Data science fundamentals
Lesson 7

Hay Kranen

Monday October 15th, 2018


しょうゆ

Today's programme

Learn improvements
Recap
Wiki API
Break
Refactoring
Pandas
Lunch break

LEARN CHANGES

RECAP

URL CHECKER

Start with imports

```
#asking for the URL
url = input("What url would you like to scrape?")
url = url.strip()
import requests
```

The is operator

```
# printing the error message if the status code i
if status is not 200:
        print(f'Something went wrong, please try a
        exit()
```

```
In [17]: a = 200

In [18]: b = 200

In [19]: a is b
Out[19]: True

In [20]: a = 404

In [21]: b = 404

In [22]: a is b
Out[22]: False

In [23]: a == b
Out[23]: True
```

Variable names

```
#ask for a url
u = input("Copy past an URL: ").strip()

#request the website for the given url
r = requests.get(u)

#put the headers in a variable
h = r.headers
```

if / else

```
if r.status_code is not 200:
        exit()

else:
        for key, value in headers.items():
                print(f'\n {key}: {value}')

        for line in content:
                print(line)
```

Wiki API

```python
if r.status_code is not 200:
        exit()

else:
        for key, value in headers.items():
                print(f'\n {key}: {value}')

        for line in content:
                print(line)
```

```python
# lists of the language codes and the display name of the languages for the print statements
language = ["en", "nl", "fr"]
languages = ["English", "Dutch", "French"]
```

```python
print(f'\n{languages[index]}: ')
```

```python
languages = {
    "en" : "English",
    "nl" : "Dutch",
    "fr" : "French"
}

for code, language in languages.items():
    print(code) # 'en'
    print(language) # 'English'
```

```python
from IPython.display import display, Image
```

```python
#Variables for our headers
title = data["title"]
desc = data["description"]
extr = data["extract"]
thumb = data["originalimage"]["source"]
img = Image(url = thumb)

if status == 200:
    print("Website is online! We'll proceed!")

    print("Title:", title)
    print("Description:", desc)
    print("Extract:", extr)
    print("Image:")
    display(img)
```

```
# beter om nog om te zetten naar key + value
# value checken of hij leeg is
```

```python
import requests
import json

title = input("Enter an article: ").strip().replace(" ", "_")
languages = {
    "en" : "English",
    "es" : "Espanol",
    "nl" : "Nederlands"
}

for lang, langname in languages.items():
    print(f"Looking up in {langname}")
    url = f"https://{lang}.wikipedia.org/api/rest_v1/page/summary/{title}"

    req = requests.get(url)

    if req.status_code != 200:
        print(f"We got an error: {req.status_code}")
        print("")
    else:
        data = json.loads(req.text)
        description = data["description"]
        extract = data["extract"]
        print(f"{title}: {description}")
        print(extract)
        print("")
```

Refactoring

the process of restructuring existing computer code—changing the factoring—without changing its external behavior

Write a program where a user can enter the number 1, 2 or 3.
All other inputs are invalid.

```python
choice = input("Choose an option: 1, 2 or 3: ")
```

```python
if choice == "1":
    print("Awesome, that is a valid option")
elif choice == "2":
    print("Awesome, that is a valid option")
elif choice == "3":
    print("Awesome, that is a valid option")
else:
    print("That is an invalid option!")
```

```python
choice = input("Choose an option: 1, 2 or 3: ")
```

```python
if choice != "1" and choice != "2" and choice != "3":
    print("That is an invalid option!")
else:
    print("Awesome, that is a valid option")
```

```python
choice = input("Choose an option: 1, 2 or 3: ")
```

```python
if choice in ["1", "2", "3"]:
    print("Awesome, that is a valid option")
else:
    print("That is an invalid option!")
```

```python
choice = input("Choose an option: 1, 2 or 3: ")
```

```python
choice = int(choice)

if choice in [1, 2, 3]:
    print("Awesome, that is a valid option")
else:
    print("That is an invalid option!")
```

```python
choice = input("Choose an option: 1, 2 or 3: ")
```

```python
VALID_CHOICES = [1,2,3]
choice = int(choice)

if choice in VALID_CHOICES:
    print("Awesome, that is a valid option")
else:
    print("That is an invalid option!")
```

```python
choice = input("Choose an option: 1, 2 or 3: ")
```

```python
VALID_CHOICES = [1,2,3]
choice = int(choice)

if choice not in VALID_CHOICES:
    print("That is an invalid option!")
    exit()

print("Awesome, that is a valid option")
```

```python
choice = input("Choose an option: 1, 2 or 3: ")
```

```python
if choice == "1":
    print("Awesome, that is a valid option")
elif choice == "2":
    print("Awesome, that is a valid option")
elif choice == "3":
    print("Awesome, that is a valid option")
else:
    print("That is an invalid option!")
```

```
Choose an option: 1, 2 or 3: 2
Awesome, that is a valid option
Awesome, that is a valid option
```

```python
VALID_CHOICES = [1,2,3]
choice = int(choice)

if choice not in VALID_CHOICES:
    print("That is an invalid option!")
    exit()

print("Awesome, that is a valid option")
```

