eGenix.com

eGenix PyRun

One file Python
Runtime

Version 2.2

Printed in Germany.

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 for Python 2.x and only around 13 MB for Python 3.x. Can be further compressed down to just 3-4 MB using e.g. upx.
- **Python 2.6, 2.7, 3.4 and 3.5 Support:** PyRun works with all popular Python versions. We chose the single source approach for achieving this, but had to drop Python 2.5 support as a result.
- **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, 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

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

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

- **install-pyrun script:** Using the bash script *install-pyrun*, we make the whole installation process even easier. It automatically downloads and installs the right version for your platform and also take care of adding setuptools and pip to the PyRun installation.

- **Install a complete Python runtime including dependencies in one go:** using our *install-pyrun* installation bash script, you can install PyRun as well as all dependencies using a single `install-pyrun -r requirements.txt` run.

- **Open Source:** PyRun is licensed under the eGenix Public License. This Python-style 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** 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 1.0.2 or later**
- **zlib 1.2 or later**
- **SQLite 3.4 or later**
- **bzip2 1.0 or later**
- **libzma 5.0 or later** (Python 3 only)

1.3 Installation

Installation of eGenix PyRun can be done in three ways:

- using the bash script *install-pyrun* for automatic installation
- downloading and installing a precompiled binary from *www.egenix.com*, or
- from sources by compiling your own version.

The following sections explain these 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

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:

```
cd ~/bin
curl -O https://downloads.egenix.com/python/install-pyrun
chmod 755 install-pyrun
```

The script is a simple bash script, so you can inspect it with any text editor.

Using install-pyrun

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), setuptools 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 setuptools and pip:

```
$ install-pyrun targetdir
Installing eGenix PyRun ...
Installing local setuptools 15.2 ...
Installing local pip 1.5.6 ...

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
eGenix PyRun 2.7.8 (release 2.0.1, default, Aug 27 2014, 00:58:48)
[GCC 4.5.1 20101208 [gcc-4_5-branch revision 167585]]
```

```
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 setuptools 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 or --minimal
           install eGenix PyRun only (no setuptools and pip)
       -l of --log
           log installation to targetdir/pyrun-installation.log
       -q or --quiet
           quiet installation
       -r or --requirements
           have pip install the given requirements (only works
           if pip is installed as part of the pyrun installation)

       --python=2.7
           install PyRun for Python version 2.6, 2.7 (default),
           3.4, 3.5
       --python-unicode=ucs2
           install PyRun for Python Unicode version
           ucs2 (default for Python 2) or ucs4 (default for Python 3)
       --pyrun=2.2.0
           install PyRun version 2.2.0 (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
       --pyrun-executable=pyrun
           symlink to and use an alternative name for the PyRun
           executable

       --setuptools-distribution=setuptools.tgz
           use the given setuptools distribution file instead of
           downloading it from PyPI
       --setuptools-version=15.2
           install the setuptools 15.2 (default); use
           --setuptools-version=latest to automatically find the
           latest version on PyPI
       --distribute-distribution=distribute.tgz
           alias for --setuptools-distribution

       --pip-distribution=pip.tgz
           use the given pip distribution file instead of
           downloading it from PyPI
```

```
        --pip-version=1.5.6
            install the pip 1.5.6 (default); use --pip-version=latest
            to automatically find the latest version on PyPI

        --help
            show this text
        --version
            show the script version
        --copyright
            show copyright
        --debug
            enable debug output
        --disable-certificate-checks
            disable certificate checks when downloading packages;
            this should normally not be needed

    Without options, the script installs eGenix PyRun, setuptools and pip
    in targetdir. If no local versions of setuptools or pip are found, the
    tools are downloaded from pypi.python.org.
```

The exact versions of pyrun, pip and setuptools listed in the above output may be different in case you downloaded a more recent version of the script.

install-pyrun is secure

install-pyrun will use `curl` as default downloader, and fallback to `wget`, if curl is not available. The script defaults to operating the downloaders in secure mode, i.e. all links are HTTPS links and all certificates are verified before proceeding with the download.

