Intro to Python for R Users

Adam Kaplan

February 25, 2022

Python vs. R

Python and R are fairly similar. This is a quick overview of the differences to help you get up to speed.

Importing packages

- Importing packages is similar: instead of `[R] library(mypackage)`, do `[Python] import mypackage`.

Importing packages

- Importing packages is similar: instead of `[R] library(mypackage)`, do `[Python] import mypackage`.
- Python also lets you import specific functions from a package: `from mypackage import cool_function`

Importing packages

- Importing packages is similar: instead of `[R] library(mypackage)`, do `[Python] import mypackage`.
- Python also lets you import specific functions from a package: `from mypackage import cool_function`
- You can also rename packages if they're too long: `import numpy as np`

Importing packages

- Importing packages is similar: instead of `[R] library(mypackage)`, do `[Python] import mypackage`.
- Python also lets you import specific functions from a package: `from mypackage import cool_function`
- You can also rename packages if they're too long: `import numpy as np`
- Installing packages is slightly different however: `[R] install.packages('mypackage')` as opposed to `[Python] pip install mypackage` OUTSIDE of Python either in the command line or Jupyter Notebooks.

What are all those dots for?

(Or, methods, attributes, and namespaces)

What are all those dots for?

(Or, methods, attributes, and namespaces)

- Dots have special meaning in Python. It's not like R, where people put dots in all sorts of names.

What are all those dots for?

(Or, methods, attributes, and namespaces)

- Dots have special meaning in Python. It's not like R, where people put dots in all sorts of names.
- Python is much more careful about keeping packages' functions attached to the functions. If the `requests` library has a function called `get`, you call it like this `requests.get()`. This reminds you where the `get` function came from and prevents you from overwriting some other package's `get` function.

What are all those dots for?

(Or, methods, attributes, and namespaces)

- Dots have special meaning in Python. It's not like R, where people put dots in all sorts of names.
- Python is much more careful about keeping packages' functions attached to the functions. If the `requests` library has a function called `get`, you call it like this `requests.get()`. This reminds you where the `get` function came from and prevents you from overwriting some other package's `get` function.
- Python is also more "object oriented" than R. Objects often have built in or attached functions, called methods.

What are all those dots for?

- Methods are called with a dot notation.

```
[R] strsplit("Adam Kaplan", " ")
```

and

```
[Python] "Adam Kaplan".split(" ")
```

What are all those dots for?

- Methods are called with a dot notation.

```
[R] strsplit("Adam Kaplan", " ")
```

and

```
[Python] "Adam Kaplan".split(" ")
```

- Objects can also have attributes, which are pieces of data attached to an object. Example: `adam.subfields = ['methods', 'international relations']`

Data Structures (Lists)

Like R's vectors, Python uses a lot of lists. These are ordered arrays. Note that Python starts with 0!

```
my_list = [x, y, z]
> my_list[0]
x
```

Data Structures (Dictionaries)

Python has a data structure called a dictionary, which are like lists that you access by key name instead of by position (think a more general form of R's dataframes). Example:

```
article = {"title": "Rivalry and Revenge",
           "author" : "Balcells",
           "year" : "2017"}
```

Data Structures (Dictionaries)

Python has a data structure called a dictionary, which are like lists that you access by key name instead of by position (think a more general form of R's dataframes). Example:

```
article = {"title": "Rivalry and Revenge",
           "author" : "Balcells",
           "year" : "2017"}

> article.keys()
['title', 'author', 'year']
```

Data Structures (Dictionaries)

Python has a data structure called a dictionary, which are like lists that you access by key name instead of by position (think a more general form of R's dataframes). Example:

```
article = {"title": "Rivalry and Revenge",
           "author" : "Balcells",
           "year" : "2017"}
```

```
> article.keys()
['title', 'author', 'year']
```

```
> article['author']
"Balcells"
```

Loops and functions

Functions are only slightly different than in R:

```
def my_function(x):
    z = (x + 2)^2
    return z
```

Loops and functions

Functions are only slightly different than in R:

```
def my_function(x):
    z = (x + 2)^2
    return z
```

Loops are fast and nice in Python, unlike in R, and are very easily done:

```
for i in my_list:
    my_function(i)
```

Loops and functions

Functions are only slightly different than in R:

```
def my_function(x):
    z = (x + 2)^2
    return z
```

Loops are fast and nice in Python, unlike in R, and are very easily done:

```
for i in my_list:
    my_function(i)
```

Pro move: list comprehensions:

```
[my_function(i) for i in my_list]
```

Whitespace

- As you can tell, Python makes heavy use of whitespace to set apart different levels of functions, for loops, etc. Use four spaces (Jupyter converts tabs to four spaces automatically.

```
def my_function(big_list):
    print(len(big_list))
    for l in big_list:
        for i in l:
            ...
    return stuff
```

Whitespace

- As you can tell, Python makes heavy use of whitespace to set apart different levels of functions, for loops, etc. Use four spaces (Jupyter converts tabs to four spaces automatically.
- No need for curly braces!

```
def my_function(big_list):
    print(len(big_list))
    for l in big_list:
        for i in l:
            ...
    return stuff
```

Scraper time

Time to scrape! Go to

https://github.com/akapl0/PML_Web_scraping