Python: An Advanced Scripting Language

By: Adam Crayton, Ashley Day, Gabby Prusse, and Zeke Pinkerton

What is it? And where did it come from? → Link

- Features
  - High-level, general purpose programming language
  - Emphasizes code readability with the use of significant indentation
  - Dynamically types and garbage-collected
  - Supports multiple programming paradigms (it is structured, object-oriented and functional)
  - No semicolons!
- History
  - Started getting worked on in the late 1980s as a successor to the ABC programming language.
  - First released in 1991 as Python 0.9.0
  - Invented by Guido van Rossum, shouldering sole responsibility for the project as the lead developer until July 2018.
  - Had major releases with Python 2.0 in 2000 and Python 3.0 in 2008.

Installation and Use

- Here is how to install Python onto your computer
- Python has a lot of cool things you can do with it!
  - Documentation Page
- Python has many uses to it. You can use it for
  - Web and Internet Development (Django, Pyramid)
  - Scientific and Numeric (SciPy)
  - Education
  - Desktop GUIs (wxWidgets, win32 extensions)
  - Software Development (SConds, Buildbot, Roundup)
  - Business Applications (Oodo, Tryton)
- Being that Python is a scripting language, you need an interpreter (which is what you installed). You must run it with tis interpreter
  - python file_name.py
  - py file_name.py

Basics of Python → GitHub Link

- Python is meant to be a very “easy-to-read” language, meaning the syntax is rather simple.
- helloworld.py
- As you can see, when printing, the method for it is print(), which prints whatever is in the parenthesis and automatically adds a new line, a lot easier than doing System.out.println() or printf(“. . .\n”)!
- Variables also don’t have types, meaning you can have a variable be anything! You can check this by calling type() on the variable
  - I.e.) x = 7 → print(type(x)) → Output: “int”
  - type_and_cast.py

Basics (continued)

- In Python, indentation is actually apart of the syntax! If you create a statement that usually indicates a block (if, for, while, etc.) requires syntax, if you don't include it, the program will throw an error and not run!
- Variable names in Python follow most of the same ground rules of most other programming languages, where the main rules are:
  - Name cannot start with a letter
  - Can only contain alphabetic, numeric, and underscore characters in the name
  - Must start with a letter or underscore
  - Cannot be a python keyword
  - Example: variable_names.py
- More examples:
  - sys.py
  - arg_sum.py

Bindings

- Python has **name bindings**. This means that we are “binding” the name to the object.
  - If we set x = 1, y = 2, then we have two bindings in the interpreter, where x points to 1 and y point to 2.
  - If we set z = x, then we have another binding in the interpreter where z points to 1 as well do it it binding a name to the object.
- This means that whenever we change a variables value, the interpreter binds the name of the variable to the objects value. This is good to reduce the possibility of data duplication.
- Because Python has name binding and it changes whenever it is set to something new, it has dynamic binding.

Scope

- Scope in Python is a lot like many other languages.
  - Variables defined in functions are unable to be accessed outside of functions
  - Global variables can be accessed anywhere, however cannot be modified inside a function call
- However, global variables can be modified inside a function with the **global** keyword.
  - The global keyword accesses the variable in a global way, allowing whatever calculations, modifications and such to GLOBALLY change the value of the variable.
  - More scope examples: scope.py

Functions

- Functions in Python are defined with the “def” keyword, where you follow it with the function name and parenthesis defining if any parameters will be passed in
- Parameters can also be arbitrary arguments where the function will get a *tuple* of arguments and access them just like an array.
- They can also only allow arbitrary **keyword** arguments where they will receive a *dictionary* of arguments
- functions.py

Classes/Object

- Despite Python being a scripting language, it supports OOP (Object-Oriented Programming), allowing us to create classes and objects!
  - Classes have a __init__() function initializes the functions attributes and sets the values (think of this as a constructor in Java or a struct in C)
  - Classes also have a __str__() function which allows you to format a class in string form, like to toString() method in java
  - The self parameter is a reference to a current instance of the class, and is used to access variables that belongs to class. However, it doesn't have to be names self, it can be named anything!
  - sample_class.py
  - recursion.py
  - intermediate_information.py

Lambda Functions

- One thing Python does is allows for distinct lambda functions. Lambda functions is a small anonymous function that allows for multiple arguments, but will only execute one expression.
  - **lambda *arguments* : *expression***
  - Lambda functions allow for similar tasks to be performed, but can also be various. For example, you could have a program that multiplies two unknown numbers together. A lambda function would allow this to become a doubler function, or maybe even a tripler function.
- lambda.py

Polymorphism and Inheritance

- Polymorphism and Inheritance are relatively similar to many other programming languages.
- polymorphism.py
- inheritance.py

Modules

- Modules is Python's version of Java packages. Python allows the creation of files that can then be imported and have their functions and variables be accessed.
- module_examples/
  - main.py ← class that imports all classes and is the "main" class
  - greeting.py ← has a greeting function
  - person.py ← has a Person dictionary with their name, age and country
  - person_hello.py ← file that implements both greeting and person (shows how you can only select things from a file)