Aske Svane Qvist

Exercises W38Mon: Digital Data Structures

You fingers should now be iching for some new data to play with. And guess what? The Internet is full of it! Every **website** you can imagine can be **turned into a dataset** for analysis and you will now learn how to do this.

1. HTML

It all starts with learning a little about the language of the World Wide Web. **HTML is the *markup language* used by web pages**. It's ubiquitous on the web; even when editing this notebook you are interacting with HTML (right click and hit "View Page Source" if you need proof).

HTML consists of a series of elements that label pieces of content. Elements tell the browser how to display the content and are defined by a start tag followed by some content and an end tag: `<tagname> Content </tagname>` .

Start tags are enclosed in angled brackets `<>` while end tags have a forward slash between the angled brackets `</>` .

> **Exercise 1.1:** What is the typical use of `<p>` , and `<h1>` tags? Look them up.

- `<p>` defines a paragraph of text
- `<h1>` defines the highest level or most important header in the HTML

> **Exercise 1.2:** Can you find a `<h1>` tag in the Kaggle mainpage (https://kaggle.com/)? What text is contained in it? *Remember you can right click and hit 'Inspect' to view the code for the elements in the site.*

"Start with more than a blinking cursor" - It is the main header on the page also written with the biggest letters

> **Exercise 1.3:** Try changing the text in the `<h1>` tag. What happens? Reload the page. What happens now?

- I could change the text on the webpage!

Elements may also contain attributes that provide additional information about elements. Attributes are always contained in the start tag and usually come in name/value pairs like name = "value":
`<tagname attribute_name = 'attribute_value'> Content </tagname>`

For the next two exercises, refer to the code below.

```
<html>
<body>

<h1>This is the main title of the webpage</h1>
<h2>This is a sub-heading</h2>
<p style="text-align:right">This is a paragraph of text.</p>

<h2>This is another sub-heading</h2>
<p>This is a paragraph of text with some words in bold.</p>
<img src="https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fme
dia.giphy.com%2Fmedia%2FkFgzrTt798d2w%2Fgiphy.gif&f=1&nofb=1" width="49
3" height="340">
<p>And that just above is an image.</p>

</body>
</html>
```

> **Exercise 1.4**: What is the attribute of the first `<p>` element?

- that is 'style' - it aligns text to the right

> **Exercise 1.5:** How many attributes does the `<img>` tag have? Can you notice something different between the `<img>` tag and the other tags?

- The `<img>` has 2 attributes: src (the source of the image?), the width, and the height.
- It does not end with the tagname but just a '>'

> **Exercise 1.6:** The `<a>` tag defines a hyperlink. The most important attribute in an `<a>` tag is the `href` attribute, which specifies the link's destination. Write some

text with a hyperlink to your most visited website. *Remember HTML is a type of markdown, so you can write your answer in a markdown cell and it will render correctly.*

```
<a href="https://www.youtube.com/">My hyperlink to YouTube</a>
```

Up to now, we have mostly seen tags that contain text. However, there are other types of tags that are used as containers for other HTML elements. The most commonly used container is the `<div>`. `<div>` tags can contain text tags such as `<h>` tags or `<p>` tags, tables, and even other `<div>` tags. These are nested divs.

Exercise 1.7: Go back to the Kaggle site (https://www.kaggle.com/) and find out how deeply nested the `<h1>` tag is.

In [ ]:
```
#<body>
#    <main>
#        <div>
#            <div>
#                <div>
#                    <section>
#                        <div>
#                            <div>
#                                <div>
#                                    <h1>
```

2. CSS

CSS stands for Cascading Style Sheets and it is the language used to style an HTML document i.e. describes how HTML elements should be displayed (think colors, font, spacing, etc).

CSS can be inserted *inline* to describe how a tag should look, *internally* at the beginning of the section to describe how all tags in that section should be displayed, or *externally*, in a different document that describes how all tags in the site should be displayed.

Sometimes you might want a same type of tag to look differently in different places of the site (ie. you want a paragraph to have a font size 18 at the top of the page and a font size 14 lower down). In that case, you can use `class` and `id` attributes to identify the tags and apply styling to them. You might wonder why you need to know this if you are studying data science, not web design. Well, if you ever want to scrape certain elements from a website, you need to identify them first, and that's when these tags a attributes become very useful to you.

Let's practice this identifying elements.

