**QAPYTH3**

**Python 3 Programming**

Instructors Guide

Version No. 2.2

[Course-level information 2](#_Toc259514147)

[Instructor’s Pack 2](#_Toc259514148)

[Course goals 3](#_Toc259514149)

[Equipment you should find in the classroom 3](#_Toc259514150)

[Course set-up 3](#_Toc259514151)

[Instructor prerequisites 4](#_Toc259514152)

[Delegate prerequisites 4](#_Toc259514153)

[Course Format and Approach 4](#_Toc259514154)

[Timing 5](#_Toc259514155)

[Chapter Summary 6](#_Toc259514156)

[0. Common introduction 6](#_Toc259514157)

[1. Introduction to Python 6](#_Toc259514158)

[2. Fundamental Variables 8](#_Toc259514159)

[3. Flow Control 9](#_Toc259514160)

[4. String Handling 10](#_Toc259514161)

[5. Collections 11](#_Toc259514162)

[6. Regular Expressions 12](#_Toc259514163)

[7. Data Storage and File Handling 14](#_Toc259514164)

[8. Functions 15](#_Toc259514165)

[9. Advanced Collections 16](#_Toc259514166)

[10. Modules and Packages 17](#_Toc259514167)

[11. Classes and OOP 18](#_Toc259514168)

[12. Error Handling and Exceptions 20](#_Toc259514169)

[13. Multitasking 21](#_Toc259514170)

[14. The Python Standard Library 22](#_Toc259514171)

[General 22](#_Toc259514172)

[Slides 22](#_Toc259514173)

[15. The Way Ahead 23](#_Toc259514174)

The Course Consultant is: Clive Darke, QA

E-mail address: [clive.darke@qa.com](mailto:clive.darke@qa.com)

Introduction

This document is intended to provide help to the existing and new lecturers of the “Python 3 Programming” course. The course layout, philosophy and structure will be discussed. This, together with the timing, potential difficulties and comments should help any lecturer in achieving good level of satisfaction when teaching this course.

Please email comments and suggestions concerning this Instructor's Guide to the course consultant.

This course is **not** an introduction to programming. It is targeted at people that are already experienced script designers and/or programmers.

Course-level information

## Instructor’s Pack

Instructors should be provided with the following items:

* QAPYTH3 Delegate Guide
* QAPYTH3 Appendices
* QAPYTH3 Exercise Guide
* QAPYTH3 Set-up Document
* QAPYTH3 Instructor Guide (this document)

If any of the items in the list above is missing, take it up with your normal contact with QA.

Errors detected since the printed edition of these manuals are mentioned in the pages which follow, and will be corrected in the next release. If you find any more please report them to the course consultant.

## Course goals

The main purpose of this course is to prepare the delegates for using the Python 3 programming language to write scripts .

## Equipment you should find in the classroom

Contact the local support staff if any of the items below are missing

- A fully installed machine per delegate, with delegate packs

- A fully installed lecturer machine

## Course set-up

All public courses run at QA locations will be pre-installed, ready for the class. That includes all hardware, software installation and configuration as well as user accounts configured with the correct environment and all files required for the practicals.

Files of interest to the instructor, including custom modules may be found in the following directories:

Windows: InstructorSupport

Linux: Setup/instructor.tgz (tar –xvzf to extract)

Note: this course is not suitable for Python 2 installations!

## Linux

## From March 2011 the build includes a Linux VM (CentOS), but this might be omitted for on-site courses. The VM is based upon the same build as is used in Linux, Perl, and shell courses, but with some improvements specifically for Python.

## It includes Python 3.6, Eclipse and IDLE, and has been included because of requests from ACO delegates.  It is self-contained and does *not* go through the installation procedure that our UNIX/Linux courses use.

## Everyone has the same username/password: qa, and the machine name is yogi. User root has the password secret. It shares the C: drive with Windows, visible through /mnt/C\_DRIVE.

## IDLE warning: if you are signed on as root, then su to another user and try and use IDLE it will fail with 'couldn't connect to display 0:0'. This is because the second user is trying to open root's display.

