

Statistical Computing

Gabriele Rovigatti

Bank of Italy

March 4, 2021



This Course

- Python basics
 - General Framework
 - Data Types
 - Syntax
 - for loops, if/else
 - Modules / Libraries
 - Functions
- Webscraping
 - Data Storage
 - HTML syntax
 - Requests package
 - Webdriver package
- Text Mining?

Today

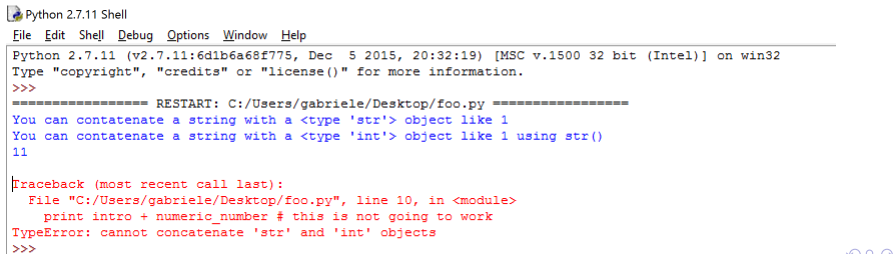
- **Python basics**
 - **General Framework**
 - **Data Types**
 - **Syntax**
 - **for loops, if/else**
 - Modules / Libraries
 - Functions
- Webscraping
 - Data Storage
 - HTML syntax
 - Requests package
 - Webdriver package
- Text Mining?

Best friends - Error Messages

In case of error, every interpreter/software will try to tell you what is wrong in order for you to fix it.

Objects and errors

```
string_number = '1'
numeric_number = 1
intro = 'You can concatenate a string with a '
print(intro + str(type(string_number)) + ' object like ' + string_number)
print(intro + str(type(numeric_number)) + ' object like ' + str(numeric_number) + ' using str()')
print(10 + numeric_number) # sum numeric objects
print(intro + numeric_number) # this is not going to work
```



```
Python 2.7.11 Shell
File Edit Shell Debug Options Window Help
Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec 5 2015, 20:32:19) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/gabriele/Desktop/foo.py =====
You can concatenate a string with a <type 'str'> object like 1
You can concatenate a string with a <type 'int'> object like 1 using str()
11
Traceback (most recent call last):
  File "C:/Users/gabriele/Desktop/foo.py", line 10, in <module>
    print intro + numeric_number # this is not going to work
TypeError: cannot concatenate 'str' and 'int' objects
>>>
```

Best friends - Google

The screenshot shows a Google search interface in a browser. The search bar contains the text "how to make a course on python web scraping". Below the search bar, there are tabs for "Tutti", "Video", "Immagini", "Notizie", "Shopping", "Altro", "Impostazioni", and "Strumenti". The search results show approximately 1,100,000 results in 0.79 seconds. The first three results are:

- Beginner's guide to Web Scraping in Python (using BeautifulSoup)**
<https://www.analyticsvidhya.com> > Business Analytics > Traduci questa pagina
22 ott 2015 - Web scraping is a technique of extracting information from websites. ... This would not only require finding out new courses, but also scrape the web for ... I prefer BeautifulSoup (python library), since it is easy and intuitive to ...
- Python Web Scraping Tutorial using BeautifulSoup - DataQuest**
<https://www.dataquest.io/blog/web-scraping-tutorial-python/> > Traduci questa pagina
17 nov 2016 - Before we get started, if you're looking for more background on APIs or the csv format, you might want to check out our Dataquest courses on ...
- First web scraper - Read the Docs**
<https://first-web-scraper.readthedocs.io/> > Traduci questa pagina
A step-by-step guide to writing a web scraper with Python. The course assumes the reader has little experience with Python and the command line, ... Python 2.7 is preferred but you can probably find a way to make most of this tutorial work ...

At the bottom of the screenshot, there is a taskbar with various application icons and a system tray showing the time as 15:49 on 23/02/2017.

