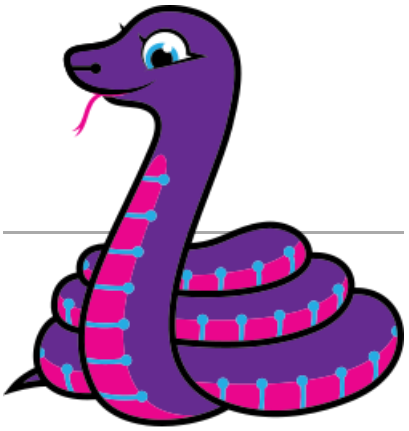
CircuitPython

CircuitPython[5] is an open-source derivative of the MicroPython programming language targeted toward students and beginners. Development of CircuitPython is supported by Adafruit Industries. It is a software implementation of the Python 3 programming language, written in C.[3] It has been ported to run on several modern microcontrollers.

CircuitPython consists of a Python compiler to bytecode and a runtime interpreter of that bytecode that runs on the microcontroller hardware. The user is presented with an interactive prompt (the REPL) to execute supported commands immediately. Included are a selection of core Python libraries. CircuitPython includes modules which give the programmer access to the low-level hardware of supported products as well as higher-level libraries for beginners.[6]

CircuitPython is a fork of MicroPython, originally created by Damien George.[7] The MicroPython community continues to discuss[8] forks of MicroPython into variants such as CircuitPython.

CircuitPython



Logo of the Blinka library, a compatibility layer for CircuitPython

Original author(s)	Adafruit Industries
Initial release	July 19, 2017[1]
Stable release	9.2.7[2] / 1 April 2025
Repository	https://github.com/adafruit/circuitpython
Written in	C[3]
Platform	Supported microcontrollers and single-board computers
Type	Python implementation
License	MIT license[4]
Website	circuitpython.org (https://circuitpython.org/)

CircuitPython is targeted to be compatible with CPython, the reference implementation of the Python programming language.[9] Programs written for CircuitPython-compatible boards may not run unmodified on other platforms such as the Raspberry Pi.[10]

Usage

CircuitPython is being used as an emerging alternative solution for microcontroller programming, which is usually done in C, C++, or assembly. The language has also seen uptake in making small, handheld video game devices.[11] Developer Chris Young has ported his infrared transmit-and-receive software to CircuitPython to provide interactivity and to aid those with accessibility issues.[12]

Community

The user community support includes a Discord chat room and product support forums.[13] A Twitter account dedicated to CircuitPython news was established in 2018.[14] A newsletter, Python on Microcontrollers, is published weekly since 15 November, 2016 by Adafruit to provide news and information on CircuitPython, MicroPython, and Python on single board computers.[15] A Reddit subreddit, r/CircuitPython, provides news on CircuitPython and related news and projects and has about 4,300 members.[16]

Hardware support

The version 9.1.0 supports a range of platforms, called "ports":[17]

- atmel-samd: Microchip SAMD21, SAMx5x
- cxd56: Sony Spresense
- espressif: Espressif ESP32, ESP32-S2, ESP32-S3, ESP32-C2, ESP32-C3, ESP32-C6
- nordic: Nordic nRF52840, nRF52833
- raspberrypi: Raspberry Pi RP2040, RP2350
- stm: ST STM32F4 chip family

These ports are considered alpha and will have bugs and missing functionality:

- broadcom: Raspberry Pi boards such as RPi 4, RPi Zero 2W (bare metal)
- litex: fomu
- mimxrt10xx: NXP i.MX RT10xxx
- renode: hardware simulator
- silabs: Silicon Labs MG24 family
- stm: ST non-STM32F4 chip families

Previous versions supported the ESP8266 microcontroller, but its support was dropped in version 4.[18]

Blinka Software Abstraction Layer

CircuitPython code may run on MicroPython or CPython using the Adafruit written Blinka compatibility layer.[19] It acts as a translation layer between CircuitPython code and underlying code. This allows CircuitPython code to run on many more devices including a wide range of single-board computers which are listed on circuitpython.org.[20] It is a pip installable Python library. The CircuitPython runtime is not used, as documented in the guide *CircuitPython Libraries on Linux and Raspberry Pi*.[21]

Modules (Libraries)

Adafruit has fostered a community which has contributed software libraries for more than 488 sensors and drivers.[22][23]

References

1. Shawcroft, Scott (19 July 2017). "CircuitPython 1.0.0!" (https://blog.adafruit.com/2017/07/19/circuitpython-1-0-0/). *Adafruit Blog*. Adafruit Industries. Retrieved 1 May 2018.
2. "Release 9.2.7" (https://github.com/adafruit/circuitpython/releases/tag/9.2.7). 1 April 2025. Retrieved 24 April 2025.
3. "adafruit/circuitpython" (https://github.com/adafruit/circuitpython). *GitHub*. Adafruit Industries. Retrieved 2 May 2018.
4. George, Damien P. (4 May 2014). "circuitpython/LICENSE" (https://github.com/adafruit/circuitpython/blob/master/LICENSE/micropython). *GitHub*. Retrieved 1 May 2018.
5. "CircuitPython is an education friendly open-source derivative of MicroPython" (https://github.com/adafruit/circuitpython). *GitHub*. Retrieved 30 April 2018.
6. "CircuitPython" (https://readthedocs.org/projects/circuitpython/). *Read the Docs*. Adafruit Industries. Retrieved 1 May 2018.
7. George, Damien (20 May 2016). "Damien P. George" (http://dpgeorge.net/). Damien P. George. Retrieved 1 May 2018.
8. "Adafruit CircuitPython" (https://forum.micropython.org/viewtopic.php?t=2894). *MicroPython Forum*. MicroPython.org. Retrieved 2 May 2018.
9. Lewis, James (14 February 2018). "Circuit Python adds Python to Microcontrollers" (https://www.baldengineer.com/circuit-python.html). *The Bald Engineer*. Retrieved 2 May 2018.
10. Ganne, Simon. "Can I use circuitPython code on my raspberry?" (https://www.element14.com/community/thread/62774/l/can-i-use-circuitpython-code-on-my-raspberry?displayFullThread=true). *Element 14 Community*. Element 14.
11. Dopieralski, Radomir. "CircuitPython LAMEBOY" (https://bitbucket.org/thesheep/circuitpython-lameboy). *BitBucket*. Retrieved 2 May 2018.
12. Young, Chris (6 June 2018). "Announcing IRLibCP — a Circuit Python Module for Infrared Transmitting and Receiving" (http://tech.cyborg5.com/2017/06/06/announcing-irlibcp-a-circuit-python-module-for-infrared-transmitting-and-receiving/). *CY's Tech Talk*. Chris Young. Retrieved 2 May 2018.
13. "Adafruit CircuitPython and MicroPython" (https://forums.adafruit.com/viewforum.php?f=60). *Adafruit Support Forums*. Adafruit Industries. Retrieved 1 May 2018.
14. "CircuitPython" (https://twitter.com/CircuitPython). *Twitter*. Adfafruit Industries. Retrieved 1 May 2018.
15. "The Python on Microcontrollers Newsletter" (https://www.adafruitdaily.com/category/circuitpython/). *Adafruit Daily*. Adafruit Industries. Retrieved 17 July 2024.
16. "r/CircuitPython" (https://www.reddit.com/r/circuitpython/). *Reddit.com*. Adafruit Industries. Retrieved 17 July 2024.
17. "CircuitPython 9.1.0" (https://github.com/adafruit/circuitpython/releases/tag/9.1.0). *GitHub*. 17 July 2024.
18. "Why are we dropping support for ESP8266?" (https://learn.adafruit.com/welcome-to-circuitpython/circuitpython-for-esp8266). *Adafruit.com*. Adafruit Industries. Retrieved 15 April 2019.
19. "Blinka" (https://circuitpython.org/blinka). *CircuitPython*. Adafruit Industries. Retrieved 17 July 2024.
20. "Blinka" (https://circuitpython.org/blinka). *CircuitPython*. Adafruit Industries. Retrieved 17 July 2024.
21. "CircuitPython Libraries on Linux and Raspberry Pi" (https://learn.adafruit.com/circuitpython-on-raspberrypi-linux). *Adafruit Learning System*. Adafruit Industries Industries. Retrieved 17 July 2024.

22. "Python on Microcontrollers Newsletter" (https://www.adafruitdaily.com/2024/07/15/python-on-microcontrollers-newsletter-circuitpython-9-1-released-arduino-adopts-qwiic-and-so-much-more-circuitpython-python-micropython-thepsf-raspberry_pi/). Retrieved 17 July 2024.
23. "CircuitPython Libraries" (https://circuitpython.org/libraries). Retrieved 17 July 2024.

External links

- CircuitPython (https://github.com/Adafruit/CircuitPython) on GitHub
- MicroPython (https://www.youtube.com/playlist?list=PLjF7R1fz_OOXrI15wuXeESA0aA4VzcWSi) playlist on YouTube • Tutorials by Tony DiCola / Adafruit

CLPython

CLPython is an implementation of the Python programming language written in Common Lisp. This project allow to call Lisp functions from Python and Python functions from Lisp. Licensed under LGPL. CLPython was started in 2006, but as of 2013, it was not actively developed and the mailing list was closed.[1]

CLPython	
Repository	github.com/metawilm/cl-python (https://github.com/metawilm/cl-python)
Platform	Common Lisp
Type	Python Programming Language Compiler
License	LGPL
Website	common-lisp.net/project/clpython/ (http://common-lisp.net/project/clpython/)

See also

Free and open-source software portal

- CPython - the default implementation of Python, written in C.
- IronPython - an implementation of Python in C# targeting .NET and Mono.
- Jython - an implementation of Python for the JVM.
- Hy - An implementation of Lisp written in Python

References

1. "CLPython - an implementation of Python in Common Lisp" (https://common-lisp.net/project/clpython/).

External links

- Official website (http://common-lisp.net/project/clpython/)
- https://github.com/metawilm/cl-python

CPython

CPython is the reference implementation of the Python programming language. Written in C and Python, CPython is the default and most widely used implementation of the Python language.

CPython can be defined as both an interpreter and a compiler as it compiles Python code into bytecode before interpreting it. It has a foreign function interface with several languages, including C, in which one must explicitly write bindings in a language other than Python.

CPython

Original author(s)	Guido van Rossum
Developer(s)	Python core developers and the Python community, supported by the Python Software Foundation
Initial release	26 January 1994
Stable release	3.13.3[1] / 8 April 2025
Repository	https://github.com/python/cpython
Written in	C, Python
Platform	42 platforms; see § Distribution
Available in	English
Type	Python Programming Language Interpreter
License	Python Software Foundation License
Website	www.python.org (https://www.python.org/)

Design

A particular feature of CPython is that it makes use of a global interpreter lock (GIL) such that for each CPython interpreter process, only one thread may be processing bytecode at a time.[2] This does not mean that there is no point in multithreading; the most common multithreading scenario is where threads are mostly waiting on external processes to complete.

This can happen when multiple threads are servicing separate clients. One thread may be waiting for a client to reply, and another may be waiting for a database query to execute, while the third thread is actually processing Python code.

However, the GIL does mean that CPython is not suitable for processes that implement CPU-intensive algorithms in Python code that could potentially be distributed across multiple cores.

In real-world applications, situations where the GIL is a significant bottleneck are quite rare. This is because Python is an inherently slow language and is generally not used for CPU-intensive or time-sensitive operations. Python is typically used at the top level and calls functions in libraries to perform specialized tasks. These libraries are generally not written in Python, and Python code in another thread can be executed while a call to one of these underlying processes takes place. The non-Python library being called to perform the CPU-intensive task is not subject to the GIL and may concurrently execute many threads on multiple processors without restriction.

Concurrency of Python code can only be achieved with separate CPython interpreter processes managed by a multitasking operating system. This complicates communication between concurrent Python processes, though the *multiprocessing* module mitigates this somewhat; it means that applications that

really can benefit from concurrent Python-code execution can be implemented with limited overhead.

The presence of the GIL simplifies the implementation of CPython, and makes it easier to implement multi-threaded applications that do not benefit from concurrent Python code execution. However, without a GIL, multiprocessing apps must make sure all common code is thread safe.

Although many proposals have been made to eliminate the GIL, the general consensus has been that in most cases, the advantages of the GIL outweigh the disadvantages; in the few cases where the GIL is a bottleneck, the application should be built around the multiprocessing structure. To help allow more parallelism, an improvement was released in October 2023 to allow a separate GIL per sub-interpreter in a single Python process and have been described as "threads with opt-in sharing".[3][4]

After several debates, a project was launched in 2023 to propose making the GIL optional from version 3.13 of Python,[5] which was released October 7, 2024, in Python 3.13.0.[6][7]

History

In 2009, a Google sponsored branch named Unladen Swallow was created to incorporate a just-in-time compiler into CPython.[8][9] Development ended in 2011 without it being merged into the main implementation,[10] though some of its code, such as improvements to the cPickle module, made it in.[11][8]

In 2021, a "specializing adaptive interpreter" was proposed, which was measured to improve performance by 10-60% by specializing commonly executed instructions displaying apparent type stability into faster, type specific instructions, and which could de-specialize instructions when necessary.[12] The SAI was first included in Python 3.11, which was measured to be 25% faster on average than Python 3.10 by the "pyperformance" benchmark suite.[13]

In 2024, an experimental Just-in-time compiler was merged into CPython’s main development branch. This early JIT sits on top of LLVM, aiming to speed up hot code paths. At the time of the merge, the compiler was still not included in CPython’s default build configurations and offered roughly equal performance to the SAI; one of the conditions for its full adoption was a performance increase of at least 5%.[14] It remains disabled by default, though users can enable it to experiment and see how Python might eventually rival other JIT’d languages.[15]

Distribution

Officially supported tier-1 platforms are Linux for 64-bit Intel using a GCC toolchain, macOS for 64-bit Intel and ARM, and Microsoft Windows for 32- and 64-bit Intel. Official tier-2 support exists for Linux for 64-bit ARM, wasm32 (Web Assembly) with WASI runtime support, and Linux for 64-bit Intel using a clang toolchain. Official supported tier-3 systems include 64-bit ARM Windows, 64-bit iOS, Raspberry Pi OS (Linux for armv7 with hard float), Linux for 64-bit PowerPC in little-endian mode, and Linux for s390x.

More platforms have working implementations, including:[16]

Unix-like

- AIX
- BSD
- Darwin
- FreeBSD
- HP-UX
- illumos
- Linux
- macOS
- NetBSD
- OpenBSD
- Plan 9
- Solaris
- Tru64

Special and embedded

- Android
- Apple iOS (support for outdated Python 3.6 and 2.7 available)
- BlackBerry 10
- GP2X
- iPodLinux
- Nintendo DS
- GameCube
- Symbian OS Series60
- Nokia 770 Internet Tablet
- Nokia N800
- Nokia N810
- Nokia N900
- Openmoko
- Palm OS
- PlayStation 2
- PlayStation 3 (FreeBSD)
- Psion
- QNX
- Sharp Zaurus
- Xbox/XBMC
- VxWorks

Other

- AROS
- OS/390
- Windows 8 and later
- z/OS

PEP 11[17] lists platforms which are not supported in CPython by the Python Software Foundation. These platforms can still be supported by external ports. These ports include:

- AtheOS (unsupported since 2.6)
- BeOS (unsupported since 2.6)
- DOS (unsupported since 2.0)
- IRIX 4 (unsupported since 2.3)
- IRIX 5 and later (unsupported since 3.2, 3.7)[18]
- Mac OS 9 (unsupported since 2.4)
- MINIX (unsupported since 2.3)
- OpenVMS (unsupported since 3.3)
- OS/2 (unsupported since 3.3)
- RISC OS (unsupported since 3.0)
- Windows: Generally, Windows versions continue to receive full tier-1 support for as long as they are still covered by Microsoft's extended support lifecycle policy. Once Microsoft's extended support period expires for an older version of Windows, the project will no longer support that version of Windows in the next major (X.Y.0) release of Python. However, bug fix releases (0.0.Z) for each release branch will retain support for all versions of Windows that were supported in the initial X.Y.0 release. Editions of Windows targeting embedded and IoT systems are considered outside the scope of Python's official support policy.
  - Windows 8 (no official support in any major releases of Python since January 10, 2023)
  - Windows 7 (no official support in any major releases of Python since January 14, 2020)
  - Windows Vista (unsupported since 3.9)
  - Windows XP (unsupported since 3.5)
  - Windows 2000 (unsupported since 3.3)
  - Windows 3.x (unsupported since 2.0)
  - Windows 9x (unsupported since 2.6)
  - Windows NT4 (unsupported since 2.6)

External ports not integrated to Python Software Foundation's official version of CPython, with links to its main development site, often include additional modules for platform-specific functionalities, like graphics and sound API for PSP and SMS and camera API for S60. These ports include:

- Amiga: AmigaPython[19]
- IBM i: iSeriesPython[20]
- DOS using DJGPP: PythonD[21]
- MorphOS: Python 2 and 3[22]
- PlayStation Portable: Stackless Python for PSP[23]
- Symbian OS: Python for S60
- Windows CE/Pocket PC: Python Windows CE port[24]
- OpenVMS: Ports of Python 3.x are maintained by VSI[25]

Enterprise Linux

These Python versions are distributed with currently-supported enterprise Linux distributions.[26] The support status of Python in the table refers to support from the Python core team, and not from the distribution maintainer.

Enterprise Linux

Distribution version	Distribution end-of-life	Python version	
Ubuntu 22.04 LTS (Jammy Jellyfish)			3.10 [1] (https://launchpad.net/ubuntu/jammy/+package/python3)
Ubuntu 20.04 LTS (Focal Fossa)	2030-04[27]	[28]	3.8[29]
Ubuntu 18.04 LTS (Bionic Beaver)	2028-04[30]	2.7[31]	3.6[32]
Ubuntu 16.04 LTS (Xenial Xerus)	2021-04-30[33]	2.7[31]	3.5[32]
Debian 12	2028-06[34]		3.11[34]
Debian 11	2026-06[35]		3.9[35]
Debian 10	2024-06[36]	2.7[37]	3.7[38]
Debian 9	2022-06-30[39]	2.7[40]	3.5[41]
Red Hat Enterprise Linux 8	2029	2.7[42]	3.6
Red Hat Enterprise Linux 7	2024-11-30[43]	2.7[44]	
CentOS 8	2029-05-31	2.7	3.6
CentOS 7	2024-06-30	2.7[45]	
SUSE Linux Enterprise Server 15	2031-07-31	2.7[46]	3.6
SUSE Linux Enterprise Server 12	2027-10-31	2.7[47]	
SUSE Linux Enterprise Server 11	2022-03-31	2.7[47]	
Old version, not maintained	Old version, still maintained	**Latest version**	

Alternatives

CPython is one of several "production-quality" Python implementations including: Jython, written in Java for the Java virtual machine (JVM); PyPy, written in RPython and translated into C; and IronPython, written in C# for the Common Language Infrastructure. There are also several experimental implementations.[48]

References

1. "Release v3.13.3" (https://github.com/python/cpython/releases/tag/v3.13.3). 8 April 2025. Retrieved 9 April 2025.
2. "Initialization, Finalization, and Threads" (https://docs.python.org/3/c-api/init.html). Python v3.8.3 documentation. Retrieved 2020-06-04.
3. Jake Edge (August 15, 2023). "A per-interpreter GIL" (https://lwn.net/Articles/941090/). *LWN*. Retrieved 2024-01-13.

4. "PEP 684 – A Per-Interpreter GIL | peps.python.org" (https://peps.python.org/pep-0684/). Retrieved 2024-01-13.
5. "PEP 703 – Making the Global Interpreter Lock Optional in CPython | peps.python.org" (https://peps.python.org/pep-0703/). *peps.python.org*. Retrieved 2023-09-17.
6. "PEP 719 – Python 3.13 Release Schedule | peps.python.org" (https://peps.python.org/pep-0719/#schedule). *peps.python.org*. Retrieved 2023-09-17.
7. "Python Release Python 3.13.0" (https://www.python.org/downloads/release/python-3130/). *Python.org*. Retrieved 2025-03-28.
8. Winter, Collin; Yasskin, Jeffrey; Kleckner, Reid (2010-03-17). "PEP 3146 - Merging Unladen Swallow into CPython" (https://peps.python.org/pep-3146/). Python.org.
9. "Unladen Swallow 2009Q1" (https://code.google.com/p/unladen-swallow/wiki/Release2009Q1). unladen-swallow, A faster implementation of Python. Retrieved 19 October 2012.
10. Kleckner, Reid (26 March 2011). "Unladen Swallow Retrospective" (https://qinsb.blogspot.com/2011/03/unladen-swallow-retrospective.html). *QINSB is not a Software Blog (qinsb.blogspot.com)*.
11. "Issue 9410: Add Unladen Swallow's optimizations to Python 3's pickle. - Python tracker" (https://bugs.python.org/issue9410). *bugs.python.org*. Retrieved 2019-08-08.
12. Mark Shannon (April 13, 2021). "PEP 659 – Specializing Adaptive Interpreter" (https://peps.python.org/pep-0659/).
13. Pablo Galindo Salgado. "What's new in Python 3.11 § PEP 659: Specializing Adaptive Interpreter" (https://docs.python.org/3/whatsnew/3.11.html#whatsnew311-pep659).
14. Brandt Bucher; Savannah Ostrowski (April 11, 2024). "PEP 744 {{subst:endash}} JIT Compilation" (https://peps.python.org/pep-0744/). Retrieved February 5, 2025.
15. "What's New in Python 3.13.3 & the Upcoming 3.14—A Beginner's Guide" (https://kritimyantra.com/blogs/whats-new-in-python-3-13-3-the-upcoming-3-14-a-beginners-guide). Retrieved 2025-04-25.
16. "PythonImplementations" (https://wiki.python.org/moin/PythonImplementations). Retrieved 19 July 2012.
17. "PEP 11 -- Removing support for little used platforms" (https://peps.python.org/pep-0011/). *Python.org*. Retrieved 2019-08-08.
18. "Irix still supported?" (https://mail.python.org/pipermail/python-dev/2009-February/086111.html). 14 February 2009.
19. "AmigaPython Home Page" (http://www.monkeyhouse.eclipse.co.uk/amiga/python/). *www.monkeyhouse.eclipse.co.uk*. Retrieved Feb 20, 2025.
20. "I | Series | Python" (https://www.iseriespython.com/). *I* | *Series* | *Python*. Retrieved Feb 20, 2025.
21. "PythonD 32bit Python for DOS and Windows" (https://www.caddit.net/pythond/). *www.caddit.net*. Retrieved Feb 20, 2025.
22. http://yellowblue.free.fr/yiki/doku.php/en:dev:python:start Python 2 and 3
23. "Google Code Archive - Long-term storage for Google Code Project Hosting" (https://code.google.com/archive/p/pspstacklesspython). *code.google.com*. Retrieved Feb 20, 2025.
24. "Python Windows CE port" (https://sourceforge.net/projects/pythonce/). *SourceForge*. Jan 22, 2018. Retrieved Feb 20, 2025.
25. "Python" (https://vmssoftware.com/products/python/). *VSI*. Retrieved 2021-08-31.
26. "Support Life Cycles for Enterprise Linux Distributions" (https://web.archive.org/web/20220830033136/https://linuxlifecycle.com/). Archived from the original (https://linuxlifecycle.com/) on 2022-08-30. Retrieved 2017-10-15.
27. "Ubuntu release cycle" (https://ubuntu.com/about/release-cycle). *Ubuntu*. Retrieved 2021-01-18.

