

Introduction to Python.

- Intended audience – is this you?
- Overview:
 - What it is and why you might like it;
 - Where to get it;
 - How to learn about it;
 - Basic usage – everything from “calculator” usage through to defining our own classes.
- Who is this guy Cian, anyway?

So what is Python?

- An easy-to-learn but powerful programming language.
- Portable, with a large standard library.
- Used to solve lots of different problems:
 - numerical computation;
 - systems automation;
 - fast network servers;
 - games.

Where do I get it?

- <http://www.python.org>
 - Downloads, documentation, etc.
 - Get a version for your favourite OS.
 - If you're running Linux, you've probably already got a Python interpreter installed: try typing
`python`
at your shell prompt.
- Note the word 'interpreter'.
- Start one now in your Redbrick shell.

How do I learn about it?

- Lots of tutorial information online:
 - <http://docs.python.org/tutorial/index.html>;
 - <http://diveintopython.org/>;
 - Other online docs at python.org.
- Dig in! Be curious. Play.
- Python does not bite.
- We'll run through the basics now.

Simple expressions.

- Try using your interpreter as a calculator.
- Comments start with the # character.
- Create a variable anywhere simply by assigning it a value.
- Various simple types:
 - Integers;
 - Floating-point numbers;
 - Strings;
 - Lists, tuples and dictionaries.

Control structures.

- Programming requires a few basic types of control flow.
 - Branching: `if ... elif ... else`
 - Looping: `while, for`
- Various other control-flow statements:
 - `break, continue, pass`
- List comprehensions.

Functions.

- We want to “package” common procedures for reuse as *functions* (or *methods*): `def`, `return`
- There are lots of useful builtins:
 - `len()`, `range()`, `dir()`

Classes and objects.

- Python supports object-oriented programming.
- A class is a type: data + operations on that data.
- An object is an instance of the type.
- Examples abound: lists, dicts, etc.
- A powerful means of abstraction.

The standard library.

- Be lazy: let other people solve your problems.
- Familiarize yourself with the library docs:
 - <http://docs.python.org/library/index.html>
- There's lots of other library and example code out there.

Things we haven't covered.

- There's lots more to Python:
 - File I/O.
 - Exceptions.
 - Lambdas or anonymous functions.
 - Defining modules.
 - Garbage collection.
 - The list goes on ...
- All of these things and more are covered by the docs or your favourite book. Dig in!

Wrapping up.

- Feel free to ask for help in Redbrick chat or on the newsgroups.
 - There are lots of knowledgeable folks around.
- Also feel free to email or hey me directly:
 - pooka@redbrick.dcu.ie
- Questions?

Introduction to Python.

- Intended audience – is this you?
- Overview:
 - What it is and why you might like it;
 - Where to get it;
 - How to learn about it;
 - Basic usage – everything from “calculator” usage through to defining our own classes.
- Who is this guy Cian, anyway?

1

Audience: smart people who are newish to programming.

We'll move reasonably fast.

Folks more familiar with programming might find this a bit slow, but hopefully there'll be notes of interest along the way.

Cian Synnott

* a.k.a. pooka@redbrick.dcu.ie

* Joined in 1996! Yes, probably some of you were five.

* Works as a Linux sysadmin, and writes a fair bit of Python as a result.

So what is Python?

- An easy-to-learn but powerful programming language.
- Portable, with a large standard library.
- Used to solve lots of different problems:
 - numerical computation;
 - systems automation;
 - fast network servers;
 - games.

2

Portable – runs natively on Windows, Linux, Mac; has been ported to e.g. Java Virtual Machine and .NET, even to some modern phones.

Standard library – comes with any implementation of Python, so can be used easily in portable code.

Where do I get it?

- <http://www.python.org>
 - Downloads, documentation, etc.
 - Get a version for your favourite OS.
 - If you're running Linux, you've probably already got a Python interpreter installed: try typing
`python`
at your shell prompt.
- Note the word 'interpreter'.
- Start one now in your Redbrick shell.

3

Interpreted language – contrast with compiled languages and note that Python does actually have a compiled representation.