Both the pypi.python.org and our downloads.egenix.com servers use HTTPS for enhanced security.

Setup a complete Python environment in one go using -r requirements.txt

install-pyrun also supports the useful pip install -r requirements.txt instruction, so that you can set up a complete Python runtime environment in one go.

Simply create a *requirements.txt file* using the *same format* as supported by pip and then pass the -r option to install-pyrun.

requirements.txt:

```
Django==1.8.0
pytz
egenix-mx-base == 3.2.8
```

Then run `install-pyrun`:

```
$ ./install-pyrun -r requirements.txt projectdir
```

Manual PyRun installation

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 (Python 2) or include/pythonX.Xm (Python 3)

The lib/pythonX.X directory is used as location of the Python libraries 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. For Python 3, the include directory has the ABI flags appended. For Python 3.4, this is include/python3.4m for most Python installations, for Python 3.5 include/python3.5m.

> Please note that while **eGenix PyRun itself is fully relocatable** after installation due to the relative search path, the tools **setuptools 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 Shipping custom external libraries with eGenix PyRun

To facilitate shipping custom external libraries with PyRun, newer versions of PyRun (2.1 and later) add an *rpath setting* to the binary which results in the dynamic linker on compatible systems to first look in these directories for shared libraries such as OpenSSL, sqlite, zlib or bzip2 libraries.

The rpath setting defines a dynamic library search path which is used if no `LD_LIBRARY_PATH` OS environment variable is set.

Default rpath

Default rpath in eGenix PyRun 2.1 and later:

- $ORIGIN
- $ORIGIN/../lib
- $ORIGIN/../lib/pythonX.X/site-packages/OpenSSL

The $ORIGIN entry is a placeholder which is replaced with the directory of the pyrun binary by the dynamic linker at load time.

The above path first looks in the same directory as the `pyrun` binary, then in the ../lib/ directory and then in the OpenSSL site-packages directory.

The latter is added to simplify using our *egenix-pyopenssl package* with eGenix PyRun. The package ships up-to-date OpenSSL libraries inside the package directory.

Override rpath at runtime

You can override this rpath by setting the OS environment variable `LD_LIBRARY_PATH`. A linker search path given in `LD_LIBRARY_PATH` will then be prepended to the search path.

Change or disable rpath setting

If you want to compile your own eGenix PyRun executable you canyou're your own default rpath by adjusting the `PYRUNRPATH` variable in the PyRun/Makefile.

1.3.6 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 NumPy, Cython and Django.

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

Customization Options

Changing installation directory

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/.

Example:

```
# Install eGenix PyRun to local dir:
make PREFIX=local-test/
```

Building a different Python version

If you want to build against a specific Python version, you can specify the version using the *make* parameter `PYTHONFULLVERSION=2.7.5`. Please have a look at the top of the `PyRun/Makefile` for supported Python versions.

Example:

```
# Build eGenix PyRun for Python 3.5.1:
make PYTHONFULLVERSION=3.5.1
```

Building against custom OpenSSL library installations

eGenix PyRun will per default look in these directories for a suitable OpenSSL installation (both include files and libraries):

- /usr/local/ssl - for Linux and many other Unix platforms
- /usr/contrib. - for HP-UX
- /usr/sfw - for Solaris/OpenIndiana
- /usr - as fallback location

It is also possible to set the OpenSSL installation directory by passing `SSL=/path/to/openssl` to the make command.

Example:

```
# Build eGenix PyRun for local OpenSSL install:
make SSL=/home/lemburg/my-openssl
```

Passing in special compile / link options

The eGenix PyRun `PyRun/Makefile` supports passing in parameters for CFLAGS and LDFLAGS, which are then used for building the Python interpreter and the frozen pyrun binary.

Example:

```
# Build eGenix PyRun with
make CFLAGS="-I /home/lemburg/my-include-files"
```

More customization options

Please have a look at the `PyRun/Makefile` for more customization options.

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.

We have applied a few customizations to the freeze tool and ship separate versions for Python 2 and Python 3 with the source distribution of eGenix PyRun.

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 extract the Python grammar files the Python stdlib, so that the lib2to3 package can work without having the files loaded from the file system and put this information into a static configuration template (PyRun/Runtime/pyrun_grammar_template.py), which is then used by lib2to3 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 Differences compared to Python

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 enhancements and known incompatibilities compared to a regular Python installation.

2.2.1 Changes compared to standard Python

This is a list of changes we have applied compared to standard Python. Unlike the compatibility fixes listed in the next sections, these are features that are not necessary to get eGenix PyRun to work, but ones that we think are slightly more useful than the standard Python choices.

Default PYTHONIOENCODING in Python 3

In Python 3.4+, the Python I/O encoding is determined by looking at the locale of the process. Depending on the used system, this can be UTF-8, ASCII or some other encoding.

Since we think it's better to either correctly set the encoding via an environment variable such as PYTHONIOENCODING and not guess, we chose to define a default for the PYTHONIOENCODING which is used, in case this environment variable is not set.

The default for PYTHONIOENCODING in eGenix PyRun 3.4+ is "utf-8:surrogateescape". This allows pyrun to work in most situations without failing prints or weird encoding exceptions.

If you do have a different setup or would like exceptions for non-UTF-8 data to be raised, you can set the PYTHONIOENCODING environment variable to override this default.

It is even possible to have eGenix PyRun revert back to the locale scanning behavior by setting the PYTHONIOENCODING variable to "" (empty string).

User site configurations disabled

Since PyRun is normally used in isolated environments in which per-user installations are not really wanted, we have disabled the flag `ENABLE_USER_SITE` in site.py of the standard library to disable setup of the user-site configurations.

This means that PyRun will not automatically search these user site-packages directories and also not install into them. It also means that sys.path search run faster and startup time is a little better as a result.

Simulating user installations

Note that PyRun's site-packages directory is always set up relative to the location of the PyRun binary, so using user site configurations should not be needed.

If you want to simulate a user site installation with PyRun, you can create a local installation of PyRun based on a system-wide one by either copying the binary or simply creating a symbolic link in a user's subdirectory.

Disabled check for Python build directory installations

PyRun will normally run outside any build directory installations. We therefore save the startup costs for the special check to detect PyRun running in the Python build directory.

Special lib/site-python directory not included in default sys.path

The site.py setup module checks for the availability of a lib/site-python directory and adds this to `sys.path` if found. Since this directory is hardly ever used and very uncommon for the isolated run-time environments for which PyRun is intended, we have disabled this addition.

2.2.2 Additional features compared to Python

Disabling HTTPS certificate verification

Python 2.7.9 and 3.4+ introduced many changes to the SSL support of Python. The Python 3.4 ssl module was backported to Python 2.7 in 2.7.9 and the default HTTPS certificate verification checks enabled per default (see *PEP 476*)

Even though this results in making Python 2.7 more secure, it also means that that Python 2.7 now needs access to a reliable and up-to-date trust store which has the trusted CA root certificates. HTTPS requests to websites or servers which don't provide certificates listed in those CA root certificates will be rejected, unless the application code explicitly permits such connections.

This will in many cases result in Python not accepting CA Cert certificates or self-signed certificates anymore, which are often used in test setups or by embedded systems.

eGenix PyRun provides a way to disable these checks using an environment variable PYRUN_HTTPSVERIFY:

`PYRUN_HTTPSVERIFY`

When set to 0, PyRun will use the unverified SSL context for HTTPS connections per default, just like Python 2.7.8 and earlier did.

Setting this variable to 1 will force the default to also verify the HTTPS certificates.

