Introduction to Python

Andrew Armstrong

University of Cincinnati
Carl H. Lindner College of Business
Center of Business Analytics

2017 September 14-15

Education

- Dual Bachelors in Computer Engineering and Mathematics
  - Michigan Technological University
- Masters in Computer Engineering
  - The Air Force Institute of Technology
  - Thesis work focusing on cryptanalysis
- Masters in Management of Technology
  - University of Texas at San Antonio
- PhD in Applied Mathematics
  - The Air Force Institute of Technology
  - Dissertation work focusing on the analysis of of astronomical data

Contact Information

- School email:
  - armstam@ucmail.uc.edu
- Personal email:
  - andrewarmstrong004@gmail.com

Course Resources

- You can find all the material for this course at the following website.

- `https://github.com/andrewarmstrong004/Introduction-to-Python`

Course Description

This course is designed to be an introduction to Python for applications in data analytics. The material will emphasize the core concepts in Python, specifically data types, data structures, and functions and how they can be implemented and used to address data analytics problems. Popular modules used in data analysis will also be covered at a high level.

Course Objectives

- Course Objectives
  - Describe the differences between the main branches of Python
  - Know and be able to utilize the basic Python data types and structures
  - Write basic Python statements such as if/elif/else, for loops, and while loops
  - Be able to import and use standard Python modules for importing, exporting, and manipulating data
  - Write a basic Python module and be able to describe how modules work
  - Incorporate commonly used modules into a program to improve its utility

Course References

- Python Basics - **Main book for the course**
  - Lutz, Mark, *Learning Python*, O'Reilly Media, 5th Edition, 2013.

- Python Analytics
  - McKinney, Wex, *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*, O'Reilly Media, 1st Edition, 2012.
  - Grus, Joel, *Data Science from Scratch: First Principles with Python*, O'Reilly Media, 1st Edition, 2015.

Course References Continued

- More Advanced Python
  - Lutz, Mark, *Programming Python: Powerful Object-Oriented Programming*, O'Reilly Media, 4th Edition, 2011.
  - Slatkin, Brett, *Effective Python: 59 Specific Ways to Write Better Python*, Addison-Wesley Professional, 1st Edition, 2015.
- Transitioning from Python 2.7x to Python 3.x
  - Ramalho, Luciano, *Fluent Python: Clear, Concise, and Effective Programming*, O'Reilly Media, 1st Edition, 2015.
- Theory
  - Hastie, Trevor et al., *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2nd Edition, 2016.

Tentative Course Outline - Day 01

- Basic introduction to Python
- Where to write Python code
- Creating your first program
- Overview of boolean and numeric types
- Introduction to function
- Control flow
- Sequences
- Sets, Mappings, and comments
- Statements in Python

Tentative Course Outline - Day 02

- Day 01 Review
- Data I/O
- Intermediate functions
- Basic Modules and Packages
- Plotting and Regression
- Introduction to Machine Learning
- Case Studies

Installing Anaconda

- Download Anaconda from
  - `https://www.continuum.io/downloads`
- Make sure that you are downloading version 2.7x
  - This should be the lower download button (should be blue)
- Anaconda bundles up Python and several commonly used modules into single file for easy download
  - List of packages included in Anaconda: `https://docs.continuum.io/anaconda/pkg-docs`
  - These packages bring a lot of functionality to Python that we can take advantage of, however, they are not all the possible Python packages that you could use
- Once you have it fully downloaded, just leave it, we will take care of it later

Python History

- Originally written by Guido van Rossum
  - Guido van Rossum worked at Centrum Wiskunde & Informatica (CWI) in the Netherlands
  - Development started over the Christmas break in 1989 as a "hobby" programming project to work on during his time off
- Python is the successor of the ABC language, which was also written by workers at CWI
- Python is named after Monty Python's Flying Circus, not after the snake

Python Development

- Python version 0.9.0, the first publicly available version, was released in February 1991 on alt.sources
  - Included classes and inheritance
  - Included core data data types `list`, `dict`, and `str`
- Version 1.0 was released in January of 1994
  - Major additions included `lamda`, `map`, `filter`, and `reduce`
- Version 2.0 was released in October of 2000
  - Introduced list comprehensions, which is a fancy way of making lists