## If the Linux VM cannot gain internet access, IDLE <F1> help cannot display the Python help text. To avoid this issue, from September 2012, the Python help text is installed locally in /var/www/html/python. External links from the help pages will still not work without internet access of course.

Python 2 .6 is installed in /usr/bin/python.

Matplotlib errors:

If you get the following message:

/usr/lib/libstdc++.so.6: version `GLIBCXX\_3.4.15' not found

This is caused by incompatible runtime libraries. Ensure the following is set:

export LD\_LIBRARY\_PATH=/usr/local/lib

## Eclipse

## Eclipse is installed on Windows and Linux for delegates who are already familiar with this IDE.

## The setup for Eclipse with the PyDev plug-in should be the same on Windows and Linux. There are two workspaces, projects and solutions, on Windows these are under C:\Labs and on Linux under /home/qa. Within these there is a project for each chapter.

## If you wish to change the editor font for demonstration purposes:

## Windows->Preferences->PyDev->Editor->ColorsandFonts->

## Basic->TextFontInstructor prerequisites

Number one, a sense of humour. Open Source is meant to be fun!

The course is run on Windows 10 with Python 3, however there is no reason why it could not be run on a major implementation of Linux, such as Red Hat or SuSE. However, at the time of writing, Python 3 is not generally bundled with these distributions, so it will probably have to be installed. Currently the exercises have not been tested on any platform other than Windows 10 and CentOS, so assumptions should not be made.

A few Python Easter eggs for the amusement of delegates ('import this' is in the course already):

from \_\_future\_\_ import braces

SyntaxError: not a chance

import antigravity

Starts a browser set to <http://xkcd.com/353/>

Note: import \_\_hello\_\_ does not seem to do anything on Python 3

IDLE hints:

Set the font to Lucida Console 16 bold

Set the Initial Windows size to Height 25

To clear the screen: print("\n"\*25)

## Delegate prerequisites

Required: solid experience of either of UNIX or Windows environments. The delegates should be accomplished programmers in any high level language – which does not include HTML or SQL. The course doesn’t spend time on first principles and delegates that don’t have a programming past struggle. Unfortunately, there is usually a number of them in the class….

## Course Format and Approach

The course format follows the standard, with a title slide, content, then finishing on a summary slide.

Additional slides often follow the summary, these cover obscure aspects, or some common pitfalls. I suggest you mention these to delegates, and use them as back-up to answer questions.

The Appendices do not necessarily follow this format, and represent additional material which may be referred to, or even taught if there is sufficient time and interest. There are no labs for material in the Appendices.

The practical sessions can have a lot of time allocated. Currently the course is a four-day event, with plenty of time to explore areas of interest to the delegates.

## Timing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Time | Day 1 | Day 2 | Day 3 | Day 4 |
| 9:00 |  | Review | Review | Review |
| 9:15 |  | 5 – Collections | 8 - Functions | 11 - Practical |
| 9:30 | 0 - Introduction |  |  |  |
| 9:45 |  |  |  |  |
| 10:00 | 1 – Intro to Python 3 |  | 8 - Practical | 12 – Error Handling and Exceptions |
| 10:15 |  | 5 – Practical |  |  |
| 10:30 |  |  |  |  |
| 10:45 | 1 – Practical |  | 9 - Advanced Collections | 12 - Practical |
| 11:00 |  |  |  |  |
| 11:15 | 2 – Fundamental Variables | 6 – Regular Expressions |  |  |
| 11:30 |  |  | 9 - Practical |  |
| 11:45 |  |  |  | 13 - Multitasking |
| 12:00 | 2 – Practical |  |  |  |
| 12:15 |  |  |  |  |
|  | ***Lunch*** | ***12:30 - 13:30*** |  |  |
| 13:30 | 3 – Flow Control | 6- Practical | 10 – Modules and Packages |  |
| 13:45 |  |  |  | 13 - Practical |
| 14:00 |  |  |  |  |
| 14:15 | 3 - Practical |  |  |  |
| 14:30 |  | 7 – Data Storage and File Handling | 10 - Practical |  |
| 14:45 |  |  |  | 14. The Python Standard Library |
| 15:00 | 4 – String Handling |  |  |  |
| 15:15 |  |  | 11 – Classes and OOP |  |
| 15:30 |  | 7 - Practical |  | Conclusion |
| 15:45 |  |  |  |  |
| 16:00 | 4 – Practical |  |  | *Good Byes* |
| 16:15 |  |  |  |  |
| 16:30 |  |  | 11 - Practical |  |
| 16:45 |  |  |  |  |

