GENERAL ASSEMBLY

PYTHON PROGRAMMING 101

Peter Goodin

Lead Data Science Instructor, GA Melbourne

Research Fellow, Neurorehabilitation, Royal Melbourne Hospital

Honorary Research Fellow, Florey Institute of Neuroscience & Mental Health

LEARNING OBJECTIVES

- Discuss the history of Python & how it can be used
- Define how Python compares to other programming languages
- Describe Python's data structures and syntax
- Demonstrate basic Python programming fundamentals to solve a real world problem
- Create a custom learning plan to build your Python skills after this workshop!

PRE-WORK

PRE-WORK REVIEW

- Bring a laptop with Anaconda installed. Scroll to your operating system version and click on the install button for Anaconda with Python 2.7.
- We will be using Jupyter Notebooks as the main IDE for the workshop. If you have installed Anaconda, then you are ready to go!
- All the code, data and slides downloaded from: https://github.com/petergoodin/ga_python101

OPENING

WHO AM I?

Email: peter.goodin@generalassemb.ly

Peter Goodin
What I do: Lead Instructor – Data Science GA Melbourne, Neuroscience of Stroke / brain damage

What have I done: Ph.D Cognitive Neuroscience – Psychoneuroimmunology of Major Depressive Disorder, Consumer Neuroscience, Condom analysis.

What I'm interested in: Tech & data science for rehabilitation after brain damage.

petergoodin

@peter_goodin

ABOUT YOU!

- Before we dive in, let’s talk a bit about you!
- Name
- What brings you to GA?
  - Current activities
  - Your Goals
  - Programming experience

THE BIG PICTURE

- What we'll cover:
  - What is Python?
  - What can Python do for me?
  - The Python environment and basic commands
  - Python Packages
  - Types, structures and functions
  - Programming (pseudocode and Python)
  - Data exploration with Python

THE BIG PICTURE

- Why this python matters:
  - Programming is a sought-after skill
  - Python is an accessible language that is widely used in data analysis.

- Why this python rocks:
- Python opens up a door to a variety of opportunities, including video games development, web dev, data science and research

WHAT IS PYTHON?

- Created by Guido Van Rossum in 1991
- Emphasises productivity and code readability
  - The language is easy to pick up and learn
  - This gentle learning curve brings makes it easier for many to contribute to production level code
  - Readable code means that almost anyone can pick up a piece of code and understand what it is doing

THE ZEN OF PYTHON

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules,
Although practicality beats purity.
Errors should never pass silently,
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it,
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

SOME CHARACTERISTICS OF PYTHON

- Object-oriented (OO): Instead of concentrating on isolated "actions", object-orientation enables us to focus on "objects" that contain data & attributes.
- Objects have specific procedures, known as methods (code) that can access and modify the attributes of the object.
- OO makes it easier to reuse code in other programs

SOME CHARACTERISTICS OF PYTHON

- High-level programming: This refers to the use of language similar to natural language makes it easy to use and automate.
- Related to the code readability mentioned earlier.

SOME CHARACTERISTICS OF PYTHON

- Dynamic Typing: Variables are assigned their type by Python contextually.
- No need to pre-specify what kind of data a particular variable will have (integer, float, string, etc).
- Variable can be re-used (both good and bad), quick to make changes, generally faster to work with.

WHY PYTHON?

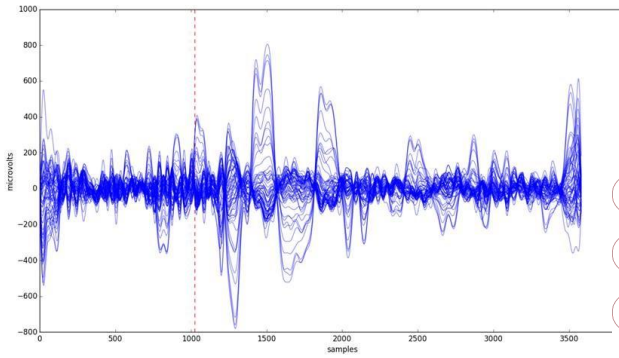
Python is:

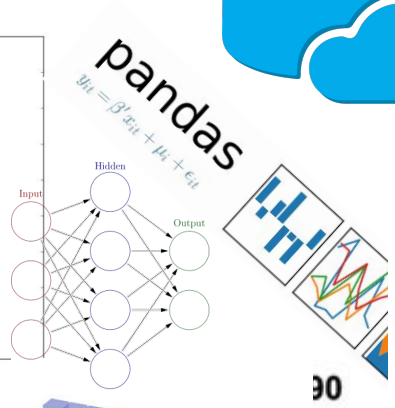
- Great for rapid prototyping and full-stack commercial applications.
- A modern, elegant, object-oriented language.
- Highly expressive, i.e., you can be more productive.
- Well documented and has an established and growing community.
- Comes with "batteries included" - in other words, Python has libraries that will help you do a ton of different tasks!