Python 2.7x vs Python 3.x

- Python 2.6 and 3.0 were released simultaneously in December of 2008
  - The split in versions was due to the fixing in some design flaws built into Python which didn't allow for complete backwards compatibility
  - Python 3 attempted to "reduce feature duplication by removing old ways of doing things"
- Python 2.7 and 3.1 releases coincided in June of 2009
  - This is the last major release of Python 2.x
- In November of 2014 it was announced that support of Python 2.7 would last until 2020

The Zen of Python

- The core philosophy of Python is summarized in The Zen of Python (PEP 20)
  - `https://www.python.org/dev/peps/pep-0020/`
  - PEP stands for Python Enhancement Proposals
- Examples from The Zen of Python include:
  - Beautiful is better than ugly
  - Explicit is better than implicit
  - Simple is better than complex
  - Complex is better than complicated
  - Readability counts

Python Features

- Python is a multi-paradigm programming language
  - Object-oriented programming
  - Structured programming
  - Functional programming
  - and many others
- Python uses dynamic typing
  - Meaning you don't have to tell Python what type of variable you are making and that the type of variable can change
- Python has built in garbage collection
  - This means Python cleans up its own memory
  - Done using reference counting and a cycle-detection garbage collector

Pythonic Way of Thinking

- Pythonic means code that doesn't just get the syntax right
  - Pythonic code follows the conventions of the Python community and uses the language in the way it is intended to be used.
- Python was made to be extended
- This means that all desired components of the language don't need to be built into the languages core
- Python was written to be easily read and fun to use
  - Many online examples use references to Monty Python for example

Python Applications

- System programming
  - System administration tools
- Graphical User Interfaces (GUIs)
  - Making pretty user interfaces
- Internet scripting
  - Networking tasks
- Component integration
  - “Gluing” different pieces of code, even from different languages, together
- Numeric and scientific computing
  - Competing with even C++ and FORTRAN

Python Applications

- Rapid prototyping
  - Fast initial development with easy transition to other languages like C
- Database programming
  - Managing large databases
- Game design
  - Creating games, even complex and attractive games
- Data mining
  - Fast and easy data mining applications
- Robotics
  - Controlling robotic systems

Python in the Real World

- Google
  - Extensively uses Python in its web search system
- YouTube
  - Largely written in Python
- Dropbox
  - Server and desktop client written primarily in Python
  - Guido van Rossum now works for Dropbox
- Raspberry Pi
  - Promotes python as its educational language
- EVE Online (a massively multiplayer online game)
  - Broadly uses python

Python in the Real World

- BitTorrent
  - First began as a Python program
- Industrial Light & Magic
  - Uses Python in the production of movies
- NSA
  - Cryptographic and intelligence analysis
- One Laptop Per Child (OLPC)
  - Used to build the user interface
- JPMorgan Chase
  - Financial market forecasting

Documentation

- The core documentation for Python is located at: `https://docs.python.org/2.7/`
- Reading this documentation can be relatively challenging, but is really worth the effort
- If you don't want to dig through all the technical background of a function, and just want the answer, Google is your friend
- Most Google searches will lead you to `https://stackoverflow.com`
  - For example: `https://goo.gl/c9pYY6`

Getting Help in Python

- `help()`
  - Gives you help about a function or class
  - Typing in just help()
  - Typing in help(list)
- `dir`
  - Tells you the attributes of the function or class
  - Type dir(list)
- `dict`
  - The attribute dictionary of a function or class
  - Type list.__dict__
- Easiest way is to Google it or look up the documentation online

Finalizing Anaconda Install

- `https://www.continuum.io/downloads`
- Adding it to your path
  - Linux: `export PATH="/Users/jsmith/anaconda2/bin:$PATH"`
  - Windows don't worry about it
- `conda create -name snowflakes`

Class Folder Setup

- Create a folder in a location easily navigated to
  - We will use this folder over the course of the day
  - This folder should be in the same drive as your Anaconda installation
- Navigate to `https://github.com/andrewarmstrong004/Introduction-to-Python`
- This site has all the material you will need for class today
- Download at least the first couple lecture and practice files
  - These files should end with .ipynb

Using the Terminal

- Open up your python terminal
  - Either type in conda
  - Or open up the terminal from your application launcher
- `>>>` means you are in a python environment
  - if you don't see that, something is wrong
- type the following into your terminal:
  - `>>> print 'Hello, World!'`
- Congratulations, your just wrote your first Python code

Using a Text Editor

- You can also just write Python code in a text editor and save it as a .py file
- You can then just run this Python file by typing in `python` and then the name of your program into your terminal
- This is not recommended, it is very hard to problem solve this way
  - Especially getting rid of nuisance whitespace

Using a Text Editor

- Open up a new blank document in a basic text editor, like notepad on windows
- type the following code into your new document:
  - `print 'Hello, World!'`
  - `print 5+7`
- Save this new file as Second_Program.py on your desktop or in an easily to navigate to folder
- Open up your Anaconda terminal and navigate to your new file
- Once there, execute the following command:
  - python Second_Program.py
- You now have written your second Python program

Integrated Development Environments (IDEs

- IDEs are very common ways of programming in Python
- Common IDEs include:
  - Spyder: `https://pythonhosted.org/spyder/`
    - Should be installed with Anaconda
  - Notepad++: `https://notepad-plus-plus.org/download/v7.3.3.html`
  - Sublime Text: `http://www.sublimetext.com/2`
  - PyCharm: `https://www.jetbrains.com/pycharm/`
    - The one I use most often
    - I believe its free for students
- Find one you like and try to learn all its bells and whistles

Integrated Development Environments (IDEs

- Open up your favorite Python IDE
  - If you haven't used Python before, open up the Spyder program that comes with Anaconda
  - If using Windows, Spyder should be located in the Anaconda folder in your Start menu
- Type the example code into the appropriate area of your IDE
  - Take note of the difference in color and formatting when using an IDE when compared to notepad (or similar)
- Save the code as Second_Program.py in an easily accessible area
- Run the code using your terminal or with the built in functions of your IDE
  - If you are using your terminal, type the following into it:
    - python Second_Program.py

Python Notebooks

- In Windows you can just open jupyter notebooks from the Start menu
- Using your terminal, navigate to your desired folder for class
- type `jupyter notebook` in the terminal

Python Notebooks

- Once you have Jupyter Notebooks open, use the file navigation system to navigate to the folder we setup earlier
  - The folder that you have been saving the .ipynb files to
- Open Lecture 00c - Third Python Program
- Use the command shift+enter to execute the code selected and move on to the next section
- Use the command ctrl+enter to execute the code selected, but stay on the same code block
- Press h when not in edit mode to see all the command shortcuts
- Execute the first three code blocks
- Change the name in the forth code block to be your own name and execute the code

Python Notebooks

- In Windows you can just open jupyter notebooks from the Start menu
- Using your terminal, navigate to your desired folder for class
- type `jupyter notebook` in the terminal

Python Notebooks

- Python notebooks are good places to write code when you are building and testing new code
- They are also very helpful when you need to share code or write up examples of code
- We will be using jupyter notebooks for nearly all the code in this class, however, they may not be optimal for writing up code for actual implementation
- For actual implementation, writing up code using an IDE is usually the prefered method
  - But even very strong python programmers use notebooks to build and test before implementation in a python file

A Note on the Importance of Whitespace

- Python is one of the few languages where whitespace matters
- This means that you can't just add random indentation to your code like you can in other languages
- Python does this to both make the code easier to read and to get away from all the parenthesis and similar symbols scattered throughout most other languages
- This becomes very important when dealing with control flow, functions, and loops

Lecture 01 - Boolean and Numeric Types

- In this section we are going to cover the most basic data types used in Python
- Boolean Types
  - boolean
    - A boolean is either True of False
    - Can also be thought of as a 1 or 0
    - Examples: True, False
- Numeric Types
  - Integer
    - This is a whole number, or a natural number
    - Examples: 0, 1, -17, 1287
  - Floating Point Number
    - This is a number with a trailing decimal point
    - Examples: 0.0, 1.0, .333333, 3.1415926
  - Complex Number
    - Numbers with an imaginary component
    - Examples: $1 + 1j$, $3j$, $2.7 + .8j$

