
ISA 414 – Managing Big Data

Lecture 6 – Introduction to Python

String Manipulation and Regular Expressions

Dr. Arthur Carvalho

arthur.carvalho@miamioh.edu



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

Announcements

- Assignment 1 is now available on Canvas
 - Deadline: Wednesday, September 15th, before 11:59 pm

Lecture Objectives

- Quick review of Homework 4
- Learn how to manipulate strings in Python
- Learn about regular expressions

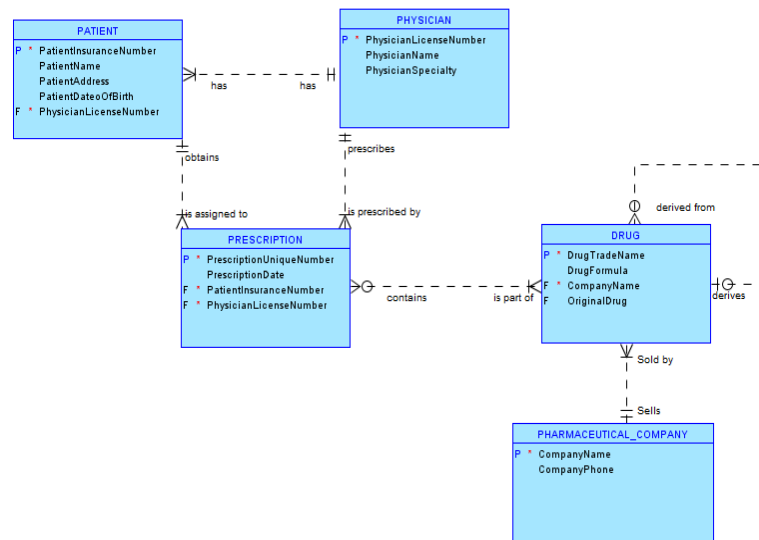
Lecture Instructions

- Download from Canvas:
 - The notebook “*Lecture 6.ipynb*”
 - The log file “*access.log.txt*”
- Make sure all the files are inside the same folder
- Open the file “*Lecture 6.ipynb*” with VS Code

Introduction to Python

- Traditional view of data (table, data frames, ...)
 - Well-defined structure

| | Return | SAT | MBA | Age | Tenure | var | var |
|----|--------|--------|-----|------|--------|-----|-----|
| 1 | -21.10 | 1211.0 | .0 | 47.0 | 4.0 | | |
| 2 | -6.38 | 1163.0 | .0 | 43.0 | 5.0 | | |
| 3 | -3.32 | 1217.0 | 1.0 | 39.0 | 3.0 | | |
| 4 | -2.27 | 1185.0 | 1.0 | 35.0 | 2.0 | | |
| 5 | -9.46 | 1030.0 | .0 | 36.0 | 5.0 | | |
| 6 | -1.28 | 1141.0 | 1.0 | 38.0 | 4.0 | | |
| 7 | 4.66 | 1019.0 | 1.0 | 38.0 | 2.0 | | |
| 8 | 7.06 | 1012.0 | .0 | 38.0 | 4.0 | | |
| 9 | -12.39 | 925.0 | 1.0 | 36.0 | 4.0 | | |
| 10 | -10.27 | 1195.0 | .0 | 46.0 | 5.0 | | |
| 11 | -3.31 | 845.0 | .0 | 33.0 | 3.0 | | |
| 12 | -7.74 | 902.0 | 1.0 | 46.0 | 5.0 | | |
| 13 | 7.49 | 1076.0 | 1.0 | 43.0 | 6.0 | | |
| 14 | -13.03 | 1320.0 | .0 | 50.0 | 6.0 | | |
| 15 | 8.92 | 1154.0 | 1.0 | 36.0 | 2.0 | | |
| 16 | 2.04 | 1213.0 | 1.0 | 40.0 | 6.0 | | |
| 17 | -.50 | 1220.0 | .0 | 33.0 | 4.0 | | |
| 18 | -.30 | 943.0 | .0 | 38.0 | 4.0 | | |
| 19 | 11.16 | 1064.0 | 1.0 | 34.0 | 1.0 | | |
| 20 | -7.61 | 789.0 | 1.0 | 46.0 | 2.0 | | |
| 21 | -1.21 | 1094.0 | 1.0 | 49.0 | 4.0 | | |
| 22 | .60 | 1212.0 | 1.0 | 40.0 | 4.0 | | |