Exercise 2.1: What class does the `<h1>` tag in the Kaggle mainpage

(https://www.kaggle.com/) belong to?

class="sc-kEqXSa sc-iqAclL sc-kBqmDu dlltzk jibPFy boQsLV"

Exercise 2.2: Go to the competitions section in the Kaggle site (https://www.kaggle.com/competitions) and look at the 'Get started' and 'Active competitions' text (using just your eyes for now). Now go to the datasets section in the Kaggle site (https://www.kaggle.com/datasets) and look at the 'Trending datasets' and 'Popular datasets' text. Do you think the text in both sections has the same font, color, size? Do you suspect the texts in both pages belong to the same `class` ? Now Inspect these elements and look at their classes. What do you find?

"Get started" - `<h2 class="sc-fFSPTT sc-bkbkJK sc-khIimk gaJJBS eraKfR gNMCgZ">Get Started</h2>` "Active competitions" - `<h2 class="sc-fFSPTT sc-bkbkJK sc-eSRwjH gaJJBS eraKfR lnhkkx">Active Competitions</h2>`

"Trending datasets" - `<h2 class="sc-fFSPTT sc-bkbkJK sc-irqbAE gaJJBS eraKfR kxZWSs">Trending Datasets</h2>`
"Popular datasets" - `<h2 class="sc-fFSPTT sc-bkbkJK sc-irqbAE gaJJBS eraKfR kxZWSs">Popular Datasets</h2>`

- The two on the first page belong to the same class. The two on the other page belong to another class. However, they look fairly alike

3. JSON

JSON stands for Java Script Object Notation. Unlike other types of data, JSON data is non-tabular, meaning that not all records need to have the same attributes. Data in JSON format is organized into collections of objects, where objects are collections of attribute-value pairs. This is very much like another data structure you already know: dictionaries.

You can therefore parse a JSON text and read in Python to access data just like you would in a dictionary. Let's see how this works.

Exercise 3.1: Open this URL (https://rickandmortyapi.com/api/episode/1) and explore what kind of data it contains (I hope there are some Rick and Morty fans in the class). Use the requests library to access the website, render it as text in a variable, and parse it using `json.loads()` . You will need to import both `requests` and `json` to do this.

Hint: You can execute `?requests.get` after importing `requests` to display the module documentation.

In [11]:
```
# Import packages
import json, requests

# Save the url
url = "https://rickandmortyapi.com/api/episode/1"

# grabbing the data with .get()
response = requests.get(url)
response
```

Out[11]:
```
<Response [200]>
```

In [10]:
```
# save as a string
data_string = response.text
data_string
```

Out[10]:
```
'{"id":1,"name":"Pilot","air_date":"December 2, 2013","episode":"S01E01","c
haracters":["https://rickandmortyapi.com/api/character/1","https://rickandm
ortyapi.com/api/character/2","https://rickandmortyapi.com/api/character/3
5","https://rickandmortyapi.com/api/character/38","https://rickandmortyapi.
com/api/character/62","https://rickandmortyapi.com/api/character/92","http
s://rickandmortyapi.com/api/character/127","https://rickandmortyapi.com/ap
i/character/144","https://rickandmortyapi.com/api/character/158","https://r
ickandmortyapi.com/api/character/175","https://rickandmortyapi.com/api/char
acter/179","https://rickandmortyapi.com/api/character/181","https://rickand
mortyapi.com/api/character/239","https://rickandmortyapi.com/api/character/
249","https://rickandmortyapi.com/api/character/271","https://rickandmortya
pi.com/api/character/338","https://rickandmortyapi.com/api/character/39
4","https://rickandmortyapi.com/api/character/395","https://rickandmortyap
i.com/api/character/435"],"url":"https://rickandmortyapi.com/api/episode/
1","created":"2017-11-10T12:56:33.798Z"}'
```

In [17]:
```
# data as json
dataJSON = json.loads(data_string)
dataJSON
```

Out[17]:

```
{'id': 1,
 'name': 'Pilot',
 'air_date': 'December 2, 2013',
 'episode': 'S01E01',
 'characters': ['https://rickandmortyapi.com/api/character/1',
  'https://rickandmortyapi.com/api/character/2',
  'https://rickandmortyapi.com/api/character/35',
  'https://rickandmortyapi.com/api/character/38',
  'https://rickandmortyapi.com/api/character/62',
  'https://rickandmortyapi.com/api/character/92',
  'https://rickandmortyapi.com/api/character/127',
  'https://rickandmortyapi.com/api/character/144',
  'https://rickandmortyapi.com/api/character/158',
  'https://rickandmortyapi.com/api/character/175',
  'https://rickandmortyapi.com/api/character/179',
  'https://rickandmortyapi.com/api/character/181',
  'https://rickandmortyapi.com/api/character/239',
  'https://rickandmortyapi.com/api/character/249',
  'https://rickandmortyapi.com/api/character/271',
  'https://rickandmortyapi.com/api/character/338',
  'https://rickandmortyapi.com/api/character/394',
  'https://rickandmortyapi.com/api/character/395',
  'https://rickandmortyapi.com/api/character/435'],
 'url': 'https://rickandmortyapi.com/api/episode/1',
 'created': '2017-11-10T12:56:33.798Z'}
```