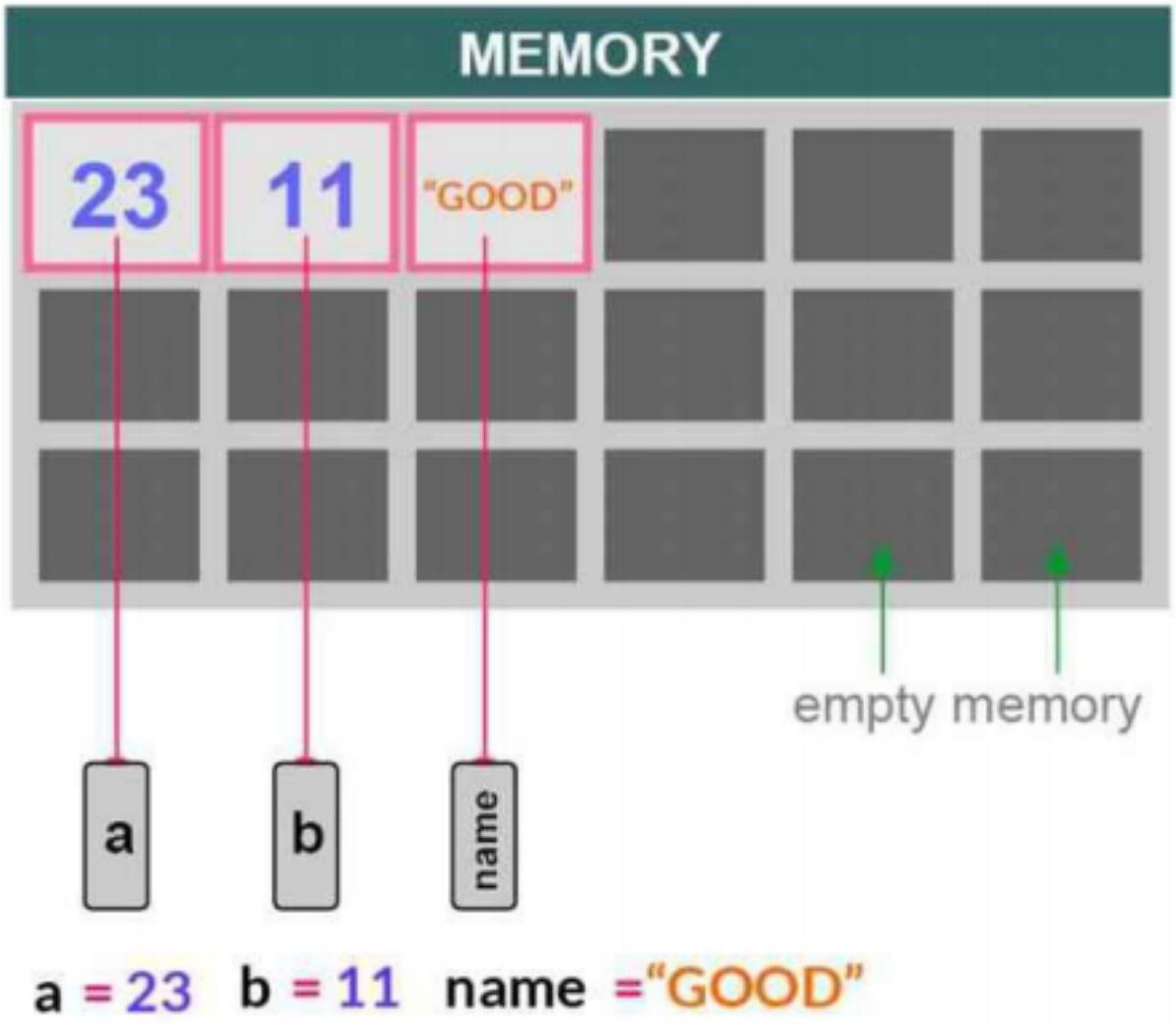
Naming Variables

- We can store information in Python by naming variables and assigning them values
- A basic example would be: `a = 7`
- In Python we do not need to type cast variables, meaning we don't need to tell Python what type of variable we are going to be needing (like an int of bool)
- We can also reassign variables to be a completely different type by just reassigning it
  - `a = 2.4`

Mutability

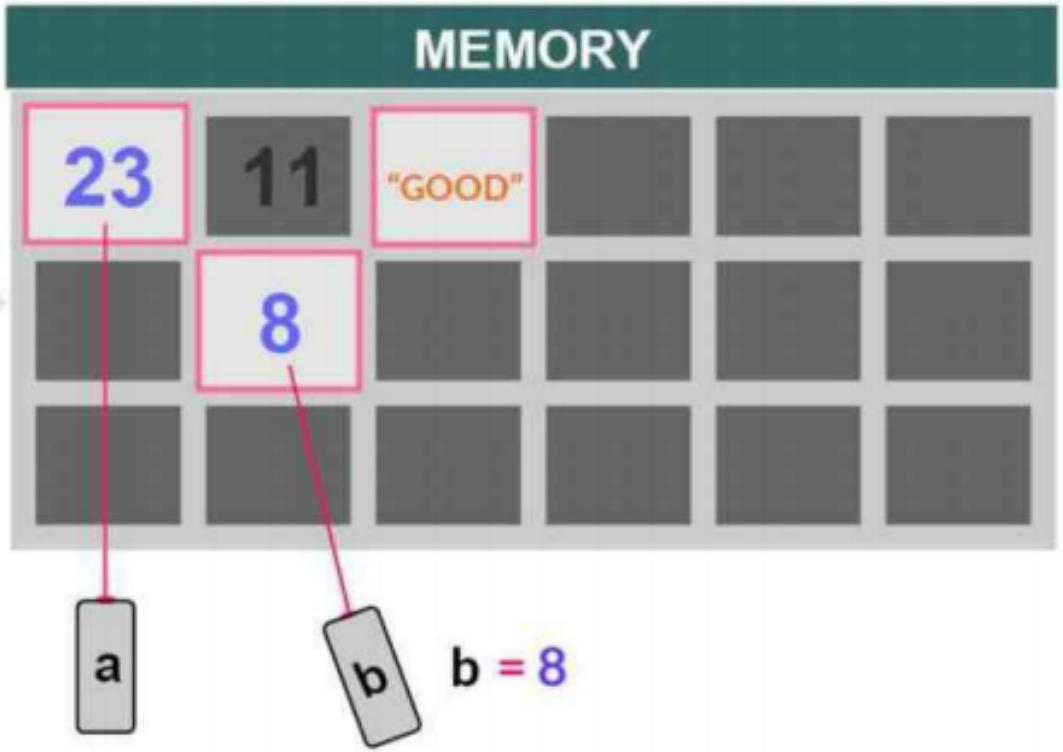
- Once a variable is assigned it creates a point in memory where it stores the data
- It then creates a pointer with the given name which points to that location
- To change what a given variable is pointing at, we then have two options
  - Change where the pointer is pointing
  - Change the value at the place the pointer is pointing
- If we can't change what is in a particular memory locations, we call that variable type immutable
- If we can change what is in a particular memory location, we call that varible type mutable

Mutability - Example



Mutability - Example

lets change the value of b from 11 to 8



Because numbers are immutable, "b" changes location to the new value.
When there is no reference to a memory location the value fades away and the location is free to use again.
This process is known as garbage collection

Lecture 01 - Boolean and Numeric Types

- The concept of mutability isn't very important to us right now because of how transparent it is when using boolean and numeric type variables
- Mutability will become more important to us when we start dealing with more complex types (strings, tuples, and lists)
- Lets now play around with some more code to see how these boolean and numeric type variables work
  - Open up Lecture 01 - Boolean and Numeric Types.ipynb using jupyter notebooks