28. "With Python 2 EOL'ed, Ubuntu 20.04 LTS Moves Along With Its Python 2 Removal - Phoronix" (https://www.phoronix.com/news/Python-2-EOL-Ubuntu-20.04). *phoronix.com*. Retrieved 2020-04-01.
29. "Binary package "python3" in ubuntu focal" (https://launchpad.net/ubuntu/focal/+package/python3). *Launchpad.net*.
30. "Ubuntu 18.04 extended to 2028" (https://www.serverwatch.com/server-news/canonical-extends-ubuntu-18.04-lts-linux-support-to-10-years.html). ServerWatch.com. 2018-11-15. Retrieved 2019-09-09.
31. "python-defaults package: Ubuntu" (https://launchpad.net/ubuntu/+source/python-defaults). Canonical Ltd. 2018-06-08. Retrieved 2018-06-08.
32. "python3-defaults package: Ubuntu" (https://launchpad.net/ubuntu/+source/python3-defaults). Canonical Ltd. 2018-06-08. Retrieved 2018-06-08.
33. Science, Carnegie Mellon University School of Computer. "Ubuntu 16.04 - End of Life in 2021 - SCS Computing Facilities - Carnegie Mellon University" (https://computing.cs.cmu.edu/news/2020/eol-ubuntu-1604). *computing.cs.cmu.edu*. Retrieved 2021-02-15.
34. "Debian 12 bookworm released" (https://www.debian.org/News/2023/20230610). *debian.org*.
35. "Debian -- News -- Debian 11 "bullseye" released" (https://web.archive.org/web/20210814210911/https://www.debian.org/News/2021/20210814.en.html). *debian.org*. Archived from the original (https://www.debian.org/News/2021/20210814.en.html) on 2021-08-14. Retrieved 2022-01-04.
36. "LTS - Debian Wiki" (https://wiki.debian.org/LTS). *wiki.debian.org*. Retrieved 2021-02-15.
37. "Debian -- Details of package python in buster" (https://packages.debian.org/buster/python). *packages.debian.org*. Retrieved 2019-09-13.
38. "Debian -- News -- Debian 10 "buster" released" (https://www.debian.org/News/2019/20190706.en.html). *debian.org*. Retrieved 2019-08-09.
39. "Debian -- News -- Debian 8 Long Term Support reaching end-of-life" (https://www.debian.org/News/2020/20200709.en.html). *debian.org*. Retrieved 2021-02-15.
40. "DistroWatch.com: Debian" (https://distrowatch.com/table.php?distribution=debian). DistroWatch.com. 2017-10-15. Retrieved 2017-10-15.
41. "Debian -- Details of package python3 in stretch" (https://packages.debian.org/stretch/python3). Retrieved 2017-12-19.
42. "Python in RHEL 8" (https://web.archive.org/web/20190510043548/https://developers.redhat.com/blog/2018/11/14/python-in-rhel-8/). *Red Hat Developer Blog*. 2018-11-14. Archived from the original (https://developers.redhat.com/blog/2018/11/14/python-in-rhel-8/) on 2019-05-10. Retrieved 2019-05-10.
43. "Red Hat Enterprise Linux Life Cycle" (https://access.redhat.com/support/policy/updates/errata). *Red Hat Customer Portal*. Retrieved 2020-04-01.
44. "DistroWatch.com: Red Hat Enterprise Linux" (https://distrowatch.com/table.php?distribution=redhat). DistroWatch.com. 2017-09-07. Retrieved 2017-10-15.
45. "DistroWatch.com: CentOS" (https://distrowatch.com/table.php?distribution=centos). DistroWatch.com. 2017-09-14. Retrieved 2017-10-15.
46. "Release Notes | SUSE Linux Enterprise Desktop/SUSE Linux Enterprise Workstation Extension 15 GA" (https://www.suse.com/releasenotes/x86_64/SUSE-SLED/15/index.html#Features.Other). *suse.com*. Retrieved 2019-08-08.
47. "DistroWatch.com: openSUSE" (https://distrowatch.com/table.php?distribution=opensuse). DistroWatch.com. 2017-10-14. Retrieved 2017-10-15.
48. Martelli, Alex (2006). *Python in a Nutshell* (2nd ed.). O'Reilly. pp. 5–7. ISBN 978-0-596-10046-9.

Further reading

- Shaw, Anthony (2021). *CPython Internals: Your Guide to the Python 3 Interpreter*. Real Python. ISBN 9781775093343.

External links

- CPython (https://github.com/python/cpython) on GitHub

Cython

FreeBSD Ports

Cython (/ˈsaɪθɒn/) is a superset of the programming language Python, which allows developers to write Python code (with optional, C-inspired syntax extensions) that yields performance comparable to that of C.[5][6]

Cython is a compiled language that is typically used to generate CPython extension modules. Annotated Python-like code is compiled to C and then automatically wrapped in interface code, producing extension modules that can be loaded and used by regular Python code using the import statement, but with significantly less computational overhead at run time. Cython also facilitates wrapping independent C or C++ code into python-importable modules.

Cython is written in Python and C and works on Windows, macOS, Linux, and FreeBSD producing C source files compatible with CPython 2.6, 2.7, and 3.3 and later versions. The Cython source code that Cython compiles (to C) can use both Python 2 and Python 3 syntax, defaulting to Python 2 syntax in Cython 0.x and Python 3 syntax in Cython 3.x. The default can be overridden (e.g. in source code comment) to Python 3 (or 2) syntax. Since Python 3 syntax has changed in recent versions, Cython may not be up to date with the latest additions. Cython has "native support for most of the C++ language" and "compiles almost all existing Python code".[7]

Cython 3.0.0 was released on 17 July 2023.[8]

Cython

Developer	Robert Bradshaw, Stefan Behnel, et al.
First appeared	28 July 2007[1]
Stable release	3.1.0-1[2] (8 May 2025) [±] (https://en.wikipedia.org/w/index.php?title=Template:Latest_stable_software_release/Cython&action=edit)
Preview release	3.0.0 beta 2 (27 March 2023[3]) [±] (https://en.wikipedia.org/w/index.php?title=Template:Latest_preview_software_release/Cython&action=edit)
Implementation language	Python
OS	Windows, macOS, Linux
License	Apache License 2.0
Filename extensions	.pyx, .pxd, .pxi [4]
Website	cython.org (https://cython.org)
Influenced by	
C, Python	

Design

Cython works by producing a standard Python module. However, the behavior differs from standard Python in that the module code, originally written in Python, is translated into C. While the resulting code is fast, it makes many calls into the CPython interpreter and CPython standard libraries to perform actual work. Choosing this arrangement saved considerably on Cython's development time, but modules have a dependency on the Python interpreter and standard library.

Although most of the code is C-based, a small stub loader written in interpreted Python is usually required (unless the goal is to create a loader written entirely in C, which may involve work with the undocumented internals of CPython). However, this is not a major problem due to the presence of the Python interpreter.[9]

Cython has a foreign function interface for invoking C/C++ routines and the ability to declare the static type of subroutine parameters and results, local variables, and class attributes.

A Cython program that implements the same algorithm as a corresponding Python program may consume fewer computing resources such as core memory and processing cycles due to differences between the CPython and Cython execution models. A basic Python program is loaded and executed by the CPython virtual machine, so both the runtime and the program itself consume computing resources. A Cython program is compiled to C code, which is further compiled to machine code, so the virtual machine is used only briefly when the program is loaded.[10][11][12][13]

Cython employs:

- Optimistic optimizations
- Type inference (optional)
- Low overhead in control structures
- Low function call overhead[14][15]

Performance depends both on what C code is generated by Cython and how that code is compiled by the C compiler.[16]

History

Cython is a derivative of the Pyrex language, but it supports more features and optimizations than Pyrex.[17][18] Cython was forked from Pyrex in 2007 by developers of the Sage computer algebra package, because they were unhappy with Pyrex's limitations and could not get patches accepted by Pyrex's maintainer Greg Ewing, who envisioned a much smaller scope for his tool than the Sage developers had in mind. They then forked Pyrex as SageX. When they found people were downloading Sage just to get SageX, and developers of other packages (including Stefan Behnel, who maintains the XML library LXML) were also maintaining forks of Pyrex, SageX was split off the Sage project and merged with `cython-lxml` to become Cython.[19]

Cython files have a `.pyx` extension. At its most basic, Cython code looks exactly like Python code. However, whereas standard Python is dynamically typed, in Cython, types can optionally be provided, allowing for improved performance, allowing loops to be converted into C loops where possible. For example:

```
# The argument will be converted to int or raise a TypeError.
def primes(int kmax):

    # These variables are declared with C types.
    cdef int n, k, i

    # Another C type
    cdef int p[1000]

    # A Python type
```

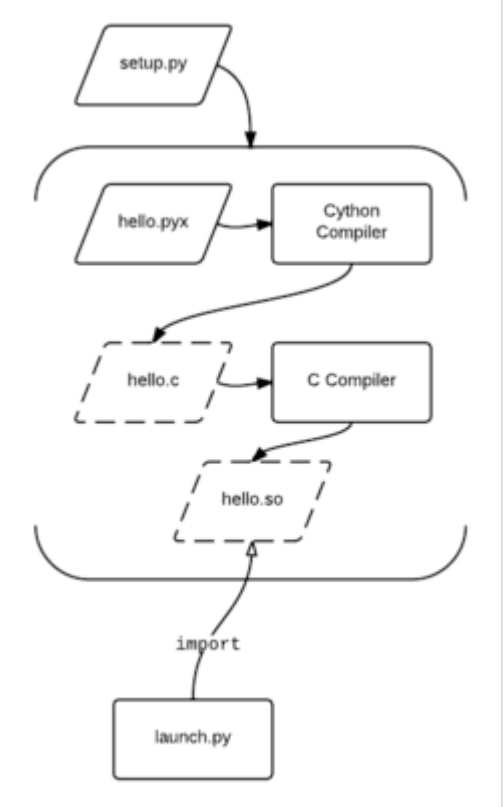
```
    result = []

    if kmax > 1000:
        kmax = 1000

    k = 0
    n = 2

    while k < kmax:
        i = 0

        while i < k and n % p[i] != 0:
            i = i + 1

        if i == k:
            p[k] = n
            k = k + 1
            result.append(n)

        n = n + 1

    return result
```

Example

A sample hello world program for Cython is more complex than in most languages because it interfaces with the Python C API and `setuptools` or other PEP517-compliant extension building facilities. At least three files are required for a basic project:

- A `setup.py` file to invoke the `setuptools` build process that generates the extension module
- A main python program to load the extension module
- Cython source file(s)

The following code listings demonstrate the build and launch process:



Hello World in Cython

```
# hello.pyx - Python module, this code will be translated
to C by Cython.
def say_hello():
    print("Hello World!")
```

```
# launch.py - Python stub loader, loads the module that
was made by Cython.

# This code is always interpreted, like normal Python.
# It is not compiled to C.

import hello
hello.say_hello()
```

```
# setup.py - unnecessary if not redistributing the code, see below
from setuptools import setup
from Cython.Build import cythonize
```

```
setup(name = "Hello world app",
      ext_modules = cythonize("*.pyx"))
```

These commands build and launch the program:

```
$ python setup.py build_ext --inplace
$ python launch.py
```

Using in IPython/Jupyter notebook

A more straightforward way to start with Cython is through command-line IPython (or through in-browser python console called Jupyter notebook):

```
In [1]: %load_ext Cython

In [2]: %%cython
   ...: def f(n):
   ...:     a = 0
   ...:     for i in range(n):
   ...:         a += i
   ...:     return a
   ...:
   ...: cpdef g(int n):
   ...:     cdef long a = 0
   ...:     cdef int i
   ...:     for i in range(n):
   ...:         a += i
   ...:     return a
   ...:

In [3]: %timeit f(1000000)
10 loops, best of 3: 26.5 ms per loop

In [4]: %timeit g(1000000)
1000 loops, best of 3: 279 µs per loop
```

which gives a 95 times improvement over the pure-python version. More details on the subject in the official quickstart page.[20]

Uses

Cython is particularly popular among scientific users of Python,[12][21][22] where it has "the perfect audience" according to Python creator Guido van Rossum.[23] Of particular note:

- The free software SageMath computer algebra system depends on Cython, both for performance and to interface with other libraries.[24]
- Significant parts of the scientific computing libraries SciPy, pandas and scikit-learn are written in Cython.[25][26]
- Some high-traffic websites such as Quora use Cython.[27]

Cython's domain is not limited to just numerical computing. For example, the lxml XML toolkit is written mostly in Cython, and like its predecessor Pyrex, Cython is used to provide Python bindings for many C and C++ libraries such as the messaging library ZeroMQ.[28] Cython can also be used to develop parallel programs for multi-core processor machines; this feature makes use of the OpenMP library.

