GENERAL ASSEMBLY

PYTHON PROGRAMMING 101

Dr. Elaina A. Hyde

Western Sydney University
Adjunct Fellow

LEARNING OBJECTIVES

- Discuss the history of Python & how it's used in different industries
- Define how Python compares to other programming languages
- Describe the benefits of a Python workflow when looking at data
- Demonstrate basic Python programming fundamentals to solve a real world problem
- Create a custom learning plan to build your data science skills after this workshop!

DATA SCIENCE 101

PRE-WORK

Pre-Work Review

- Bring a laptop with Anaconda installed. Use Python 3.

- We will be using Jupyter Notebooks for this workshop, if you have installed Anaconda, then you are ready to go.

OPENING

About Me: Pre data science (Astrophysics)

- PhD. May 2010-2014, Macquarie University, Australia, **PhD in Astronomy and Astrophysics** obtained September 2014
- MSc. August 2007 - August 2009, The University of Amsterdam, Amsterdam, the Netherlands, MSc obtained September 2009; **Masters in Astronomy and Astrophysics**
- MSc. Fall 2005 - Fall 2006, Jointly at the Ludwig Maximilian University, Munich, Germany, and the Max Planck Institute for Extraterrestrial Physics, 2005-2006: **Masters in Physics.** Awarded the **Marie Curie Fellowship** prize**.**
- B.S. Fall 2000 - Summer 2005, University of Arizona, Tucson Arizona, U.S.A, **Bachelor of Science** degree obtained May 2005; major: **Physics/ Astronomy**, minor: **Optical Engineering/ Planetary Sciences**

About Me: Data Science

- Lead Instructor: Data Science Immersive, General Assembly, Sydney Australia [April 2017 – June 2017]
- As the lead instructor, I am responsible for organizing and disseminating all the course material as well as creating new material as needed and managing the students and their capstone projects.
- In this course, I teach basic Data Science with some advanced techniques. Some of the topics I teach in this course are: Git, Python, Data Structures, Inferential and Descriptive Statistics, Probability, EDA processes, numpy, visualization, SQL, Tableau, Decision Trees and Random Forest, Machine Learning, Classification (KNN, logistic, etc), Clustering (kmeans vs DBscan), Regression (polynomial, logistic, etc.), hypothesis testing, bootstrapping, PCA and NLP, to name just a few.

ABOUT YOU

- Who are you?
- Why are you interested in Python?
- Have you done any programming before?
- Do you have any applications in mind?
- Fun Fact?

OUR EXPECTATIONS

- You're ready to take charge of your learning experience.
- You're curious and excited about Python!
- You've installed Anaconda with Python 2.7.

THE BIG PICTURE

- **What we'll cover:**
  - Why Python?
  - What can Python do for me?
  - Implementing Python into your workflow
  - Python Libraries
  - Programming (pseudocode and Python)
  - Dive into data with Python

THE BIG PICTURE

- **Why this topic matters:**
  - Programming is a sought-after skill
  - Python has been gaining popularity (why will see why!)

- **Why this topic rocks:**
  - Python opens up a door to a variety of opportunities, from data science to research

INTRODUCTION

WHAT IS PYTHON AND WHAT CAN IT DO FOR ME?

WHAT IS PYTHON?

- Created by Guido Van Rossum in 1991
- Emphasizes 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

SOME CHARACTERISTICS OF PYTHON

- Interpreted language: Code implementations execute instructions without having to translate them into machine-language instruction.

- In contrast, compiled code is executed by the computer's CPU (hardware)

- Interpreted code may run less quickly, but can be executed on multiple platforms without modification

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

ACTIVITY: CHECK YOUR KNOWLEDGE

EXERCISE

DIRECTIONS

1. Check with your classmate
    1. What is Python?
    2. What does readability mean
    3. Is there a difference between compiled and interpreted languages?
    4. What is object orientation?

DELIVERABLE

Explain some common terms related to Python programming

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?

