Deploying Python Programs on Windows Platforms

Adrian P. Robson

1 March 2016

Contents

1 Introduction

This report explains how a Python application program can be 'installed' on a Windows computer.

- without distributing it with a specialised installation tool, such as Inno Setup,
- or converting it to an `exe` file with a tool like py2exe.

This approach requires that python programs are distributed as files of plain text. If you do not want what is effect open source code, you will have to choose

a different way to distribute your programs. Module files are discussed, but packages are not considered in this report.

Three types of application are considered: Graphical User Interface (GUI), Console User Interface (CUI) and command line. The first two have a similar treatment, but command line invocation needs a little more work. Consult the summary section (§6) for a short overview. Other sections look at the process in more detail: the need for Python to be installed is explained (§5) and some sample installation instructions are given (§7). Strategies for invoking GUI and CUI programs are described together(§2), but command line execution is looked at separately (§3).

2 GUI and CUI Program Execution

GUI application can be very easy to use, but programs with a CUI are much simpler to write, and for many applications a CUI can be more than adequate.

A *program* is one or more files that contain Python language text (scripts). One of these is the entry point of the program, which we will call the *main file*. Other program files are *modules*, which are collections of useful function definitions.

A Python program can be started by double left clicking its main file icon[1]. However, it is often more convenient to to invoke a Python program with a *file shortcut* that can be put into the Program Start menu or on the Desktop (See§2.1).

Graphical Usue Interface programs should have the type `pyw` to prevent a console window being opened when the program starts.

Console interface applications have to run in a terminal window, so they must have a `py` file type. Unfortunately, the terminal window closes when the program finishes, possibly hiding important information. To prevent this, *CUI applications should pause before they finish* (See §2.2).

2.1 File Shortcuts

File shortcuts are good for invoking GUI and CUI applications. They are normally put in the Start menu or on the Desktop, and double left clicking them will start the program.

There are a few of ways to make a shortcut for a Python program. Here are a couple:

- 1. Right click on the program file, then left click Create shortcut.
  2. A new shortcut will appear in the folder. Rename it as required, and move (drag and drop) it to where you want it to be. The program file name in a shortcut is an absolute path, so is okay to move the shortcut anywhere you want to, but it will probably not work if it is transfered to a different computer say as part of a folder structure.

[1]This only works if file type associations have been set for `py` and `pyw`, which should be so if Python has been correctly installed.

- This method is rather longer but puts the shortcut directly into the Start menu:

  1. Left click Start, right click All programs, then left click Open or Open all users.
  2. A Start Menu window will appear. Open the `Programs` folder.
  3. Open the folder where you want the shortcut to be shown, creating a new folder if required.
  4. Right click in the folder window, select new and then left click Shortcut.
     (a) Browse to the Python program file, select it, then left click `Next`.
     (b) Type in a name for the shortcut, then left click Finish.

     The new shortcut will be created.

The working folder of a shortcut can be changed by modifying the Start in box in the shortcut's properties window. However, as with the program, the path entered must be absolute. Shortcuts can be combined with batch files (§3.1) to achieve more complicated behaviour.

2.2 Pausing a Console Application

A console application started from a shortcut runs in a window with command line input and output. Unfortunately, the window closes when the program ends, possibly hiding important information. *Putting a pause at the end of the program* is an easy way to fix this:

```
input("Press enter to close window:")
```

Unfortunately, this can be incompatible with using the program from the command line, where a pause is not wanted. A solution is to use a command line parameter to control the program's behaviour. Assuming that there are no other arguments, put this statement at the end of the program:

```
if len(sys.argv) == 2 and sys.argv[1].lower() == "-pause":
    print()
    input("Press enter to close window:")
```

Then in the program's shortcut properties add a pause flag to the the end of the path in the Target box like this:

```
C:\Users\...\myprog.py -pause
```

In the equivalent `bat` file, the pause argument is omitted.

3 Command Line Execution

Command line execution is starting a program by typing its name in an already open terminal window. User interaction then continues as it would with a shortcut started CUI program. Without help the *full relative file name and type* of its main file has to be used. However, a neat alternatively is to use a `bat` script file to start the program with just a simple command name (§3.1). For this to work the folder containing the `bat` file must be in the system path (§3.2).

Shortcuts are no help in this context. They can be used from the command line but their type (`lnk`) has to be given (!).

