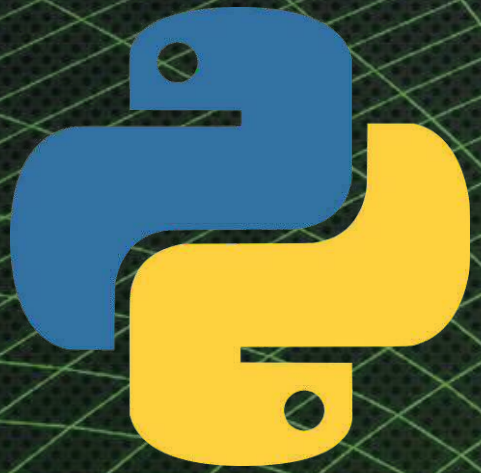

PYTHON DEEP DIVE

PART 1

This course is about

the Python language and built-in types

the standard library

becoming an expert Python developer

idiomatic Python

The Zen of Python

Tim Peters

`import this`

PEP 20

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!

Quick Refresher

Python's type hierarchy

variables

conditionals

functions

loops

break, continue and try

classes

Variables and Memory

what are variables? → symbols for memory addresses

memory

Python memory management → reference counting, garbage collection

mutability → function arguments, shared references

what is equality of two objects?

Python memory optimizations → interning, peephole

Numeric Types

integers

rationals

floats → binary representations exactness rounding
equality measures of closeness, approximate equality

Decimals → alternative to floats exactness precision rounding

complex numbers → `cmath` standard library

Numeric Types - Booleans

associated Truth values → every object has one

precedence and short-circuiting

Boolean operators → what they <u>really</u> do
using in context of associated truth values

comparison operators → identity, value equalities ordering

Functions

higher-order functions

docstrings and annotations

Lambdas

introspection

functional programming
- → map, filter, zip
- → reducing functions
- → partial functions

Functions - Arguments

positional arguments

keyword-only arguments

default values → caveats

packing and unpacking

variable positional arguments

variable keyword-only arguments

Functions - Scopes and Closures

global and local scopes

nested scopes

closures

nested closures

Decorators

decorators

nested decorators

parameterized decorators

stacked decorators

class decorators

decorator classes

applications → memoization, single dispatch, logging, timing

Tuples as Data Structures

tuples are not just read-only lists

data structures

packing and unpacking

named tuples

augmenting named tuples

Modules, Packages and Namespaces

what are modules?

what are packages?

how do the various imports work?

how to manipulate namespaces using packages

zip archives

`__main__`

Extras

will keep growing over time

important new features of Python 3.6 and later

best practices

random collection of interesting stuff

additional resources

send me your suggestions!