Best friends - Stack Overflow

The screenshot shows a web browser window displaying a Stack Overflow question. The browser's address bar shows the URL: `http://stackoverflow.com/questions/41215636/python-requests-and-beautifulsoup4-colle`. The Stack Overflow logo and navigation links (Questions, Jobs, Documentation, Tags, Users) are visible at the top. The question title is "Python requests and BeautifulSoup4, collecting only the href links". The question body contains a code snippet for using BeautifulSoup and requests to extract hrefs from a webpage. The user's comment below the code states: "I would like to print only the hrefs but I cannot seem to figure this out. I've looked at different videos and can't get it. What am I doing wrong? I know the above code is printing the contents of the 'a' tag but I need just the href's." The right sidebar shows the question was asked 2 months ago, viewed 38 times, and is active. Below this, a "Related" section lists other questions. The bottom of the image shows the Windows taskbar with various application icons and the system clock showing 16:30 on 23/02/2017.

beautifulsoup - Python re... X +

http://stackoverflow.com/questions/41215636/python-requests-and-beautifulsoup4-colle Cerca

stackoverflow Questions Jobs Documentation BETA Tags Users Search... Log In Sign Up

Python requests and BeautifulSoup4, collecting only the href links

```
from bs4 import BeautifulSoup
import requests

url = "https://www.brightscope.com/ratings"
headers = {'User-Agent': 'Mozilla/5.0'}
page = requests.get(url)
soup = BeautifulSoup(page.text, "html.parser")

data = soup.find_all('li', {"class": "more-data"}) + soup.findAll('li', {"class": "more-data topt
for item in data:
    print(item('a'))
```

I would like to print only the hrefs but I cannot seem to figure this out. I've looked at different videos and can't get it. What am I doing wrong? I know the above code is printing the contents of the "a" tag but I need just the href's.

python beautifulsoup html-parsing python-requests

asked 2 months ago
viewed 38 times
active 2 months ago

Related

- 1 How to extract text, with the link and another text python
- 2 How to ignore empty lines, next_sibling in BeautifulSoup
- 9 pip install requests exception BeautifulSoup4 exception

Chiedimi qualcosa 16:30 23/02/2017

Why Python?

- Python is a high-level, interpreted language
- No need to pre-specify the variable type
- “Poor” syntax - faster to learn
- Clear, readable, understandable
- Freely available, very popular (i.e., a lot of support!)
- Not good for memory-intensive tasks, not as fast as C/C++
- Continuously evolving, extremely flexible (from machine learning to webscraping)

Python Interactive Console

After installing Python

Ubuntu

Windows 10

- Linux/Mac: from

terminal
- Windows: from

prompt

Typing `$python` in the terminal/prompt should yield

```
Python 3.5.2 (default, Nov 17 2016, 17:05:23)
```

```
[GCC 5.4.0 20160609] on linux
```

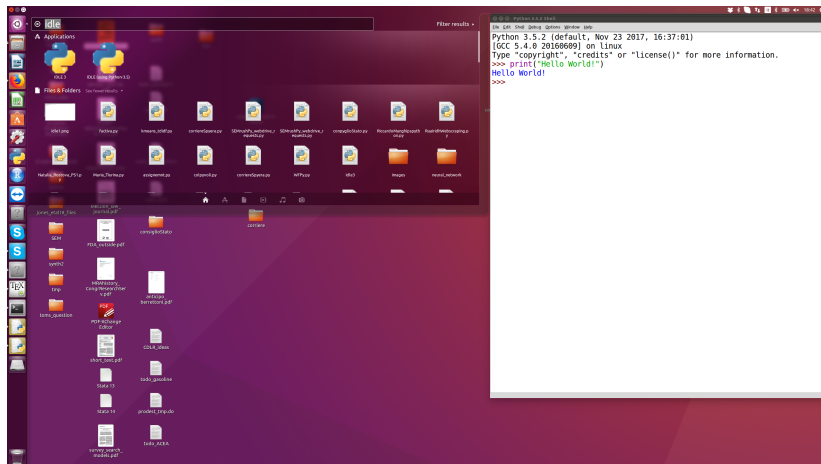
```
Type "help", "copyright", "credits" or "license" for more info
```

```
>>>
```

Different python versions (type `$python2.7` or `$python3`)