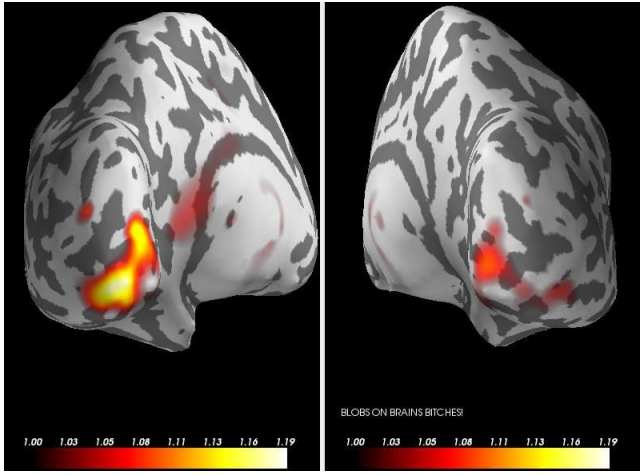
WHY PYTHON?

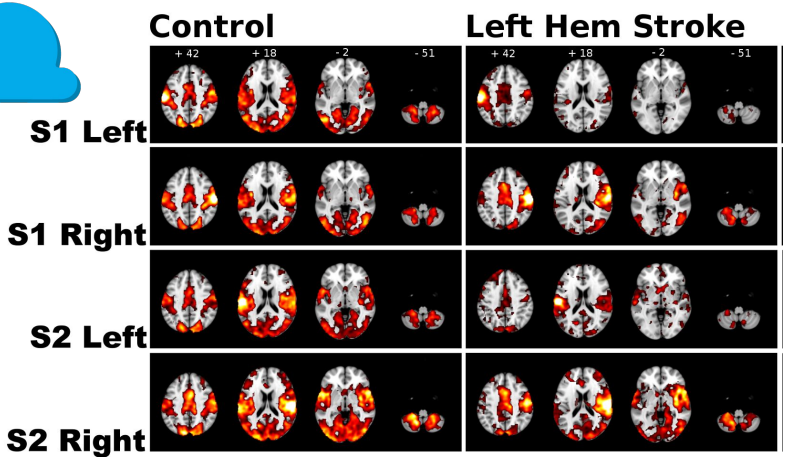
- Basically everything can be done with Python.
- Relatively easy to learn (still have to work at it though!)
- If something can't be done, you can probably create an extension for it.
- Things can be done quickly!
  - For example a program that takes you weeks in C++, might take you a day in Python.

HOW DO I USE PYTHON?









PYTHON ENVIRONMENT & BASIC COMMANDS

PYTHON Vs OTHER LANGUAGES

- Python is often compared to other interpreted languages, such as:
  - Java,
  - JavaScript,
  - Perl,
  - Tcl, or
  - Smalltalk.

- Comparisons to compiled languages like C++, Common Lisp and Scheme can also be enlightening.

PYTHON PROGRAMMING

- Let us what a Python program looks like.
- Starting with the typical `"Hello World!"` program:
  - In essence, we are writing code to print the message "Hello World!" in the screen.

PYTHON PROGRAMMING

Let’s see the C++ version:

C++

```cpp
#include<iostream>
using namespace std;

int main()
{
    cout << "Hello World!";
    return 0;
}
```

- What do the components of the code do?

PYTHON PROGRAMMING

What about Java?

Java

```java
public class HelloWorld
{
    public static void main (String[] args)
    {
        System.out.println("Hello, world!");
    }
}
```

- Once again, it does exactly the same at the C++ version – Prints Hello, world! to the screen

And in Python?

PYTHON PROGRAMMING

And in Python?