# Chapter Summary

## 0. Common introduction

You may wish to display the course contents, and briefly walk through the subjects covered. You may also wish to introduce the course materials, and the three manuals. Now is a good time for personal introductions, and to check prerequisites.

It helps to get the delegates to state the operating system they use. This allows pitching the discussions for the rest of the course. For example, if there is nobody with Linux or UNIX, you can virtually skip mentioning it.

## Introduction to Python

**Slides**

#### **What is Python?**

#### **What is Python 3?**

Python 3.2 is the current release. Main significant improvement so far is improvements to threading. The philosophy surrounding the GIL has been rewritten.

#### **Why Python?**

Note that there are no Python 3 versions of Jython or IronPython at the time of writing.

#### **Why Python?**

#### **Why Python?**

At the time of writing (August 2012), this is the status of the listed products on Python 3:

wxPython No version (GUI toolkit) 2.7 support just released. In beta test.

PyQt Released

NumPy Released

SciPy Released

DISLIN Released

JSON Implemented as part of the Python 3 base (Javascript Object Notation)

Django Released (Framework)

Pyjamas No information

Zope Released

PostgreSQL is available, and at least one mysql (not specifically tested by QA)

#### **Performance downsides**

Psyco is deprecated in favour of PyPy - Psyco will probably never work with Python 3 or 64-bit since maintenance has ceased.

Google was developing *Unladen Swallow*, however this project seems to be in doubt. It appears that some of the improvements might see their way into 3.3, but the swallow might be more like a parrot - passed-on and joined the choir invisible, bereft of life it rests in peace, etc.

#### **The community**

#### **Running Python interactively**

#### **Python scripts**

#### **Python help**

#### **Anatomy of a Python script**

#### **Modules**

Here's a simple module demo:

import webbrowser

webbrowser.open("http://www.python.org")

#### **Functions and builtins**

#### **Summary**

#### **Python builtin functions (1)**

#### **Python builtin functions (2)**

#### **Labs**

The purpose of these labs is to get familiar with the environment.

Please ensure you point out to delegates the labs directory, which, as well as solutions, contains some code templates and test data. The notes directory contains all the code from the slides.

Note that you might have to introduce

sys.path.append('C:/labs').

Don't spend too long on these labs, make sure everyone can at least create and run a script. Encourage fast-trackers to browse the documentation.

**Note:** On Windows cmd.exe, line endings are "\r\n", and in Python 3.2.1 a bug (<http://bugs.python.org/issue11272>) strips out the new-line but not the "\r". The symptoms are:

reply = input("Please enter a value:")

print("<",reply,"> was input")

we get:

Please enter a value:one

> was input

The "\r" has the effect of hiding the characters before it. There are several solutions, one is to remove all trailing whitespace:

reply = input("Please enter a value:").**rstrip()**

## 2. Fundamental Variables

**Slides**

#### **Python is Object Oriented**

#### **Python variables**

#### **Variable names**

#### **Type specific methods**

In IDLE:

>>> x = 'hello'

Contrast:

>>> dir(x)

>>> help(x)

>>> print x.\_\_doc\_\_

#### **Operators and type**

#### **Augmented assignments**

#### **Python types**

Note that this is an animated slide. Animations:

1. A grey box appears around sequences
2. That box, and the py3 logo, disappear
3. A horizontal line appears dividing immutable and mutable object types