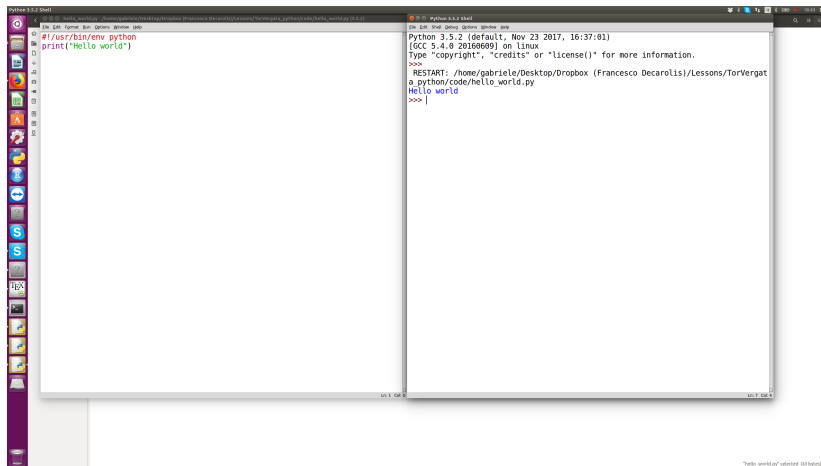
Python IDLE/Shell

The Integrated Development and Learning Environment comes by default with Python



Write Codes: Hello World!

```
#!/usr/bin/env python  
print("Hello world")
```



The screenshot shows a Linux desktop environment. On the left, a text editor window titled 'hello_world.py' contains the following code:

```
#!/usr/bin/env python  
print("Hello world")
```

On the right, a terminal window titled 'Python 3.5.2 Shell' shows the execution of the script. The prompt is 'Python 3.5.2 (default, Nov 23 2017, 16:37:01) [GCC 5.4.0 20160609] on linux'. The user enters '>>>' and the prompt changes to 'RESTART: /home/gabriele/Desktop/Dropbox (Francesco Decarolis)/Lessons/TorVergata/python/code/hello_world.py'. The user enters 'Hello world' and the prompt changes to '>>>'. The output of the script is 'Hello world'.

Strings

```
1 # this is a string
2 str_sentence = 'This is a very simple string'
3 # this is a string, too, but multi-line
4 print(''How many lines must
5 a multi-line string have?''')
6 # this is a multi-line string with different quotes
7 print("""Let's see whether we can make
8 a multi-line object as well""")
9 # print() and concatenate methods
10 x = 'heaven'
11 z = 'hell'
12 print(x)
13 print("Can you tell %s from %s?" % (x,y))
```

Strings are **iterable** objects

[Iterables - Definition](#)

Numbers

```
1 # Integers
2 birth_year = 1899
3 birth_year_from_string = int0("1899")
4
5 # Floating Points
6 pi = 3.14159265
7 pi = float("3.14159265")
8
9 # Fixed Points
10 from decimal import Decimal
11 price = Decimal("0.02")
```

Lists

The most basic data structure in Python is the sequence. Each element of a sequence is assigned a number - its position or index. The first index is zero, the second index is one, and so forth. The most common sequences are the lists.

```
1 # Initiate empty lists
2 empty_list = []
3 numeric_list = [1,1,2,3,5,8]
4 string_list = ['Fibonacci', 'Series', 'In', 'List']
5
6 # append() method
7 numeric_list.append(13)
8
9 # extend() method
10 numeric_list.extend([21,34])
```

append() vs. extend()

Working with Lists

```
1 numeric_list = [1,1,2,3,5]
2
3 # length of lists
4 len(numeric_list)
5
6 # identify list elements
7 numeric_list[0]
8 numeric_list[0:2]
9 numeric_list[4:5]
10
11 # multiple elements with different indices: for loop trick
12 indices = [0,3]
13 new_list = [ numeric_list[i] for i in indices ]
```

Dictionaries

Dictionaries are (unordered) collections of items. Dictionary elements are key:value pairs. keys can be integers or strings.

```
1         # dict() method
2 my_dict = dict()
3 # or curly brackets
4 my_dict = {
5     1 : 'third',
6     2 : 'second',
7     3 : 'first'}
8 # keys can be integers or strings
9 tcid_ym = {'first' : 3, 'second': 2, 'third' : 1}
10
11 print(my_dict[1])
12 print(tcid_ym['first'])
```