- Extremely fun to develop in :)
- Basically everything can be done with Python.
- 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.


DISCUSSION
Using Python

DEMO

IMPLEMENTING A PYTHON WORKFLOW

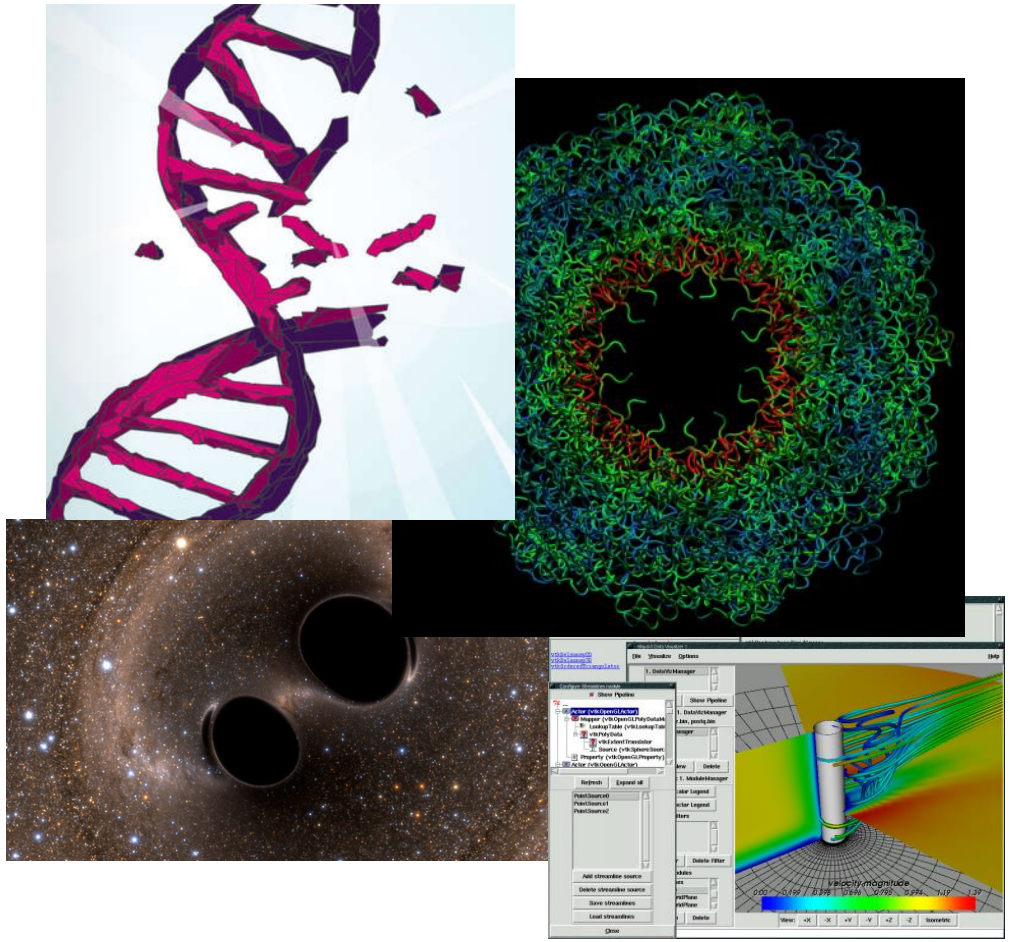
WHERE IS PYTHON USED?

- Everywhere! Industry & Academia
- Art Middleware – Tools and Plugins
- Research
- Web Development and Web Applications
- Game Development
- Windows Applications

- You name it!

EXAMPLES

- Industry
  - Drug discovery
  - Financial services
  - Films and special effects
- Academia
  - Gravitational waves
  - Scientific visualisation
  - Biomolecule simulation



PYTHON V OTHER LANGUAGES

- Python is often compared to other interpreted languages, such as:
  - Java,
  - JavaScript,
  - Perl,
  - Tcl, or
  - Smalltalk.
- Comparisons to C++, Common Lisp and Scheme can also be enlightening.


