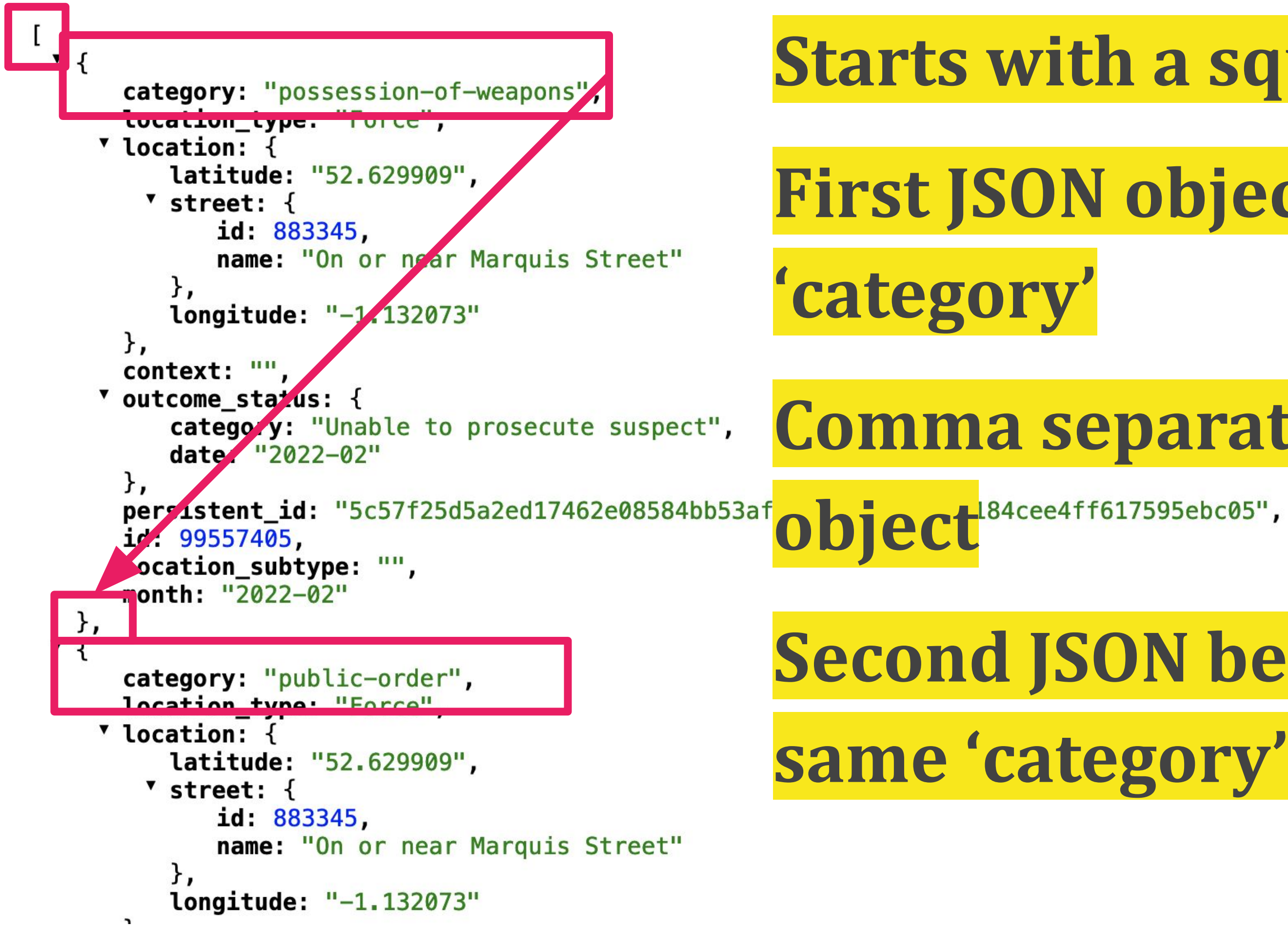
```
[ ]  #drill down into the 'result' branch and then the 'codes' sub-branch
     #and then the 'ccg' sub-sub-branch!
     json['result']['codes']['ccg']

     'E38000220'
```

https://github.com/paulbradshaw/pythonin12parts/blob/main/part9/12pandas_api.ipynb

[Lists] of {JSON}

- Lots of APIs return not just one JSON object, but a **list** of them (one for each row of data)
- Look for a square bracket at the start, and commas between each JSON object



```
[
▼{
    category: "possession-of-weapons",
    location_type: "Force",
  ▼ location: {
        latitude: "52.629909",
      ▼ street: {
            id: 883345,
            name: "On or near Marquis Street"
        },
        longitude: "-1.132073"
    },
    context: "",
  ▼ outcome_status: {
        category: "Unable to prosecute suspect",
        date: "2022-02"
    },
    persistent_id: "5c57f25d5a2ed17462e08584bb53af[illegible]184cee4ff617595ebc05",
    id: 99557405,
    location_subtype: "",
    month: "2022-02"
},
▼{
    category: "public-order",
    location_type: "Force",
  ▼ location: {
        latitude: "52.629909",
      ▼ street: {
            id: 883345,
            name: "On or near Marquis Street"
        },
        longitude: "-1.132073"
```

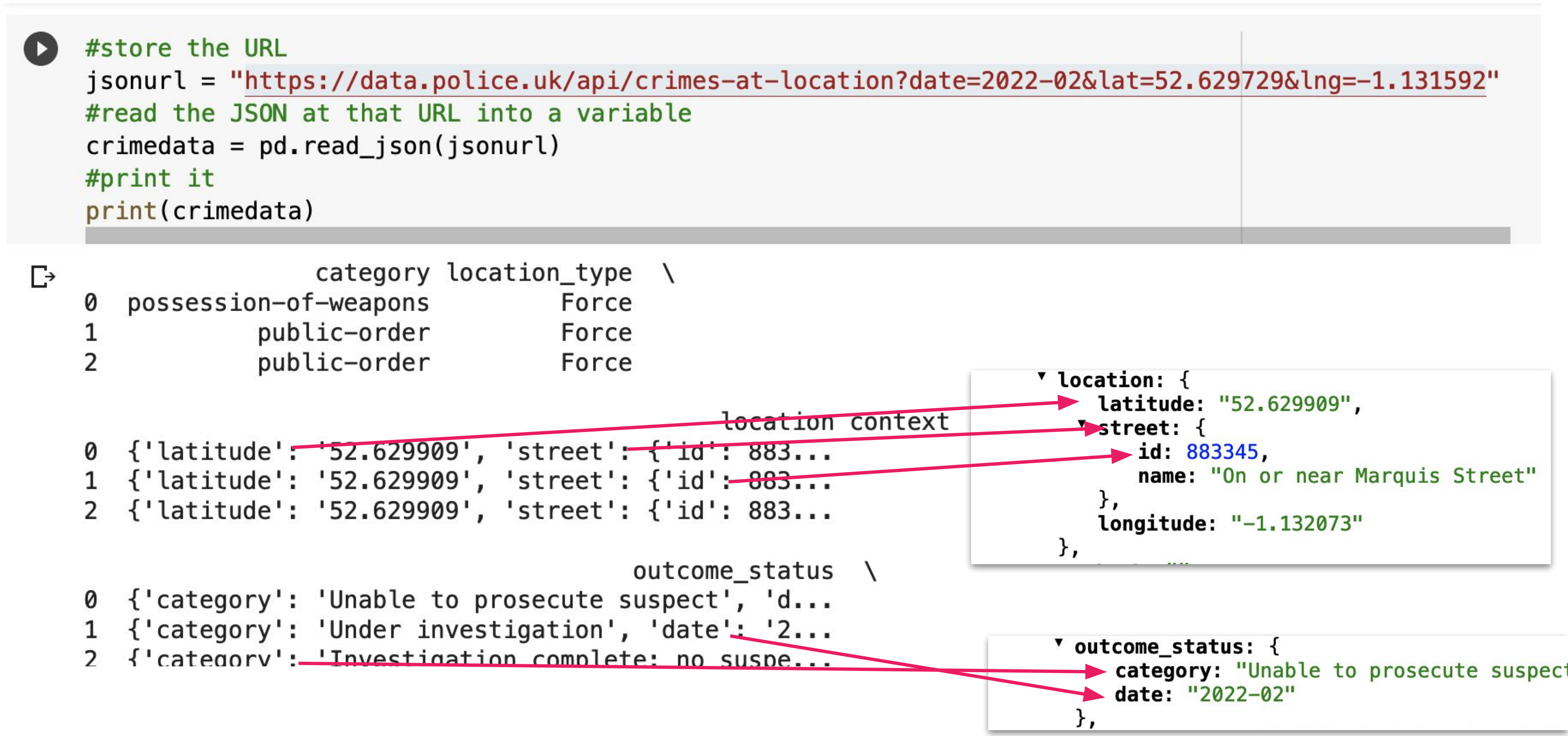
Starts with a square bracket

First JSON object, first key is 'category'

Comma separates each JSON object

Second JSON begins with the same 'category' key

https://data.police.uk/api/crimes-at-location?date=2022-02&lat=52.629729&lng=-1.131592



```python
#store the URL
jsonurl = "https://data.police.uk/api/crimes-at-location?date=2022-02&lat=52.629729&lng=-1.131592"
#read the JSON at that URL into a variable
crimedata = pd.read_json(jsonurl)
#print it
print(crimedata)
```

```
                  category location_type  \
0  possession-of-weapons         Force
1           public-order         Force
2           public-order         Force

                                        location  context
0  {'latitude': '52.629909', 'street': {'id': 883...
1  {'latitude': '52.629909', 'street': {'id': 883...
2  {'latitude': '52.629909', 'street': {'id': 883...

                                    outcome_status  \
0  {'category': 'Unable to prosecute suspect', 'd...
1  {'category': 'Under investigation', 'date': '2...
2  {'category': 'Investigation complete; no suspe...
```

```
▾ location: {
      latitude: "52.629909",
    ▾ street: {
          id: 883345,
          name: "On or near Marquis Street"
      },
      longitude: "-1.132073"
  },
```

