IST FP6-004779

PYPY

Researching a Highly Flexible and Modular Language Platform and Implementing it by Leveraging the Open Source Python Lanugage and Community

STREP

IST Priority 2

D04.1: First Partial Python Implementation on top of CPython

Due date of deliverable: May 2005

Actual Submission date: 23th December 2005

Start date of Project: 1st December 2005 Duration: 2 years

Lead Contractor of this WP: Strakt

Authors: Jacob Hallén (AB Strakt), Christian Tismer (tismerysoft)

Revision: final

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)

Dissemination Level: PU (Public)

Abstract

This document describes the initial public release of the PyPy interpreter, version 0.6, released on Friday, 20 May 2005.

Contents

1 Executive Summary

The initial release of the PyPy interpreter is built to run on top of the CPython interpreter. While able to correctly execute a large number of programs, it does so very slowly. This is expected, since it is an interpreter running on top of another interpreter. It also lacks all forms of optimisation. Benchmarks show the initial release to be approximately 2000 times slower than the CPython interpreter.

2 Introduction

2.1 Purpose of this Document

This document describes the release 0.6 version of the PyPy interpreter. It serves to document when a useable interpreter on top of CPython was released and to what extent it was compatible with CPython.

2.2 Scope of this Document

This document describes basic interpretation of Python code. It does not deal with any aspects of the translation of the PyPy interpreter into lower level code nor any aspects of optimisation, even though such elements were part of the 0.6 release of the project.

2.3 Related Documents

This document is a stand alone document. For a description of how basic interpretation of Python code is done, please see deliverable *04.2 Complete Python implementation running on top of CPython*.

3 Details of the Release

The first public release of the PyPy interpreter was presented on Fri 20 May 2005, with version number 0.6.0. It was followed on Sat 21 May 2005 by a 0.6.1 bugfix release, which fixed a trivial bug in the earlier publication.

The releases are available as http://codespeak.net/download/pypy/pypy-0.6.0.tar.bz2 and http://codespeak.net/download/pypy/pypy-0.6.1.tar.bz2 respectively.

Apart from features relating to other workpackages, this deliverable completes WP 4, task 1:

> *Implement an interpreter that is able to accept the complete Python language specification, built according to the general modularity goals described in the body of the text. This task specifically includes research and implementation work leading to:*
>
> - *the Object Space interface.*
> - *a bytecode interpreter, accepting the standard CPython bytecode format.*

> - *a "standard" object space implementing the core Python objects' normal semantics.*
>
> *This task excludes the standard extension modules and the parser and bytecode compiler, which for the purpose of testing are borrowed from the underlying CPython interpreter on top of which our interpreter runs.*

It also provides parts of WP 4, task 2:

> *Port the standard Python "builtin" library:*
>
> - *types (there are some 100 of them)*
> - *built-in functions*
> - *built-in modules*
> - *other built-in objects, e.g. exception classes.*
>
> *Research and decide on a case-by-case basis how to implement each one within PyPy in a simple and efficient way. The common options are to either re-implement them inside the interpreter where appropriate, or to provide a pure Python replacement (which in some cases already exists and can be borrowed).*

Approximately 80% of the builtins were ported for the 0.6.0 release. Most notably, support for unicode strings and arbitrary length integers were missing in the publication.