DISCUSSION
Programming
languages

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
print("hello world")
```

A very very simple program: one line of code that will print the string 'Hello World!.

It is easy to read and understand.

PYTHON PROGRAMMING

Let's see the C++ version:

C++

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

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

- It does exactly the same thing, but it requires us to write a lot more information!

What about Java?

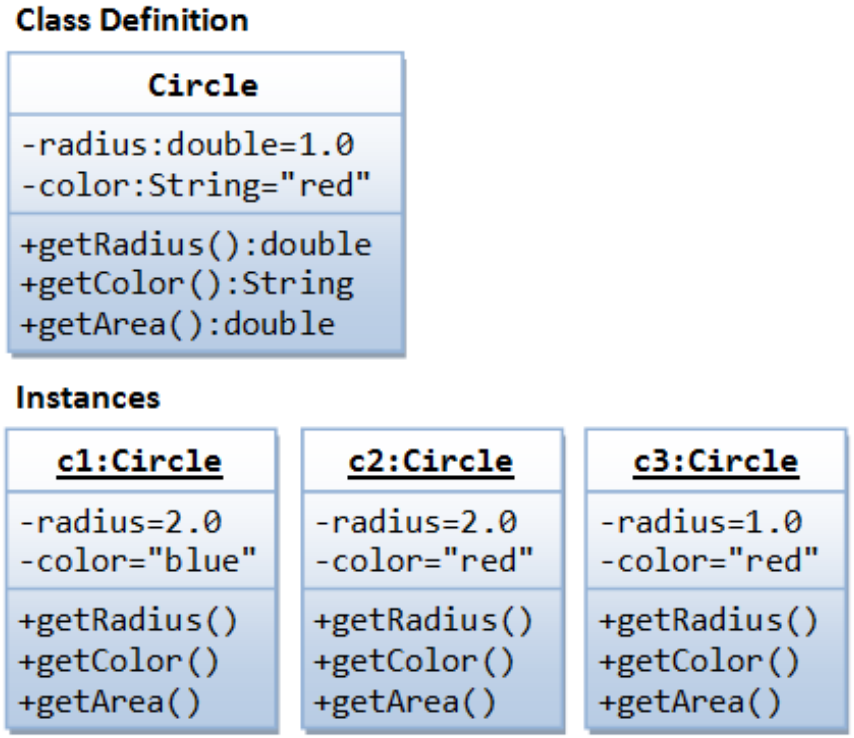
Java

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

- Once again, it does exactly the same, but it takes a lot more code!

CLASS

- In Object Oriented Programming, a **`class`** is a template definition of the methods and variables in a particular kind of `object`.
- In other words, an **`object`** is a specific instance of a **`class`**
