DSC 201: Data Analysis & Visualization

Python Variables and Functions

Dr. David Koop
Chicago Food Inspections

- Data: Information about food facility inspections in Chicago
- Data Source: https://data.cityofchicago.org/Health-Human-Services/Food-Inspections/4ijn-s7e5/data
- Fields: Name, Facility Type, Risk, Violations, Location
- What questions are interesting about this dataset?
- Example Notebook:
Chicago Food Inspections

• Two important facets of data analysis and visualization (and this course):
  - What questions did we analyze?
  - What tasks did we do?

• Learn the techniques to perform the tasks we need to do to answer the questions about data

• Being a Python or R pro is great, but it doesn't help us understand data unless we also know how to uncover the correct questions
  - May mean asking the experts who created/captured the data
  - Use your own general knowledge, too
Anaconda

- [www.continuum.io/downloads](http://www.continuum.io/downloads)
- Use Python 3.5 version *(not 2.7)*
- Can Navigator to install packages or start Jupyter Notebook
- Can also use the shell to do this:
  - `$ jupyter-notebook`
  - `$ conda install <pkg_name>`
Jupyter Notebook

- An interactive programming environment
- Runs in your web browser…
- … but does not require an internet connection
- Displays results (even interactive maps) inline
- Originally designed for Python
- Supports other languages, too
- You decide how to divide code into executable cells
- Shift+Enter to execute a cell
Jupyter Notebook

• Starts with a directory view
• Create new notebooks using the right menu (Choose Python3)
  - Originally has name "Untitled"
  - Click on "Untitled" to change the name (do this!)
• Save a notebook using the command under the File menu
• Shutting down the notebook requires quitting the kernel
  - Web browser is interface to display code and results
  - **Kernel** actually runs the code: usually see messages in a console/terminal window
  - Ctrl+C in this window ask you if you want to shut down the kernel
Jupyter Notebook

• Open a notebook by going back to the main screen and clicking on it like you would in a desktop view
• Past results are displayed, but that does not mean they are defined
• Use "Run All" or "Run All Above" to re-execute past work
  - If you shut down the kernel, all of the data and variables you defined need to be redefined (so you need to re-run all)
  - **Watch Out—Order Matters**: If you went back and re-executed cells in a different order than they are shown, doing "Run All" may not produce the same results!
• Edit mode (green) versus Command mode (blue == **Be Careful**)
Jupyter Notebook

• Can write code or plain text (can be styled Markdown)
  - Choose the type of cell using the dropdown menu
• Cells break up your code, but all data is **global**
  - Defining a variable $a$ in one cell means that variable is accessible in any other cell
  - This includes cells above the cell $a$ was defined in!
• Remember Shift+Enter to execute
• Enter just adds a new line
• Use `?<function_name>` for help
• Use Tab for auto-complete or suggestions
• Tab also indents, and Shift+Tab unindents
Jupyter Notebook Documentation

• Notebook Quick Start Guide
• Notebook Basics
Python

• Started in December 1989 by Guido van Rossum
• “Python has surpassed Java as the top language used to introduce U.S. students to programming…” (ComputerWorld, 2014)
• Python and R are the two top languages for data science
• High-level, interpreted language
• Supports multiple paradigms (OOP, procedural, imperative)
• Help programmers write readable code
• Use less code to do more
• Lots of libraries for python
  - Designed to be extensible
  - Easy to wrap code from other languages like C/C++
• Open-source with a large, passionate community
Learning Python Resources

• https://software-carpentry.org/lessons.html
• https://wiki.python.org/moin/BeginnersGuide
• https://learnxinyminutes.com/docs/python3/
• http://www.pythontutor.com
• http://www.python-course.eu
• http://thepythonguru.com
• https://wiki.python.org/moin/IntroductoryBooks
• https://en.wikibooks.org/wiki/A_Beginner%27s_Python_Tutorial
Print and Comments

• `print("Hello World")`

• Can also print variables:
  ```python
  name = "Jane"
  print("Hello,", name)
  ```

• In the notebook, putting a variable as the last line of a cell will display it

• Any text after `#` is a comment in python
  - Until the end of the line (just like `//` in Java)
  - Remember, can also have entire cells that are text in a notebook
Python Variables and Types

• No type declaration necessary
• Variables are names, not memory locations
  
  ```python
  a = 0
  a = "abc"
  a = 3.14159
  ```
• Don't worry about types, but think about types
• Strings are a type
• Integers are as big as you want them
• Floats can hold large numbers, too (double-precision)
Python Math

- Standard Operators: +, -, *, /, %
- Division "does what you want" (new in v3)
  - \(-5 / 2 = 2.5\)
  - \(-5 // 2 = 2 \# \text{use } // \text{ for integer division}\)
- Shortcuts: +=, -=, *=
- No ++, --
- Exponentiation (Power): **
- Order of operations and parentheses:
  - \(-4 - 3 - 1\)
  - \(-4 - (3 - 1)\)
- "abc" + "def"
- "abc" * 3
Python Strings

- Strings can be delimited by single or double quotes
  - "abc" and 'abc' are exactly the same thing
  - Easier use of quotes in strings: "Joe's" or 'He said "Stop!"'

- String concatenation: "abc" + "def"

- Repetition: "abc" * 3

- Special characters: \n \t like Java

- Indexing:
  
  ```python
  a = "abcdef"
  a[0]
  ```

- Slicing: a[1:3]

- Format:
  
  ```python
  name = "Jane"
  print("Hello, {}".format(name))
  ```
Exercise

• Given variables x and y, print the long division answer of x divided by y with the remainder.

• Examples:
  - x = 11, y = 4 should print "2R3"
  - x = 15, y = 2 should print "7R1"
Loops

- `while <condition>:`
  - `<indented block>`
  # end of while block (indentation done)

- Remember the colon!

- `a = 5`
  - `while a > 0:`
    - `print(a)`
    - `a -= 2`

- `a > 0` is the condition

- Python has standard boolean operators (`<, >, <=, >=, ==, !=`)  
  - What does a boolean operation return?  
  - Linking boolean comparisons (`and, or`)
Conditionals

- if, else
  - Again, indentation is required

- elif
  - Shorthand for else: if:

- Same type of boolean expressions
Functions

• Calling functions is as expected:
  mul(2,3) # computes 2*3 (mul from operator package)

- Values passed to the function are parameters
- May be variables!
  a = 5
  b = 7
  mul(a,b)

• print is a function
  print("This line doesn't end", end="")
  print("See it continues")

- end is also a parameter, but this has a different syntax
- Keyword argument!
Why do we create and use functions?
Defining Functions

• **def** keyword

• Arguments have names but **no types**
  
  ```python
  def hello(name):
      print("Hello {}".format(name))
  ```

• Can have defaults:
  
  ```python
  def hello(name="Jane Doe"):  # default
      print("Hello {}".format(name))
  ```

• With defaults, we can skip the parameter

  ```python
  hello() or hello("John")
  ```

• Also can pick and choose arguments:
  
  ```python
  def hello(name1="Joe", name2="Jane"):  # default
      print("Hello {} and {}".format(name1, name2))
  hello(name2="Mary")
  ```
Return statement

• Return statement gives back a value:
  
  ```python
def mul(a, b):
    return a * b
  ```

• Variables changed in the function won't be updated:
  
  ```python
def increment(a):
    a += 1
    return a

b = 12
b = increment(b)
c = increment(b)
print(b, c)
```