See also

- PyPy
- Numba

References

1. Behnel, Stefan (2008). "The Cython Compiler for C-Extensions in Python" (https://www.behnel.de/cythonEP2008/cython-ep2008.html). *EuroPython* (28 July 2007: official Cython launch). Vilnius/Lietuva.
2. "Release 3.1.0-1" (https://github.com/cython/cython/releases/tag/3.1.0-1). 8 May 2025. Retrieved 25 May 2025.
3. *Cython Changelog* (https://github.com/cython/cython/blob/61c079e/CHANGES.rst), cython, 15 May 2023, retrieved 19 May 2023
4. "Language Basics — Cython 3.0.0a9 documentation" (https://cython.readthedocs.io/en/latest/src/userguide/language_basics.html#cython-file-types). *cython.readthedocs.io*. Retrieved 9 September 2021.
5. "Cython - an overview — Cython 0.19.1 documentation" (https://docs.cython.org/src/quickstart/overview.html). Docs.cython.org. Retrieved 21 July 2013.
6. Smith, Kurt (2015). *Cython: A Guide for Python Programmers* (https://shop.oreilly.com/product/0636920033431.do). O'Reilly Media. ISBN 978-1-4919-0155-7.
7. "FAQ · cython/cython Wiki" (https://github.com/cython/cython). *GitHub*. Retrieved 11 January 2023.
8. "Cython Changelog" (https://cython.readthedocs.io/en/latest/src/changes.html). *cython.org*. Retrieved 21 July 2023.
9. "Basic Tutorial — Cython 3.0a6 documentation" (https://cython.readthedocs.io/en/latest/src/tutorial/cython_tutorial.html). *cython.readthedocs.io*. Retrieved 11 December 2020.
10. Oliphant, Travis (20 June 2011). "Technical Discovery: Speeding up Python (NumPy, Cython, and Weave)" (https://technicaldiscovery.blogspot.com/2011/06/speeding-up-python-numpy-cython-and.html). Technicaldiscovery.blogspot.com. Retrieved 21 July 2013.
11. Behnel, Stefan; Bradshaw, Robert; Citro, Craig; Dalcin, Lisandro; Seljebotn, Dag Sverre; Smith, Kurt (2011). "Cython: The Best of Both Worlds" (https://research.google.com/pubs/pub36727.html). *Computing in Science and Engineering*. **13** (2): 31–39. Bibcode:2011CSE....13b..31B (https://ui.adsabs.harvard.edu/abs/2011CSE....13b..31B). doi:10.1109/MCSE.2010.118 (https://doi.org/10.1109%2FMCSE.2010.118). hdl:11336/13103 (https://hdl.handle.net/11336%2F13103). S2CID 14292107 (https://api.semanticscholar.org/CorpusID:14292107).
12. Seljebot, Dag Sverre (2009). "Fast numerical computations with Cython" (http://conference.scipy.org/proceedings/SciPy2009/paper_2). *Proceedings of the 8th Python in Science Conference*. pp. 15–22. doi:10.25080/GTCA8577 (https://doi.org/10.25080%2FGTCA8577).
13. Wilbers, I.; Langtangen, H. P.; Ødegård, Å. (2009). Skallerud, B.; Andersson, H. I. (eds.). "Using Cython to Speed up Numerical Python Programs" (https://web.archive.org/web/20170104170007/http://simula.no/research/sc/publications/Simula.SC.578/simula_pdf_file). *Proceedings of MekIT'09*: 495–512. Archived from the original (http://simula.no/research/sc/publications/Simula.SC.578/simula_pdf_file) (PDF) on 4 January 2017. Retrieved 14 June 2011.

14. "wrapper benchmarks for several Python wrapper generators (except Cython)" (https://web.archive.org/web/20150404154630/http://telecom.inescporto.pt/~gjc/pybindgen-benchmarks/). Archived from the original (http://telecom.inescporto.pt/~gjc/pybindgen-benchmarks/) on 4 April 2015. Retrieved 28 May 2010.
15. "wrapper benchmarks for Cython, Boost.Python and PyBindGen" (https://web.archive.org/web/20160303224104/http://www.behnel.de/cycppbench/). Archived from the original (https://behnel.de/cycppbench/) on 3 March 2016. Retrieved 28 May 2010.
16. "Cython: C-Extensions for Python" (http://cython.org/index.html). Retrieved 22 November 2015.
17. "Differences between Cython and Pyrex" (https://github.com/cython/cython/wiki/DifferencesFromPyrex). *GitHub*.
18. Ewing, Greg (21 March 2011). "Re: VM and Language summit info for those not at Pycon (and those that are!)" (https://mail.python.org/pipermail/python-dev/2011-March/109642.html) (Message to the electronic mailing-list `python-dev`). Retrieved 5 May 2011.
19. Says Sage and Cython developer Robert Bradshaw at the Sage Days 29 conference (22 March 2011). "Cython: Past, Present and Future" (https://www.youtube.com/watch?v=osjSS2Rrvm0). Archived (https://ghostarchive.org/varchive/youtube/20211221/osjSS2Rrvm0) from the original on 21 December 2021. Retrieved 5 May 2011 – via YouTube.
20. "Building Cython code" (https://cython.readthedocs.io/en/latest/src/quickstart/build.html). *cython.readthedocs.io*. Retrieved 24 April 2017.
21. "inSCIght: The Scientific Computing Podcast" (https://web.archive.org/web/20141010032300/http://inscight.org/2011/03/31/episode_/) (Episode 6). Archived from the original (http://inscight.org/2011/03/31/episode_/) on 10 October 2014. Retrieved 29 May 2011.
22. Millman, Jarrod; Aivazis, Michael (2011). "Python for Scientists and Engineers" (https://escholarship.org/uc/item/93s2v2s7). *Computing in Science and Engineering*. **13** (2): 9–12. Bibcode:2011CSE....13b...9M (https://ui.adsabs.harvard.edu/abs/2011CSE....13b...9M). doi:10.1109/MCSE.2011.36 (https://doi.org/10.1109%2FMCSE.2011.36).
23. Guido Van Rossum (21 March 2011). "Re: VM and Language summit info for those not at Pycon (and those that are!)" (https://mail.python.org/pipermail/python-dev/2011-March/109634.html) (Message to the electronic mailing-list `python-dev`). Retrieved 5 May 2011.
24. Erocal, Burcin; Stein, William (2010). "The Sage Project: Unifying Free Mathematical Software to Create a Viable Alternative to Magma, Maple, Mathematica and MATLAB". *Mathematical Software – ICMS 2010* (https://wstein.org/papers/icms/icms_2010.pdf) (PDF). Lecture Notes in Computer Science. Vol. 6327. Springer Berlin / Heidelberg. pp. 12–27. CiteSeerX 10.1.1.172.624 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.172.624). doi:10.1007/978-3-642-15582-6_4 (https://doi.org/10.1007%2F978-3-642-15582-6_4). ISBN 978-3-642-15581-9.
25. "SciPy 0.7.2 release notes" (https://web.archive.org/web/20160304052117/http://docs.scipy.org/doc/scipy/reference/release.0.7.2.html). Archived from the original (http://docs.scipy.org/doc/scipy/reference/release.0.7.2.html) on 4 March 2016. Retrieved 29 May 2011.
26. Pedregosa, Fabian; Varoquaux, Gaël; Gramfort, Alexandre; Michel, Vincent; Thirion, Bertrand; Grisel, Olivier; Blondel, Mathieu; Prettenhofer, Peter; Weiss, Ron; Dubourg, Vincent; Vanderplas, Jake; Passos, Alexandre; Cournapeau, David (2011). "Scikit-learn: Machine Learning in Python". *Journal of Machine Learning Research*. **12**: 2825–2830. arXiv:1201.0490 (https://arxiv.org/abs/1201.0490). Bibcode:2011JMLR...12.2825P (https://ui.adsabs.harvard.edu/abs/2011JMLR...12.2825P).
27. "Is Quora still running on PyPy?" (https://www.quora.com/Is-Quora-still-running-on-PyPy/answer/Alex-Yakunin).
28. "ØMQ: Python binding" (https://www.zeromq.org/bindings:python).

External links

- Official website (https://cython.org)
- Cython (https://github.com/cython) on GitHub

MicroPython

MicroPython is a software implementation of a programming language largely compatible with Python 3, written in C, that is optimized to run on a microcontroller.[2][3]

MicroPython consists of a Python compiler to bytecode and a runtime interpreter of that bytecode. The user is presented with an interactive prompt (the REPL) to execute supported commands immediately. Included are a selection of core Python libraries; MicroPython includes modules which give the programmer access to low-level hardware.[4]

MicroPython does have an inline assembler, which lets the code run at full speed, but it is not portable across different microcontrollers.

The source code for the project is available on GitHub under the MIT License.[5]

MicroPython

Developer(s)	Damien P. George
Initial release	3 May 2014
Stable release	1.25.0 / 15 April 2025
Repository	github.com/micropython/micropython (https://github.com/micropython/micropython)
Written in	C
Platform	ARM Cortex-M, STM32, ESP8266, ESP32, 16-bit PIC, Unix, Microsoft Windows, Zephyr, JavaScript, RP2040
License	MIT license[1]
Website	micropython.org (https://micropython.org)

History

MicroPython was originally created by the Australian programmer Damien George, after a successful Kickstarter-backed campaign in 2013.[6] The original Kickstarter campaign released MicroPython with an STM32F4-powered development board "pyboard". In the meantime MicroPython has been developed to support a number of ARM based architectures.[7] The ports supported in the mainline are ARM Cortex-M (many STM32[8] boards, RP2040 boards, TI CC3200/WiPy, Teensy boards, Nordic nRF series, SAMD21 and SAMD51), ESP8266, ESP32,[9] 16-bit PIC, Unix, Windows, Zephyr, and JavaScript.[10] Also, there are many forks for a variety of systems and hardware platforms not supported in the mainline.[11]

In 2016, a version of MicroPython for the BBC Micro Bit was created as part of the Python Software Foundation's contribution to the Micro Bit partnership with the BBC.[12]

In July 2017, MicroPython was forked to create CircuitPython, a version of MicroPython with emphasis on education and ease of use. MicroPython and CircuitPython support somewhat different sets of hardware (e.g. CircuitPython supports Atmel SAM D21 and D51 boards, but dropped support for ESP8266). As of version 4.0, CircuitPython is based on MicroPython version 1.9.4.[13]

In 2017, Microsemi made a MicroPython port for RISC-V (RV32 and RV64) architecture.[14]

In April 2019, a version of MicroPython for the Lego Mindstorms EV3 was created.[15]

In January 2021, a MicroPython port for the RP2040 (ARM Cortex-M0+, on Raspberry Pi Pico and others) was created.[16]

A Casio FX-9860GIII calculator which was introduced in 2020, and came with built-in MicroPython

Features

Ability to run Python

MicroPython has the ability to run Python, allowing users to create simple and easy-to-understand programs.[17] MicroPython supports many standard Python libraries, supporting more than 80% of the features of Python's most used libraries.[17] MicroPython was designed specifically to support the typical performance gap between microcontrollers and Python.[18] Python code is able to directly access and interact with hardware, with increased hardware possibilities that are not available using a normal Python application that is run on an operating system.[19]

Code portability

MicroPython's utilisation of hardware abstraction layer (HAL) technology allows developed code to be portable among different microcontrollers within the same family or platform and on devices that support and can download MicroPython. Programs are often developed and tested on high-performance microcontrollers and distributed with the final application used on lower-performance microcontrollers.[20]

Modules

MicroPython offers functionality, once new code has been written, to create a frozen module and use it as a library which can be a part of developed firmware. This feature assists with avoiding repetitive downloading of the same, already error-free, tested code into a MicroPython environment. This type of module will be saved to a microcontroller's modules directory for compiling and uploading to the microcontroller where the library will be available using Python's import command to be used repeatedly.[20]

Read–eval–print loop

The read–eval–print loop (REPL) allows a developer to enter individual lines of code and have them run immediately on a terminal.[21] Linux-based and macOS systems have terminal emulators that can be used to create a direct connection to a MicroPython device's REPL using a serial USB connection. The REPL

assists with the immediate testing of parts of an application as each part of the code can be run and the results visually examined. Once different parts of code are loaded into the REPL, additional REPL features can be used to experiment with that code's functionality.[17]

Helpful REPL commands (once connected to a serial console):[21]

- CTRL + C: keyboard interrupt
- CTRL + D: reload
- `help()`: help message
- `help("modules")`: lists built-in modules
- `import board` ↵ Enter `dir(board)`: lists all the pins on your microcontroller board that are available to be used in a program's code

Limitations

Although MicroPython fully implements Python language version 3.4 and much of 3.5, it does not implement all language features introduced from 3.5 onwards,[22] though some new syntax from 3.6 and more recent features from later versions, e.g. from 3.8 (assignment expressions) and 3.9. It includes a subset of the standard library.[23]

MicroPython has more limited hardware support in the microcontroller market than other popular platforms, like Arduino with a smaller number of microcontroller choices that support the language.[18] MicroPython does not include an integrated development environment (IDE) or specific editor unlike other platforms.[18]

Syntax and semantics

MicroPython's syntax is adopted from Python, due to its clear and easy-to-understand style and power.[24] Unlike most other programming languages less punctuation is used with fewer syntactical machinations in order to prioritise readability.[17]

Code blocks

MicroPython adopts Python's code block style, with code specific to a particular function, condition or loop being indented.[17] This differs from most other languages which typically use symbols or keywords to delimit blocks.[17] This assists with the readability of MicroPython code as the visual structure mirrors the semantic structure. This key feature is simple but important as misused indentation can result in code executing under a wrong condition or an overall error from the interpreter.[17]

A colon (:) is the key symbol used to indicate the ending of a condition statement.[17] The indent size is equivalent to one tab or 4 spaces.