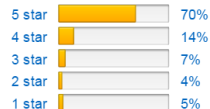
Introduction to Python

➤ Unstructured data: *text*

Customer Reviews

★★★★☆ 8,597

4.4 out of 5 stars



Share your thoughts with other customers

Write a customer review

[See all 8,597 customer reviews](#)

Top Customer Reviews

★★★★☆ **Flawed But Ultimately Thought-Provoking and Worthwhile**

By [Kenny O.](#) on March 1, 2003

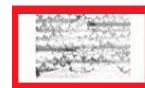
Format: Paperback

Yes, much of what negative reviewers of this book have to say is true: the writing is blunt and simple, the characters lack depth and complexity, it is quite male-focused in its subject matter and language, it has a bunch of quasi-religious mumbo-jumbo, and so on. This book should not be put on the list of great literature for the ages. There are doubtless many novels that cover subject matter from this book far more artfully. As I read the book, I was aware of its hokeyness and lack of redeeming literary qualities. I am, in fact, usually the first person to criticize books that read like this.

And yet, I have to say - and I feel a bit sheepish about this - that I found it meaningful, even profound at times. How can I say this, given my criticisms? First of all, unlike many reviewers, I did not approach this book with great expectations. No one told me that this was Shakespeare or Tolstoy; I had never even heard of it until it was recommended to me recently. And by the end of page 2, I had adjusted my expectations further. This clearly was not going to be winning the Booker prize.

But I found the book moving in its simple way. The characters deliver their statements without subtlety, but subtlety is more a literary virtue than a philosophical one. In fact, I essentially came to view this work as a life philosophy expressed as a fable, so I didn't particularly mind that its messages were not buried far beneath the surface.

Are those messages novel? No, but what of it? Novelists have been recycling themes for centuries, because many themes are of enduring interest and relevance. The point is, the messages are worthwhile and deserving of consideration. [Read more](#)



Don't fly @BritishAirways. Their customer service is horrendous.

Promoted by

9/2/13, 7:57 PM

Introduction to Python

➤ (Semi) structured data: *web logs*

- Try opening the file *access.log.txt* with VS Code

```
1 79.133.215.123 - - [14/Jun/2014:10:30:13 -0400] "GET /home HTTP/1.1" 200 1671 "-"  
  "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)  
  Chrome/35.0.1916.153 Safari/537.36"  
2 162.235.161.200 - - [14/Jun/2014:10:30:13 -0400] "GET  
  /department/apparel/category/featured%20shops/product/adidas%20Kids'%20RG%20III%20M  
  id%20Football%20Cheat HTTP/1.1" 200 1175 "-" "Mozilla/5.0 (Macintosh; Intel Mac  
  OS X 10_9_3) AppleWebKit/537.76.4 (KHTML, like Gecko) Version/7.0.4  
  Safari/537.76.4"  
3 39.244.91.133 - - [14/Jun/2014:10:30:14 -0400] "GET /department/fitness HTTP/1.1"  
  200 1435 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3) AppleWebKit/537.36  
  (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36"  
4 150.47.54.136 - - [14/Jun/2014:10:30:14 -0400] "GET  
  /department/fan%20shop/category/water%20sports/product/Pelican%20Sunstream%20100%20  
  Kayak/add_to_cart HTTP/1.1" 200 1932 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X  
  10_9_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36"
```

Introduction to Python

➤ (Semi) structured data: *web logs*

- Not in a standard tabular format, but the data nonetheless contain precious information
 - One can explore certain patterns in order to impose a tabular structure

```
1 79.133.215.123 - - [14/Jun/2014:10:30:13 -0400] "GET /home HTTP/1.1" 200 1671 "-"  
   "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)  
   Chrome/35.0.1916.153 Safari/537.36"
```