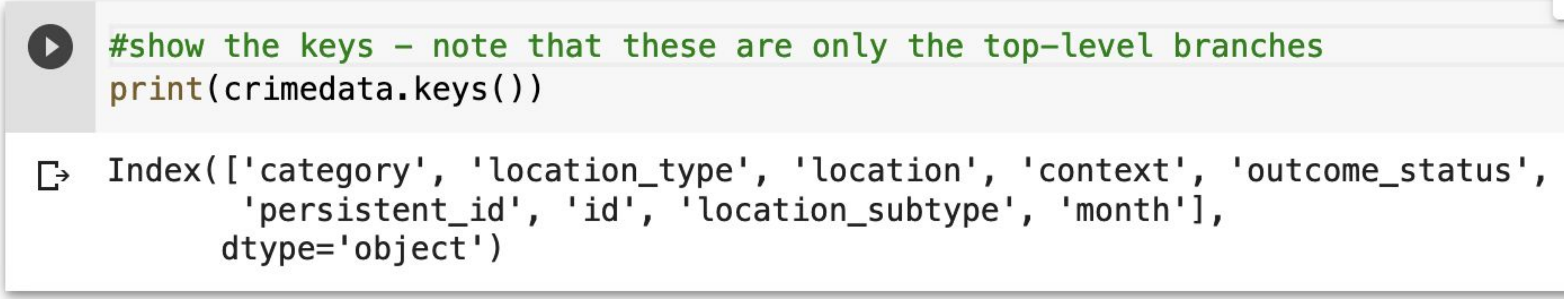
```
▾ outcome_status: {
      category: "Unable to prosecute suspect
      date: "2022-02"
  },
```

https://github.com/paulbradshaw/pythonin12parts/blob/main/part9/12pandas_api.ipynb

Keys & indices in JSON lists

- With data imported from lists of JSON, you may need to combine keys **and** an index, e.g. **jsondata[‘category’][0][‘location’]**

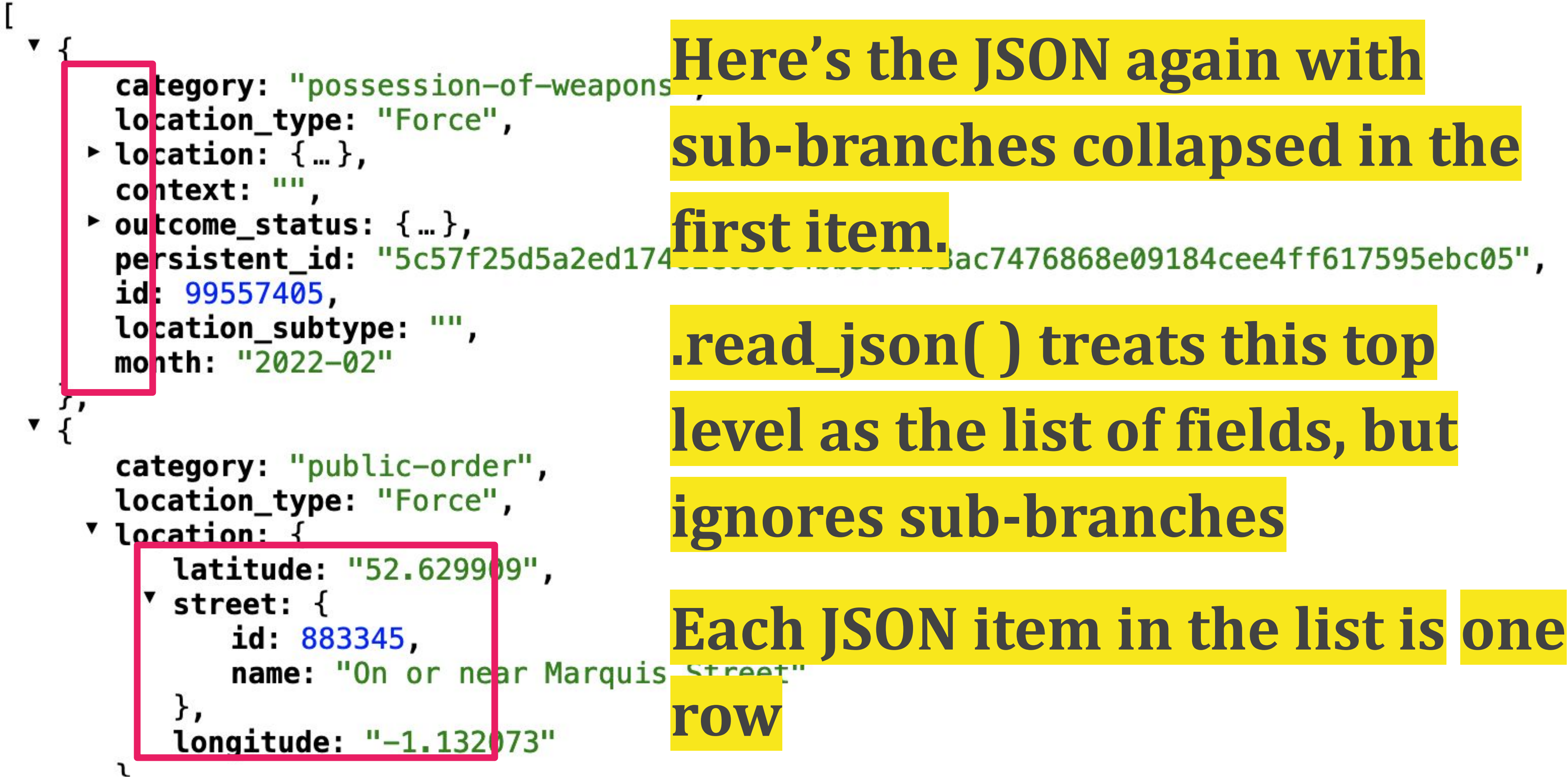
  (Category branch, 1st item, location branch)



```
#show the keys - note that these are only the top-level branches
print(crimedata.keys())
```

```
Index(['category', 'location_type', 'location', 'context', 'outcome_status',
       'persistent_id', 'id', 'location_subtype', 'month'],
      dtype='object')
```

https://github.com/paulbradshaw/pythonin12parts/blob/main/part9/12pandas_api.ipynb