Note: Future PyRun versions may adopt a possible new Python environment variable to do the same. See *http://bugs.python.org/issue23857*

2.2.3 Compatibility fixes applied to Python

In order to get some things to work in PyRun, we had to apply a few compatibility fixes to the Python distributions:

Frozen modules always have a __file__ attribute

In standard Python 2, the attribute is always set to "<frozen>". In standard Python 3, the attribute does not exist at all.

This poses many problems to software relying on this attribute for reporting and debugging purposes (e.g. to show the location of an error) or produce coverage reports.

Since it's impossible to fix all this third party software, we chose to alwas add the attribute to frozen modules. The attribute is set to "<pyrun>/package/module.py" for all modules frozen into pyrun. This allows most uses of the attribute to work without problems.

Some modules also use this attribute to locate external resources relative to the module location. Since the attribute does not point to an existing path, this will fail, so special work-arounds have to put in place for such software.

On the positive side, several such modules do take the situation into account that the path does not exist and fall back to other solutions.

Python system configuration included in pyrun_config.py

Since the build time information is normally read from the Python installation files, we had to find a way to make this available without going to the file system.

The trick was to use a generated module pyrun_config.py which has the Python config information and can also patch it at runtime to accommodate for the relocation feature of pyrun.

Note: Python 3 and later versions of Python 2.7 adopted this trick for sysconfig as well.

Integrated lib2to3 grammar files

The lib2to3 package comes with its own pgen2 parser. This parser uses a special set of grammar files for Python's grammar which are embedded into the package.

Since we cannot ship those files in the file system, we chose to stored the grammar pickles in a pyrun_grammar.py helper module and patched the package to use these pickles instead.

This allows lib2to3 to work without having to read the grammar files and makes on-the-fly conversions during installation possible. An important example of a package which uses this technique for Python 3 support is setuptools.

Added list of available lib2to3 fixes to config

The lib2to3 package scans its own fixes/ directory for available fix modules. In a frozen package this doesn't work, so we preprocessed the list, put it into the pyrun_config.py module and patched lib2to3 to load it from there.

Bug fixes to Python stdlib modules

While porting eGenix PyRun to Python 3.4 we found a couple of bugs in stdlib modules which we patch in pyrun for the supported Python versions and also reported upstream. Please see the change log on the *eGenix product web page* for details.

Statically linked standard library modules

In order to have the standard library extensions linked into the pyrun binary instead of having them built as separate module, eGenix PyRun defines additional entries for them in the Modules/Setup file.

Several modules are documented there, but not all of them. We extracted the relevant information from the Python setup.py file and added it manually to the Modules/Setup file for each supported Python version.

Since the Modules/Setup process is not nearly as flexible as the setup.py process, configuration of the modules is sometimes very complicated. For this reason we have punted on a few packages such as the ctypes package and instead added them to the distributed binaries as separately installable dynamic shared modules.

Note that it is possible to customize these Module/Setup files further in case you would like to add other third party extensions to the pyrun binary.

2.2.4 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
- _decimal (in Python 3)

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.

2.2.5 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.

2.2.6 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)

For the lib2to3 package, which also needs external files, we have patched the package and included the necessary grammar files in the `pyrun` binary, since it is used by several Python packages during installation on Python 3.

Running frozen packages with -m

The `pkgutil` module which is needed to implement the `-m` option in Python only has limited support for frozen modules in Python 2.7 and 3.4. It has no support for frozen packages. 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.

> Please note that installing eGenix PyRun is much easier using the `install-pyrun` bash script we provide, which automates the following steps. See 1.3.2 Quick install using install-pyrun for instructions on how to obtain and use this script.

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

```
mkdir tmp
cd tmp
wget http://downloads.egenix.com/python/egenix-pyrun-2.0.0-
py2.7_ucs2.linux-i686.tgz
tar xvfz egenix-pyrun-2.0.0-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-
2.1.tar.gz
cd setuptools-2.1
../bin/pyrun2.7 setup.py install
cd ..
rm -rf setuptools-2.1
```

Installing pip is just as easy:

```
wget http://pypi.python.org/packages/source/p/pip/pip-1.4.1.tar.gz
cd pip-1.4.1
../bin/pyrun2.7 setup.py install
cd ..
rm -rf pip-1.4.1
```

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
eGenix PyRun 2.7.6 (release 2.0.0, default, Jun 13 2014, 20:12:35)
[GCC 4.5.0 20100604 [gcc-4_5-branch revision 160292]]
Thank you for using eGenix PyRun. Type "help" or "license" for
details.