- The **`object`** contains actual values instead of variables.

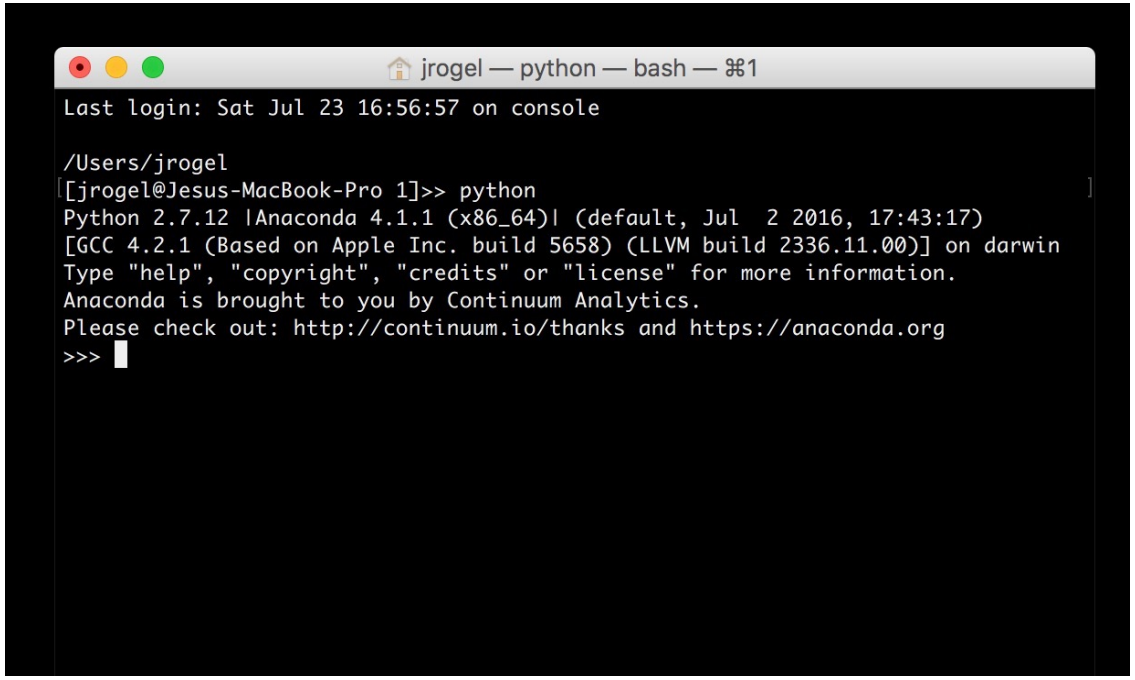


PYTHON: INTERACTIVE SHELLS V SCRIPTS

- 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”

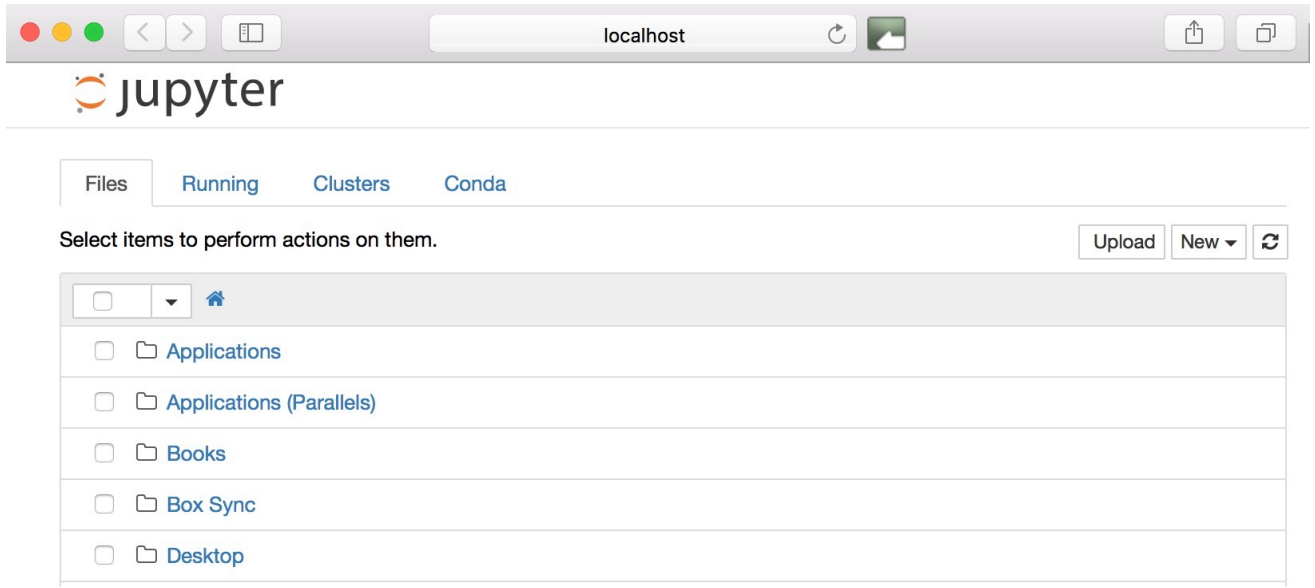


PYTHON SHELL

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

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 V 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 (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)

Academic Alternatives:
Emacs
Vim

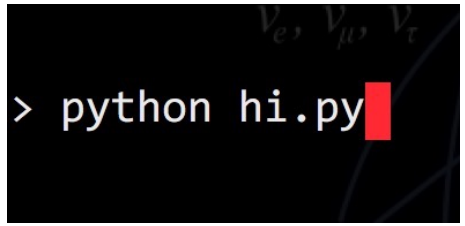

CHECK
Programming
Concepts

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:



PYTHON SCRIPT

- Unlike other languages, there's no `main()` function that gets run automatically
- the `main()` function is implicitly all the code at the top level.
- A more sophisticated version of the “Hello World!” program is as follows:

Python