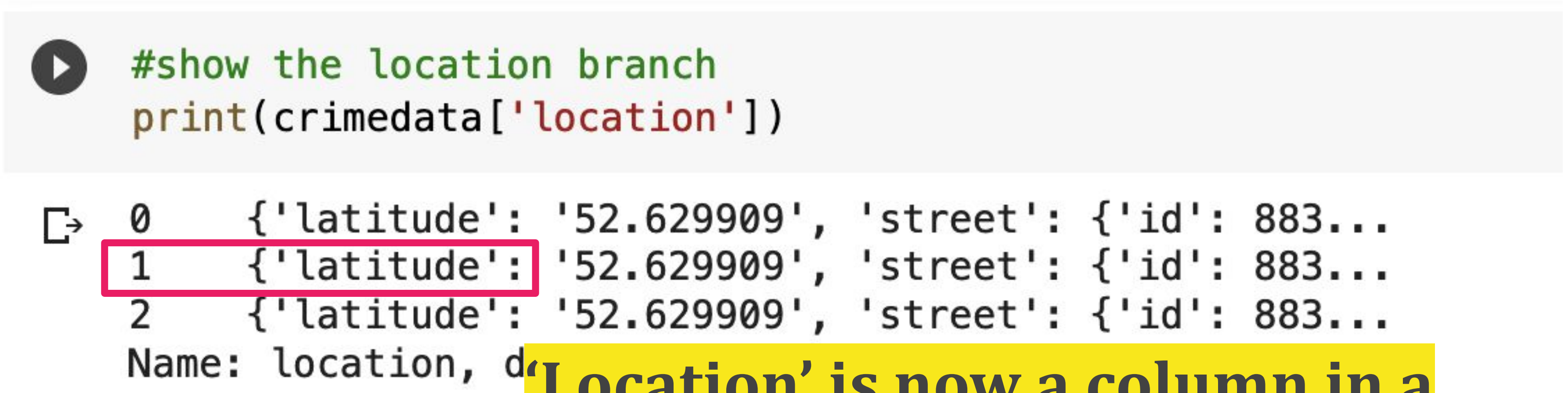
```
[
 ▼ {
      category: "possession-of-weapons
      location_type: "Force",
    ▸ location: {…},
      context: "",
    ▸ outcome_status: {…},
      persistent_id: "5c57f25d5a2ed174                3ac7476868e09184cee4ff617595ebc05",
      id: 99557405,
      location_subtype: "",
      month: "2022-02"
   },
 ▼ {
      category: "public-order",
      location_type: "Force",
    ▼ location: {
         latitude: "52.629909",
       ▼ street: {
            id: 883345,
            name: "On or near Marquis Street"
         },
         longitude: "-1.132073"
      }
```

Here's the JSON again with sub-branches collapsed in the first item.

.read_json( ) treats this top level as the list of fields, but ignores sub-branches

Each JSON item in the list is one row

https://data.police.uk/api/crimes-at-location?date=2022-02&lat=52.629729&lng=-1.131592



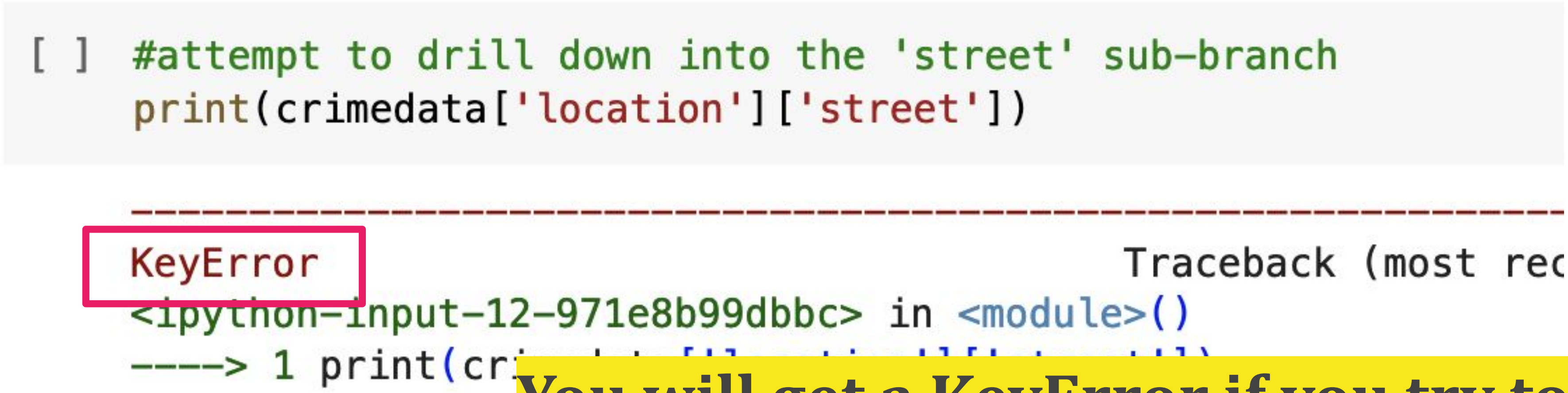
```
#show the location branch
print(crimedata['location'])
```

```
0     {'latitude': '52.629909', 'street': {'id': 883...
1     {'latitude': '52.629909', 'street': {'id': 883...
2     {'latitude': '52.629909', 'street': {'id': 883...
Name: location, d
```

'Location' is now a column in a dataframe - it doesn't have any keys, only rows/items (1st, 2nd)