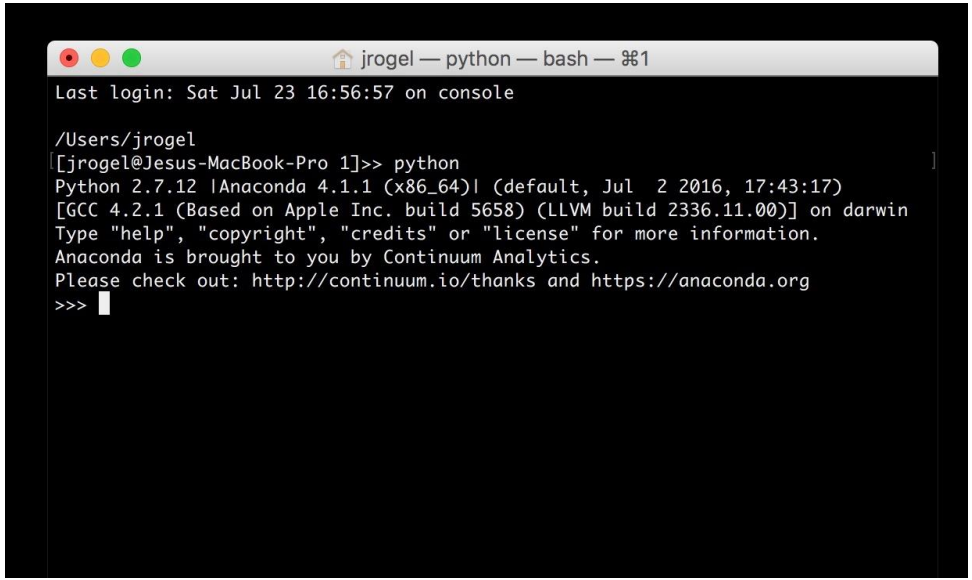
```python
print("hello world")
```

PYTHON: INTERACTIVE SHELLS Vs. SCRIPTS

- In our “Hello World!” python program, we are assuming that we are using an interactive shell,
- In other words, we are writing code that is executed immediately by the Python interpreter.
- We are able to "interact" with the results of the commands we pass. We can do this using a:
  - Python shell
  - iPython shell
  - Jupyter notebook

PYTHON SHELL

- A python shell is similar to a Command Line Terminal and it can be launched by typing:
- **“python”**



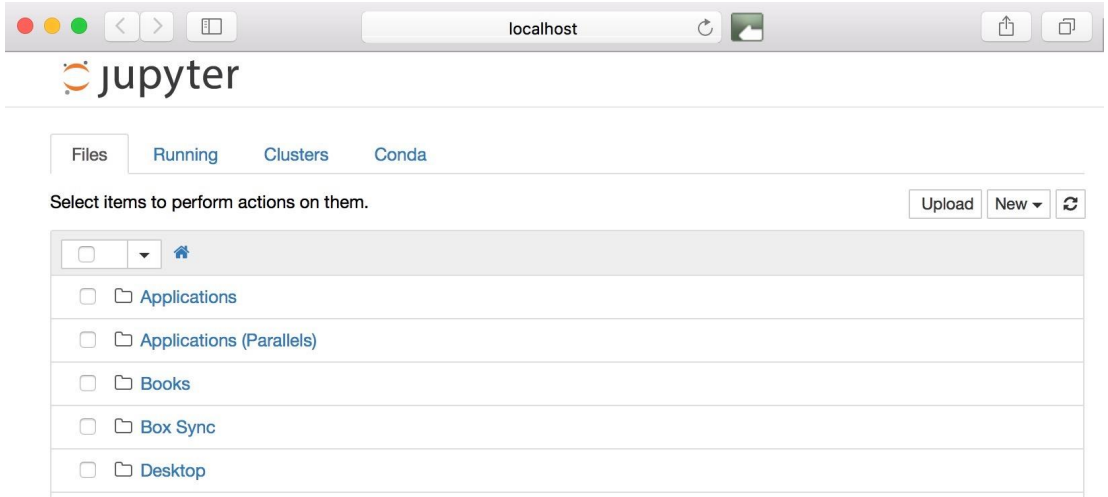
iPYTHON SHELL

- A python shell is more interesting than a plain terminal, providing syntax colouring and shortcuts to interact with our code. It can be launched with:
- **“ipython"**

```
jrogel — python ◂ python.app /Applications/anaconda/bin/ipython — bash — ⌘1

/Users/jrogel
[jrogel@Jesus-MacBook-Pro 5]>> ipython
Python 2.7.12 |Anaconda 4.1.1 (x86_64)| (default, Jul  2 2016, 17:43:17)
Type "copyright", "credits" or "license" for more information.

IPython 4.2.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: 1+1
Out[1]: 2

In [2]: print("Hello World!")
Hello World!

In [3]:
```

JUPYTER NOTEBOOK

- A Jupyter notebook is a web interface that let's us use formatting along side our code. It is the extremely common and very useful! You can launch it by typing:
- **"jupyter notebook"**



PYTHON: INTERACTIVE SHELLS Vs. SCRIPTS

- Sometimes we do not need to interact with our Python code.

