Python vs. R

Zoe Vernon, Andrew Vaughn, James Duncan

10/14/22

Table of contents

Comparing the Behavior of Python and R **1**
- Instructions . . . 1
  - No shows . . . 3
- Questions . . . 3
  - Main Questions . . . 3
  - Additional Questions . . . 3

PDF

Comparing the Behavior of Python and R

This week, we will be exploring the behavior of Python for several common actions that we have previously covered in R. The final product of today's work will be a PDF which is uploaded as a group submission to Gradescope.

For quick reference, there is some example Python code in `syntax.py`.

Instructions

> **!** Important
>
> Please carefully read the full instructions before starting the assignment, as you will need to decide with your group how to organize and combine your efforts.

Your group is expected to work on this for ~**60 minutes** (so office hours will be cut short by about 10 minutes today).

It's OK if you don't get through all of the questions in one hour (i.e., you can just submit what you have at the end of section), but it should be clear that you put some thought and effort into the questions your group worked on.

> **Tip**
>
> Do your best to document your efforts and insights as you go.

1. We will separate into groups of (ideally) 3 or (if necessary) 4. I'll try to make sure that each group has at least one person with some Python experience, so **please wait to get started** until we've figured that out.
2. Groups should try to answer every question in the Main Questions section. Here are some **options** for working through these questions:
   - Option A – **Divide, consult, conquer**: Individual group members work on different questions, consulting with each other as they go or as needed.
   - Option B – **Many minds, one task**: The group works together, discussing and answering each question sequentially.
   - Option C – **Go with the flow**: Feel free to use a flexible mixture of Option A and Option B, working separately on some tasks and uniting on others as you see fit.
3. At the end of section (say, the last ~10-15 minutes), combine the solutions into **one PDF**. Each group member will then **individually submit** copies of this PDF on Gradescope.
   - Make sure to include the name of each group member at the top of the PDF.
   - How you create the PDF is up to you but some **options** are:
     a. (recommended) Using **Google Docs**.
        - *Separate documents, then combine*: Each group member uses a separate Google Doc, then one person combines those documents into a final shared document.
        - *One document*: Everyone adds to a single document at the same time.
        - You can take screenshots of code / outputs that you run in Jupyter or RStudio (alternatively, the IPython or R consoles), or just copy paste code and outputs directly to the document.
        - When your final shared document is ready, each group member will export it to PDF and make individual submission to Gradescope.

     b. Create an **R Markdown** file with the code and render to PDF. This may be more of a headache to combine if each person is working separately.

     c. Create a **Jupyter Notebook** and export to PDF.
        - In JupyterLab, **File > Save and Export Notebook As... > PDF**.
        - In the classic Jupyter Notebook interface, **File > Download as > PDF via LaTeX (.pdf)**.

No shows

If you miss lab this week, you can form a group of 2-3 students and submit this assignment on Gradescope by Monday October 17th at 10pm.

Questions

Main Questions

> Tip
>
> Ideally, you will make it through all of these, although if you run out of time that is okay.

1. Do Python functions behave like pass-by-value or pass-by-reference? In other words, if you pass in an object and modify it, does that affect the value of the object in the environment from which the function was called? Check this for a scalar, a list, and a `numpy` array.
2. If you copy a list, dictionary, or `numpy` array in Python, are the values copied or does the new object just use the same memory as the original object?
3. How are `NA`s handled in Python lists? What about in `numpy` arrays?
4. Do Python functions use promises/lazy evaluation?
5. How does variable scoping work in Python - does it use lexical scoping and look for variables in the environment where a function was defined?
6. Consider the relative efficiency of `for` loops versus vectorized calculations for `numpy` arrays and see how it compares to the equivalent operation in R.
7. Can lists and `numpy` arrays be modified in place, without copying the object?
8. Consider whether Python allows you to have functions and variables in the global environment that have the same names as functions/variables in packages or in modules (e.g., make a file `test.py` in your working directory that you can import using `import test`). Consider `math.cos` and create your own `cos` function. How does this compare to how R finds objects?

Additional Questions

> Tip
>
> Work on these if you finish quickly/are curious

1. Can you create a closure with embedded data, like we did in R?
2. Can you determine if the speed of looking up values in a dictionary varies with the size of the dictionary (this will indicate if something like hashing is going on or if the look up has to scan through all the elements).
3. Compare the Python debugger to R's debugger.

4. If you create classes and objects in Python's object-oriented system, what are the similarities and differences relative to R's R6 system? There is a brief section on object-oriented programming in Python in the lab 6 materials.