>>> import mx.DateTime
```

```
>>> mx.DateTime.now()
<mx.DateTime.DateTime object for '2013-06-13 20:26:30.62' at
7f1845a6a300>
>>>
```

eGenix PyRun will just as well install other packages such as Django, Trac, Numpy, Cython, etc.

We've tried to make PyRun as compatible with existing packages as possible, so everything should mostly work out of the box for you.

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/ /pyrun3.4 -h
Usage: pyrun [pyrunoptions] <script> [parameters]

Version: 3.4.3 (release 2.1.0, default, May  8 2015, 21:43:45)
[GCC 4.5.1 20101208 [gcc-4_5-branch revision 167585]]

Available pyrun options:

-b:   run the given <script> file as bytecode
-c:   compile and run <script> directly as Python code
-d:   enable debug mode (-dd for level 2)
-h:   show this help text
-i:   enable interactive mode
-m:   import and run a module <script> available on PYTHONPATH
-s:   ignore user site; PyRun always ignores user site configs
-u:   open stdout/stderr in unbuffered mode
-v:   run in verbose mode (-vv for level 2)
-B:   don't write byte code files
-E:   ignore environment variables (only PYTHONPATH)
-O:   run in optimized mode (-OO also removes doc-strings)
-R:   not implemented; use PYTHONHASHSEED instead
-S:   skip running site.main() and disable support for .pth files
-V:   print the pyrun version and exit
-3:   not implemented; only for compatibility with Python

Most Python environment variables are supported.

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.

Available options

`-b <script>`

Run the given <script> file as bytecode.

`-c <script>`

Compile and run <script> directly as Python code.

`-d`

Enable debug mode. Use -dd for level 2. This generates more debug output to stderr when PyRun starts and also sets Python's debug flag accordingly.

`-h`

Show the help text.

`-i`

Enable interactive mode. PyRun will enter interactive mode after running the script when this option is used.

`-m <script>`

Import and run a module <script> available on PYTHONPATH. The semantics are the same as those of Python itself.

`-s`

Ignore user site configurations. Since PyRun always ignores user site configurations, this flag has no effect.

`-u`

Open stdout/stderr in unbuffered mode. Note that PyRun can only emulate this by reopening sys.stdin and sys.stdout in unbuffered mode.

`-v`

Run PyRun in verbose mode. Use -vv for level 2. The Python verbose flag is set accordingly.

`-B`

Tells PyRun to not write byte code files for compiled files (i.e. .pyc or .pyo files and creating __pycache__ directories for Python 3.4+).

`-E`

Ignore environment variables. PyRun can only apply this for the PYTHONPATH variable, since other variables will already been read during Python startup.

`-O`

Run in optimized mode. -OO also removes doc-strings.

`-R`

Randomize the string hash function. Since this cannot be implemented in PyRun, it stops with an error. Use PYTHONHASHSEED instead.

`-S`

Skip running `site.main()` and disable support for .pth files.

`-V`

Print the PyRun version and exit.

`-3`

Not implemented. This flag is ignored to stay compatible with Python.

3.2 Debugging eGenix PyRun Installations

eGenix PyRun supports the Python -v and -d command line switches to give you extra information about how processing is done. Both switches enable additional output from eGenix PyRun as well as Python itself, since the command line switches are passed to Python.

Please note however that these switches are enabled in Python by the eGenix PyRun command line emulation which is written in Python. As a result, they are not in effect in the early setup stages of PyRun startup and don't include the early startup time information available in Python itself.

3.2.1 More information about PyRun paths and settings

If you need to debug PyRun installations, you can use -dd to have PyRun display paths and setting variables at startup time. Here's an example output:

```
$ bin/pyrun2.7 -dd
### PyRun Debug Information
```

```
# Name and version
pyrun_name = 'pyrun'
pyrun_version = '2.7.6'
pyrun_libversion = '2.7'
pyrun_release = '2.0.0'
pyrun_build = '(release 2.0.0, default, Jun 13 2014, 21:47:11)
\n[GCC 4.5.1 20101208 [gcc-4_5-branch revision 167585]]'

# Files and directories
pyrun_executable = '/tmp/pyrun2/bin/pyrun2.7'
pyrun_dir = '/tmp/pyrun2/bin'
pyrun_binary = 'pyrun2.7'
pyrun_prefix = '/tmp/pyrun2'
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:     /tmp/pyrun2/bin
pyrun:   sys.path after adjusting it (before cleanup):
pyrun:     /tmp/pyrun2/lib/python2.7/site-packages/setuptools-2.1-
py2.7.egg
pyrun:     /tmp/pyrun2/lib/python2.7/site-packages/pip-1.4.1-
py2.7.egg
pyrun:     /tmp/pyrun2
pyrun:     /tmp/pyrun2/lib/python2.7
pyrun:     /tmp/pyrun2/lib/python2.7/lib-dynload
pyrun:     /tmp/pyrun2/lib/python2.7/site-packages
pyrun:   sys.path final version:
pyrun:     /tmp/pyrun2/lib/python2.7/site-packages/setuptools-2.1-
py2.7.egg
pyrun:     /tmp/pyrun2/lib/python2.7/site-packages/pip-1.4.1-
py2.7.egg
pyrun:     /tmp/pyrun2
pyrun:     /tmp/pyrun2/lib/python2.7
pyrun:     /tmp/pyrun2/lib/python2.7/lib-dynload
pyrun:     /tmp/pyrun2/lib/python2.7/site-packages
pyrun: Importing site.py
pyrun:   sys.path before importing site:
pyrun:     /tmp/pyrun2/lib/python2.7/site-packages/setuptools-2.1-
py2.7.egg
pyrun:     /tmp/pyrun2/lib/python2.7/site-packages/pip-1.4.1-
py2.7.egg
pyrun:     /tmp/pyrun2
pyrun:     /tmp/pyrun2/lib/python2.7
pyrun:     /tmp/pyrun2/lib/python2.7/lib-dynload
pyrun:     /tmp/pyrun2/lib/python2.7/site-packages
pyrun:   sys.path after importing site:
pyrun:     /tmp/pyrun2/lib/python2.7/site-packages/setuptools-2.1-
py2.7.egg
pyrun:     /tmp/pyrun2/lib/python2.7/site-packages/pip-1.4.1-
py2.7.egg
```

```
pyrun:     /tmp/pyrun2
pyrun:     /tmp/pyrun2/lib/python2.7
pyrun:     /tmp/pyrun2/lib/python2.7/lib-dynload
pyrun:     /tmp/pyrun2/lib/python2.7/site-packages
eGenix PyRun 2.7.6 (release 2.0.0, default, Jun 13 2014, 21:47:11)
[GCC 4.5.1 20101208 [gcc-4_5-branch revision 167585]]
Thank you for using eGenix PyRun. Type "help" or "license" for
details.

>>>
```

The -d and -dd switches will also enable the Python debug settings, so you can use this to get additional debug information from Python or extensions compiled with debug support.

3.2.2 Debugging imports and import searches

If you need to debug imports and get more information about how eGenix PyRun executes the scripts, the -v command line switch is useful. It tells PyRun to output additional useful information to stderr and also enable the Python verbose flag which results in Python writing import searches and GC cleanup details to stderr.