Interpreter means that rapid prototyping is easy. Also makes learning pretty painless – you can try things out quickly.

Get them all logged into Redbrick and take a brief tour around the interpreter: ctrl-C, ctrl-D etc.

How do I learn about it?

- Lots of tutorial information online:
 - <http://docs.python.org/tutorial/index.html>;
 - <http://diveintopython.org/>;
 - Other online docs at python.org.
- Dig in! Be curious. Play.
- Python does not bite.
- We'll run through the basics now.

4

Dive Into Python is good for more experienced programmers who want to pick up a new language.

Bite: Despite the name, Python is not a very restrictive/constrictive language ...

We'll cover a good deal of the material in the official Python tutorial at speed in this workshop. Work through it at your leisure later to solidify your knowledge.

Simple expressions.

- Try using your interpreter as a calculator.
- Comments start with the # character.
- Create a variable anywhere simply by assigning it a value.
- Various simple types:
 - Integers;
 - Floating-point numbers;
 - Strings;
 - Lists, tuples and dictionaries.

5

Some calculator-style expressions – include use of +, -, /, *, %, **. Note difference between integer and floating point operations.

Comment a line. Note that comments will be most used in interpreted source files; break out of interpreter and give an example.

Create some variables and give more expression examples.

Demonstrate strings with ', “, “””. Demonstrate +, *, automatic concatenation, % 'printf-style'.

Show some list examples. Demonstrate subscripting, slicing. On strings too. Demonstrate immutability of strings vs. lists. Introduce tuples. Introduce dicts.

Control structures.

- Programming requires a few basic types of control flow.
 - Branching: `if ... elif ... else`
 - Looping: `while, for`
- Various other control-flow statements:
 - `break, continue, pass`
- List comprehensions.

6

Show examples of 'if' statements. Discuss boolean operators `==, !=, <, <=, >=, and, or, not, in, is`.

Note grouping of statements by indentation rather than `begin, end` or `{ }`.

Just mention `break` and `continue`, no need to go into them in detail.

Demonstrate `pass`.

Demonstrate some simple list comprehensions, and explain where they come from.

Functions.

- We want to “package” common procedures for reuse as *functions* (or *methods*): `def`, `return`
- There are lots of useful builtins:
 - `len()`, `range()`, `dir()`

7

Demonstrate some simple function definitions and usage.

Show the usage of builtins.

Mention the various fun you can have with argument lists, but don't go into it in detail.

Note that methods are for object-oriented programming in Python.

Classes and objects.

- Python supports object-oriented programming.
- A class is a type: data + operations on that data.
- An object is an instance of the type.
- Examples abound: lists, dicts, etc.
- A powerful means of abstraction.

8

Define and demonstrate the use of a simple class.

Demonstrate the difference between classes and objects.

Talk briefly about references & immutability again.

Demonstrate builtin objects using lists & the methods on them.

The standard library.

- Be lazy: let other people solve your problems.
- Familiarize yourself with the library docs:
 - <http://docs.python.org/library/index.html>
- There's lots of other library and example code out there.

9

Bring up the library docs and do a quick tour through.

Demonstrate the use of the library using a simple website monitoring script (urllib + re).

Mention other libraries: matplotlib, scipy, pygame, etc.

Things we haven't covered.

- There's lots more to Python:
 - File I/O.
 - Exceptions.
 - Lambdas or anonymous functions.
 - Defining modules.
 - Garbage collection.
 - The list goes on ...
- All of these things and more are covered by the docs or your favourite book. Dig in!

10

A quick tour – lots of sights we missed.

Basic features are simple, but there's lots more.

Pick a project, some program you want, and figure out how to do it in Python. Feel free to ask for help.

Wrapping up.

- Feel free to ask for help in Redbrick chat or on the newsgroups.
 - There are lots of knowledgeable folks around.
- Also feel free to email or hey me directly:
 - pooka@redbrick.dcu.ie
- Questions?

11

Note that Python is used all over the place these days in open-source projects and in industry. It's a useful language to be familiar with.

Not at all difficult to learn, and Redbrick can provide plenty of support.