3.1 Public Announcement of the Release

This is a copy of the release announcement:

```
The PyPy 0.6 release
++++++++++++++++++++

*The PyPy Development Team is happy to announce the first
public release of PyPy after two years of spare-time and
half a year of EU funded development.  The 0.6 release
is eminently a preview release.*

What it is and where to start
+++++++++++++++++++++++++++++

Getting started:    http://codespeak.net/pypy/index.cgi?doc/getting_started.html

PyPy Documentation: http://codespeak.net/pypy/index.cgi?doc

PyPy Homepage:      http://codespeak.net/pypy/

PyPy is a MIT-licensed reimplementation of Python written in
Python itself.  The long term goals are an implementation that
is flexible and easy to experiment with and retarget to
different platforms (also non-C ones) and such that high
performance can be achieved through high-level implementations
of dynamic optimisation techniques.

The interpreter and object model implementations shipped with 0.6 can
be run on top of CPython and implement the core language features of
```

```
Python as of CPython 2.3.  PyPy passes around 90% of the Python language
regression tests that do not depend deeply on C-extensions.  Some of
that functionality is still made available by PyPy piggy-backing on
the host CPython interpreter.  Double interpretation and abstractions
in the code-base make it so that PyPy running on CPython is quite slow
(around 2000x slower than CPython), this is expected.

This release is intended for people that want to look and get a feel
into what we are doing, playing with interpreter and perusing the
codebase.  Possibly to join in the fun and efforts.

Interesting bits and highlights
+++++++++++++++++++++++++++++++

The release is also a snap-shot of our ongoing efforts towards
low-level translation and experimenting with unique features.

* By default, PyPy is a Python version that works completely with
  new-style-classes semantics.  However, support for old-style classes
  is still available.  Implementations, mostly as user-level code, of
  their metaclass and instance object are included and can be re-made
  the default with the ‘‘--oldstyle‘‘ option.

* In PyPy, bytecode interpretation and object manipulations
  are well separated between a bytecode interpreter and an
  *object space* which implements operations on objects.
  PyPy comes with experimental object spaces augmenting the
  standard one through delegation:

  * an experimental object space that does extensive tracing of
    bytecode and object operations;

  * the ’thunk’ object space that implements lazy values and a ’become’
    operation that can exchange object identities.

  These spaces already give a glimpse in the flexibility potential of
  PyPy.  See demo/fibonacci.py and demo/sharedref.py for examples
  about the ’thunk’ object space.

* The 0.6 release also contains a snapshot of our translation-efforts
  to lower level languages.  For that we have developed an
  annotator which is capable of inferring type information
  across our code base.  The annotator right now is already
  capable of successfully type annotating basically *all* of
  PyPy code-base, and is included with 0.6.

* From type annotated code, low-level code needs to be generated.
  Backends for various targets (C, LLVM,...) are included; they are
  all somehow incomplete and have been and are quite in flux. What is
  shipped with 0.6 is able to deal with more or less small/medium examples.


Ongoing work and near term goals
++++++++++++++++++++++++++++++++

Generating low-level code is the main area we are hammering on in the
next months; our plan is to produce a PyPy version in August/September
```

```
that does not need to be interpreted by CPython anymore and will
thus run considerably faster than the 0.6 preview release.

PyPy has been a community effort from the start and it would
not have got that far without the coding and feedback support
from numerous people.   Please feel free to give feedback and
raise questions.

    contact points: http://codespeak.net/pypy/index.cgi?contact

    contributor list: http://codespeak.net/pypy/index.cgi?doc/contributor.html

have fun,

    Armin Rigo, Samuele Pedroni,

    Holger Krekel, Christian Tismer,

    Carl Friedrich Bolz


    PyPy development and activities happen as an open source project
    and with the support of a consortium funded by a two year EU IST
    research grant. Here is a list of partners of the EU project:

        Heinrich-Heine University (Germany), AB Strakt (Sweden)

        merlinux GmbH (Germany), tismerysoft GmbH(Germany)

        Logilab Paris (France), DFKI GmbH (Germany)

        ChangeMaker (Sweden)
```

3.2 Compliance Test Results

The following list shows to what extent the 0.6.1 release passed the Python compliance tests. These tests were taken from the 2.3.4 release of CPython and in some cases modified by the PyPy project in order to remove implementation dependent details from the tests. In a few cases standard library modules implemented in Python that are provided in the 2.3.4 release of CPython had to be modified. All modified tests and modified modules are found in the *lib-python/modified-2.3.4/* subdirectory of the release. The complete original standard library including tests are found in the *lib-python/2.3.4/* subdirectory of the release. This makes it very easy to compare directories and files in order to determine exactly which changes have been necessary to make.