But each row does have its own 'latitude' key, 'street' key etc.

https://github.com/paulbradshaw/pythonin12parts/blob/main/part9/12pandas_api.ipynb



```
[ ]  #attempt to drill down into the 'street' sub-branch
     print(crimedata['location']['street'])
```

```
-----------------------------------------------------------
KeyError                                  Traceback (most rec
<ipython-input-12-971e8b99dbbc> in <module>()
----> 1 print(cr
```

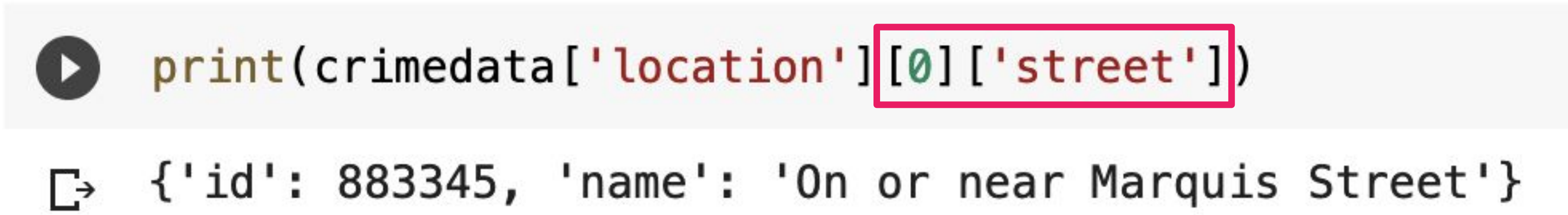
You will get a KeyError if you try to drill down into a sub-branch - because the sub-branches exist in each row - and you need to go into a specific row first

https://github.com/paulbradshaw/pythonin12parts/blob/main/part9/12pandas_api.ipynb

KeyError!

KeyError means the key doesn’t exist *here*

This is because the branch doesn’t have a sub-branch - it’s a **list** of JSON objects which **each** has a key

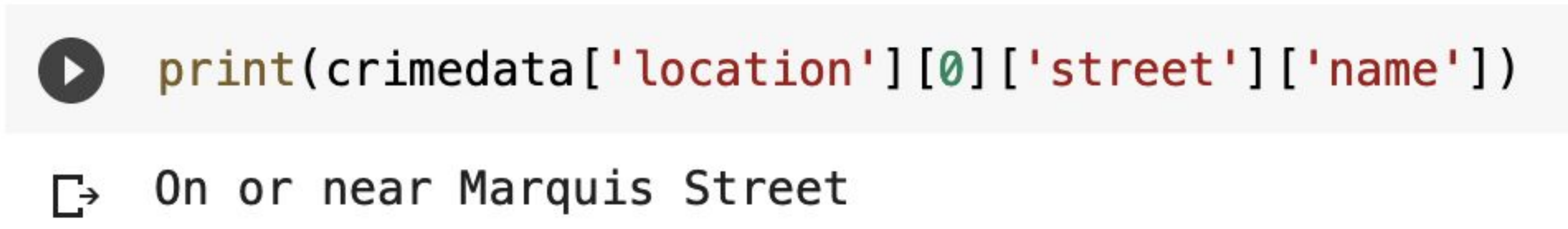


```
print(crimedata['location'][0]['street'])
```

```
{'id': 883345, 'name': 'On or near Marquis Street'}
```

So specify the index first, before the sub-branch you want to drill down to

https://github.com/paulbradshaw/pythonin12parts/blob/main/part9/12pandas_api.ipynb



```
print(crimedata['location'][0]['street']['name'])
```

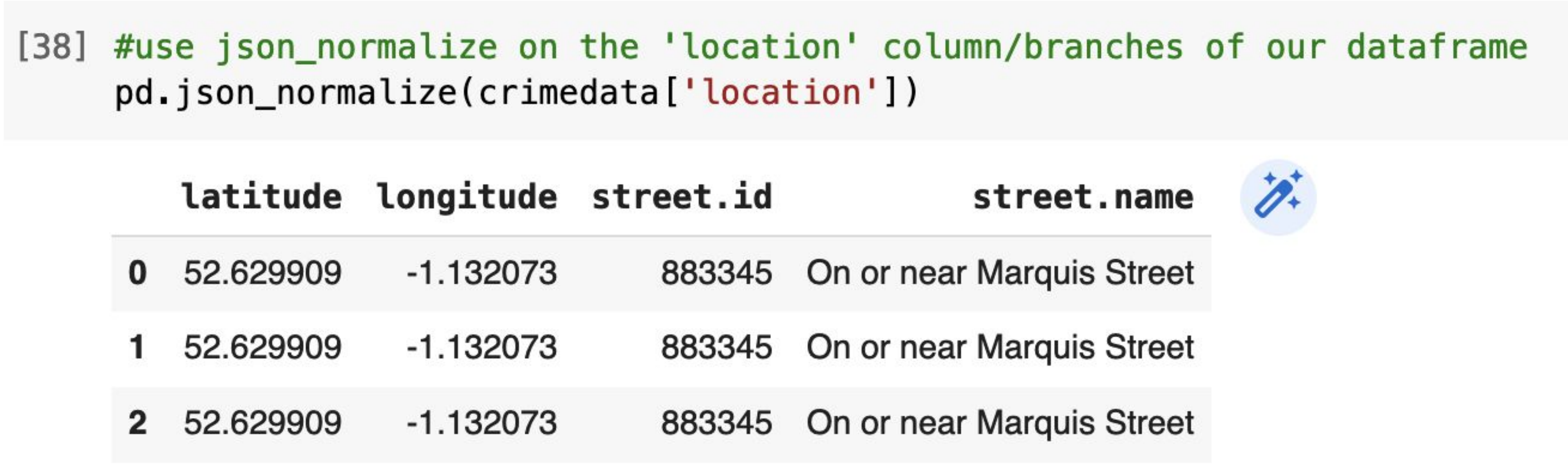
```
On or near Marquis Street
```

Unless there are further lists-within-items, you can add keys for further sub-sub-branches

https://github.com/paulbradshaw/pythonin12parts/blob/main/part9/12pandas_api.ipynb