Operations

MicroPython has the ability to perform various mathematical operations using primitive and logical operations.[19]

Supported operations[19]

Type	Operator	Name	Example
Arithmetic	`+`	Addition	`variable + 1`
	`-`	Subtraction	`variable - 1`
	`*`	Multiplication	`variable * 4`
	`/`	Division	`variable / 4`
	`%`	Modulo division	`variable % 4`
Comparison	`==`	Equals	`expression1 == expression2`
	`!=`	Not equal	`expression1 != expression2`
	`<`	Less than	`expression1 < expression2`
	`>`	Greater than	`expression1 > expression2`
	`<=`	Less than or equals	`expression1 <= expression2`
	`>=`	Greater than or equals	`expression1 >= expression2`
Logical	`&`	bitwise and	`variable1 & variable2`
	`|`	bitwise or	`variable1 | variable2`
	`^`	bitwise exclusive or	`variable1 ^ variable2`
	`~`	bitwise complement	`~variable1`
	`and`	logical and	`variable1 and variable2`
	`or`	logical or	`variable1 or variable2`

Libraries

MicroPython is a lean and efficient implementation of Python with libraries similar to those in Python.[25] Some standard Python libraries have an equivalent library in MicroPython renamed to distinguish between the two. MicroPython libraries are smaller with less popular features removed or modified to save memory.[19]

The three types of libraries in MicroPython:[19]

- derived from a standard Python library (built-in libraries)
- specific MicroPython libraries
- specific libraries to assist with hardware functionality

MicroPython is highly customisable and configurable, with language differing between each board (microcontroller) and the availability of libraries may differ. Some functions and classes in a module or the entire module may be unavailable or altered.[19]

Standard Python libraries in MicroPython[4]

Library name	Description
array	operations on arrays
cmath	provides math functions for complex numbers
gc	garbage collector
math	provides basic math operations for floating-point numbers
sys	system-level functions; provides access to variables used by the interpreter
binascii	functions for converting between binary and ASCII
collections	operations for collections and container types that hold various objects
errno	provides access to error codes
hashlib	operations for binary hash algorithms
heapq	operations to implement heap queue algorithm
io	operations for handling input/output streams
json	handles conversion between JSON documents and Python objects
os	functions for filesystem access and basic operating system functions
re	implements regular expression matching operations
select	functions for handling events on multiple streams
socket	connecting to sockets (networks), providing access to socket interface
struct	performs conversions to Python objects by packing and unpacking primitive data types
time	provides time and date function, including measuring time intervals and implementing delays
zlib	operations to decompress binary data

MicroPython-specific libraries[4]

Library name	Description
framebuf	provides a frame buffer that can be used to create bitmap images to be sent to a display
machine	functions assisting with accessing and interacting with hardware blocks
micropython	access and control of MicroPython internals
network	assists with installing network driver, allowing interactions through networks
ctypes	access binary data structures

Custom MicroPython libraries

When developers begin to create a new application, standard MicroPython libraries and drivers may not meet the requirements, with insufficient operations or calculations. Similar to Python, there is the possibility of extending MicroPython's functionality with custom libraries which extend the ability of the existing libraries and firmware.[20]

In MicroPython, files ending with .py take preference over other library aliases which allows users to extend the use and implementation of the existing libraries.[19]

Supporting hardware

As MicroPython's implementation and popularity continues to grow, more boards have the ability to run MicroPython. Many developers are building processor specific versions that can be downloaded onto different microcontrollers.[19] Installing MicroPython on microcontrollers is well documented and user-friendly.[20] MicroPython allows interactions between microcontroller hardware and applications to be simple, allowing access to a range of functionality while working in a resource constrained environment, with a strong level of responsiveness.[17]

The two types of boards used to run MicroPython:[19]

- MicroPython loaded when manufactured, meaning only MicroPython can be run.
- boards that have firmware that allows MicroPython to be installed to the board.

Executing code

To move a program onto a MicroPython board, create a file and copy it onto the microcontroller in order to execute. With the hardware connected to a device, such as a computer, the board's flash drive will appear on the device allowing files to be moved to the flash drive. There will be two existing python files, boot.py and main.py that are typically not modified, main.py may be modified if you wish to run the program every time the microcontroller is booted, otherwise, programs will be run using the REPL console.[19]

Pyboard

The pyboard is the official MicroPython microcontroller board which fully supports MicroPython's software features. The pyboard's hardware features include:[4]

- microcontroller (MCU consisting of CPU, flash ROM and RAM)
- microUSB connector
- micro-SD card slot
- IO pins
- switches, LEDs, servo ports, real time clock, accelerometer

The booting process

The pyboard contains an internal drive with filesystem named `/flash` which is stored within the board's flash memory, additionally, a microSD card can be inserted into a slot and is accessible through `/sd`. When booted up, a pyboard must select a filesystem to boot from either `/flash` or `/sd` with the current directory being set to either `/flash` or `/sd`. By default, if an SD card is inserted, `/sd` will be used, if not, `/flash` is used. If needed, the use of the SD card for the booting process can be avoided by creating an empty file called `/flash/SKIPSD` which is stored on the board flash memory to skip the SD card for the booting process.[4]

Boot modes

When the pyboard is powered up normally or the reset button is pressed then the pyboard is booted in a standard mode, meaning that the boot.py file is executed, then the USB configured and finally the python program will run.[4]

There is an ability to override the standard boot sequence through holding down the user switch whilst the board is in the booting process and then pressing reset as you continue to hold the user switch. The pyboard's LEDs will flick between modes and once the LEDs have reached the mode wanted by the user, they can let go of the user switch and the board will boot in the specific mode.[4]

the boot modes are:[4]

- standard boot: green LED only (runs boot.py then python program)
- safe boot: orange LED only (does not run any scripts during boot-up)
- filesystem reset: green and orange LED together (resets flash drive to factory state and boots in safe mode) – used as a fix when filesystem is corrupted

Errors

- if red and green LEDs flash alternatively then the python script has an error, and you must use the REPL to debug.
- if all 4 LEDs cycle on and off then there is a hard fault which cannot be recovered from and requires a hard reset.[4]

Programming examples

Source:[19]

Hello world program:

```
# Print to serial console

print("Hello, World!")
```

Importing + turning on a LED:

```
import pyb

# Turn LED on

pyb.LED(1).on()
```

Reading a file + loop:

```
import os

# Open and read a file

with open("/readme.txt") as f:
    print(f.read())
```

Bytecode

MicroPython includes a cross compiler which generates MicroPython bytecode (file extension *.mpy*). The Python code can be compiled into the bytecode either directly on a microcontroller or it can be precompiled elsewhere.

MicroPython firmware can be built without the compiler, leaving only the virtual machine which can run the precompiled *mpy* programs.

Implementation and uses

MicroPython is utilised through firmware being loaded by standard software onto a particular microcontroller into flash memory, communicating using a terminal application loaded onto a computer that emulates a serial interface.[20]

The main uses of MicroPython can be generalised into 3 categories:[20]

- **educational purposes**: using MicroPython's read–eval–print Loop (REPL) to interact with a microcontroller, it is possible to visually explain the concepts of data processing and communicating with boards in a simpler way than more complicated programming languages.
- **developing and testing device and sensor designs**: MicroPython offers verified, bug-free, and thoroughly tested reference implementations of interfaces used in microcontrollers solving a common developer's task of implementing peripheral communication setup and control. MicroPython offers direct and interactive accessibility to device registers which makes it easy to verify functionality and develop and test hardware parts and devices and algorithms for control and acquiring data from a device.
- **monitoring and configuring tool for design of complex applications**: certain applications require specific applications on high performing microcontrollers. MicroPython is able to assist with state monitoring and set-up of system parameters.

Implementation of MicroPython can differ depending on the availability of standard and supporting libraries and the microcontroller's flash memory and RAM size.[20]

See also

- Espruino
- CircuitPython

References

1. George, Damien P. (4 May 2014). "micropython/LICENSE at master · micropython/micropython" (https://github.com/micropython/micropython/blob/master/LICENSE). *GitHub*. Retrieved 11 February 2017.
2. Venkataramanan, Madhumita (6 December 2013). "Micro Python: more powerful than Arduino, simpler than the Raspberry Pi" (https://www.wired.co.uk/article/micro-python). *Wired*. Retrieved 15 December 2016.

