eGenix PyRun

One file Python
Runtime

Version 1.2

Contents

1. Introduction

eGenix PyRun™ is our one-file, portable, no-install version of Python, making the distribution of a Python interpreter to run based scripts and applications to Unix based systems as simple as copying a single file.

1.1 Features

- **Small footprint:** only about 11 MB in size. Can be compressed down to just 3-4 MB using e.g. upx.
- **Full Python Support:** PyRun works with Python 2.5, 2.6 and 2.7.
- **Cross Platform Support:** PyRun runs on the following platforms: Linux, FreeBSD and Mac OS X.
- **Full 64-bit Support:** PyRun can be built on both 32-bit and 64-bit builds on all supported platforms.
- **Highly Portable Codebase:** in addition to the already supported platforms for PyRun, we provide custom porting services for more exotic platforms.
- **Easy Installation:** Simply drop the executable into a directory and start using it. No installers, no packagers and only a minimal set of dependencies needs to be provided.
- **Fully Relocatable:** eGenix PyRun uses relative search paths, so you can easily move the installation around.
- **Compatible with setuptools/distribute, easy_install, pip:** Great care was taken to make sure that PyRun can be used with setuptools et al.[1]
- **Perfect virtualenv replacement:** PyRun provides an even better level of isolation from the system installed Python version. Instead of using symlinks and other tricks to create a virtual Python

[1] Please note that setuptools/distribute and pip are *not* designed to be relocatable.

environments, PyRun comes with a complete Python runtime and thus doesn't need to play any tricks.

- **Open Source:** PyRun is licensed under the eGenix Public License. This makes it possible to integrate PyRun into other open-source or commercial products

- **Commerical Support:** eGenix provides commercial support for PyRun, in case you need custom builds, help with the integration or need problem solving support.

1.2 System Requirements

eGenix PyRun integrates a few standard Python extension modules, which rely on system provided third-party libraries to be available on the build system as development versions, and on the target system as binary versions.

1.2.1 Source Installations

eGenix PyRun needs these third party tools to be available on the target machines:

- **OpenSSL** 0.9.8/1.0.0 or later

- **zlib** 1.2 or later

- **SQLite** 3.4 or later

- **bzip2** 1.0 or later

Future versions of PyRun will add more flexibility to the build process to make the requirements more customizable.

If you want to compile PyRun yourself, you will also need the development packages of the above tools, a C compiler and a GNU make compatible make installed.

Please note that the binary versions of these libraries also need to be available on the PyRun installation target systems. Most modern Unix systems have the above libraries installed per default.

1.2.2 Binary Installations

The readily built version of eGenix PyRun we make available on our website were built using these library versions:

- OpenSSL 1.0.0
- zlib 1.2
- SQLite 3.4
- bzip2 1.0

1.3 Installation

Installation of eGenix PyRun can be done in two ways:

- from sources by compiling your own version, or
- downloading and installing a precompiled binary.

The following sections explain both options.

1.3.1 Windows Installer

eGenix PyRun currently **does not support Windows platforms.**

We will look into making eGenix PyRun compatible with Windows in one of the upcoming releases.

1.3.2 Quick Install using install-pyrun or: How to replace virtualenv with eGenix PyRun

To simplify the installation of eGenix PyRun and make it as easy to use as *virtualenv*, we've created a shell script called **install-pyrun** that you can *download from our server*. It is also distributed in the source archives of PyRun in the PyRun/ folder.

After download of the script, you have to make it executable and place it on your path.