```
test-2.7-ucs2/bin> ./pyrun -v
import site # frozen
import traceback # frozen
import sysconfig # frozen
import re # frozen
import sre_compile # frozen
import _sre # builtin
import sre_parse # frozen
import sre_constants # frozen
import _locale # builtin
# zipimport: found 135 names in
/home/lemburg/egenix/projects/PyRun/test-2.7-
ucs2/lib/python2.7/site-packages/setuptools-2.1-py2.7.egg
import code # frozen
import codeop # frozen
import __future__ # frozen
dlopen("/home/lemburg/egenix/projects/PyRun/test-2.7-
ucs2/lib/python2.7/lib-dynload/readline.so", 2);
import readline # dynamically loaded from
/home/lemburg/egenix/projects/PyRun/test-2.7-
ucs2/lib/python2.7/lib-dynload/readline.so
eGenix PyRun 2.7.9 (release 2.1.0, default, Apr 28 2015, 12:55:41)
[GCC 4.5.1 20101208 [gcc-4_5-branch revision 167585]]
Thank you for using eGenix PyRun. Type "help" or "license" for
details.

>>>
```

If you additionally want to know where PyRun searches for imports, increase verbosity by using the -vv flag:

```
test-2.7-ucs2/bin> ./pyrun -vv
import site # frozen
```

```
import traceback # frozen
import sysconfig # frozen
import re # frozen
import sre_compile # frozen
import _sre # builtin
import sre_parse # frozen
import sre_constants # frozen
import _locale # builtin
# zipimport: found 135 names in
/home/lemburg/egenix/projects/PyRun/test-2.7-
ucs2/lib/python2.7/site-packages/setuptools-2.1-py2.7.egg
# trying /home/lemburg/egenix/projects/PyRun/test-2.7-
ucs2/lib/python2.7/site-packages/pip-1.4.1-
py2.7.egg/sitecustomize.so
# trying /home/lemburg/egenix/projects/PyRun/test-2.7-
ucs2/lib/python2.7/site-packages/pip-1.4.1-
py2.7.egg/sitecustomizemodule.so
# trying /home/lemburg/egenix/projects/PyRun/test-2.7-
ucs2/lib/python2.7/site-packages/pip-1.4.1-
py2.7.egg/sitecustomize.py
# trying /home/lemburg/egenix/projects/PyRun/test-2.7-
ucs2/lib/python2.7/site-packages/pip-1.4.1-
py2.7.egg/sitecustomize.pyc
…
import readline # dynamically loaded from
/home/lemburg/egenix/projects/PyRun/test-2.7-
ucs2/lib/python2.7/lib-dynload/readline.so
eGenix PyRun 2.7.9 (release 2.1.0, default, Apr 28 2015, 12:55:41)
[GCC 4.5.1 20101208 [gcc-4_5-branch revision 167585]]
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

EGENIX.COM PUBLIC LICENSE AGREEMENT

Version 1.1.0

This license agreement is based on the Python CNRI License Agreement, a widely accepted open-source license.

1. Introduction

This "License Agreement" is between eGenix.com Software, Skills and Services GmbH ("eGenix.com"), having an office at Pastor-Loeh-Str. 48, D-40764 Langenfeld, Germany, and the Individual or Organization ("Licensee") accessing and otherwise using this software in source or binary form and its associated documentation ("the Software").

2. License

Subject to the terms and conditions of this eGenix.com Public License Agreement, eGenix.com hereby grants Licensee a non-exclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use the Software alone or in any derivative version, provided, however, that the eGenix.com Public License Agreement is retained in the Software, or in any derivative version of the Software prepared by Licensee.

3. NO WARRANTY

eGenix.com is making the Software available to Licensee on an "AS IS" basis. SUBJECT TO ANY STATUTORY WARRANTIES WHICH CAN NOT BE EXCLUDED, EGENIX.COM MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, EGENIX.COM MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

4. LIMITATION OF LIABILITY

EGENIX.COM SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF THE SOFTWARE FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR OTHER PECUNIARY LOSS) AS A RESULT OF USING, MODIFYING OR

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