Tip: json_normalize( ) for branches

- **pd.json_normalize( )** can ‘flatten’ sub-branches — but only to the next level
- Sub-branches are combined with main branch to create column names, e.g. street.id is the ‘id’ branch of ‘street’



```
[38] #use json_normalize on the 'location' column/branches of our dataframe
     pd.json_normalize(crimedata['location'])
```

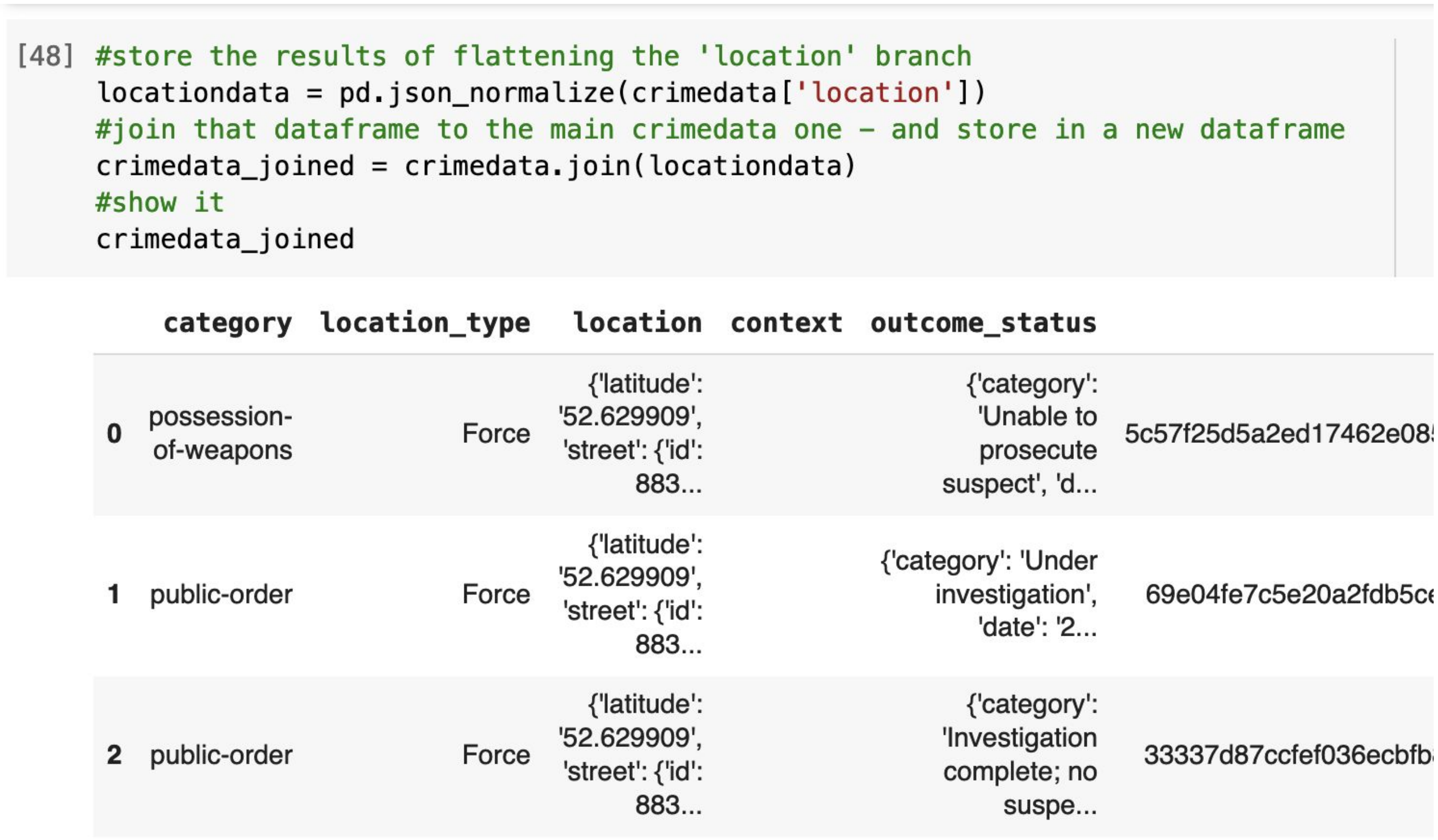
	latitude	longitude	street.id	street.name
0	52.629909	-1.132073	883345	On or near Marquis Street
1	52.629909	-1.132073	883345	On or near Marquis Street
2	52.629909	-1.132073	883345	On or near Marquis Street

Note this doesn’t include the ‘top level’ of the original dataframe

https://github.com/paulbradshaw/pythonin12parts/blob/main/part9/12pandas_api.ipynb