Refactor snacknames
Take the code you wrote for the first 'snacknames' assignment (not the multidimensional one) and refactor. Use everything you have learned so far. If you didn't make the exercise, use the **snacknames-refactor.py** file from the Github repo.

Just make sure that the output is the same as the original program!

Original assignment
Loop through three predefined friends, print out their name, the length of the name and ask for the favourite snack of this friend. Save this snack somehow.

After the first loop, loop again and print the name of the friend and their favourite snack.

Tips
* You're free to use any data type you want for this assignment, like a **dict** or a multidimensional **list.**
* You're also free to all the other things you have learned, like F-strings.

PANDAS

ASSIGNMENT 4

Simple types and methods (**int**, **str**, **bool**)
Comparisons and **if** statements
for and **while** loops
Complex types and methods (**list**, **dict**)

```python
with open("paintings.csv") as f:
    lines = f.read().splitlines()
    paintings = []

    for item in paintings:
        painting = item.split(",")
        paintings.append(painting)
```

```python
import csv

with open("paintings.csv") as f:
    reader = csv.reader(f)
    paintings = list(reader)
```

```
[['Salvator Mundi', 'Leonardo da Vinci', '450.3'],
 ['Interchange', 'Willem de Kooning', '300'],
 ['The Card Players', 'Paul Cézanne', '250'],
 ['Nafea Faa Ipoipo', 'Paul Gauguin', '210'],
 ['Number 17A', 'Jackson Pollock', '200']]
```

```
Salvator Mundi,Leonardo da Vinci,450.3
Interchange,Willem de Kooning,300
The Card Players,Paul Cézanne,250
Nafea Faa Ipoipo,Paul Gauguin,210
Number 17A,Jackson Pollock,200
```

```
Salvator Mundi,Leonardo da Vinci,450.3^M
Interchange,Willem de Kooning,300^M
The Card Players,Paul Cézanne,250^M
Nafea Faa Ipoipo,Paul Gauguin,210^M
Number 17A,Jackson Pollock,200
```

```
Salvator Mundi;Leonardo da Vinci;450.3
Interchange;Willem de Kooning;300
The Card Players;Paul Cézanne;250
Nafea Faa Ipoipo;Paul Gauguin;210
Number 17A;Jackson Pollock;200
```

```
"Salvator Mundi","Leonardo da Vinci",450.3
"Interchange","Willem de Kooning",300
"The Card Players","Paul Cézanne",250
"Nafea Faa Ipoipo","Paul Gauguin",210
"Number 17A","Jackson Pollock",200
```

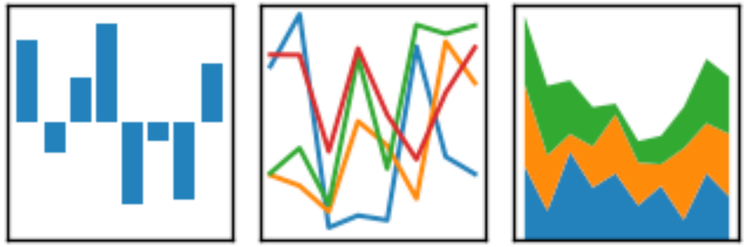
```python
import csv

with open("paintings.csv") as f:
    reader = csv.reader(f)
    paintings = list(reader)
```

```
[['Salvator Mundi', 'Leonardo da Vinci', '450.3'],
 ['Interchange', 'Willem de Kooning', '300'],
 ['The Card Players', 'Paul Cézanne', '250'],
 ['Nafea Faa Ipoipo', 'Paul Gauguin', '210'],
 ['Number 17A', 'Jackson Pollock', '200']]
```