Working with Dictionaries

Elements in dictionaries are called by either `[key]` or by the `get()` method. The latter returns *None* instead of *KeyError* if the key is not in the dictionary

```
1 # Initiate a dictionary
2 lessons = {
3     1 : 'Basics',
4     2 : 'Syntax',
5     3 : 'Webscraping'
6 }
7 lessons[1] # 'Basics'
8 lessons[1] = 'Basics + Syntax Elements' # CHANGE item
9 lessons.get(1) # 'Basics + Syntax Elements'
10 lessons[116] = 'Machine Learning' # ADD item
11 print(lessons) # {1: 'Basics + Syntax', 2: 'Syntax', 3: 'Webscraping', 116: 'Machine Learning'}
12 lessons.get(115) # NULL
13 lesson[115] # KeyError: 115
```


Working with Dictionaries / 2

```
1 # length of a dictionary is the number of "key : value" pairs
2 len(lessons) # 4
3 # 'in' is used to check whether a key exists in a dictionary (returns a bool)
4 4 in lessons # True
5 # update() method adds new "key : value" pairs to the dictionary
6 additional_lessons = {4 : 'Statistical Computing', 5 : 'Research'}
7 lessons.update(additional_lessons)
8 # pop() method: removes the item and returns the value
9 lessons.pop(2) # 'Syntax'
10 # del: removes the item
11 del(lessons[3])
12 # .values(), .keys() and items() methods: return values, keys and tuples as lists
13 lessons.items() # dict_items([(1, 'Basics + Syntax Elements'), (116, 'Machine Learning')])
14 lessons.values() # dict_values(['Basics + Syntax Elements', 'Machine Learning'])
15 lessons.keys() # dict_keys([1, 116, 5, 4])
16 # clear() method deletes all items at once
17 lessons.clear()
18 # del: removes the dictionary itself
19 del(lessons)
```

Many ways of coding - same outcome

Beginner

```
print("""There are 2 things to do in order to start working with Python:
1) download and install it at "http://www.python.it/download/";
2) follow closely a very good tutorial""")
```

Intermediate

```
print("""There are %g things to do in order to start working with Python: \n
%g) download and install it at "http://www.python.it/download/"; \n
%g) follow closely a very good tutorial."" " % (2,1,2))
```

Advanced

```
def printer(steps):
    p = range(steps)[1::]
    return(max(p),min(p),p[1])
if __name__ == '__main__':
    print("""There are %g things to do in order to start working with Python: \n
    %g) download and install it at "http://www.python.it/download/";\n
    %g) follow closely a very good tutorial. "" " % printer(3))
```

Indentation - when blanks do matter

In Python, leading blanks (*indentation*) are crucial in loops!

Indentation

```
for i in range(3):
print('This is invalid syntax!') #after a 'for','if',ect. Python expects at least one indentend command

# blanks matter, but not everywhere
b0=1
b1 = 1
b2 = 1
print(b0, b1, b2) #these are all equivalent
b = list() #initiate an empty list, equivalent to b = []
for i in range(3):
    b.append(i)
    print('We are INSIDE the loop, hence will print ' + str(b)) #it will print the list b at each iteration
print('We are OUTSIDE the loop, hence correctly prints ' + str(b)) #it will print b once
```

Run

For loops

The Python `for` statement iterates over the members of a sequence - or numbers, or a string - in order, executing a block of code each time.

```
1 for iterating_var in sequence:  
2     statement(s)
```

The first item in the sequence is assigned to the iterating variable `iterating_var`. Next, the statements block is executed. Each item in the list is assigned to `iterating_var`, and the `statement(s)` block is executed until the entire sequence is exhausted.

For Loops - Examples

```
1 for letter in 'Python':      # First Example
2     print('Current Letter :', letter)
3
4 fruits = ['banana', 'apple', 'mango']
5 for fruit in fruits:        # Second Example
6     print('Current fruit :', fruit)
7
8 print("Good bye!")
```

if..elif..else

An else statement can be combined with an if statement. An else statement contains the block of code that executes if the conditional expression in the if statement resolves to 0 or a FALSE value.