3. Yegulalp, Serdar (5 July 2014). "Micro Python's tiny circuits: Python variant targets microcontrollers" (http://www.infoworld.com/article/2608012/python/micro-python-s-tiny-circuits--python-variant-targets-microcontrollers.html). InfoWorld. Retrieved 15 December 2016.
4. "MicroPython - Python for microcontrollers" (https://micropython.org/). *micropython.org*. Retrieved 12 August 2017.
5. "MicroPython on GitHub" (https://github.com/micropython/micropython). *GitHub*. 7 February 2022.
6. "Micro Python: Python for microcontrollers" (https://www.kickstarter.com/projects/214379695/micro-python-python-for-microcontrollers). *Kickstarter*. 26 February 2016. Retrieved 15 December 2016.
7. Beningo, Jacob (11 July 2016). "Prototype to production: MicroPython under the hood" (http://www.edn.com/electronics-blogs/embedded-basics/4442357/Prototype-to-production---MicroPython-under-the-hood). EDN Network. Retrieved 15 December 2016.
8. "MicroPython on Nucleo STM32, STM32F411CE, and STM32F401CC: flashing firmware and basic tools" (https://mischianti.org/micropython-on-nucleo-stm32-stm32f411ce-and-stm32f401cc-flashing-firmware-and-basic-tools/). *Mischianti*. August 2023.
9. "MicroPython with esp8266 and esp32: flashing firmware and programming with basic tools" (https://mischianti.org/micropython-with-esp8266-and-esp32-flashing-firmware-and-programming-with-basic-tools-1/). *Mischianti*. 7 June 2023.
10. George, Damien P. "micropython/ports at master · micropython/micropython" (https://github.com/micropython/micropython/tree/master/ports). *GitHub*. Retrieved 22 October 2019.
11. Sokolovsky, Paul. "Awesome MicroPython" (https://github.com/pfalcon/awesome-micropython#forks-and-variants). *GitHub*. Retrieved 22 October 2019.
12. Williams, Alun (7 July 2015). "Hands on with the BBC Micro-Bit user interface" (http://www.electronicsweekly.com/news/design/embedded-systems/video-bbc-micro-bit-user-interface-2015-07/). *ElectronicsWeekly.com*. Retrieved 8 July 2015.
13. Shawcroft, Scott (22 May 2019). "CircuitPython 4.0.1 released!" (https://blog.adafruit.com/2019/05/22/circuitpython-4-0-1-released/). *Adafruit Blog*. Adafruit Industries. Retrieved 11 June 2019.
14. "RISC-V Poster Preview — 7th RISC-V Workshop" (https://content.riscv.org/wp-content/uploads/2017/12/RISC-V-Poster-Preview.pdf) (PDF). 28 November 2017. Retrieved 17 December 2018.
15. "LEGO releases MicroPython for EV3 based on ev3dev and Pybricks" (https://www.ev3dev.org/news/2019/04/13/ev3-micropython/). *www.ev3dev.org*. Retrieved 21 April 2020.
16. "Meet Raspberry Silicon: Raspberry Pi Pico now on sale at $4" (https://www.raspberrypi.org/blog/raspberry-pi-silicon-pico-now-on-sale/). *www.raspberrypi.org*. 21 January 2021. Retrieved 21 January 2021.
17. Alsabbagh, Marwan (2019). *MicroPython Cookbook*. Birmingham, UK: Packt Publishing.
18. Bruno, P. (25 November 2021). "An Introduction to MicroPython" (https://all3dp.com/2/micropython-beginner-s-guide/). *All3DP*. Retrieved 9 May 2022.
19. Bell, Charles (2017). *MicroPython for the Internet of Things*. Berkeley, USA: Apress.
20. Gaspar, G.; Kuba, P.; Flochova, J.; Dudak, J.; Florkova, Z. (2020). *Development of IoT applications based on the MicroPython platform for Industry 4.0 implementation*. 2020 19th International Conference on Mechatronics – Mechatronika (ME). pp. 1–7.
21. Rembor, K. "The REPL" (http://learn.adafruit.com/welcome-to-circuitpython/the-repl). *Welcome to CircuitPython!*. Adafruit Learning System. Retrieved 9 May 2022.
22. "MicroPython differences from CPython — MicroPython latest documentation" (https://docs.micropython.org/en/latest/genrst/index.html). *docs.micropython.org*.
23. "MicroPython - Python for microcontrollers" (https://www.micropython.org). *micropython.org*.

24. Wang, L.; Li, Y.; Zhang, H.; Han, Q.; Chen, L. (2021). *An Efficient Control-flow based Obfuscator for Micropython Bytecode*. 2021 7th International Symposium on System and Software Reliability (ISSSR). pp. 54–63.
25. Khamphroo, M.; Kwankeo, N.; Kaemarungsi, K.; Fukawa, K. (2017). *MicroPython-based educational mobile robot for computer coding learning*. 2017 8th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES). pp. 1–6.

External links

- Official website (https://micropython.org)
- micropython (https://github.com/micropython/micropython) on GitHub
- GOTO 2016 • MicroPython & the Internet of Things • Damien George (https://www.youtube.com/watch?v=EvGhPmPPzko&t=938s) on YouTube
- MicroPython (https://www.youtube.com/playlist?list=PLjF7R1fz_OOXrI15wuXeESA0aA4VzcWSi) playlist on YouTube • Tutorials by Tony DiCola / Adafruit

Numba

Numba is an open-source JIT compiler that translates a subset of Python and NumPy into fast machine code using LLVM, via the llvmlite Python package. It offers a range of options for parallelising Python code for CPUs and GPUs, often with only minor code changes.

Numba was started by Travis Oliphant in 2012 and has since been under active development at its repository in GitHub (https://github.com/numba/numba) with frequent releases. The project is driven by developers at Anaconda, Inc., with support by DARPA, the Gordon and Betty Moore Foundation, Intel, Nvidia and AMD, and a community of contributors on GitHub.

Example

Numba can be used by simply applying the `numba.jit` decorator to a Python function that does numerical computations:

```
import numba
import random

@numba.jit
def monte_carlo_pi(n_samples: int) -> float:
    """Monte Carlo"""
    acc = 0
    for i in range(n_samples):
        x = random.random()
        y = random.random()
        if (x**2 + y**2) < 1.0:
            acc += 1
    return 4.0 * acc / n_samples
```

The just-in-time compilation happens transparently when the function is called:

```
>>> monte_carlo_pi(1000000)
3.14
```

Numba's website (https://numba.pydata.org) contains many more examples, as well as information on how to get good performance from Numba.

Numba logo

Original author(s)	Continuum Analytics
Developer(s)	Community project
Initial release	15 August 2012
Stable release	0.61.2[1] / 9 April 2025
Repository	github.com/numba/numba (https://github.com/numba/numba)
Written in	Python, C
Operating system	Cross-platform
Platform	x86-64, ARM64, POWER
Type	Technical computing
License	BSD 2-clause
Website	numba.pydata.org (http://numba.pydata.org/)

Numba CUDA

Developer(s)	NVIDIA
Stable release	0.4.0 / January 27, 2025[2]
Repository	github.com/NVIDIA/numba-cuda (https://github.com/NVIDIA/numba-cuda)
Platform	NVIDIA GPU
License	BSD 2-clause

Website	nvidia.github.io/numba-cuda/ (https://nvidia.github.io/numba-cuda/)

GPU support

Numba can compile Python functions to GPU code. Initially two backends are available:

- NVIDIA CUDA, see numba.readthedocs.io/en/stable/cuda/index.html (https://numba.readthedocs.io/en/stable/cuda/index.html)
- AMD ROCm HSA, see numba.pydata.org/numba-doc/dev/roc (https://numba.pydata.org/numba-doc/dev/roc)

Since release 0.56.4,[3] AMD ROCm HSA has been officially moved to unmaintained status and a separate repository stub (https://github.com/numba/numba-rocm) has been created for it.

Alternative approaches

Numba is one approach to make Python fast, by compiling specific functions that contain Python and NumPy code. Many alternative approaches for fast numeric computing with Python exist, such as Cython, Pythran (https://pythran.readthedocs.io/en/latest/), and PyPy.

References

1. "Release Numba 0.61.2" (https://github.com/numba/numba/releases/tag/0.61.2). 9 April 2025. Retrieved 24 April 2025.
2. "Tags · NVIDIA/numba-cuda" (https://github.com/NVIDIA/numba-cuda/tags). Retrieved February 16, 2025.
3. "Release Notes — Numba 0.56.4+0.g288a38bbd.dirty-py3.7-linux-x86_64.egg documentation" (https://numba.readthedocs.io/en/stable/release-notes.html#version-0-54-0-19-august-2021).

IronPython

IronPython is an implementation of the Python programming language targeting the .NET and Mono frameworks. The project is currently maintained by a group of volunteers at GitHub. It is free and open-source software, and can be implemented with Python Tools for Visual Studio, which is a free and open-source extension for Microsoft's Visual Studio IDE.[2][3]

IronPython is written entirely in C#, although some of its code is automatically generated by a code generator written in Python.

IronPython is implemented on top of the Dynamic Language Runtime (DLR), a library running on top of the Common Language Infrastructure that provides dynamic typing and dynamic method dispatch, among other things, for dynamic languages.[4] The DLR is part of the .NET Framework 4.0 and is also a part of Mono since version 2.4 from 2009.[5] The DLR can also be used as a library on older CLI implementations.

IronPython

Original author(s)	Jim Hugunin, Microsoft
Developer(s)	Dino Viehland, .NET Foundation
Initial release	September 5, 2006[1]
Stable release	3.4.2 / December 20, 2024
Preview release	3.4.0-beta1 / April 30, 2022
Repository	github.com/IronLanguages/ironpython3 (https://github.com/IronLanguages/ironpython3)
Written in	C#
Operating system	Windows, Linux, macOS
Platform	.NET Framework, .NET, Mono
Type	Python programming language implementation
License	Apache License 2.0
Website	ironpython.net (https://ironpython.net)

Status and roadmap

Jim Hugunin created the project and actively contributed to it up until Version 1.0 which was released on September 5, 2006.[6] IronPython 2.0 was released on December 10, 2008.[7] After version 1.0 it was maintained by a small team at Microsoft until the 2.7 Beta 1 release. Microsoft abandoned IronPython (and its sister project IronRuby) in late 2010, after which Hugunin left to work at Google.[8] The project is currently maintained by a group of volunteers at GitHub.

- Release 2.0, released on December 10, 2008, and updated as 2.0.3 on October 23, 2009, targets CPython 2.5.[9] IronPython 2.0.3 is only compatible up to .NET Framework 3.5.
- Release 2.6, released on December 11, 2009, and updated on April 12, 2010, targets CPython 2.6.[10] IronPython 2.6.1 versions is binary compatible only with .NET Framework 4.0. IronPython 2.6.1 must be compiled from sources to run on .NET Framework 3.5. IronPython 2.6.2, released on October 21, 2010, is binary compatible with both .NET Framework 4.0 and .NET Framework 3.5.
- Release 2.7 was released on March 12, 2011 and it targets CPython 2.7.[11]

- Release 2.7.1 was released on October 21, 2011 and it targets CPython 2.7.[12]
- Release 2.7.2.1 was released on March 13, 2012. It enables support for ZIP file format libraries, SQLite, and compiled executables.[13]
- Release 2.7.4 was released on September 7, 2013.[14]
- Release 2.7.5 was released on December 6, 2014 and mostly consists of bug fixes.[15]
- Release 2.7.6 was released on August 21, 2016 and only consists of bug fixes.[16]
- Release 2.7.7 was released on December 7, 2016 and only consists of bug fixes.[17]
- Release 2.7.8 was released on February 16, 2018 and consists of bug fixes, reorganized code, and an updated test infrastructure (including significant testing on Linux under Mono). It is also the first release to support .NET Core.[18]
- Release 2.7.9 was released on October 9, 2018 and consists of bug fixes, reorganized code. It is intended to be the last release before IronPython 3.[19]
- Release 2.7.10 was released on April 27, 2020 and adds .NET Core 3.1 support.[20]
- Release 2.7.11 was released on November 17, 2020 and resolves issues when running on .NET 5.
- Release 2.7.12 was released on January 21, 2022 and resolves issues with .NET 6 and removes support for .NET core 2.1[21]
- Release 3.4.0 was released on December 12, 2022 and is the first release to support Python 3.x. [22]

Differences with CPython

There are some differences between the Python reference implementation CPython and IronPython.[23] Some projects built on top of IronPython are known not to work under CPython.[24] Conversely, CPython applications that depend on extensions to the language that are implemented in C are not compatible with IronPython ,[25] unless they are implemented in a .NET interop. For example, NumPy was wrapped by Microsoft in 2011, allowing code and libraries dependent on it to be run directly from .NET Framework.[26]

Silverlight

IronPython is supported on Silverlight (which is deprecated by Microsoft and already has lost support in most web browsers[27]). It can be used as a scripting engine in the browser just like the JavaScript engine.[28] IronPython scripts are passed like simple client-side JavaScript scripts in `<script>`-tags. It is then also possible to modify embedded XAML markup.

The technology behind this is called Gestalt.

```
// DLR initialization script.
<script src="http://gestalt.ironpython.net/dlr-latest.js" type="text/javascript"></script>

// Client-side script passed to IronPython and Silverlight.
<script type="text/python">
    window.Alert("Hello from Python")
</script>
```

The same works for IronRuby.

License

Until version 0.6, IronPython was released under the terms of Common Public License.[29] Following recruitment of the project lead in August 2004, IronPython was made available as part of Microsoft's Shared Source initiative. This license is not OSI-approved but the authors claim it meets the open-source definition.[30] With the 2.0 alpha release, the license was changed to the Microsoft Public License,[31] which the OSI has approved. The latest versions are released under the terms of the Apache License 2.0.

Interface extensibility

One of IronPython's key advantages is in its function as an extensibility layer to application frameworks written in a .NET language. It is relatively simple to integrate an IronPython interpreter into an existing .NET application framework. Once in place, downstream developers can use scripts written in IronPython that interact with .NET objects in the framework, thereby extending the functionality in the framework's interface, without having to change any of the framework's code base.[32]

IronPython makes extensive use of reflection. When passed in a reference to a .NET object, it will automatically import the types and methods available to that object. This results in a highly intuitive experience when working with .NET objects from within an IronPython script.

