Dictionary

Saturday, September 26, 2020 10:16 PM

Dictionary

Data Camp

- Each key in a dictionary is unique and immutable, so any string combination or value (i.e. True) works as a key, but lists cannot serve as strings
- Pandas is a high level data manipulation tool created by Wes McKinney built on the Numpy package
- You can extend your numpy toolkit

Key Takeaways

When do you use a List vs. a Dictionary?

List	Dictionary
Select, update and remove: [ ]	Select, update and remove: { }
Indexed by range of numbers	Indexed by unique keys
Collection of values order matters select entire subsets	Lookup table with unique keys

- Dataframes are 2D displays of information (i.e. spreadsheet or a table)

This can be created by using a dictionary:

```
dict = {
    "country":["Brazil", "Russia", "India", "China", "South Africa"],
    "capital":["Brasilia", "Moscow", "New Delhi", "Beijing", "Pretoria"],
      "area":[8.516, 17.10, 3.286, 9.597, 1.221]
 "population":[200.4, 143.5, 1252, 1357, 52.98] }
```

- keys (column labels)
- values (data, column by column)

```
import pandas as pd
brics = pd.DataFrame(dict)
```

Writing your row indexes names:

```
brics.index = ["BR", "RU", "IN", "CH", "SA"]
```

DataFrame

```
brics
```

	country	capital	area	population
BR	Brazil	Brasilia	8.516	200.40
RU	Russia	Moscow	17.100	143.50
IN	India	New Delhi	3.286	1252.00
CH	China	Beijing	9.597	1357.00
SA	South Africa	Pretoria	1.221	52.98

- Notice the different data types in each column (string for country and capital and integer for area and population)

This can be done by importing a csv:

- brics.csv

```
,country,capital,area,population
BR,Brazil,Brasilia,8.516,200.4
RU,Russia,Moscow,17.10,143.5
IN,India,New Delhi,3.286,1252
CH,China,Beijing,9.597,1357
SA,South Africa,Pretoria,1.221,52.98
```

```
brics = pd.read_csv("path/to/brics.csv")
brics
```

	Unnamed: 0	country	capital	area	population
0	BR	Brazil	Brasilia	8.516	200.40
1	RU	Russia	Moscow	17.100	143.50
2	IN	India	New Delhi	3.286	1252.00
3	CH	China	Beijing	9.597	1357.00
4	SA	South Africa	Pretoria	1.221	52.98

To specify that the first column is the index name, use the index argument when importing:

```
brics = pd.read_csv("path/to/brics.csv", index_col = 0)
brics
```

	country	population	area	capital
BR	Brazil	200	8515767	Brasilia
RU	Russia	144	17098242	Moscow
IN	India	1252	3287590	New Delhi
CH	China	1357	9596961	Beijing
SA	South Africa	55	1221037	Pretoria

iloc and loc

- Indexing into dataframes

Jupyter Notebook

- It's perfectly possible to chain square brackets to select elements. To fetch the population for Spain from europe, for example, you need:

```
europe = { 'spain': { 'capital':'madrid', 'population':46.77 },
           'france': { 'capital':'paris', 'population':66.03 },
           'germany': { 'capital':'berlin', 'population':80.62 },
           'norway': { 'capital':'oslo', 'population':5.084 } }
europe['spain']['population']
```

- **Import CSV into Dataframes**

```
# Import cars data
import pandas as pd
cars = pd.read_csv('cars.csv', index_col = 0)

# Print out drives_right value of Morocco
print(cars.loc[['MOR'],['drives_right']])

# Print sub-DataFrame
print(cars.loc[['RU','MOR'], ['country','drives_right']])
```

- **Series vs. Dataframes**

Series is a type of list in pandas which can take integer values, string values, double values and more. But in Pandas Series we return an object in the form of list, having index starting from 0 to n, Where n is the length of values in series.

Series can only contain single list with index, whereas dataframe can be made of more than one series or we can say that a dataframe is a collection of series that can be used to analyse the data.

Creating a series:

```
import pandas as pd
import matplotlib.pyplot as plt

author = ['Jitender', 'Purnima', 'Arpit', 'Jyoti']

auth_series = pd.Series(author)
print(auth_series)

=====================================

0    Jitender
1     Purnima
2       Arpit
3       Jyoti
dtype: object
```

- **Creating a Dataframe from Seriesprint**

```
import pandas as pd
import matplotlib.pyplot as plt

author = ['Jitender', 'Purnima', 'Arpit', 'Jyoti']
article = [210, 211, 114, 178]

auth_series = pd.Series(author)
article_series = pd.Series(article)

frame = { 'Author': auth_series, 'Article': article_series }

result = pd.DataFrame(frame)

print(result)

=======================================

     Author  Article
0  Jitender      210
1   Purnima      211
2     Arpit      114
3     Jyoti      178
```

Building a DF and making it look good

```
import pandas as pd

# Build cars DataFrame
names = ['United States', 'Australia', 'Japan', 'India', 'Russia', 'Morocco', 'Egypt']
dr =  [True, False, False, False, True, True, True]
cpc = [809, 731, 588, 18, 200, 70, 45]
cars_dict = { 'country':names, 'drives_right':dr, 'cars_per_cap':cpc }
cars = pd.DataFrame(cars_dict)
print(cars)

# Definition of row_labels
row_labels = ['US', 'AUS', 'JPN', 'IN', 'RU', 'MOR', 'EG']

# Specify row labels of cars
cars.index = row_labels
```

iloc and loc

- Indexing into dataframes
  - Square brackets
    - Column access `brics[["country", "capital"]]`
    - Row access: only through slicing `brics[1:4]`
  - loc (label-based)
    - Row access `brics.loc[["RU", "IN", "CH"]]`
    - Column access `brics.loc[:, ["country", "capital"]]`
    - Row & Column access

```
brics.loc[
["RU", "IN", "CH"],
["country", "capital"]
]
```

- Printing out columns using slicing [row range:column range]

```
cars.loc[:, 'country']
cars.iloc[:, 1]
cars.loc[:, ['country','drives_right']]
cars.iloc[:, [1, 2]]
```

Slice objects: [specific row #s, specific column #]

```
   a     b     c     d
0  1     2     3     4
1  100   200   300   400
2  1000  2000  3000  4000
```

```
>>> df.iloc[[0, 2], [1, 3]]
     b     d
0    2     4
2  2000  4000
```

With *slice* objects.

```
>>> df.iloc[1:3, 0:3]
      a     b     c
1   100   200   300
2  1000  2000  3000
```

With a boolean array whose length matches the columns.

- Notice how the length range [1:3] excludes the 3rd one so it's a range of two

- No specified second item refers specifically to rows

```
>>> df.iloc[[0]]
   a  b  c  d
0  1  2  3  4
>>> type(df.iloc[[0]])
<class 'pandas.core.frame.DataFrame'>
```

```
>>> df.iloc[[0, 1]]
     a    b    c    d
0    1    2    3    4
1  100  200  300  400
```

With a *slice* object.

- [[ ]] double brackets makes a dataframe
- [] single bracket makes a panda series

```
# Definition of row_labels
row_labels = ['US', 'AUS', 'JPN', 'IN', 'RU', 'MOR', 'EG']

# Specify row labels of cars
cars.index = row_labels
print(cars)

# Print cars again
```