3.2.1 Core tests

Total test compliance	92.08%
Test modules passed completely	88.15%
Test modules (partially) failed	11.85%
Test modules timeout	0.00%

Details

Percent failed	Test
100.00%	test_imp
100.00%	test_import
100.00%	test_importhooks
25.00%	test_repr
100.00%	test_profile
100.00%	test_tarfile
100.00%	test_userstring
100.00%	test_inspect
100.00%	test_extcall
3.19%	test_pickletools
5.92%	test_generators
100.00%	test___all__
100.00%	test_pyclbr
0.00%	test_future1
0.00%	test_future2
0.00%	test_slice
0.00%	test_unpack
0.00%	test_softspace
0.00%	test_opcodes
0.00%	test_augassign
0.00%	test_new
0.00%	test___future__
0.00%	test_global
0.00%	test_longexp
0.00%	test_multifile
0.00%	test_math
0.00%	test_long_future
0.00%	test_future3
0.00%	test_eof
0.00%	test_atexit
0.00%	test_shutil
0.00%	test_hash
0.00%	test_MimeWriter
0.00%	test_contains
0.00%	test_traceback
0.00%	test_pkgimport
0.00%	test_dircache
0.00%	test_scope
0.00%	test_netrc
0.00%	test_future

Percent failed	Test
0.00%	test_unary
0.00%	test_hexoct
0.00%	test_grammar
0.00%	test_time
0.00%	test_codeop
0.00%	test_syntax
0.00%	test_warnings
0.00%	test_sys
0.00%	test_filecmp
0.00%	test_pkg
0.00%	test_dummy_threading
0.00%	test_httplib
0.00%	test_compile
0.00%	test_call
0.00%	test_whichdb
0.00%	test_dummy_thread
0.00%	test_fnmatch
0.00%	test_urlparse
0.00%	test_profilehooks
0.00%	test_operator
0.00%	test_os
0.00%	test_cmath
0.00%	test_glob
0.00%	test_userdict
0.00%	test_copy_reg
0.00%	test_fileinput
0.00%	test_mimetypes
0.00%	test_univnewlines
0.00%	test_bool
0.00%	test_isinstance
0.00%	test_StringIO
0.00%	test_binop
0.00%	test_imaplib
0.00%	test_string
0.00%	test_trace
0.00%	test_copy
0.00%	test_userlist
0.00%	test_anydbm
0.00%	test_rfc822
0.00%	test_str
0.00%	test_robotparser
0.00%	test_cgi

Percent failed	Test
0.00%	test_htmllib
0.00%	test_calendar
0.00%	test_mimetools
0.00%	test_richcmp
0.00%	test_getopt
0.00%	test_doctest2
0.00%	test_cfgparser
0.00%	test_sgmllib
0.00%	test_shlex
0.00%	test_textwrap
0.00%	test_compare
0.00%	test_doctest
0.00%	test_pprint
0.00%	test_urllib2
0.00%	test_cookie
0.00%	test_difflib
0.00%	test_shelve
0.00%	test_urllib
0.00%	test_dumbdbm
0.00%	test_bisect
0.00%	test_fpformat
0.00%	test_marshal
0.00%	test_hmac
0.00%	test_iter
0.00%	test_htmlparser
0.00%	test_tokenize
0.00%	test_builtin
0.00%	test_sets
0.00%	test_heapq
0.00%	test_pow
0.00%	test_bufio
0.00%	test_pickle
0.00%	test_sort
0.00%	test_long
0.00%	test_datetime

3.2.2 Non core tests

Total test compliance	24.90%
Test modules passed completely	20.33%
Test modules (partially) failed	77.24%

Test modules timeout	2.44%

Details

Percent failed	Test
100.00%	test_threaded_import
100.00%	test_al
100.00%	test_grp
100.00%	test_gdbm
100.00%	test_dl
100.00%	test_cl
100.00%	test_mpz
100.00%	test_zipfile
100.00%	test_audioop
100.00%	test_hotshot
100.00%	test_curses
100.00%	test_queue
100.00%	test_cd
100.00%	test_fork1
100.00%	test_bsddb
100.00%	test_dbm
100.00%	test_sax
100.00%	test_macostools
100.00%	test_plistlib
100.00%	test_ioctl
100.00%	test_aepack
100.00%	test_zipimport
100.00%	test_weakref
100.00%	test_email
100.00%	test_email_codecs
100.00%	test_pwd
100.00%	test_imageop
100.00%	test_winsound
100.00%	test_crypt
100.00%	test_gl
100.00%	test_sunaudiodev
100.00%	test_imgfile
100.00%	test_timing
100.00%	test_symtable
100.00%	test_nis
100.00%	test_zlib
100.00%	test_threadedtempfile
100.00%	test_winreg