With the script, you can automate the whole installation process easily. In order to install the set of eGenix PyRun (together with the Python include files and some extra C extensions from the stdlib that are not compiled into PyRun), distribute and pip to a new directory targetdir, just run the following command:

```
install-pyrun targetdir
```

This will then download a suitable eGenix PyRun binary distribution for your platform, install distribute and pip:

```
$ install-pyrun targetdir
Installing eGenix PyRun ...
Installing distribute ...
Installing pip ...

eGenix PyRun was installed in targetdir

To run eGenix PyRun, use targetdir/bin/pyrun

$ cd targetdir
$ bin/pyrun -c 'print "Hello World!"'
Hello World!
$ bin/pyrun
pyrun 2.7.4 (release 1.2.0, default, Apr 25 2013, 20:21:13)
[GCC 4.5.0 20100604 [gcc-4_5-branch revision 160292]]
Thank you for using eGenix PyRun. Type "help" or "license" for
details.

>>>
```

The script comes with a set of options to customize the version, platform, Unicode variant and also allows disabling installation of distribute and pip as well as permit installation from a local PyRun archive.

Simply run the script with option -h to see all options:

```
$ install-pyrun -h

Install eGenix PyRun in a given target directory.

USAGE:
       install-pyrun [options] targetdir

OPTIONS:
       -m: install eGenix PyRun only (no distribute and pip)
       -l: log installation to targetdir/pyrun-installation.log
       -q: quiet installation

       --python=2.7
           install PyRun for Python version 2.7 (default), 2.6, 2.5
       --python-unicode=ucs2
           install PyRun for Python Unicode version ucs2 (default)
           or ucs4
       --pyrun=1.0.0
           install PyRun version 1.0.0 (default)
       --platform=linux-i686
           install PyRun for the given platform; this is usually
```

```
        auto-detected
    --platform-list
        list available platform strings

    --pyrun-distribution=pyrun.tgz
        use the given PyRun distribution file; this overrides
        all other distribution selection parameters
    --distribute-distribution=distribute.tgz
        use the given distribute distribution file instead of
        downloading it
    --pip-distribution=pip.tgz
        use the given pip distribution file instead of
        downloading it

    --help
        show this text
    --version
        show the script version
    --copyright
        show copyright
    --debug
        enable debug output

Without options, the script installs eGenix PyRun, distribute and
pip in targetdir.
```

If you'd rather install eGenix PyRun manually, you can do so by downloading the prebuilt binaries yourself. Please see the next section for details.

1.3.3 Prebuilt Binary Distribution

Simply download a suitable binary distribution for the version of Python and platform you need and extract it to a base directory where you'd like PyRun to live. Then use the eGenix PyRun executable like you'd use a regular Python interpreter, e.g.

```
./bin/pyrun2.7 myscript.py
```

You can also put pyrun into the shebang of the script, e.g.

```
#!/usr/bin/env pyrun2.7
# Hello World Demo
print "Hello World!"
```

Please note that the binary distributions contain more than just the PyRun executable. They also come with a few extra standard library extensions which are normally not part of PyRun and the include files needed to compile Python extensions for use with PyRun.

If you are just looking for the plain single-file PyRun executable, only extract the file bin/pyrunX.X (with X.X being the underlying Python version, e.g. 2.7) from the binary distributions.

1.3.4 eGenix PyRun Directory Structure

If you want to install other libraries or packages for use in eGenix PyRun, you need to pay a little more attention to where you copy the executable. eGenix PyRun assumes the following directory layout **relative to the executable** (with X.X being the underlying Python version, e.g. 2.7):

- bin/pyrunX.X
- lib/pythonX.X
- include/pythonX.X

The lib/pythonX.X directory is used as location of the Python libaries and automatically put on `sys.path` in case it is available. Optional packages installed through distutils or setuptools will go into the corresponding lib/pythonX.X/site-packages/ directory.

The lib/pythonX.X directory may also contain Python standard library extension modules in the lib-dynload/ sub-directory which are not integrated into the eGenix PyRun executable. The prebuilt binary distributions come with a set of such extensions.

The include/pythonX.X directory is only needed in case you want to compile Python C extensions. It is available as part of the prebuilt binary distributions we make available.

Please note that while **eGenix PyRun itself is fully relocatable** after installation due to the relative search path, the tools **setuptools/distribute and pip are not**. They hard-code the paths into their scripts, so you can not relocate a PyRun installation, after these tools have been installed.

1.3.5 Building you own eGenix PyRun binary

In order to build your own version, simply download the above source archive, untar/unzip it to a temporary directory and follow these steps:

```
cd egenix-pyrun-*
cd PyRun
make
make install
```

This will download the Python source distribution and start a build of eGenix PyRun. The result will be installed to the directory /usr/local/ using the directory layout as described above.

If you'd like to build a binary distribution archive, use the following commands instead:

```
cd egenix-pyrun-*
cd PyRun
make distribution
```

You can then pick up the distribution archive from the dist/ directory.

Testing

To run some simple tests, please use the test-distribution target:

```
make test-distribution
```

This will install the distribution you just built, install it locally in a test directory and run a few tests, including pip installations of sizeable packages such as Cython and Django.

Note that only the installation itself is tested, not the packages themselves.

Customization Options

If you would rather install to a different directory, you can add the make parameter `PREFIX=/path/to/pyrun/`, This will cause make to install eGenix PyRun in /path/to/pyrun/.

If you want to build against a specific Python version, you can specify the version using the *make* parameter `PYTHONFULLVERSION=2.7.4`. Please have a look at the `PyRun/Makefile` for more ways of customizing the setup.

For future versions of eGenix PyRun, we plan to make the setup customizable via a top-level setup.py file so you can use Python to trigger the build, customize the included standard lib extension modules and installation.

2. eGenix PyRun Internals

eGenix PyRun uses the standard Python freeze tool, which you can find the Tools/freeze/ directory of the Python source code distribution to combine the Python interpreter with a large subset of the Python standard library into a single-file Python runtime.

2.1 PyRun Building Parts

In order to use freeze.py for creating PyRun, we had to implement these steps:

- we created a freeze.py template (PyRun/Runtime/pyrun_template.py) which provides a mostly compatible command line interface to the standard Python interpreter and references the standard library modules that we wanted to include in PyRun,
- we extract the Python configuration information from the Python Makefile and configure files and put this information into a static configuration template (PyRun/Runtime/pyrun_config_template.py), which is then used by sysconfig.py to load the configuration,
- we created a script (PyRun/makepyrun.py) which creates all the necessary pyrun*.py files from the templates,
- we added patches to Python (PyRun/Runtime/Python-*.patch) and the Modules/Setup files (PyRun/Runtime/Setup.PyRun-*) to be able to statically link in extension modules that would normally be built as shared modules and to provide a pure-Python implementation of the Python command line interface.

2.1.1 The PyRun Makefile

The PyRun/Makefile extracts the Python source code, applies the patches and adds the Modules/Setup file.

It then creates the pyrunX.X.py freeze.py template, the pyrun_config.py module and runs PyRun/Runtime/freeze/freeze.py on the generated pyrunX.X.py file.

freeze.py then generates the frozen module versions and a Makefile in PyRun/Runtime/Makefile which can then be used to build the pyrunX.X executable.

The PyRun/Makefile also takes care of installing the executable together with the include files and optional shared modules built during the process; as well as packaging the builds into binary .tar.gz files, which can simply be extracted anywhere in the file system to "install" eGenix PyRun.

2.1.2 Adding Python modules/packages to eGenix PyRun

The easiest way to have modules or whole packages added to PyRun is to modify PyRun/Runtime/pyrun_extras.py and import them in that file.

freeze.py will then automatically find the modules and referenced packages, freeze and add them to PyRun.

Alternatively, you can also edit the PyRun/Runtime/makepyrun.py file and add the modules/packages in the configuration section near the top of that module.

In a future version of eGenix PyRun, we're going to simplify this process so that you can pass the modules to include as parameter to the build script.

2.1.3 Removing Python modules/packages from eGenix PyRun

If you want to further reduce the PyRun file size, you can remove additional modules/packages from the frozen binary by editing the PyRun/Runtime/makepyrun.py file and adding the modules/packages to the remove lists.

In some cases, this may not be enough to completely remove the modules/packages, e.g. if you still have other modules in PyRun which reference the removed modules/packages, freeze.py is going to re-add them in the module search process.

To overcome this limitation, you will have to additionally add the modules/packages to the PyRun/Makefile `EXCLUDES` variable.

In a future version of eGenix PyRun, we're going to simplify this process so that you can pass the modules to exclude as parameter to the build script.

2.2 Known Incompatibilities

eGenix PyRun provides a robust production runtime environment, but has to make some compromises due to the way it is built.

This section explains the known incompatibilities compared to a regular Python installation.

2.2.1 Standard library modules not linked into PyRun

We have deliberately excluded a number of standard library modules that are either to complicated to build, have license issues or are not needed often enough in our use cases to warrant including in eGenix PyRun:

- *dbm modules
- crypt
- readline
- parser
- tkinter
- _multiprocessing
- all test packages and sub-packages
- ctypes
- nis
- audio modules

The modules are still built (if the needed development files are found on the build system) and packaged in the eGenix PyRun distribution files, so you can use them, if you need to.

However, they are not statically linked into the PyRun executable, so when moving this file around, you have to make sure that the relative directory structure expected by PyRun (see 1.3.4 eGenix PyRun Directory Structure) makes it possible to find those shared modules.

2.2.2 Include files not included in PyRun executable

For obvious reasons, we cannot include the Python include header files in the PyRun executable, since the compiler/preprocessor will have to find them in order to use them.

We do include the include files in the distribution packages and install-pyrun will also install them, so it's possible to compile Python C extensions if you use one of those distribution forms.

Compiling C extensions is not possible without the include header files, so the single-file PyRun runtime executable is not enough to compile Python C extensions.

2.2.3 Some things that don't work

There are a couple of tricks which Python modules sometimes play, which don't work with frozen modules.

File access to resources in packages

Some standard library modules/packages come with non-Python resource files such as binary .exe stubs or data files. Since freeze.py will only find Python modules, these files are not included in the frozen PyRun executable and since the frozen modules/packages don't live in the file system, access to such resource files is not possible via the module/package path.

Examples of such modules/packages:

- idellib
- distutils' bdist_wininst (the rest of distutils works fine)
- pkgutil on frozen packages (it works on frozen modules in Python 2.7)
- several test modules/packages (not included anyway)

Running frozen packages with -m

The pkgutil modules only has limited support for frozen modules in Python 2.7 and no support for frozen packages in Python. As a result, running `pyrun2.7 -m timeit` works, but e.g. `pyrun2.7 -m lib2to3 --help` doesn't.

Python test suite issues

Running the Python test suite shows some strange issues which we have not yet tracked down:

- Some test modules hang when run with regrtest.py, e.g. test_docxmlrpc
- Test modules cause strange errors (mostly encoding errors) when run with regrtest.py. Running the test modules directly doesn't show these errors. Some preliminary investigation suggests that these issues could be caused by regrtest.py modifying the module `__path__` entries.

Not all Python command line options available

Because eGenix PyRun has to emulate the command line options in Python, it is difficult to emulate some Python command line option which take effect in the very early stages of the interpreter startup phase.

For some options like e.g. -d (debug level) and -O (optimization level), we have added helpers/patches to Python to make it possible adjusting them from Python.

Fortunately, most of the more exotic options are not used in production runtime environments for which eGenix PyRun is designed.

3. Examples of Use

Here's a short session installing setuptools, pip and our egenix-mx-base package in an eGenix PyRun installation.

First, we install a PyRun version for 32-bit Linux:

```
mkdir tmp
cd tmp
wget http://downloads.egenix.com/python/egenix-pyrun-1.2.0-
py2.7_ucs2.linux-i686.tgz
tar xvfz egenix-pyrun-1.2.0-py2.7_ucs2.linux-i686.tgz
```

Now we have a ready to use Python runtime in tmp/, using the default eGenix PyRun directory layout, including some optional extra shared libraries and the include files needed for compiling Python C extensions.

Now, we install setuptools into this runtime:

```
wget
http://pypi.python.org/packages/source/s/setuptools/setuptools-
0.6c11.tar.gz
cd setuptools-0.6c11
../bin/pyrun2.7 setup.py install
cd ..
rm -rf setuptools-0.6c11
```

Installing pip is just as easy:

```
wget http://pypi.python.org/packages/source/p/pip/pip-1.1.tar.gz
cd pip-1.1
../bin/pyrun2.7 setup.py install
cd ..
rm -rf pip-1.1
```

You can then install other Python packages using the usual installation methods:

```
bin/pip install egenix-mx-base
```

and the packages are available to eGenix PyRun just as they would in a regular Python installation:

```
$ bin/pyrun2.7
pyrun 2.7.4 (default, Jun 24 2012, 20:37:17)
[GCC 4.2.1 (SUSE Linux)]
Thank you for using eGenix PyRun. Type "help" or "license" for
details.

>>> import mx.DateTime
>>> mx.DateTime.now()
<mx.DateTime.DateTime object for '2012-06-28 19:26:30.62' at
7f1845a6a300>
>>>
```

3.1 eGenix PyRun Command Line Options

These are eGenix PyRun's command line options. They are shown if you start the PyRun executable with option `-h`:

```
$ bin/pyrun2.7 -h
Usage: pyrun [pyrunoptions] <script> [parameters]

Version: 2.7.4 (release 1.2.0, default, Apr 25 2013, 20:21:13)
[GCC 4.5.0 20100604 [gcc-4_5-branch revision 160292]]

Available pyrun options:

-h:   show this help text
-v:   run in verbose mode
-i:   enable interactive mode
-m:   import and run a module <script> available on PYTHONPATH
-c:   compile and run <script> directly as Python code
-b:   run the given <script> file as bytecode
-E:   ignore environment variables (only PYTHONPATH)
-S:   skip running site.main() and disable support for .pth files
-O:   run in optimized mode (-OO also removes doc-strings)
-u:   open stdout/stderr in unbuffered mode
-d:   enable debug mode
-V:   print the pyrun version and exit

Without options, the given <script> file is loaded and run.
Parameters are passed to the script via sys.argv as normal.
```

The exact output is subject to changes between eGenix PyRun versions.

The meaning of most options is similar to the Python interpreter command line options of the same name.

Note that not all options are available, since it is difficult to emulate them in pure Python. Even some of the above options were only possible using patches to Python, e.g. the -d and -O options.

3.2 Debugging eGenix PyRun Installations

If you need to debug PyRun installations, you can use -dd to have PyRun display paths and setting variables at startup:

```
$ bin/pyrun2.7 -dd

### PyRun Debug Information

# Name and version
pyrun_name = 'pyrun'
pyrun_version = '2.7.4'
pyrun_libversion = '2.7'
```

```
pyrun_release = '1.2.0'
pyrun_build = '(release 1.2.0, default, Apr 26 2013, 18:10:38)
\n[GCC 4.5.0 20100604 [gcc-4_5-branch revision 160292]]'

# Files and directories
pyrun_executable = '/usr/local/bin/pyrun2.7'
pyrun_dir = '/usr/local/bin'
pyrun_binary = 'pyrun2.7'
pyrun_prefix = '/usr/local'
pyrun_bindir = 'bin'

# Options
pyrun_verbose = 0
pyrun_debug = 2
pyrun_as_module = False
pyrun_as_string = False
pyrun_bytecode = False
pyrun_ignore_environment = False
pyrun_ignore_pth_files = False
pyrun_interactive = False
pyrun_unbuffered = False
pyrun_optimized = 0

pyrun: Setting up sys.path
pyrun:   sys.path before adjusting it (compile time version):
pyrun:     /usr/local/bin
pyrun:     /usr/local/lib/python2.7
pyrun:   sys.path after adjusting it (before cleanup):
pyrun:     /home/lemburg
pyrun:     /home/lemburg/bin
pyrun:     /home/lemburg/lib
pyrun:     /usr/local/lib/python2.7
pyrun:     /usr/local/lib/python2.7/lib-dynload
pyrun:     /usr/local/lib/python2.7/site-packages
pyrun:   sys.path final version:
pyrun:     /home/lemburg
pyrun:     /home/lemburg/bin
pyrun:     /home/lemburg/lib
pyrun:     /usr/local/lib/python2.7
pyrun:     /usr/local/lib/python2.7/lib-dynload
pyrun:     /usr/local/lib/python2.7/site-packages
pyrun: Importing site.py
pyrun:   sys.path before importing site:
pyrun:     /home/lemburg
pyrun:     /home/lemburg/bin
pyrun:     /home/lemburg/lib
pyrun:     /usr/local/lib/python2.7
pyrun:     /usr/local/lib/python2.7/lib-dynload
pyrun:     /usr/local/lib/python2.7/site-packages
pyrun:   sys.path after importing site:
pyrun:     /home/lemburg
pyrun:     /home/lemburg/bin
pyrun:     /home/lemburg/lib
pyrun:     /usr/local/lib/python2.7
pyrun:     /usr/local/lib/python2.7/lib-dynload
pyrun:     /usr/local/lib/python2.7/site-packages
pyrun 2.7.4 (release 1.2.0, default, Apr 26 2013, 18:10:38)
[GCC 4.5.0 20100604 [gcc-4_5-branch revision 160292]]
Thank you for using eGenix PyRun. Type "help" or "license" for
details.

>>>
```

4. Support

eGenix.com is providing commercial support for this package. If you are interested in receiving information about this service please see the *eGenix.com Support Conditions*.

5. Copyright & License

This software is covered by the ***eGenix.com Public License Agreement***, which is included in the following section. The text of the license is also included as file "LICENSE" in the package's main directory.

Since eGenix PyRun also pulls in Python, the respective *Python license* also applies to the resulting pyrun binary. The Python license is included as file "LICENSE.Python" in the package's main directory as well as the *eGenix Third-Party License* document.

In simple words, you are free to use the software without paying fees or royalties as long as you give proper attribution and keep the license documents together with the software. Please see the license document for details and consult a lawyer if you have legal questions.

By downloading, copying, installing or otherwise using the software, you agree to be bound by the terms and conditions of the following *eGenix.com Public License Agreement*.

DISTRIBUTING THE SOFTWARE, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE EXCLUSION OR LIMITATION MAY NOT APPLY TO LICENSEE.

5. Termination

This License Agreement will automatically terminate upon a material breach of its terms and conditions.

6. Third Party Rights

Any software or documentation in source or binary form provided along with the Software that is associated with a separate license agreement is licensed to Licensee under the terms of that license agreement. This License Agreement does not apply to those portions of the Software. Copies of the third party licenses are included in the Software Distribution.

7. General

Nothing in this License Agreement affects any statutory rights of consumers that cannot be waived or limited by contract.

Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between eGenix.com and Licensee.

If any provision of this License Agreement shall be unlawful, void, or for any reason unenforceable, such provision shall be modified to the extent necessary to render it enforceable without losing its intent, or, if no such modification is possible, be severed from this License Agreement and shall not affect the validity and enforceability of the remaining provisions of this License Agreement.

This License Agreement shall be governed by and interpreted in all respects by the law of Germany, excluding conflict of law provisions. It shall not be governed by the United Nations Convention on Contracts for International Sale of Goods.

This License Agreement does not grant permission to use eGenix.com trademarks or trade names in a trademark sense to endorse or promote products or services of Licensee, or any third party.

The controlling language of this License Agreement is English. If Licensee has received a translation into another language, it has been provided for Licensee's convenience only.

8. Agreement

By downloading, copying, installing or otherwise using the Software, Licensee agrees to be bound by the terms and conditions of this License Agreement.

For question regarding this License Agreement, please write to:

eGenix.com Software, Skills and Services GmbH

Pastor-Loeh-Str. 48

D-40764 Langenfeld

Germany