Do I need this could it be a footnote

3.1 Startup Scripts

Batch (`bat`) files can be used to start a Python program from the command line without having to giving a file type. They offer a versatile way to start Python programs: the working folder can be changed, pre or post processing can be performed, alternative versions of Python can be used. As a simple example, a file called `demo.bat` could contain:

```
@echo off
rem Wrapper for demonstration program.
demo/demo2.py demo start.ini %1 %2 %3
```

With this, the command `demo` will invoke a program called `demo2.py`, which is in sub-folder `demo`. It is given `start.ini` as a argument, and it will accept up to three further arguments, which can be given with the command.

The use of `bat` files is not restricted to command line execution. They can be combined with shortcuts for CUI and GUI programs.

3.2 System Path

The Windows `PATH` variable is used by the operating system to locate programs started from the command line. It contains a list of folders that are searched to find the required executable. Adding a folder to this list is done as follows:

1. Left click Start menu, Computer (in left column), System properties (top menu bar), Advanced system settings (right column) and then Environment Variables (Button).
2. This opens a dialogue window: Locate and select `PATH` in the System Variables list, and left click the Edit (button).
3. This opens another dialogue window: Change the string in its Variable value text box by adding a semicolon and the *absolute path* of the new folder to the end of the string.
4. Repeatedly left click okay to close all dialogues and commit the change.

Take great care not to change or delete and other variables.

4 Program Location

Python programs can be put anywhere if a shortcut or `bat` file is used to invoke them, but scattering programs about the file system is not advised. A better idea is to put them in a dedicated folder. There are a couple places where such a folder might be sensibly put:

Program Files folder is the place where most installed applications are stored. This is probably the best location for normal installation. There might also be a `Program Files (x86)` folder, which holds 32-bit executables[2]. For consistency, use `Program Files` if there is a choice.

[2] 64-bit versions of Microsoft Windows have `Program Files` and `Program Files (x86)` folders for backward compatibility.

Documents folder is easy to access and is normally included in a standard backup. This is be good method for developers. Using a single root folder for all programs and other executables is recommended. This cam of course have many sub-folders to organise the programs. (See §4.2 for example of how this might be done.)

If command line execution is used, then the folder holding the program's `bat` file must be in the system path (see §3.2).

4.1 Module and Other Files

4.1.1 Modules

Modules are most simply stored in the same folder as the main program, or in sub-folders. The location affects the module's `import` name. So the module `mymodule` in sub-folder `modules` would be loaded with

```
import modules.mymodule
```

where module folders are referenced relative to the program's folder[3].

4.1.2 Data files

When a program creates a data file with a relative path, it is put in the *working folder*. If the program is started with a shortcut, the working folder will by default be where the Shortcut was created. If it is run as a console application via a `bat` file, the default working folder will where the `bat` file is stored. Starting the program from the command line makes the working folder the console window's current folder.

The working folder can be changed: For shortcuts modify the Start in box in the shortcut properties window, but an absolute path must be given.

4.2 Folder Strategies

Here we look at ways to store a Python program in the target computer's file system in more detail than in §4 above. For this purpose, a program consists of a main script file and zero or more module files. There can also be data, documentation and `bat` files.

There is no definitive scheme for storing Python script files, but a number of examples are given below:

Two separate applications in Program Files:

The program `myapp` is a GUI application. It is started with a shortcut in the Start Menu. The program `myapp2.py` is a command line program.

```
Program Files\
   ...
   pyapp\
       myapp.pyw        ← shortcut to this file
       myappmod.py
```

[3]There is a `PYTHONPATH` environment variable and some Python start parameters that can affect the module search path, but these are not considered here.

```
pyapp2\              ← folder in system path
    myapp2.py
    myapp2.bat       ← for command line execution
```

A program that is a single file in Documents:

The program `myprog` in a consol interface application, and it is invoked from the command line. The folder `bin`[4] is used to store programs and other executables.

```
Documents\
   ...
   bin\              ← folder in system path
      ...
      myprog.py      ← shortcut to this file
      myprog.bat     ← for command line execution
```

A single program that has multiple files in Documents:

The program `myprog\main` has a console interface and is also executed from the command line. It has a couple of modules and a configuration file.

```
Documents\
   bin\              ← folder in system path
      myprog\          main.py      ← shortcut to this file
         module1.py       ← module
         module2.py       ← module
         myprog.ini       ← configuration file
      myprog.bat       ← for command line execution
```