Introduction to Python

➤ (Semi) structured data: *web logs*

- Let's impose a tabular structure on the data
 - Assignment 1

```
1 79.133.215.123 - - [14/Jun/2014:10:30:13 -0400] "GET /home HTTP/1.1" 200 1671 "-"  
"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/35.0.1916.153 Safari/537.36"
```

| IPs | Date | Request | Page | HTTP_Version | First_Code | Second_Code | User_Agent |
|----------------|----------------------|---------|-------|--------------|------------|-------------|--|
| 79.133.215.123 | 14/Jun/2014:10:30:13 | GET | /home | 1.1 | 200 | 1671 | Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.3 6 (KHTML, like Gecko) Chrome/35.0.1916. 153 Safari/537.36 |

Introduction to Python


- One needs to manipulate characters (strings) in order to impose some structure on textual data
 - We will next learn a series of commands to manipulate and process textual data in Python
 - Highly relevant when collecting and pre-processing textual data
 - Web scrapping, text mining

Introduction to Python

➤ Let's start by loading the data

```
file = open('access.log.txt', 'r')  
web_logs = file.readlines()
```

- The variable `web_logs` is a list containing 360,000 elements (log lines)

| Name | ▲ Type | Size | Value |
|--|---------------|--------|--|
| file | TextIOWrapper | | <_io.TextIOWrapper name='access.log.txt' mo |
|  web_logs | list | 360000 | ['79.133.215.123 - - [14/Jun/2014:10:30:13 -04 |

Introduction to Python

- Let's look at the first element in `web_logs`

`web_logs[0]`

- Let's calculate the number of characters in the first log entry

`len(web_logs[0])`

Introduction to Python

- Now let's start imposing a tabular structure on our data
 - First step: obtain the variable IP
 - Question 1 a), Assignment 1
 - Which pattern can we explore?
 - How can one retrieve IP addresses without hardcoding positions?

```
1 79.133.215.123 - - [14/Jun/2014:10:30:13 -0400] "GET /home HTTP/1.1" 200 1671 "-"  
  "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)  
  Chrome/35.0.1916.153 Safari/537.36"  
2 162.235.161.200 - - [14/Jun/2014:10:30:13 -0400] "GET  
  /department/apparel/category/featured%20shops/product/adidas%20Kids'%20RG%20III%20M  
  id%20Football%20Cheat HTTP/1.1" 200 1175 "-" "Mozilla/5.0 (Macintosh; Intel Mac  
  OS X 10_9_3) AppleWebKit/537.76.4 (KHTML, like Gecko) Version/7.0.4  
  Safari/537.76.4"
```

Introduction to Python

- First approach to obtain IP addresses: split the input log data into several parts based on another string
 - In our case, split each log entry based on the presence of blank spaces
 - Function: `split()`
 - Example: splitting the first log entry
`words = web_logs[0].split(" ")`
 - Let's look at the content of `words`



79.133.215.123
-
-
[14/Jun/2014:10:30:13
-0400]
"GET
/home
HTTP/1.1"
200
1671
"_"
"Mozilla/5.0
(Windows
NT

Introduction to Python

- Let's now retrieve all the IP addresses
 - Using a for-loop

```
IP = []
```

```
for log in web_logs:
```

```
    words = log.split()
```

```
    IP.append(words[0])
```

Introduction to Python

- Let's now retrieve all the IP addresses
 - Using list comprehension

```
IP = [web_logs[i].split(" ")[0] for i in range(len(web_logs))]
```


Introduction to Python

➤ Second approach to obtain IP addresses


■ Procedure

1. Locate the index (position) of the first blank space
2. Obtain all characters from a log entry starting at position 0 and ending at the position right before the position of the first blank space

IP address:

position 0 to 13

first space: position 14



```
1 79.133.215.123 - - [14/Jun/2014:10:30:13 -0400] "GET /home HTTP/1.1" 200 1671 "-"
  "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/35.0.1916.153 Safari/537.36"
```