The else statement is an optional statement and there could be at most only one else statement following if.

```
1 if expression:
2     statement(s)
3 else:
4     statement(s)
```

Dictionary Comprehension

- is a concise way to generate a dictionary from an iterable object;
- consists of a (key: value) pair followed by *for* statement

```
1 # generate the square of the fibonacci series
2 fibonacci_list = [1,1,2,3,5,8,13] # ITERABLE object
3 fibonacci_sq_dict = {x : x*x for x in fibonacci_list}
4 fibonacci_sq_dict # {1: 1, 2: 4, 3: 9, 5: 25, 8: 64, 13: 169} *** look at t
5 # the equivalent for loop reads
6 fibonacci_sq_dict_long = {} # initiate the dict
7 for x in fibonacci_list:
8     fibonacci_sq_dict_long[x] = x*x
9 # we may want to add more if and for statements
10 fibonacci_sq_dict_odd = {x : x*x for x in fibonacci_list if x%2 == 1}
11 fibonacci_sq_dict_odd
```

Exercises

- Write a Python script to concatenate following dictionaries to create a new one [solution](#)
 - dict1 = {1:10, 2:20}
 - dict2 = {3:30, 4:40}
 - dict3 = {5:50, 6:60}
- Write a Python script to generate and print a dictionary that contains the square of numbers between 1 and n (`n=int(input("Input a number "))`) allows you to input integers). `n = 10` [solution](#)
- Write a Python program to sum all the items in the previous dictionary [solution](#)
- Write a Python program to map two lists into a dictionary [solution](#)
 - colors = ['red', 'green', 'blue']
 - value = ['#FF0000', '#008000', '#0000FF']
- Write a Python program to create a dictionary from a string. Sample string : "python for webscraping" [solution](#)

Ubuntu - Install

```
gabriele@gabriele-HP-EliteBook-840-G3:~$ sudo apt-get install python2.7
```

```
gabriele@gabriele-HP-EliteBook-840-G3:~$ sudo apt-get install python3
```

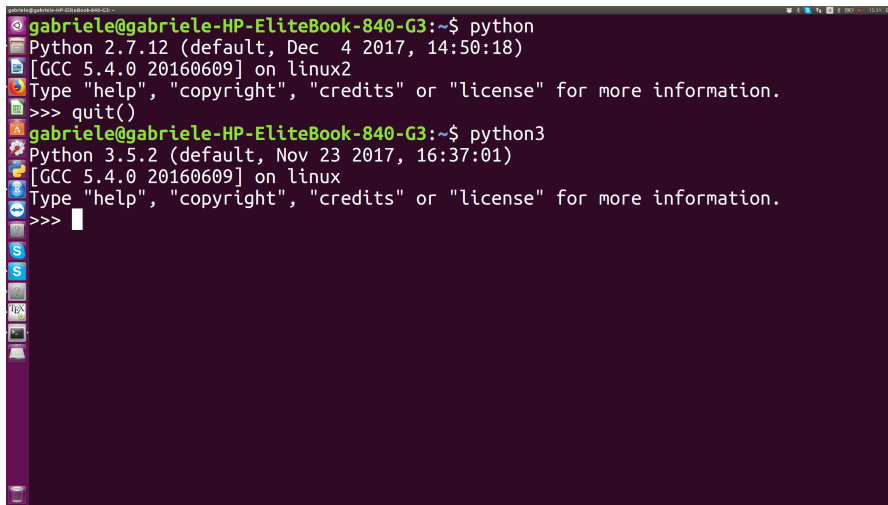
Windows - Install

The screenshot shows a Windows desktop with a Taskbar on the left containing icons for various applications. The main window is a web browser displaying the Python.org website. The website has a dark blue header with the Python logo and navigation links: About, Downloads, Documentation, Community, Success Stories, News, and Events. A search bar is located on the right. The main content area features the heading "Download the latest version for Windows" and two buttons: "Download Python 3.6.4" and "Download Python 3.7.14". Below these buttons, there are links for "Wondering which version to use?", "Looking for Python with a different OS?", and "Want to help test development versions of Python?". A section titled "Looking for a specific release?" lists Python releases by version number and release date, with a "Click for more" link for each release.