Multiple programs from the same developer with shared modules:

The programs `myProgA` and `myProgB` are executed from the command line. They are written by NepsWeb, and share `moduleA.py` and `moduleB.py`.

```
Documents\
   bin\              ← folder in system path
      myProgA.bat      ← for command line
      myProgB.bat      ← for command line
      nepsweb\       ← holds all the files from NepsWeb
         readme.txt       ← documentation
         module1.py       ← shared module
         module2.py       ← shared module
         myProgA.py
         myProgB.py
```

Shared modules in a sub-folder:

The programs `myProgC` and `myProgD` have GUIs. They are written and distributed by NepsWeb and use modules in the `network` folder.

[4]The `bin` folder name comes from the Unix operating system. Unix programs are often written in a language like C, which is complied and linked to create an executable machine code or *binary* file. So the directory where these executable files are stored is called `bin`. However, a `bin` directory is typically used to store anything that can be executed, including scripts and Python code (!).

```
bin\
   nepsweb\          ← holds all the files from NepsWeb
      myProgC.pyw          ← shortcut to this file
      myProgD.pyw          ← shortcut to this file
      network\       ← holds the network module files from NepsWeb
         lan1.py           ← shared module
         wan2.py           ← shared module
```

Here, the program `myProgD` would load the module `lan1` with the statement `import network.lan1` for example.

5 Python Language Installation

The methods described in §2 and §3 require that a suitable version of the Python language is present on the target computer. This requirement should be given in the installation instructions. It can just be stated as something like: "Python 3.2 or later must be installed." Additionally, instructions on how to obtain the correct version can be given[5], or a copy of a `python-x.x.x.msi` installation file can be included in the distribution.

5.1 Multiple Pythons

Sometimes we need to have more that one version of Python available, most often because we have legacy Python 2 applications. If this is the case, then we might have to explicitly state which version of Python we want. So in a batch file we would typically invoke Python 2.7 with

```
C:\Python27\python demo\demo.py
```

File shortcuts can be adapted by changing the Target box in their properties window. Simply put `C:\Python27\python`, separated by a space, in front of the file name's absolute path.

6 Summary

Python programs can be easily installed without special tools or scripts. A typical procedure is as follows:

1. Download and install Python language software if this has not already been done. (§5)
2. Create a folder called `bin` in the standard `Documents` folder if this has not already been done. Put the `bin` folder in the system path if command line invocation is wanted. (§3.2)
3. Copy the Python program files to a folder in `bin`, and put any `bat` files in `bin`. GUI main files should be of type `pyw` and console applications of type `py`.
4. Create a shorttcut to the main program file, and put it in the Start menu. (§2.1)

[5]Python 2 and 3 can be downloaded from `www.python.org/downloads`,

7 Sample Installation Instructions

We are not using an installation tool so the process relies on written instructions. The following is an example of some installation instructions for a simple Python GUI program with a couple of modules and a configuration file.

```
word2tex                                                Windows 7 Installation
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Word2tex is a program for converting Microsoft Word documents to Tex. User
documentation can be found in the word2tex\docs folder after installation
is complete.

You will need word2tex.zip for the installation. A basic knowledge of
how to use the Windows desktop is assumed.

This installation requires that Python 3.2 or later is installed. The
latest version of Python can be obtained from:

         https://www.python.org/downloads/

The program and associated files will be put in a folder called
Documents\bin, but this an be changed if desired.

1. Create a 'bin' folder in your Documents folder if it does not already
   exist.

2. Put a copy of wor2tex.zip into bin the and extract it. A word2tex
   folder will be created. Delete the zip file.

3. Put the program in the Start Menu by:

   (a) Left click Start, right click All Programs, then left click Open
       or Open all Users.

   (b) A Start Menu window will appear. Open the Programs folder.

   (c) Right click in the folder window, select New and then left click
       Shortcut.

        i. Browse to the bin folder of step 1. Select the word2tex.pyw
           file, then click Next.
       ii. Type in a word2tex as a name for the shortcut, then click
           Finish. A new shortcut will be created.

   (d) Close the folder window.

4. Start the program by going to Start/All Programs and clicking on the
   word2tex icon.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

[pydistro.tex 3.0]