Examples

The following IronPython script manipulates .NET Framework objects. This script can be supplied by a third-party client-side application developer and passed into the server-side framework through an interface. Note that neither the interface, nor the server-side code is modified to support the analytics required by the client application.

```
from BookService import BookDictionary

booksWrittenByBookerPrizeWinners = [book.Title for book in BookDictionary.GetAllBooks()
                                    if "Booker Prize" in book.Author.MajorAwards]
```

In this case, assume that the .NET Framework implements a class, **BookDictionary**, in a module called **BookService**, and publishes an interface into which IronPython scripts can be sent and executed.

This script, when sent to that interface, will iterate over the entire list of books maintained by the framework, and pick out those written by Booker Prize-winning authors.

What's interesting is that the responsibility for writing the actual analytics reside with the client-side developer. The demands on the server-side developer are minimal, essentially just providing access to the data maintained by the server. This design pattern greatly simplifies the deployment and maintenance of complex application frameworks.

The following script uses the .NET Framework to create a simple Hello World message.

```
import clr
clr.AddReference("System.Windows.Forms")

from System.Windows.Forms import MessageBox
MessageBox.Show("Hello World")
```

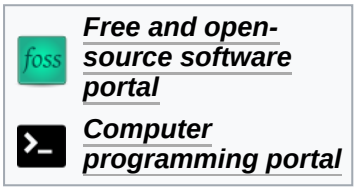
Performance

The performance characteristics of IronPython compared to CPython, the reference implementation of Python, depends on the exact benchmark used. IronPython performs worse than CPython on most benchmarks taken with the PyStone script but better on other benchmarks.[33] IronPython may perform better in Python programs that use threads or multiple cores, as it has a JIT compiler, and also because it doesn't have the Global Interpreter Lock.[34][35]

Overview and key features

1. **Integration with .NET**: IronPython allows you to use .NET libraries and frameworks directly in your Python code. This means you can leverage the extensive .NET ecosystem and access features that are specific to .NET environments.
2. **Dynamic Language Runtime (DLR)**: IronPython runs on the Dynamic Language Runtime, which is a set of services that support dynamic typing and dynamic method invocation in .NET languages.
3. **Interoperability**: You can call .NET code from IronPython and vice versa. This makes it possible to integrate Python scripts with existing .NET applications or use .NET components within Python projects.
4. **Syntax and Semantics**: IronPython aims to be as close as possible to the standard Python language (CPython), though there might be minor differences due to the underlying .NET platform.
5. **Performance**: While IronPython provides good performance for many applications, it might not be as fast as CPython for some tasks, particularly those that rely heavily on Python's native libraries.
6. **Compatibility**: IronPython is compatible with Python 2.x, but it does not support Python 3.x features. This means that some newer Python libraries or syntax may not be available.

See also



- Boo – a language for the .NET Framework and Mono with Python-inspired syntax and features borrowed from C# and Ruby
- Cobra
- IronScheme
- Jython – an implementation of Python for the Java Virtual Machine
- Cython

- PyPy – a self-hosting interpreter for the Python programming language
- Tao Framework
- Unladen Swallow – A (now-defunct) branch of CPython that aimed to provide superior performance using an LLVM-based just-in-time compiler

References

1. "CodePlex Archive" (https://web.archive.org/web/20171226082609/http://ironpython.codeplex.com/releases/view/423). Archived from the original (http://ironpython.codeplex.com/releases/view/423) on 2017-12-26. Retrieved 2014-05-30.
2. "IronPython.net" (http://www.ironpython.net). Retrieved 2013-07-03.
3. "Python Tools for Visual Studio- Home" (https://web.archive.org/web/20180126035502/http://pytools.codeplex.com/). *Python Tools for Visual Studio*. Archived from the original (http://pytools.codeplex.com/) on 2018-01-26. Retrieved 2013-07-03.
4. "Dynamic Language Runtime Overview" (http://msdn.microsoft.com/en-us/library/dd233052.aspx). Microsoft. Retrieved 2014-04-01.
5. "2009-07-02 Marek Safar · mono/Mono@340222f" (https://github.com/mono/mono/commit/340222ffe8b958cd22d9eb0388488f326845b363). *GitHub*.
6. "Jim Hugunin's blog: IronPython 1.0 released today!" (http://blogs.msdn.com/hugunin/archive/2006/09/05/741605.aspx). 2006-09-05. Retrieved 2006-12-14.
7. "Release dates for ironpython" (http://www.codeplex.com/IronPython/Release/ProjectReleases.aspx?ReleaseId=8365). 2008-12-10. Retrieved 2009-01-25.
8. Clarke, Gavin (2010-10-22). "Microsoft cuts loose Iron languages" (https://www.theregister.co.uk/2010/10/22/microsoft_kills_dynamic_languages_projects/). The Register. Retrieved 2012-04-05.
9. "2.0.3" (https://web.archive.org/web/20171226082603/http://ironpython.codeplex.com/releases/view/30416). ironpython.codeplex.com. Archived from the original (http://ironpython.codeplex.com/releases/view/30416) on 2017-12-26. Retrieved 2010-10-16.
10. "2.6" (https://web.archive.org/web/20180113101734/http://ironpython.codeplex.com/releases/view/36280). ironpython.codeplex.com. Archived from the original (http://ironpython.codeplex.com/releases/view/36280) on 2018-01-13. Retrieved 2010-10-16.
11. "2.7" (https://web.archive.org/web/20180102233741/http://ironpython.codeplex.com/releases/view/54498). ironpython.codeplex.com. Archived from the original (http://ironpython.codeplex.com/releases/view/54498) on 2018-01-02. Retrieved 2011-03-12.
12. "2.7.1" (https://web.archive.org/web/20171226084555/https://ironpython.codeplex.com/releases/view/62475). ironpython.codeplex.com. Archived from the original (http://ironpython.codeplex.com/releases/view/62475) on 2017-12-26. Retrieved 2011-12-30.
13. "2.7.2.1" (https://web.archive.org/web/20171226082357/http://ironpython.codeplex.com/releases/view/74478). ironpython.codeplex.com. Archived from the original (http://ironpython.codeplex.com/releases/view/74478) on 2017-12-26. Retrieved 2012-03-24.
14. "2.7.4" (https://web.archive.org/web/20180116003231/http://ironpython.codeplex.com/releases/view/90087). ironpython.codeplex.com. Archived from the original (http://ironpython.codeplex.com/releases/view/90087) on 2018-01-16. Retrieved 2014-12-07.
15. "2.7.5" (https://web.archive.org/web/20180126040129/http://ironpython.codeplex.com/releases/view/169382). ironpython.codeplex.com. Archived from the original (http://ironpython.codeplex.com/releases/view/169382) on 2018-01-26. Retrieved 2014-12-07.
16. "2.7.6" (https://github.com/IronLanguages/main/releases/tag/ipy-2.7.6.3). github.com. Retrieved 2016-08-21.
17. "2.7.7" (https://github.com/IronLanguages/main/releases/tag/ipy-2.7.7). github.com. Retrieved 2018-01-05.

18. "2.7.8" (https://github.com/IronLanguages/ironpython2/tree/ipy-2.7.8). github.com. Retrieved 2018-01-05.
19. "2.7.9" (https://github.com/IronLanguages/ironpython2/tree/ipy-2.7.9). github.com. Retrieved 2018-10-09.
20. "IronLanguages/ironpython2" (https://github.com/IronLanguages/ironpython2). *GitHub*. Retrieved 2020-06-26.
21. "Releases · IronLanguages/ironpython2" (https://github.com/IronLanguages/ironpython2/releases). *GitHub*. Retrieved 2022-08-08.
22. "Releases · IronLanguages/ironpython3" (https://github.com/IronLanguages/ironpython3/releases). *GitHub*. Retrieved 2023-07-09.
23. "Differences between IronPython 1.0 and CPython 2.4.3" (http://www.codeplex.com/IronPython/Wiki/View.aspx?title=Differences&referringTitle=Home). Microsoft. 2007-12-18. Retrieved 2008-02-09.
24. Foord, Michael. "New Project: Implementing .NET Libraries in Pure Python" (https://web.archive.org/web/20080830011024/http://lists.ironpython.com/pipermail/users-ironpython.com/2008-January/006297.html). Archived from the original (http://lists.ironpython.com/pipermail/users-ironpython.com/2008-January/006297.html) on 2008-08-30. Retrieved 2008-02-09.
25. Eby, Phillip (15 October 2005). "Children of a Lesser Python" (http://dirtsimple.org/2005/10/children-of-lesser-python.html). Retrieved 2008-07-09.
26. "NumPy and SciPy for .NET" (https://www.infoq.com/news/2011/07/NumPy-NET). Retrieved 2019-04-05.
27. "Silverlight 5 System Requirements" (https://www.microsoft.com/getsilverlight/locale/en-us/html/installation-win-SL5.html). *www.microsoft.com*. Retrieved 2019-11-16.
28. "Write browser applications in Python" (https://web.archive.org/web/20130317012431/http://ironpython.net/browser/). *IronPython.net*. Archived from the original (http://ironpython.net/browser/) on 2013-03-17.
29. "Original IronPython homepage" (https://web.archive.org/web/20100223101738/http://www.ironpython.com/old.html). 2004-07-28. Archived from the original (http://www.ironpython.com/old.html) on February 23, 2010. Retrieved 2007-05-13.
30. "Shared Source License for IronPython" (http://www.codeplex.com/IronPython/Project/License.aspx?LicenseHistoryId=129). 2006-04-28. Retrieved 2007-05-13.
31. "Microsoft permissive license" (http://www.codeplex.com/IronPython/Project/License.aspx?LicenseHistoryId=2866). 2007-04-28. Retrieved 2007-05-13.
32. "Using .NET objects from IronPython in Resolver One" (https://web.archive.org/web/20090114131717/http://www.resolversystems.com/documentation/index.php/Dot_Net_Objects_in_the_Grid). Archived from the original (http://www.resolversystems.com/documentation/index.php/Dot_Net_Objects_in_the_Grid) on 2009-01-14. Retrieved 2008-11-18.
33. "IronPython Performance Report" (https://archive.today/20130119180607/http://ironpython.codeplex.com/Wiki/View.aspx?title=IP26RC1VsCPy26Perf&referringTitle=Home). Archived from the original (http://ironpython.codeplex.com/Wiki/View.aspx?title=IP26RC1VsCPy26Perf&referringTitle=Home) on January 19, 2013. Retrieved 2009-10-05.
34. "IronPython at python.org" (https://wiki.python.org/moin/IronPython). python.org. Retrieved 2011-04-04. "*IronPython has no GIL and multi-threaded code can use multi core processors.*"
35. "Python's Hardest Problem, Revisited" (https://web.archive.org/web/20151031074238/http://www.jeffknupp.com/blog/2013/06/30/pythons-hardest-problem-revisited/). Archived from the original on 2015-10-31. Retrieved 2015-07-15.

External links

- Official website (https://ironpython.net)

Jython

Jython is an implementation of the Python programming language designed to run on the Java platform. It was known as **JPython** until 1999.[3]

Overview

Jython programs can import and use any Java class. Except for some standard modules, Jython programs use Java classes instead of Python modules. Jython includes almost all of the modules in the standard Python programming language distribution, lacking only some of the modules implemented originally in C. For example, a user interface in Jython could be written with Swing, AWT or SWT. Jython compiles Python source code to Java bytecode (an intermediate language) either on demand or statically.