Tip: use .join( ) to combine results

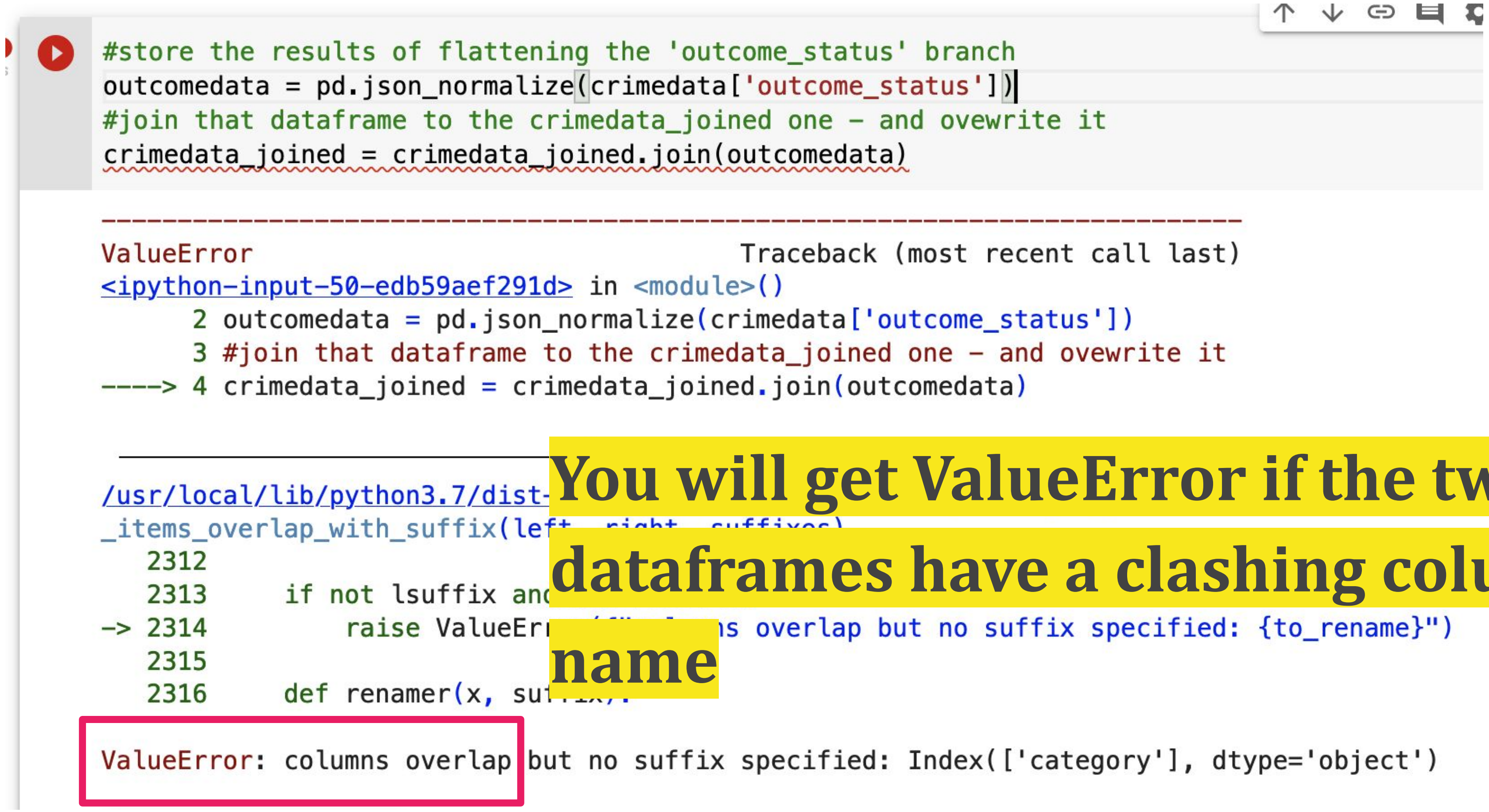
- Use a dataframe's **.join( )** method to specify another dataframe to join to it, e.g. **df1.join(df2)**
- Note: change column names if they clash or add **rsuffix='_branch'**



```
[48] #store the results of flattening the 'location' branch
     locationdata = pd.json_normalize(crimedata['location'])
     #join that dataframe to the main crimedata one - and store in a new dataframe
     crimedata_joined = crimedata.join(locationdata)
     #show it
     crimedata_joined
```

	category	location_type	location	context	outcome_status	
0	possession-of-weapons	Force	{'latitude': '52.629909', 'street': {'id': 883...		{'category': 'Unable to prosecute suspect', 'd...	5c57f25d5a2ed17462e08
1	public-order	Force	{'latitude': '52.629909', 'street': {'id': 883...		{'category': 'Under investigation', 'date': '2...	69e04fe7c5e20a2fdb5ce
2	public-order	Force	{'latitude': '52.629909', 'street': {'id': 883...		{'category': 'Investigation complete; no suspe...	33337d87ccfef036ecbfb

https://github.com/paulbradshaw/pythonin12parts/blob/main/part9/12pandas_api.ipynb



```
#store the results of flattening the 'outcome_status' branch
outcomedata = pd.json_normalize(crimedata['outcome_status'])
#join that dataframe to the crimedata_joined one - and ovewrite it
crimedata_joined = crimedata_joined.join(outcomedata)
```