- Instead, we may want to execute a program and simply get results.

In those cases, we need to create a Python script.

- To do so, we can use a text editor of our choice and save the code in a file with extension "***.py***".

SCRIPT EDITORS AND IDE

- Some common plain text editors include:
  - Atom
  - NotePad (Win), NotePad++ (Win)
  - TextEdit (Mac), TextWrangler (Mac).
  - Nano, Gedit (Linux, Mac)

SCRIPT EDITORS AND IDE

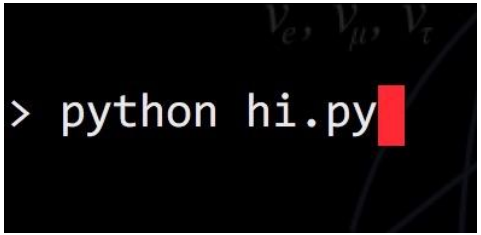
- Integrated development environments (IDEs) provide comprehensive facilities for computer programmers involved in software development.
  - PyCharm
  - Eclipse with PyDev
  - Atom
  - Spyder (included in Anaconda)

SIMPLE PYTHON SCRIPT

- A barebones script for the "Hello World!" program (saved to a file called `***hi.py***`) looks like this:

```python
print("hello world")
```

- To run the script by passing it as a command to the Python interpreter we need to write:



TASK

YOUR TURN!

SIMPLE PYTHON SCRIPT

Tasks

1. Using ipython, write a simple python script that prints out:
   a. “Hi! My name is *insert your name here*”

2. Open a text editor
   a. Write the command used in 1 and save the file as name.py
   b. Execute name.py by typing python name.py in the command line

COMMON PYTHON PACKAGES

PACKAGES

- Packages of code written to solve particular set of problems
- Can be installed with: **pip install <package name>**
- or **conda install <package name>**

- Ever used Excel? How do you fancy working with data structured in a similar way, but better graphics and less hassle? Try pandas

- Does your application require the use of advanced mathematical or numerical operations using arrays, vectors or matrices? Try SciPy (scientific python) and NumPy (numerical python)

PACKAGES

- Are you interested in using python in a data science workflow to exploit machine learning in your applications? Look no further than Scikit-learn

- Are you tired of boring-looking charts? Are you frustrated looking for the right menu to move a label in your plot? Take a look at the visuals offered by matplotlib and Seaborn

- Want interactive plots that can be used for dashboards or online? Try Bokeh

PACKAGES

- Is your boss asking about significance testing and confidence intervals? Are you interested in descriptive statistics, statistical tests, or plotting functions? Well statsmodels offers you that and more.

- All the data you require is available freely on the web but there is no download button and you need to scrape the website? You can extract data from HTML using Beautiful Soup and automate web browsing using Selenium!

- Want to make games? Try Pygame!

Guided Practise

DATA TYPES & VARIABLES

INSTRUCTIONS

- We recommend using a Jupyter notebook for this practice.

1. Open Jupyter: in a terminal type: **`jupyter notebook`**
2. Navigate to an appropriate folder where your work will be saved
3. On the top-right-hand-side click in the button called "New" and select "**Python 2**" or "**Root**" (depending on your installation of Python)
4. Voilà, you are ready to type the commands we will cover

IMPORTING A MODULE

- We need to import the functionality of packages and modules before we can use them
- Here we import the “math” module to use mathematical functions:

Python

```python
import math
x = math.cos(2 * math.pi)
print(x)

from math import *

log(10)

log(10,2)
```

TYPES, VARIABLES, ASSIGNMENT

- Like any other programming language, we need to use types and variables and be able to assign values to them

Python

```python
# variable assignments
x = 1.0
my_variable = 12.2
type(x)

y = 1
type(y)

b1 = True
type(b1)

s = "String"
type(s)
```

YOUR TURN

- Try the following in your Jupyter Notebook:

Python

```python
import types
print(dir(types))

1+2, 1-2, 1*2, 1/2

1.0+2.0, 1.0-2.0, 1.0*2.0, 1.0/2.0

# Comment

# Comparison: >, <, <=, <=, ==
2 > 1

# Testing for equality
2 == 2
```

LISTS

- Lists are collections of objects
- They can be changed

Python

```python
l = [1,2,3,4]
print(type(l))
print(l)
print(l)
print(l[1:3])
print(l[::2])

# Python starts counting from 0
print(l[0])
```

TUPLES

- Tuples are very similar to lists, but
- they cannot be changed

Python

```python
point = (10, 20)
print(point, type(point))

x, y = point
print("x =", x)
print("y =", y)
```

DICTIONARIES

- Dictionaries combine keys with values in pairs
- Like in a dictionary, you can search the keys to obtain their corresponding value

Python

```
1 params = {"parameter1" : 1.0, "parameter2" : 2.0,
2 "parameter3" : 3.0,}
3 print(type(params))
4 print(params)
```

INDEPENDENT PRACTICE

THINKING LIKE A PROGRAMMER

PSEUDOCODE

- Pseudocode allows us to represent a program concisely.
- The only thing you need is a statement to show where you are starting and where you are ending a program.
- Calculate and print the average of three numbers: 5, 10, and 15.

```
1  Start
2      num1 = 5
3      num2 = 10
4      num3 = 15
5      sum = num1 + num2 + num3
6      average = sum/3.0
7      print average
8  End
```

ACTIVITY: WRITE YOUR OWN (PSEUDO)CODE

DIRECTIONS

Create some pseudocode for one of the following tasks:

1. Create a short script that will calculate the area circle with radius r
2. Calculate and print the square of a number. If the number is larger than 10 also calculate the cube
3. List the letters in the sentence "Python is awesome"

PYTHON PROGRAMMING FUNDAMENTALS

INSTRUCTIONS

- We recommend using a Jupyter notebook for this practice.

1. Open Jupyter: in a terminal type: **`jupyter notebook`**
2. Navigate to an appropriate folder where your work will be saved
3. On the top-right-hand-side click in the button called "New" and select **"Python 2"** or **"Root"** (depending on your installation of Python)
4. Voilà, you are ready to type the commands we will cover

PYTHON PROGRAMMING FUNDAMENTALS

- Understanding core programming concepts and why they are used is just as important as knowing how to write code.
- Before we start,we'll review some basic programming concepts:
- Variables: A symbolic name that stores a value (some specific piece of information). Variables have different types.
- For example: **r = 3**

PYTHON PROGRAMMING FUNDAMENTALS

- Control Structures: A block of programming that analyses variables and chooses a direction in which to go based on given parameters.

- The term flow control details the direction the program takes (which way program control “flows”). It determines how a computer will respond when given certain conditions and parameters. Some typical structures include:

  - If statement
  - For loop
  - Functions

PYTHON PROGRAMMING FUNDAMENTALS

- Data Structures: Data structure is a particular way of storing and organising data in a computer so that it can be used efficiently. Some examples include:

  - Lists ← Native
  - Tuples ← Native
  - Dictionaries ← Native
  - Arrays ← Numpy
  - Matrices ← Numpy
  - Dataframes ← Pandas

PYTHON PROGRAMMING FUNDAMENTALS

- Syntax: The syntax of a programming language is the set of rules that define the combinations of symbols that are considered to be correctly structured programs in that language

- The interrelationship of all of these elements make it possible for us to write programs to implement algorithms and solve problems

IF

An `if` statement is a conditional structure that, if proved true, performs a function or displays information.

Think of this as a decision that moves the flow of your program depending on the answer to a TRUE-FALSE question.

In pseudocode:

```
IF a person is older than 18
THEN they can drive
ELSE they cannot drive
```

In Python we can write:

Python

```python
if age_person > 18:
    return "They can drive"
else:
    return "They cannot drive"
```

Another example:

Python

```python
A = 10
B = 100
if A>B:
    print("A is larger than B")
elif A==B:
    print("A is equal to B")
else:
    print("A is smaller than B")
```

FOR LOOP

A loop statement in programming performs a predefined set of instructions or tasks while or until a predetermined condition is met.

Think of this as a repetitive action that has to be performed until further notice.

In pseudocode:

```
FOR each user of a service in a list
   PRINT greet them
```

In Python we can write:

Python

```python
users = ["Jeff", "Jay", "Theresa"]

for user in users:
    print("Hello %s" % user)
```

FOR LOOP

Tip: When creating a for loop, make sure it's condition will always be met to help prevent an endless loop!