Write it yourself	Use a library / module
More work	Less work (in principle)
More control	Less control
Use tools that you know	Learn the library first
You’re on your own	Benefit from other people’s work
Do your own version management	Pay attention to the correct version
Read the standard documentation	Library documentation

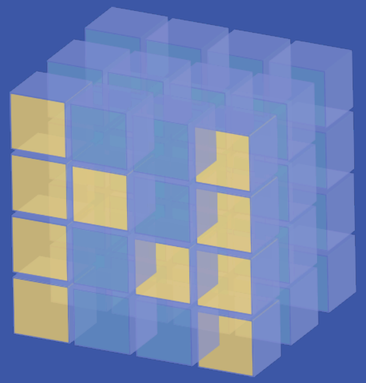
pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



jupyter

matplotlib



NumPy

Input → **Transformation** → **Output**

Input	Transformation	Output
CSV files	Cleanup	CSV files
JSON files	Analysis	JSON files
Excel files	Calculation	Excel files
API data (JSON data)	Filtering	API data (JSON data)
Python data (list, dict)	Grouping	Python data (list, dict)
		Visualisations

```python
with open("paintings.csv") as f:
    lines = f.read().splitlines()
    paintings = []

    for item in paintings:
        painting = item.split(",")
        paintings.append(painting)
```

```python
import csv

with open("paintings.csv") as f:
    reader = csv.reader(f)
    paintings = list(reader)
```

```python
paintings = pd.read_csv("paintings.csv")
```

```python
with open("paintings.csv") as f:
    lines = f.read().splitlines()
    paintings = []

    for item in paintings:
        painting = item.split(",")
        paintings.append(painting)
```

```python
paintings = pd.read_csv("paintings.csv")
```

```python
with open("movies.json") as f:
    movies = json.load()
```

```python
paintings = pd.read_json(“movies.json”)
```

Input

```
import pandas as pd
df = pd.read_json("movies.json")
```

Transformation

```
df_1983 = df[df["year"] == 1983]
del df_1983["actors"]
del df_1983["genres"]
```

Output

```
df_1983.to_csv("movies_1983.csv")
```

Pandas

Input

Transformation

Output

```python
# Import the library
import pandas as pd

# Read the JSON file to a Dataframe
df = pd.read_json("movies.json")

# Select only movies that were made in 1983
df_1983 = df[df["year"] == 1983]

# Drop the 'actors' and 'genres' columns
del df_1983["actors"]
del df_1983["genres"]

# Save to a new CSV file
df_1983.to_csv("movies_1983.csv")
```