Release version	Release date	Download	Click for more
Python 3.4.8	2016-02-09	Download	Release Notes
Python 3.5.9	2016-02-09	Download	Release Notes
Python 3.6.4	2017-12-19	Download	Release Notes
Python 3.6.5	2017-10-09	Download	Release Notes
Python 3.6.7	2017-09-19	Download	Release Notes
Python 3.7.14	2021-09-14	Download	Release Notes
Python 3.6.7	2017-08-08	Download	Release Notes

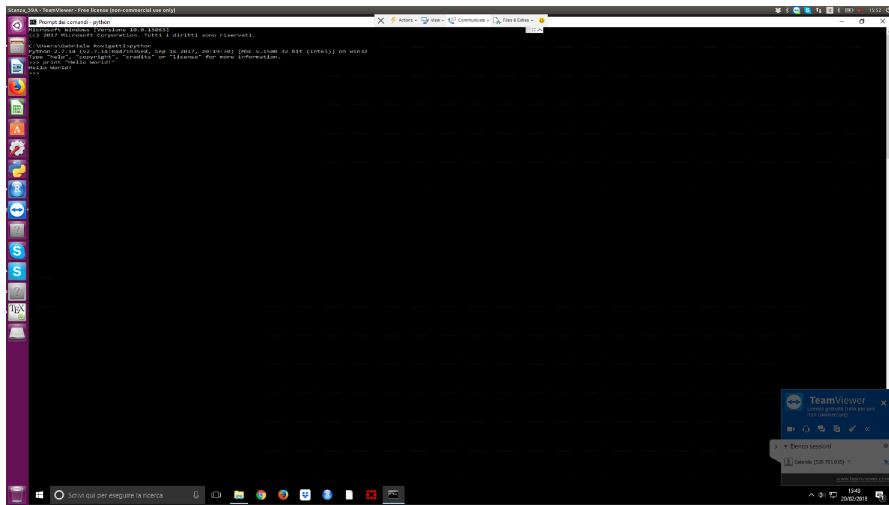
back

Linux - Terminal

A terminal window with a dark purple background and a vertical sidebar of application icons on the left. The terminal text shows the execution of 'python' and 'python3' commands, displaying version information and GCC details for Linux. The prompt is 'gabriele@gabriele-HP-EliteBook-840-G3:~\$'.

```
gabriele@gabriele-HP-EliteBook-840-G3:~$ python
Python 2.7.12 (default, Dec  4 2017, 14:50:18)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()
gabriele@gabriele-HP-EliteBook-840-G3:~$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

Windows - Prompt



```
Stanza_30A - TeamViewer - Free license (non-commercial use only)
Python 2.7.14 (tags/v2.7.14:150728e, Sep 10 2017, 00:10:10) [AMD64 (x64)] on win32
Type "help()", "copyright()", "credits()" or "license()" for more information.
>>> print "Hello World!"
Hello World!
>>>
```

back

Definition of Iterable [Back](#)

Iteration is a general term for taking each item of something, one after another. In Python, **iterator** and **iterable** have specific meanings.

- An **iterator** is an object with a `next` (Python 2) or `__next__` (Python 3) method:
 - Whenever you use a `for` loop, or `map`, or a list comprehension, etc. in Python, the `__next__` method is called automatically to get each item from the iterator
- An **iterable** is an object that has an `__iter__` method which returns an iterator, or which defines a `__getitem__` method that can take sequential indexes starting from zero (and raises an `IndexError` when the indexes are no longer valid)

append() vs. extend() method

The `list.append()` method appends **one** object to a list. The `list.extend()` method extends a list by appending elements from an iterable.

```
1 my_list = ['this', 'is']
2 addendum1 = ['a', 'list']
3 addendum2 = ['of', 'strings']
4 print(my_list.extend(addendum1))
5 print(my_list.append(addendum2))
```

back