Introduction to Python

- Locating the index (position) of the first blank space
 - One can do this using **regular expressions**
 - Very powerful language with its own syntax and semantics
 - Concise and powerful method for *searching for a specific string pattern*
 - Example: how can you automatically search for names of video games in a given text?
 - Common pattern: The <something> of <something>
 - The <World> of <Warcraft>
 - The <Lord> of <the Rings>
 - The <Legend> of <Zelda>
 - The <House> of <the Dead>
 -

Introduction to Python

- Locating the index (position) of the first blank space
 - The pattern we are looking for is very simple
 - The first occurrence of blank space
 - The function `search` from the `re` module returns the first position of a searched pattern
 - Let's search for the first occurrence of the pattern containing only the blank space in the first log entry

```
import re
hits = re.search(" ", web_logs[0])
pos_first_space = hits.start(0)
```

Introduction to Python

- Locating the index (position) of the first blank space
 - If we wanted all the positions containing single blank spaces, we would then use the `finditer` function

```
hits= re.finditer(" ", web_logs[0])
```

```
all_space_positions = [m.start(0) for m in hits]
```

Introduction to Python

- Now, we can obtain all characters from position 0 to *pos_first_space* by substring a string
 - Notation: `string[start:end:step]`
 - Remember that the element in the end position is not included

```
string = web_logs[0]
```

```
first_ip = string[0:pos_first_space:1]
```

Introduction to Python

- Even though we have focused on the first log line, it is possible to extend what we did to all logs

Get the result of the search
for the first space

← For each log (line) in web_logs

```
all_pos_first_space = [re.search(" ", log).start(0) for log in web_logs]
```

Access one log (str) Substring the log For each value from 1 to 360,000

```
IPs = [web_logs[i][0:all_pos_first_space[i]:1] for i in range(len(web_logs))]
```

Introduction to Python

➤ More on regular expressions

- These may (or may not) be useful in Assignment 1
- Finding the first position of the string HTTP

```
all_pos_HTTP = [re.search("HTTP", log).start(0) for log in web_logs]
```

- Finding the first position of a double quotation mark (")

```
all_pos_first_quote = [re.search("\"", log).start(0) for log in web_logs]
```

- Finding the first position of a left square bracket ([)

- The need for escaping will be explained in Lecture 8

```
all_pos_first_bracket = [re.search("[", log).start(0) for log in web_logs]
```

Introduction to Python

- What about finding the position of the second blank space
 - Can you do the same using list comprehension

```
all_pos_second_space = []
```

```
for logs in web_logs:
```

```
    hits = re.finditer(" ", logs)
```

```
    positions = [m.start(0) for m in hits]
```

```
    all_pos_second_space.append(positions[1])
```

- We will learn about more advanced regular expressions in upcoming lectures

Introduction to Python

➤ Potentially useful “trick” for Assignment 1

- You can always remove an obtained variable from the original data set after collecting that variable
- Example: suppose you obtained the variable **IP** by following the previous steps. You can then remove **IP** from the original data set:

```
modified_logs = [web_logs[i][all_pos_first_space[i]+1:] for i in range(len(web_logs))]
```


i-th log Substring *i*-th log from after first space to the end

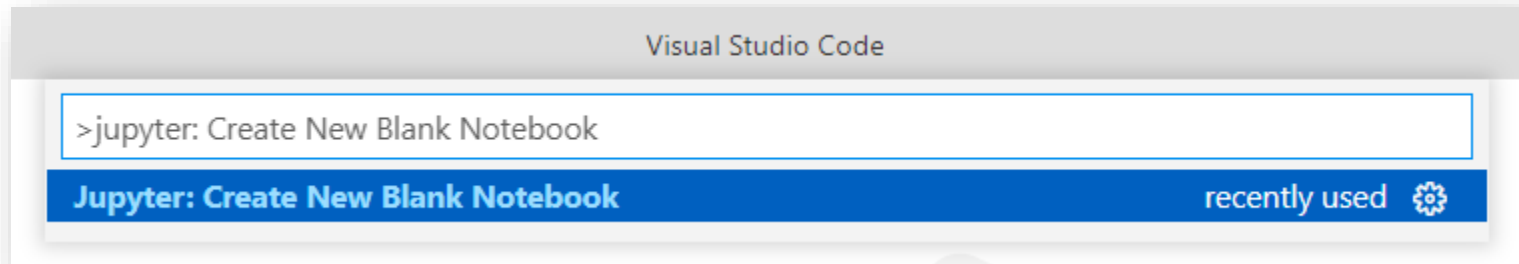
The above might make your life easier

- *E.g.*, you will never have to search for the second blank space

Assignment 1: Head Start

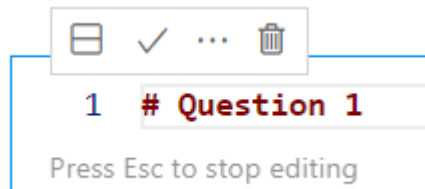
Assignment 1

- Let's put things together to get a head start in Assignment 1
 - Remember: you must submit a *.ipynb* file
 - Start by creating a new Jupyter notebook
 - Press F1 and type “*Jupyter: Create New Blank Notebook*”

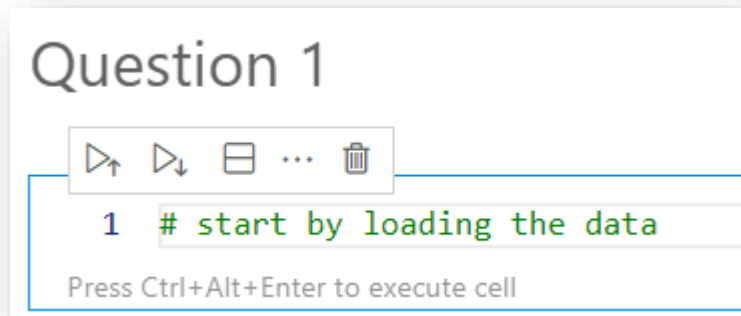


Assignment 1

- Add question-related comments using Markdown



- Add Python code with explanatory notes (optional)



Assignment 1

- Write the Python code to load the required data set

Question 1

```
1 # start by loading the data
2 file = open('access.log.txt', 'r')
3 web_logs = file.readlines()

[1] ✓ 0.2s
```




- Let's add the solution to Q1 - a)

Q1-a)

```
1 import re
2
3 # finding all the positions of the first space
4 all_pos_first_space = [re.search(" ", log).start(0) for log in web_logs]
5
6 # substring all logs from position to first space
7 IP = [web_logs[i][0:all_pos_first_space[i]:1] for i in range(len(web_logs))]
```

Assignment 1

- Run the previous code chunk for the sake of testing

| PROBLEMS OUTPUT TERMINAL JUPYTER: VARIABLES DEBUG CONSOLE | | | | |
|--|---------------|--------|--|--|
| | | | | |
| Name | ▲ Type | Size | Value | |
|  all_pos_first_spac | list | 360000 | [14, 15, 13, 13, 13, 12, 13, 13, 14, 13, 14, 11, 14, | |
| file | TextIOWrapper | | <_io.TextIOWrapper name='access.log.txt' mo | |
|  IP | list | 360000 | ['79.133.215.123', '162.235.161.200', '39.244.91 | |
|  web_logs | list | 360000 | ['79.133.215.123 - - [14/Jun/2014:10:30:13 -04] | |

Assignment 1

- Save your source code
 - *File -> Save*
 - *E.g., “Assignmet_1.ipynb”*
 - This is what you will submit on Canvas after you are done with all the questions

Assignment 1

- Remember that, unless otherwise stated, you can solve any question however you want to
 - Use any modules, techniques, functions, ...
 - Obviously, whatever you do must be accurate

Summary

- We learned how to manipulate textual data
- This was the last lecture on the basics of Python
 - We will continue to learn about Python in the next lectures, but in a much more applied context
 - Mostly, using modules
- Next lecture: collecting data via web scrapping

Copyright 2021 Arthur Carvalho. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed without explicit written consent.