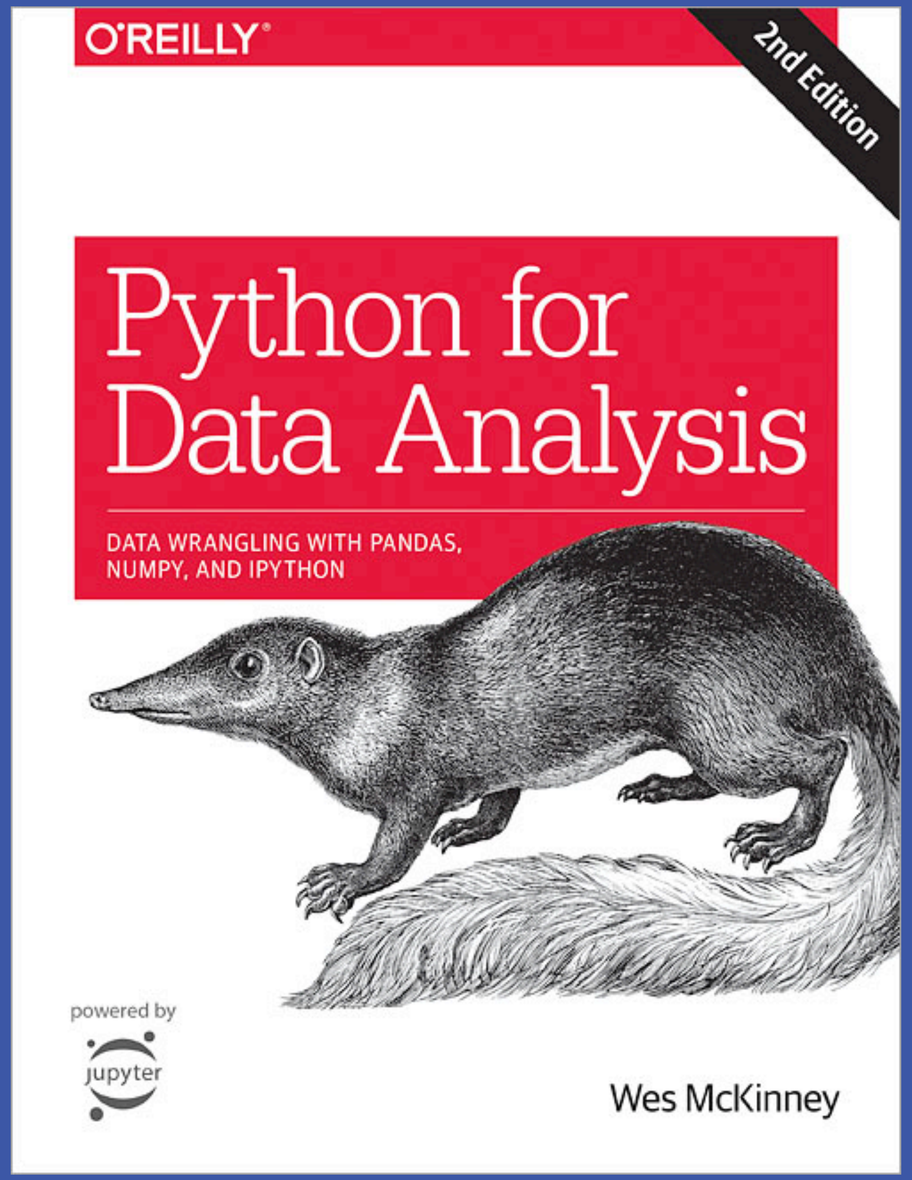
"Vanilla" Python

Input

Transformation

Output

```python
# Import json and csv libraries
import json
import csv

# Open movies.json and convert to a list of dicts
with open("movies.json") as f:
    movies = json.load(f)

# Also create a new list to hold the filtered movies
new_movies = []

# Loop over movies
for movie in movies:
    # Delete the 'actors' and 'genres' keys
    del movie["actors"]
    del movie["genres"]

    # Check if year is 1983
    if movie["year"] == 1983:
        # Add to the new list
        new_movies.append(movie)

# Open the 'movies_1983.csv' file for writing
with open("movies_1983.csv", "w") as f:
    # Get the fieldnames, this is required when also writing a header
    # We get them from the first item in new_movies
    fieldnames = new_movies[0].keys()

    # Create a new CSV 'dictionary writer'
    writer = csv.DictWriter(f, fieldnames = fieldnames)

    # Loop over the movies in the new_movies list
    for row in new_movies:
        # And write them to the CSV file
        writer.writerow(row)
```


O'REILLY
2nd Edition
Python for Data Analysis
DATA WRANGLING WITH PANDAS, NUMPY, AND IPYTHON
powered by
jupyter
Wes McKinney

Series

```
[
    "Tinus",
    "Barrie",
    "Hans"
]
```

Dataframes

```
[
    {
        "name" : "Tinus",
        "snack" : "Mars"
    },
    {
        "name" : "Barrie",
        "snack" : "Oreo"
    },
    {
        "name" : "Hans",
        "snack" : "Twix"
    }
]
```

jupyter

Temperatures

Create a new Jupyter Notebook that shows some interesting statistics about the **temperatures.csv** file (found in the Github repo)

* Import the **pandas** library as **pd**
* Read the csv file using the **read_csv()** method to a new Dataframe
* Show the first five entries using the **head()** method
* Use the **describe()** method to show general statistics about the temperature
* Show all days where the temperature was above 22 degrees
* Show all days where the temperature was below -3 degrees
* Add a new column called **freezing** that contains a boolean (**True** or **False**) if the temperature in that row is beneath zero degrees. Print the first **ten** rows.
* Use **plot()** to show a line chart of the temperature

Tips

* Make sure to put your **temperatures.csv** file in the same directory as your notebook.

Extended use

* When you're done try exploring other options and functions of the **pandas** library.

Type	Examp	Conve
Integ	42,	**int**()
Float	3.14,	**float**
Strin	"Hell	**str**()
Boole	True	**bool**(

```
age = 20

if age < 20:
    print("option 1")
elif age <= 20 and age > 20:
    print("option 2")
else:
    print("option 3")
```

Compilation
Interpretation