#### **Switching types**

#### **Python lists introduced**

#### **Python tuples introduced**

#### **Python dictionaries introduced**

#### **Summary**

#### **Python operators**

#### **Python reserved words**

**Labs**

## 3. Flow Control

**Slides**

**Python conditionals**

**Indentation**

**What is truth?**

**Boolean and logical operators**

**Chained comparisons**

**Sequence and collection tests**

**Object types**

**A note on Exception Handling**

**While loops**

**Loop control statements**

**For loops**

**enumerate**

**Counting 'for' loops**

**Zipping through multiple lists**

**Conditional expressions**

Delegates will be less than enthusiastic about these. Point out that they are used inside Lambda functions which we will see later

**Unconditional flow control**

If you are confused about the difference between exit() and sys.exit() then try:

print(type(exit))

print(type(sys.exit))

**Summary**

**Labs.**

## Question 3 suggests is\_decimal(), note that this is Python 3 specific (a Unicode function is Python 2.7). For Python 2 use is\_digit(), which is not exactly the same.

## 4. String Handling

**Slides**

**Python 3 strings**

**The print function**

**Cooking strings**

**String concatenation**

**"Quotes"**

**String methods**

**String tests**

**String formatting**

**String formatting examples**

The # before x inserts the 0x, so:

fred = '{:x}.format(3735928559) gives: deadbeef

**Other string formatting aids**

**Slicing a string**

**String methods - split and join**

**Summary**

**More string formatting examples**

**String formatting - old style**

**String % formatting examples**

Worth mentioning if you have Python 2 delegates

**Exercise**

In greek.py, the hex value could be better expressed as “{:#06x}”

**Labs**

### Here is an extra exercise you might like to demo:

### import unicodedata

### greek = ['Alpha','Beta','Gamma','Delta','Epsilon','Zeta','Eta',

### 'Theta', Iota','Kappa','Lamda','Mu','Nu','Xi','Omicron',

### 'Pi','Rho','Final Sigma','Sigma','Tau','Upsilon',

### 'Phi','Chi','Psi','Omega']

### 

### for cname in greek:

### try:

### lc = unicodedata.lookup("GREEK SMALL LETTER " +

### cname.upper())

### uc = unicodedata.lookup("GREEK CAPITAL LETTER " +

### cname.upper())

### print("{0:<12s}: {1} {2}".format(cname,lc,uc))

### 

### except (UnicodeEncodeError,KeyError) as err:

### print(err)

### 

### 5. Collections

**Slides**

**Python types – reminder**

**Useful tuple operations**

**Python lists**

**Tuple & list slicing**

**Extended iterable unpacking**

Perl people will like this….

**Adding items to a list**

Worth noting that adding items to the end of a list is more efficient than at the start. Use collection.deque if this is required.

**Removing items by position**

**Removing list items by content**

**Sorting**

**Miscellaneous list methods**

**List methods**

Reference only

**Sets**

**Exploiting sets**

**Set operators**

**Python dictionaries**

**Dictionary values**

**Dictionary methods**

Reference only

**View objects - examples**

**Summary**

**Labs**

##### You might like to introduce pprint (discussed in the final chapter), particularly if Perl people are looking for a Data::Dumper.

##### **Exercise 4**

Brute Force could use enumerate - added new solution.6. Regular Expressions

See also the Advanced Regular Expressions appendix.

**Slides**

**Regular expressions**

**Python regular expressions**

**Elementary extended RE meta-characters**

**Regular expression objects**

**Example program: searching input files**

**Regular expression substitution**

**Regular expression split**

**Matching alternatives**

**Anchors**

**Class shortcuts**

**Flags**

New with 3.1

**Repeat quantifiers**

**Quantifiers**

**Back-references**

**Global matches**

**Summary**

**Labs**

7. Data Storage and File Handling

**Slides**

**New file objects**

**Reading files into Python**

**Reading tricks**

**Binary mode**

**Writing to files from Python**

**Standard streams**

**More tricks**

**Random access**

3.1 Constants SEEK\_SET, SEEK\_CUR, SEEK\_END

**Python Pickle Persistence**

**Pickle Protocols**

3.1 Additions to notes

**Build some shelves**

**Compression**

**Summary**

Note that these slides have been recently moved here:

**Database interface overview**

**Example - SQLite from Python**

**Labs**

## Try to encourage delegates to use a set or dictionary solution.8. Functions

**Slides**

**Python functions**

**Function parameters**

**Variadic functions**

**Assigning default values to parameters**

**Named (keyword) parameters**

**Enforcing named parameters**

**Returning objects from a function**

**Variables in functions**

**Nested functions**

**Variables in nested functions**

**Find the bugs!**

**Function documentation**

**Lambda functions**

**Lambda as a sort key**

**Lambda in re.sub**

**Summary**

**Bugs fixed!**

**Function attributes**

**Function annotation**

**Labs**

## 9. Advanced Collections

**Slides**

**Advanced list functions - filter**

**List comprehensions**

**Set and dictionary comprehensions**

**Lazy lists**

**Generators**

**Generator objects and next - coroutines**

**Copying collections - problem**

**Copying collections - slice solution?**

**Copying collections - deepcopy solution**

**Summary**

**Labs**

## The C module should be available as an installer executable in:

labs/Getprocs-1.0.win32-py3.2.exe

If you are not running on a compatible version of Python, then you can build it from the GetProcs subdirectory.

## 10. Modules and Packages

If you are running a demo from IDLE, don't forget to reload the module each time:

>>> import imp

>>> imp.reload(*modulename*)

Note differences with Python 3.2, in particular \_\_pycache\_\_.

**Slides**

**What are modules and packages?**

**Multiple source files**

**How does Python find a module?**

**Importing a module**

**Importing names**

**Directories as packages**

**Writing a module**

**Module documentation**

**Testing a module**

**The Python debugger**

**The Python profiler**

**Summary**

**Labs**

## 11. Classes and OOP

**Slides**

**Object-Oriented terminology**

**Object-Oriented Programming**

**Using objects**

**A little Python OO**

**Defining classes**

**Defining methods**

**Constructing an object**

**Special methods**

**Operator overload special methods**

**Special methods – example**

**Properties**

**Properties and decorators**

References:

http://www.artima.com/weblogs/viewpost.jsp?thread=240808

http://www.ibm.com/developerworks/linux/library/l-cpdecor.html

PEP318

The concept of decorators is rather advanced, so don't worry delegates too much about how property() is actually being called, concentrate on the effect.

**Class methods**

**Inheritance**

**Inheritance terminology**

**Inheritance example**

**Summary**

**Metaclasses and ABC**

**Labs**

You may have to explain that the solution has some lines commented out for different questions.

## 12. Error Handling and Exceptions

**Slides**

**Error Handling and Exceptions**

**Writing to stderr**

**Exception handling**

**Exception syntax**

**Multiple exceptions**

**Exception arguments**

**The finally block**

**Order of execution**

**The Python exception hierarchy**

**assert**

**The raise statement**

**Raising our own Exceptions**

**Getting tracebacks**

New slide

**Summary**

**Context managers - with**

13. Multitasking

**Slides**

**Family life**

**Creating a process from Python**

**Old interface examples**

**Waiting for a child**

**Using the subprocess module**

**Popen method**

**Running a basic process**

**Capturing the output**

**Passing data through a pipe**

**Processes and threads**

**Very basic threads in Python**

**Synchronisation objects in threading**

**Simple use of Lock**

The code comes from an "Advanced Python" sockets lab

**The trouble with threads**

**Using the multiprocessing module**

Emphasis is on compatibility with threading module, which includes Queues which follows.

**Queue objects**

**Queue objects example(2)**

**Summary**

14. The Python Standard Library

General

There are 62 scripts in Tools\Scripts, probably worth taking a quick look.

Slides

**The Standard Library**

**Example - converting Python 2 scripts to Py3**

**Pretty Printer - a useful utility**

**Operating System interfaces - os and friends**

**os.open example**

**System specific attributes - sys**

**Signal handling - signal**

**Converting a signal to an exception**

**Configuration files**

**The configparser module**

**The datetime module and friends**

**datetime example**

**The platform module**

**External function interface – ctypes**

**Win32 ctypes example**

**\_\_future\_\_**

**Other modules**

**Summary**

15. The Way Ahead

#### **Recommended books**

#### **What have we achieved?**

#### **What next?**

#### **In closing...**