> **Exercise 3.2:** When did Rick and Morty's pilot air? Remember you can now access the data as a dictionary.

In [19]:
```
# Access air_date with key name
dataJSON["air_date"]
```

Out[19]:
```
'December 2, 2013'
```

> **Exercise 3.3:** How many characters appeared on episode 1 of Rick and Morty?

In [22]:
```
# Access character list
character_URLs = dataJSON["characters"]
len(character_URLs)
```

Out[22]:
```
19
```

Exercise 3.4: Yes, there are many characters in Rick and Morty. But two of them are the protagonists (see the show title). Repeat what you did in exercise 3.1 to get information about Rick (https://rickandmortyapi.com/api/character/1) and Morty (https://rickandmortyapi.com/api/character/2).

In [25]:
```
# RICK

# Save the url
rick = "https://rickandmortyapi.com/api/character/1"
# grabbing the data with .get()
response = requests.get(rick)
# as string
rick_string = response.text
# data as json
dataRick = json.loads(rick_string)

# MORTY

# Save the url
morty = "https://rickandmortyapi.com/api/character/2"
# grabbing the data with .get()
response = requests.get(morty)
# as string
morty_string = response.text
# data as json
dataMorty = json.loads(morty_string)
```

Exercise 3.4.1: What is Rick's full name? And Morty's?

In [29]:
```
print(dataRick["name"])
print(dataMorty["name"])
```

```
Rick Sanchez
Morty Smith
```

Exercise 3.4.2: Do you think both Rick and Morty appear on the same number of episodes? Check that out in the data.

In [37]:
```
len(dataMorty["episode"]) == len(dataRick["episode"])
```

Out[37]: True

They appeared in the same number of episodes

> **Exercise 3.5:** Save one of the .json files you imported back to your computer.

Hint: Use the json `.dump()` method.

In [43]:
```
# save the data as json files
with open('data/dataRick.json', 'w') as f:
    json.dump(dataRick, f)

with open('data/dataMorty.json', 'w') as f:
    json.dump(dataMorty, f)
```

4. Reflections

> Write a brief paragraph reflecting on your experience today. How do you think you can use these new skills to acquired data from the internet? What did you struggle with? What did you enjoy? What surprised you?

This was a super nice exercise. I think a foundational understanding how HTML and json work will be crucial when we start learning to scrape. I learned a lot of new things. I was struggling a bit with the syntax and had to google around. Luckely, I went over some of the exercises in the readings for today - so getting the data from a url was pretty much plug-n-play.

5. Bonus Exercise: Putting it all together

Now that you know a lot of random Rick and Morty trivia, you are ready to crush any Pub Quiz. But most importantly, now that you know the basics of HTML, CSS and JSON, you can make a basic website!

> **Exercise 5.1**: Design a profile about your favorite Rick and Morty character. Use the data from exercise 2.4 to write the content. Include different HTML tags to organize the data in hierarchies, and use CSS attributes to add some life to the website. Include a picture and a link.

Hint: Don't forget HTML is a markup language, so you can write your code in a Markdown cell and it will render it correctly.

Morty!

My favorite character from 'Rick and Morty'

Morty looks like this

You can find more information about him here:
Morty Smith (https://da.wikipedia.org/wiki/Morty_Smith)

> Mortimer "Morty" Smith is one of the title characters and the secondary protagonist of the American animated television series Rick and Morty. Created by Justin Roiland and Dan Harmon, Morty is an anxious 14-year-old based on Michael J. Fox's Marty McFly from Back to the Future.[3] Known for his awkward, anxious, second-guessing, and doubtful personality and low sense of self-esteem, the character has been well-received. He is the good-natured and impressionable grandson of alcoholic mad scientist Rick Sanchez, the son of Jerry and Beth Smith, and the younger brother of Summer Smith, who can be easily manipulated.
> wikipedia.com (https://da.wikipedia.org/wiki/Morty_Smith)

In [ ]: