**Python Workshop**

Prerequisites:

1. Previous programming experience (either in Python or in another language)
2. Set up the environment:
   1. Install the latest version of [Anaconda](https://www.anaconda.com/download/) for Python **3.6**
   2. Lunch a Jupyter notebook
   3. Create a notebook and run *print("hello world")* in the first cell.
   4. Take off the internet.

In every section:

* Brief intro
* Try it yourself
* What would be the output? What’s wrong with this?
* Ask me something that you don’t know.

Agenda:

# Introduction

## About me

I am a GA/ web developer at lubin, AI Club president, currently living in Jersey City, working on several deep learning academic projects, last semester MS program focused on machine learning. That’s the reason why I’m doing the master and the club.

Previously, I worked as a Software Engineer for about 3 years, wearing multiple hats, front-end, back-end, self-employed, android developer, manager, BI analyst. I have worked with consultancies, marketing organizations, and freelance projects.

I’m from Venezuela so I also speak Spanish.

My website [vmvargas.com](http://vmvargas.com)

Visit <https://bit.ly/aiclub-signup/> besides workshop like this we hold talks, forums, writings.

About you? What’s your main language?

## About Python

Python is an incredibly efficient language: your programs will do more in fewer lines of code than many other languages would require. Python’s syntax will also help you write “clean” code. Your code will be easy to read, easy to debug, and easy to extend and build upon compared to other languages.

People use Python for many purposes: to make games, build web applications, solve business problems, and develop internal tools at all kinds of interesting companies. Python is also used heavily in scientific fields for academic research and applied work.

Python Developers Survey 2017 Results

<https://www.jetbrains.com/research/python-developers-survey-2017/>

## About the workshop

Why this workshop?

What will I learn?

What’s out of the scope? Python 2 and below, Testing a function, Testing a class

Workshop Agenda

PYTHON FOUNDATIONS

# Getting Started

## Tools that we will use today

* Python: interpreted cross-platform high-level programming language for general-purpose programming.

useful commands:

Where python/anaconda

python hello\_world.py

import sys

if len(sys.argv)>1 and sys.argv[1]:

salutation = sys.argv[1]

* Pip:

is Python's official package manager, and is most commonly used to install packages published on the Python Package Index (PyPI).

* Conda:

Anaconda package manager that automates the process of installing, updating, and removing packages.

pip is a general-purpose manager for Python packages;

conda is a language-agnostic cross-platform environment manager

* Anaconda:

Free and open source distribution that comes with a lot of tools right of the box, it comes with either Python or R, a package manager called conda, and a lot of other libraries and packages pre-installed, these packages are usually related to analytics and scientific and help to plot and compute large amounts of data.

Anaconda is an option that people prefer because it simplify issues related to installing different python version of python, creating different environments, privileges issues, etc.

If you are on Windows, use Anaconda Prompt instead of Windows Command Prompt.﻿

* Miniconda: another software distribution

Miniconda is essentially an installer for an empty conda environment, containing only Conda and Python, so that you can install what you need from scratch.

* IPython:

Currently, IPython fulfill 2 roles: being the python backend to the jupyter notebook, (aka kernel) an an interactive python shell.

* Jupyter:

Project Jupyter was created from iPython Notebooks with the intention to be robust and language agnostic (Julia, and R notebooks).

* Git:

To publish workshop’s sections (.ipynb files)

<https://help.github.com/articles/working-with-jupyter-notebook-files-on-github/>

## Set-up the pipeline

Test your environment.

Run a hello\_world.py

## Variables and simple data types

Question:

message = 'One of Python's strengths is its diverse community.'

print(message)

SyntaxError: invalid syntax

## Introduction to Lists

In this section and the next you’ll learn what lists are and how to start working with the elements in a list. Lists allow you to store sets of information in one place, whether you have just a few items or millions of items.

Lists are one of Python’s most powerful features readily accessible to new programmers, and they tie together many important concepts in programming.