```python
def main():
    print("hello world")

if __name__ == '__main__':
    main()
```

DEMO

VISUALIZING 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_Demo_Plotly.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

GUIDED PRACTICE

INSTALLING & CONFIGURING COMMON PYTHON LIBRARIES

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

PACKAGES

- Libraries 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 yo 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

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

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

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

```
1  import math
2  x = math.cos(2 * math.pi)
3  print(x)
4
5  from math import *
6
7  log(10)
8
9  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

```python
params = {"parameter1" : 1.0, "parameter2" : 2.0,
"parameter3" : 3.0,}
print(type(params))
print(params)
```

INDEPENDENT PRACTICE

APPLYING PYTHON TO SAMPLE DATA

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.

```
Start
    num1 = 5
    num2 = 10
    num3 = 15
    sum = num1 + num2 + num3
    average = sum/3.0
    print average
End
```

ACTIVITY: WRITE YOUR OWN (PSEUDO)CODE

DIRECTIONS

1. Create some pseudocode for 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"

DELIVERABLE

Pseudocode explaining the necessary steps to achieve one of the tasks above

BREAK

INTRODUCTION

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 organizing data in a computer so that it can be used efficiently. Some examples include:

  - Lists
  - Tuples
  - Arrays
  - Matrices
  - Dataframes

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"
```

IF

Another example:

```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
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 (PSEUDO)CODE

DIRECTIONS

1. Translate the your pseudocode for the tasks we saw before

    1. Calculate and print the square of a number. If the number is larger than 10 also calculate the cube.

    2. List the letters in the sentence "Python is awesome”

DELIVERABLE

Python code that executes the tasks above

GUIDED PRACTICE

DIVE INTO SOME DATA WITH PYTHON

INSTRUCTIONS

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

1. Save the file called **`Python101_Part2_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 PRACTICE

PUTTING IT ALL TOGETHER!

WHAT ARE YOU INTERESTED IN?

- Given your interests and knowledge, which are you more interested in learning about:
  - Practical applications of Python?
  - Python fundamentals?

OPTION 1

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

1. Save the file called `Python101_Part2_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

OPTION 2: PYTHON FUNDAMENTALS

EXERCISE

DIRECTIONS

1. Write Python code to complete the following tasks:

    1. A recipe you are reading states how many grams you need for the ingredient. Unfortunately, your store only sells items in ounces. Create a program to convert grams to ounces. [ounces = (28.3495231)grams]

    2. Read in a Fahrenheit temperature. Calculate and display the equivalent centigrade temperature. The following formula is used for the conversion:

        C = (5 / 9) * (F – 32)`

    3. Calculate the amount obtained by investing the principal P for N years at the rate of R. The following formula is used for the conversion:

        A = P * (1 + R) ^ N

DELIVERABLE

Python code that executes the tasks above

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 is particularly great for data manipulation
  - Python programming basics include: types, variables, functions, and more!

TAKEAWAYS

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:

- Go through Learn Python the hard way
- Familiarize yourself with the language by going through A Beginner's Python Tutorial

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

WHAT SHOULD YOU DO NEXT?

For anyone looking for a challenge :)

- Challenge yourself by tackling the Python Challenge

Q&A