Lecture 02 - Basic Functions

- At this point we can produce code to create a specif output, but if we want to repeat the same functionality with different input we would need to completely reproduce the code to do that
- To speed along the process we can create what are called functions
- Functions are blocks of code that can be reused by calling the function name and providing it some input

Lecture 02 - Basic Functions

- We use the keyword `def` to create a function
- `def` is then followed by the new functions name
- After the function name comes an open and close parenthesis (which can contain optional input variables) and a ':'
- To get information back out of a function we use the keyword `return`
- Lets walk through how to create some basic functions
- Open up Lecture 02 - Basic Functions.ipynb using jupyter notebooks

Lecture 03 - Control Flow

- We now have convenient methods for reproducing code, however, are programs are currently static (meaning they aren't changing)
- We need a way to change what a program does while it is executing
- To do this we use what is called an if statements
- In Python if statements must have an if statement (which evaluates to True or False)
- If statements then also have optional pieces of elif and else
- Lets look at how we can use if statements to change what our code does on the fly
- Ope up Lecture 03 - Control Flow.ipynb using jupyter notebooks

Lecture 04 - Sequences

- Sequences
  - Strings (Immutable)
    - A sequence of characters
  - Tuples (Immutable)
    - A collection of mixed data types
  - Lists (Mutable)
    - A collection of mixed data types
    - Probably your most commonly used data structure

Lecture 05 - Sets, Mappings, and Comments

- Sets
  - Sets (Mutable)
    - Unordered collection of data with no duplicates
  - Frozen Sets (Immutable)
    - Unordered collection of data with no duplicates

- Mappings
  - Dictionaries (Mutable)
    - A collection of key-value pairs

Lecture 06 - Statements

- Assignments
  - How we assign values to variables
- Expression Statements
  - How to call a function or method
- Print statements
  - Printing information to the user
- If Statements
  - Managing control flow
- For loops
  - Iterates through a collection or iterable object
  - This will commonly be a list
- While loops
  - Loops while a conditions continues to be true
  - Stops when the condition becomes false

Lecture 08 - Data IO

- For other data analysis tools there are only a limited number of ways to open external data sources
- Since Python is strongly focused on the use of external modules, there are a whole slew of methods for importing data
- Depending on your particular application, you may want to use different import functions
- In this quick class we will only focus on a few of some of the more widely used methods

Lecture 09 - Intermediate Functions

- We have already created some rudimentary functions
- Lets take some of the more advanced concepts we have learned and combine them with functions to make more complex and useful pieces of code

Lecture 10 - Basic Modules and Packages

- Creating a module
  - How we create code for later use by other programs
- Importing a module
  - How we bring in old code to a new program
- Commonly used Modules in Python
  - Numpy
  - Random
  - OS

Lecture 11 - Plotting and Regression

- Plotting
  - One of the first things you want to do when you get a new data set is explore it
  - One of the best ways to do that is to plot the data in a variety of ways to see what types of patterns exist in it
  - The most common plotting tool in Python is matplotlib
  - Lets look as some simple matplotlib examples to see what we can create
  - Lets also look at another data exploration tool Seaborn to see what else is out there

Lecture 11 - Plotting and Regression Cont.

- Regression
  - Regression is one of the most commonly used prediction methods in data science
  - There are a large number of programs dedicated to performing only regression analysis
  - There are also a large number of Python packages dedicated to regression analysis
  - One of the easiest to use, and which uses the same format as R, is Statsmodels
  - Lets look at a few examples of how we can use statsmodels for regression

Lecture 12 - Introduction to Machine Learning

- Classification
  - Supervised learning
  - You know possible classes
- Clustering
  - Unsupervised learning
  - You don't know the different classes
- Dimensionality Reduction
  - Getting variable counts to manageable levels

Conclusion

- I hope I have helped you to learn some very basic Python programming today
- We went through a lot of material very quickly, so don't feel bad if you didn't catch everything
- If you want to learn to program I would recommend Python as a first programming language
- Python is also a power tool for data analytics and general purpose programming
- If you want to learn more about Python there are a large number of useful resources as books, online tutorials, and online courses
- If you have any pressing questions, you can reach out to me at:
  - andrewarmstrong004@gmail.com