## Working with Lists

You learned how to make a simple list, and you learned to work with the individual elements in a list. In this chapter you’ll learn how to loop through an entire list using just a few lines of code regardless of how long the list is.

Looping allows you to take the same action, or set of actions, with every item in a list. As a result, you’ll be able to work efficiently with lists of any length, including those with thousands or even millions of items.

## If Statements

## Dictionaries

Because dictionaries can store an almost limitless amount of information, I’ll show you how to loop through the data in a dictionary. Additionally, you’ll learn to nest dictionaries inside lists, lists inside dictionaries, and even dictionaries inside other dictionaries.

Understanding dictionaries allows you to model a variety of real-world objects more accurately.

## User Input and while Loops

Most programs are written to solve an end user’s problem. To do so, you usually need to get some information from the user.

For a simple example, let’s say someone wants to find out whether they’re old enough to vote. If you write a program to answer this question, you need to know the user’s age before you can provide an answer. The program will need to ask the user to enter, or input, their age; once the program has this input, it can compare it to the voting age to determine if the user is old enough and then report the result.

In this chapter you’ll learn how to accept user input so your program can then work with it. When your program needs a name, you’ll be able to prompt the user for a name. When your program needs a list of names, you’ll be able to prompt the user for a series of names. To do this, you’ll use the input() function.

You’ll also learn how to keep programs running as long as users want them to, so they can enter as much information as they need to; then, your program can work with that information. You’ll use Python’s while loop to keep programs running as long as certain conditions remain true.

## Functions

In this chapter you’ll learn to write functions, which are named blocks of code that are designed to do one specific job.

When you want to perform a particular task that you’ve defined in a function, you call the name of the function responsible for it. If you need to perform that task multiple times throughout your program, you don’t need to type all the code for the same task again and again; you just call the function dedicated to handling that task, and the call tells Python to run the code inside the function. You’ll find that using functions makes your programs easier to write, read, test, and fix.

* 1. Defining a Function
  2. Passing Arguments
  3. Return Values
  4. Passing a List
  5. Passing an Arbitrary Number of Arguments
  6. Storing Your Functions in Modules
  7. Styling Functions

## Classes

Object-oriented programming is one of the most effective approaches to writing software. In object-oriented programming you write classes that represent real-world things and situations, and you create objects based on these classes.

When you write a class, you define the general behavior that a whole category of objects can have.

Making an object from a class is called instantiation, and you work with instances of a class.

You’ll also write classes that extend the functionality of existing classes, so

* 1. Creating and Using a Class
  2. Working with Classes and Instances
  3. Inheritance
  4. Importing Classes
  5. The Python Standard Library
  6. Styling Classes

## Files and Exceptions

Here you’ll learn to work with files so your programs can quickly analyze lots of data. You’ll learn to handle errors so your programs don’t crash when they encounter unexpected situations.

You’ll learn about exceptions , which are special objects Python creates to manage errors that arise while a program is running. You’ll also learn about the json module, which allows you to save user data so it isn’t lost when your program stops running. Learning to work with files and save data will make your programs easier for people to use.

Users will be able to choose what data to enter and when to enter it. People can run your program, do some work, and then close the program and pick up where they left off later.

Learning to handle exceptions will help you deal with situations in which files don’t exist and deal with other problems that can cause your programs to crash. This will make your programs more robust when they encounter bad data, whether it comes from innocent mistakes or from malicious attempts to break your programs. With the skills you’ll learn in this chapter, you’ll make your programs more applicable, usable, and stable.

* 1. Reading from a File
  2. Writing to a File
  3. Exceptions
  4. Storing Data

1. Common questions on python:
   1. Python Style Guide
   2. Difference between del, remove and pop on lists.
   3. None, \_, other types of null
   4. Python garbage collector?
   5. Import python php differences
   6. Python i++
   7. Some Jupyter magic commands

Part 2:

1. A Data visualization Project
2. A warming up for the challenge

<https://ehmatthes.github.io/pcc/>