Percent failed	Test
100.00%	test_pep277
100.00%	test_mmap
100.00%	test_frozen
100.00%	test_ossaudiodev
100.00%	test_signal
100.00%	test_asynchat
100.00%	test_bsddb185
100.00%	test_normalization
100.00%	test_linuxaudiodev
100.00%	test_bsddb3
100.00%	test_gzip
100.00%	test_gc
100.00%	test_rgbimg
100.00%	test_thread
100.00%	test_select
100.00%	test_bz2
100.00%	test_threading
100.00%	test_poll
100.00%	test_getargs
100.00%	test_locale
100.00%	test_pyexpat
100.00%	test_csv
100.00%	test_pep247
100.00%	test_capi
100.00%	test_pty
100.00%	test_macfs
100.00%	test_xreadline
100.00%	test_strop
100.00%	test_scriptpackages
100.00%	test_resource
100.00%	test_unicode_file
100.00%	test_getargs2
100.00%	test_structseq
100.00%	test_socket_ssl
100.00%	test_socket
100.00%	test_socketserver
100.00%	test_rotor
100.00%	test_minidom
100.00%	test_binhex
100.00%	test_regex
100.00%	test_charmapcodec
100.00%	test_logging

Percent failed	Test
100.00%	test_timeout
11.11%	test_uu
40.00%	test_xmlrpc
100.00%	test_xmllib
100.00%	test_parser
25.00%	test_binascii
100.00%	test_sundry
100.00%	test_array
10.00%	test_urllibnet
100.00%	test_gettext
35.48%	test_random
12.77%	test_re
2.38%	test_strptime
100.00%	test_ucn
100.00%	test_unicodedata
100.00%	test_unicode
0.00%	test_bastion
0.00%	test_pep263
0.00%	test_openpty
0.00%	test_fcntl
0.00%	test_stringprep
0.00%	test_wave
0.00%	test_base64
0.00%	test_popen
0.00%	test_ntpath
0.00%	test_mailbox
0.00%	test_largefile
0.00%	test_macpath
0.00%	test_quopri
0.00%	test_commands
0.00%	test_dis
0.00%	test_posix
0.00%	test_errno
0.00%	test_posixpath
0.00%	test_md5
0.00%	test_mhlib
0.00%	test_xpickle
0.00%	test_popen2
0.00%	test_strftime

4 Glossary of Abbreviations

The following abbreviations may be used within this document:

4.1 Technical Abbreviations:

AST	Abstract Syntax Tree
CPython	The standard Python interpreter written in C. Generally known as "Python". Available from www.python.org.
codespeak	The name of the machine where the PyPy project is hosted.
docutils	The Python documentation utilities.
GenC backend	The backend for the PyPy translation toolsuite that generates C code.
GenLLVM backend	The backend for the PyPy translation toolsuite that generates LLVM code.
Graphviz	Graph visualisation software from AT&T.
Jython	A version of Python written in Java.
LLVM	Low Level Virtual Machine - a compiler infrastructure available from University of Illinois at Urbana-Champaign
LOC	Lines of code.
Object Space	A library providing objects and operations between them, available to the bytecode interpreter via a well-defined API.
Pygame	A Python extension library that wraps the Simple Directmedia Library - a cross-platform multimedia library designed to provide fast access to the graphics framebuffer and audio device.
ReST	reStructuredText, the plaintext markup system used by docutils.
RPython	Restricted Python; a less dynamic subset of Python in which PyPy is written.
Standard Interpreter	The subsystem of PyPy which implements the Python language. It is divided in two components: the bytecode interpreter, and the standard object space.
Standard Object Space	An object space which implements creation, access and modification of regular Python application level objects.

4.2 Partner Acronyms:

DFKI	Deutsches Forschungszentrum für künstliche Intelligenz
HHU	Heinrich Heine Universität Düsseldorf
Strakt	AB Strakt
Logilab	Logilab
CM	Change Maker
mer	merlinux GmbH
tis	Tismerysoft GmbH