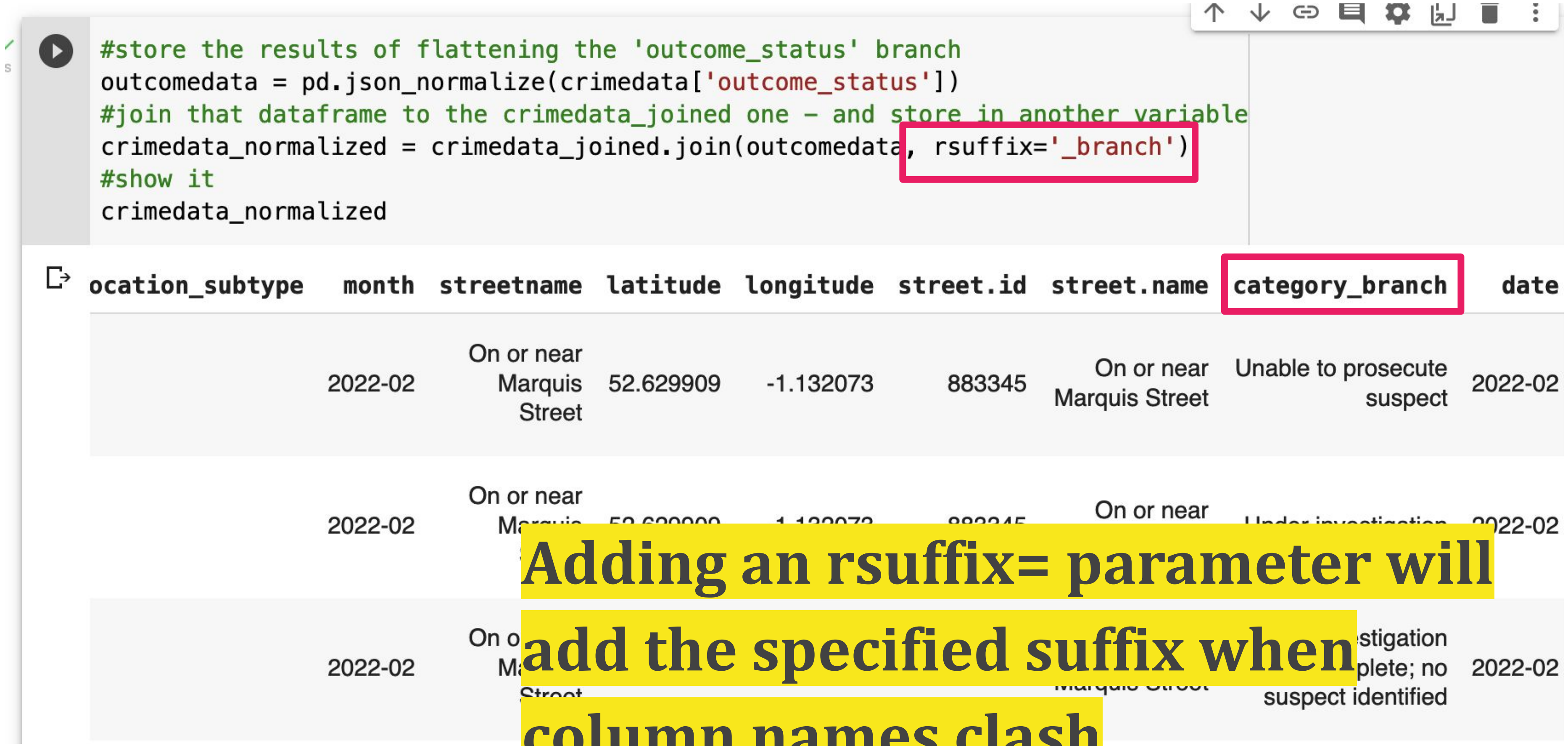
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-50-edb59aef291d> in <module>()
      2 outcomedata = pd.json_normalize(crimedata['outcome_status'])
      3 #join that dataframe to the crimedata_joined one - and ovewrite it
----> 4 crimedata_joined = crimedata_joined.join(outcomedata)


/usr/local/lib/python3.7/dist-
_items_overlap_with_suffix(left, right, suffixes)
   2312
   2313     if not lsuffix an
-> 2314         raise ValueErr                 ns overlap but no suffix specified: {to_rename}")
   2315
   2316     def renamer(x, su

ValueError: columns overlap but no suffix specified: Index(['category'], dtype='object')
```

You will get ValueError if the two dataframes have a clashing column name

https://github.com/paulbradshaw/pythonin12parts/blob/main/part9/12pandas_api.ipynb



```
#store the results of flattening the 'outcome_status' branch
outcomedata = pd.json_normalize(crimedata['outcome_status'])
#join that dataframe to the crimedata_joined one - and store in another variable
crimedata_normalized = crimedata_joined.join(outcomedata, rsuffix='_branch')
#show it
crimedata_normalized
```

ocation_subtype	month	streetname	latitude	longitude	street.id	street.name	category_branch	date
	2022-02	On or near Marquis Street	52.629909	-1.132073	883345	On or near Marquis Street	Unable to prosecute suspect	2022-02
	2022-02	On or near Marquis	52.629909	-1.132073	883345	On or near	Under investigation	2022-02
	2022-02	On o Ma Street				Marquis Street	stigation plete; no suspect identified	2022-02

Adding an rsuffix= parameter will add the specified suffix when column names clash

https://github.com/paulbradshaw/pythonin12parts/blob/main/part9/12pandas_api.ipynb

Key points

- JSON uses **{key : value}** pairs, sometimes with sub-branches
- Drill down into keys using square brackets like so: myjson[‘address’][‘postcode’]
- If it’s a [list] you’ll need to use an index