Let us see other examples. Can you explain what the program is doing?

Python

```python
for x in [1,2,3]:
    print(x)

for key, value in params.items():
    print(key + " = " + str(value))
```

LIST COMPREHENSION

List comprehension is an elegant way to define and create list in Python. It uses a for loop inside the definition of the list itself.

Let’s take a look at one, and see if you can figure out what is happening:

Python

```python
l1 = [x**2 for x in range(0,5)]
```

FUNCTIONS

A function is a group of instructions used by programming languages to return a single result or a set of results.

Functions are a convenient way to divide our code into useful blocks, providing us with order, making the code more readable and reusable.

FUNCTIONS

Here is how you define a function in Python:

```python
def function_name(input1, input2...):
    1st block of instructions
    2nd block of instructions
    ...
```

Let’s define a function that returns the square of the input value:

```python
def square(x):
    """
    Return the square of x.
    """
    return x ** 2
```

FUNCTIONS

We can call this function as follows:

```python
var1 = 7

var2 = square(var1)

print(var2)
```

DEMO

WRITING PROGRAMS IN PYTHON

WRITING PROGRAMS IN PYTHON

- Python is an interpreted language, which means you can run the program as soon as you make changes to the file.
- This makes iterating, revising, and troubleshooting programs is much quicker than many other languages.
- Let us create a complete program that will calculate the area circle with radius r.

Python

```python
# We are importing the value of pi from
# that module - Easy to read, right?
from math import pi

def circ_area(r):
    return pi * r**2

r = 3
area = circ_area(r)
print(area)
```

ACTIVITY: WRITE YOUR OWN CODE!

DIRECTIONS

Translate the your pseudocode into Python code!

1. Create a short script that will calculate the area circle with radius r.
   Hints: import math
   pi
   x**2

2. Calculate and print the square of a number. If the number is larger than 10 also calculate the cube.
   Hints: if
   x**2
   x**3

3. List the letters in the sentence "Python is awesome"
   Hints: set
   count

GUIDED PRACTISE

LET'S EXPLORE SOME DATA

INSTRUCTIONS

- Let's open a new Jupyter notebook for this practice.
- We will work in pairs.

1. Save the file called **Python101_GuidedPractice.ipynb** in a known location in your file system
2. Start Jupyter (you know how now) and navigate to the location where you saved the file
3. Open the file by clicking on the name
4. Voilà, you can start the Guided Practice

INDEPENDENT PRACTISE

YOUR TURN!

OPTION 1

- Let's open a Jupyter notebook for this practice.

1. Save the file called **Python101_IndPractice.ipynb** in a known location in your file system
2. Start Jupyter (you know how now) and navigate to the location where you saved the file
3. Open the file by clicking on the name
4. Voilà, you can start the Independent Practice

CONCLUSION

REVIEW & RECAP

In this workshop, we've covered the following topics:

- Python as a popular, flexible programming language
- Python has applications in many different areas
- Python programming basics including:
    - variable assignment (x = 1)
    - types (lists, tuples, dictionaries)
    - functions (def func_name)

- Python is particularly great for data manipulation

LEARNING PLAN

Evaluate your python programming skills! How confident are you with:

- Python Syntax
- Programming Fundamentals
- Data Analysis

WHAT SHOULD YOU DO NEXT?

For beginner programmers:

- Familiarise yourself with the language with an Introduction to Python
- Go through Learn Python the hard way
- And Automate the boring stuff

WHAT SHOULD YOU DO NEXT?

For existing programmers who are new to Python, try these:

- Read the information in Moving to Python from other languages
- Python for Java Developers
- Python for MATLAB users

WANT MORE?

General Assembly offers courses that use Python!

Check out our:

- Part-time Data Science Course
- Data Science Immersive Course

Q&A

EXIT TICKETS

DON’T FORGET TO FILL OUT YOUR EXIT TICKET

Demo

VISUALISING DATA

- Is it important?
- When is it needed?
- What does it add?
- What tools do you use?

INSTRUCTIONS

- We recommend using a Jupyter notebook for this demo.

1. Save the file called ***Python_101_DemoBokeh.ipynb*** in a known location in your file system
2. Open Jupyter: in a terminal type: ***`jupyter notebook`***
3. Navigate to the folder where you saved your file in step 1.
4. Click on the name of the file
5. Voilà, you are ready to follow the demo