Part One

Introductory Python

Why use Python?

- Python is one of the easiest languages to learn.
- It allows for rapid development and testing - essential abilities for the security professional
- There are a ton of prewritten modules that you can easily import.

Who uses Python?

- Eve Online - a very popular MMORPG was written with a Python variation called stackless.
- Battlefield - one of the most popular FPS franchises in the world has used Python for add-ons and chunks of the games' core functionality.
- SymPy - a handy Python library which is often used in the scientific world for quick algebra.

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!
—Tim Peters

Python, C, and ASM

There are 3 widely recognized levels of programming: low-level (Assembly) , high-level (C) , and scripting (Python, Perl, et al.). There are a few important characteristics of each:

- low-level - Closest to the processor, hardest to write/read, most efficient and powerful.
- high-level - A comfortable middle ground, low-level enough to compile into Assembly and get some of the power, but there are some efficiency losses when the compiler optimizes poorly.
- scripting - Furthest from the processor, these usually require some sort of interpreter (and are often called interpreted languages as a result), they're inefficient relative to low or high - level languages, but are much easier to read and write

The Python interpreter

- A fully-fledged, Turing-complete virtual computer (Not exactly, but it’s easier to think of it as one), the Python Interpreter is what executes a python program.
- Can be used to quickly script a task.
- Has a built in destructor and garbage-collector

The Python Interpreter (CONT)

Operates on a REP loop.

- R - read, take in a line of code
- E - Evaluate, turn the code into something the computer can understand, and execute it.
- P - Print, show the result

The Python Interpreter Demo

You may not understand all of this, it’s alright. This is just to demonstrate the interpreter and its utility.

dir() & help()

- dir() - returns the list of names in the current scope
- dir(object) - lists the object's attributes
- help() - starts the interactive "help" routine on an interpreter
- help(object) - Looks up the item as the name of a module, function, class, method, keyword, or documentation topic, and a help page is printed on the console.

Python Manuals (pydocs)

A glorious series of documents detailing everything native to Python. If it arrived in your Python download, it’s covered in help. These docs are phenomenal, and should be used often.

LET'S GET STARTED!!!