Jython

Initial release	January 17, 2001[1]
Stable release	2.7.4[2] / 18 August 2024
Repository	github.com/jython/jython/ (https://github.com/jython/jython/)
Written in	Python and Java
Operating system	Cross-platform
Platform	Java virtual machine
Type	Python Interpreter
License	Python Software Foundation License (for older releases see License terms)
Website	www.jython.org (https://www.jython.org/)

History

Jython was initially created in late 1997 to replace C with Java for performance-intensive code accessed by Python programs, moving to SourceForge in October 2000. The Python Software Foundation awarded a grant in January 2005. Jython 2.5 was released in June 2009.[4]

Status and roadmap

The most recent release is Jython 2.7.4. It was released on August 18, 2024 and is compatible with Python 2.7.[5]

Python 3 compatible changes are planned in Jython 3 Roadmap.[6]

Although Jython implements the Python language specification, it has some differences and incompatibilities with CPython, which is the reference implementation of Python.[7][8]

License terms

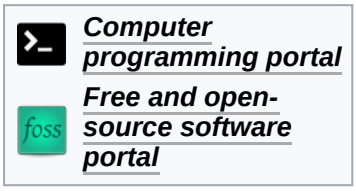
From version 2.2 on, Jython (including the standard library) is released under the Python Software Foundation License (v2). Older versions are covered by the *Jython 2.0, 2.1 license* and the *JPython 1.1.x Software License*.[9]

The command-line interpreter is available under the Apache Software License.

Usage

- JBoss Application Server's command line interface scripting using Jython
- Oracle Weblogic Server Scripting Tool uses Jython
- IBM Rational development tools allow Jython scripting
- IBM WebSphere Application Server tool scripting with *wsadmin* allows using Jython and Jacl
- ZK – a Java Ajax framework that allows glue logic written in Jython
- Ignition - A software development platform focused on HMI and SCADA[10]
- Ghidra - a reverse engineering tool developed by the NSA allows plugins to be written in Java or Jython
- openHAB - home automation software

See also



- List of JVM languages
- IronPython – an implementation of Python for .NET and Mono
- PyPy – a self-hosting interpreter for the Python programming language.
- JRuby – similar project for the Ruby programming language.
- GraalVM - a polyglot runtime written in Java, has a Python 3 implementation

References

1. Wierzbicki, Frank (March 22, 2015). "jython: 3d8067c56a1d NEWS" (https://hg.python.org/jython/file/tip/NEWS). Retrieved March 28, 2015.
2. "Release 2.7.4" (https://github.com/jython/jython/releases/tag/v2.7.4). August 18, 2024. Retrieved August 22, 2024.
3. "JythonFaq/GeneralInfo - JythonWiki" (https://wiki.python.org/jython/JythonFaq/GeneralInfo). April 3, 2014. Retrieved March 28, 2015.
4. Wierzbicki, Frank (June 16, 2009). "Jython 2.5.0 Final is out!" (http://fwierzbicki.blogspot.com/2009/06/jython-250-final-is-out.html). Retrieved July 2, 2009.
5. "News" (https://www.jython.org/news.html). *Jython*. Retrieved April 19, 2020.

6. "Jython 3 Roadmap" (https://www.jython.org/jython-3-roadmap.html). *Jython*. Retrieved October 14, 2022.
7. "JythonFaq" (https://wiki.python.org/jython/JythonFaq/GeneralInfo#IsJythonthesamelanguageasPython.3F). Jython's project. Retrieved July 5, 2009.
8. "Differences between CPython and Jython" (https://web.archive.org/web/20231007235503/https://jython.sourceforge.net/archive/21/docs/differences.html). Jython's project. Archived from the original (http://jython.sourceforge.net/archive/21/docs/differences.html) on October 7, 2023. Retrieved July 5, 2009.
9. "The Jython License" (https://web.archive.org/web/20181009152125/http://www.jython.org/Project/license.html). Jython's project. Archived from the original (https://www.jython.org/Project/license.html) on October 9, 2018. Retrieved February 9, 2008.
10. "Introducing Ignition - Ignition User Manual 7.9 - Ignition Documentation" (https://docs.inductiveautomation.com/display/DOC79/Introducing+Ignition). *docs.inductiveautomation.com*. Retrieved April 24, 2019.

External links

- Official website (https://www.jython.org/)

Psyco

Psyco is an unmaintained specializing just-in-time compiler for pre-2.7 Python originally developed by Armin Rigo and further maintained and developed by Christian Tismer. Development ceased in December, 2011.[1]

Psyco ran on BSD-derived operating systems, Linux, Mac OS X and Microsoft Windows using 32-bit Intel-compatible processors. Psyco was written in C and generated only 32-bit x86-based code.

Although Tismer announced on 17 July 2009 that work was being done on a second version of Psyco,[2] a further announcement declared the project "unmaintained and dead" on 12 March 2012 and pointed visitors to PyPy instead.[3] Unlike Psyco, PyPy incorporates an interpreter and a compiler that can generate C, improving its cross-platform compatibility over Psyco.

Psyco

Developer(s)	Armin Rigo, Christian Tismer
Final release	1.6 / December 16, 2007
Repository	bitbucket.org/arigo/psyco (https://bitbucket.org/arigo/psyco)
Written in	C, Python
Operating system	Cross-platform
Platform	32-bit x86 only
Type	Just-in-time compiler
License	MIT License
Website	psyco.sourceforge.net (http://psyco.sourceforge.net)

Speed enhancement

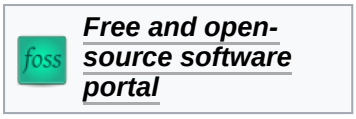
Psyco can noticeably speed up CPU-bound applications. The actual performance depends greatly on the application and varies from a slight slowdown to a 100x speedup.[4][5][6][7] The average speed improvement is typically in the 1.5-4x range, making Python performance close to languages such as Smalltalk and Scheme, but still slower than compiled languages such as Fortran, C or some other JIT languages like C# and Java.[8]

Psyco also advertises its ease of use: the simplest Psyco optimization involves adding only two lines to the top of a script:[9]

```
import psyco
psyco.full()
```

These commands will import the psyco module, and have Psyco optimize the entire script. This approach is best suited to shorter scripts, but demonstrates the minimal amount of work needed to begin applying Psyco optimizations to an existing program.

See also



- PyPy
- Unladen Swallow
- Cython
- YARV (Yet another Ruby VM)

References

1. "psyco / Commits" (https://bitbucket.org/arigo/psyco/commits/all). *Bitbucket*. Armin Rigo.
2. Tismer, Christian (17 July 2009). "[pypy-dev] ANN: psyco V2" (https://mail.python.org/pipermail/pypy-dev/2009-July/005288.html). *pypy-dev mailing list*.
3. "Psyco Homepage" (http://psyco.sourceforge.net/).
4. "Python Psyco benchmarks" (https://web.archive.org/web/20080606020118/http://shootout.alioth.debian.org/gp4/benchmark.php?test=all&lang=psyco). Archived from the original (http://shootout.alioth.debian.org/gp4/benchmark.php?test=all&lang=psyco) on 2008-06-06. Retrieved 2008-04-24.
5. "Python Psyco Homepage at sourceforge" (http://psyco.sourceforge.net/introduction.html). Retrieved 2009-03-04.
6. "A beginners guide to using Python for performance computing at scipy.org" (https://web.archive.org/web/20090311070246/http://www.scipy.org/PerformancePython). Archived from the original (http://www.scipy.org/PerformancePython) on 2009-03-11. Retrieved 2009-03-04.
7. "Charming Python: Make Python run as fast as C with Psyco" (http://www.ibm.com/developerworks/library/l-psyco.html). *IBM*. Retrieved 2009-03-04.
8. "Boxplot Summary" (https://web.archive.org/web/20110603231929/http://shootout.alioth.debian.org/gp4/benchmark.php?test=all&lang=all&d=data&gcc=on&gpp=on&java=on&csharp=on&psyco=on&mzscheme=on&vw=on&python=on&calc=calculate&box=1). Archived from the original (http://shootout.alioth.debian.org/gp4/benchmark.php?test=all&lang=all&d=data&gcc=on&gpp=on&java=on&csharp=on&psyco=on&mzscheme=on&vw=on&python=on&calc=calculate&box=1) on 2011-06-03. Retrieved 2009-10-16.
9. Rigo, Armin. "Quick examples" (http://psyco.sourceforge.net/psycoguide/node8.html). *The Ultimate Psyco Guide*. Retrieved 3 June 2011.

External links

- Psyco (https://sourceforge.net/projects/psyco/) on SourceForge
- David Mertz's IBM developerWorks article: Make Python run as fast as C with Psyco (http://www-106.ibm.com/developerworks/linux/library/l-psyco.html)
- psyco notes, Poor Yorick (https://archive.today/20121217175529/https://www.pooryorick.com/secure/wiki/Pub/Psyco)

PyPy

PyPy (/ˈpaɪpaɪ/) is an implementation of the Python programming language.[2] PyPy often runs faster than the standard implementation CPython because PyPy uses a just-in-time compiler.[3] Most Python code runs well on PyPy except for code that depends on CPython extensions, which either does not work or incurs some overhead when run in PyPy.

PyPy itself is built using a technique known as meta-tracing, which is a mostly automatic transformation that takes an interpreter as input and produces a tracing just-in-time compiler as output. Since interpreters are usually easier to write than compilers, but run slower, this technique can make it easier to produce efficient implementations of programming languages. PyPy's meta-tracing toolchain is called RPython.

PyPy officially supports Python 2.7 and 3.10[4] and has a few differences in implementations compared to CPython.[5]

Initial release	mid 2007
Stable release	7.3.19[1] (26 February 2025)
Repository	github.com/pypy/pypy (https://github.com/pypy/pypy)
Written in	RPython
Operating system	Cross-platform
Type	Python interpreter and compiler toolchain
License	MIT
Website	pypy.org (https://pypy.org/)

Details and motivation

PyPy aims to provide a common translation and support framework for producing implementations of dynamic languages, emphasizing a clean separation between language specification and implementation aspects. It also aims to provide a compliant, flexible and fast implementation of the Python programming language using the above framework to enable new advanced features without having to encode low-level details into it.[6][7]

RPython

The PyPy interpreter itself is written in a restricted subset of Python called RPython (Restricted Python).[8] RPython puts some constraints on the Python language such that a variable's type can be inferred at compile time.[9]

The PyPy project has developed a toolchain that analyzes RPython code and translates it into a form of byte code, which can be lowered into C. There used to be other backends in addition to C (Java, C#, and JavaScript), but those suffered from bitrot and have been removed. Thus, the recursive logo of PyPy is a snake swallowing itself since the RPython is translated by a Python interpreter. The code can also be run untranslated for testing and analysis, which provides a nice test-bed for research into dynamic languages.

It allows for pluggable garbage collectors, as well as optionally enabling Stackless Python features. Finally, it includes a just-in-time (JIT) generator that builds a just-in-time compiler into the interpreter, given a few annotations in the interpreter source code. The generated JIT compiler is a tracing JIT.[10]

RPython is now also used to write non-Python language implementations, such as Pixie.[11]

Project status

PyPy as of version 7.3.17 is compatible with two CPython versions: 2.7 and 3.10.[12][13] The first PyPy version compatible with CPython v3 is PyPy v2.3.1 (2014).[14] The PyPy interpreter compatible with CPython v3 is also known as PyPy3.

PyPy has JIT compilation support on 32-bit/64-bit x86 and 32-bit/64-bit ARM processors.[15] It is tested nightly on Windows, Linux, OpenBSD and Mac OS X. PyPy is able to run pure Python software that does not rely on implementation-specific features.[16]

There is a compatibility layer for CPython C API extensions called CPyExt, but it is incomplete and experimental. The preferred way of interfacing with C shared libraries is through the built-in C foreign function interface (CFFI) or ctypes libraries.

History

PyPy is a followup to the Psyco project, a just-in-time specializing compiler for Python, developed by Armin Rigo between 2002 and 2010. PyPy's aim is to have a just-in-time specializing compiler with scope, which was not available for Psyco. Initially, the RPython could also be compiled into Java bytecode, CIL and JavaScript, but these backends were removed due to lack of interest.

PyPy was initially a research and development-oriented project. Reaching a mature state of development and an official 1.0 release in mid-2007, its next focus was on releasing a production-ready version with more CPython compatibility. Many of PyPy's changes have been made during coding sprints.

- In August 2008, PyPy was able to run some popular Python libraries like Pylons,[17] Pyglet,[18] Nevow[19] and Django.[20]
- On 12 March 2010, PyPy 1.2 was released, focusing on speed. It included a working, though not yet stable, just-in-time compiler.[21]
- On 30 April 2011, PyPy version 1.5 was released, which reached compatibility with CPython 2.7.[22]
- On 9 May 2013, PyPy 2.0 was released, which introduced alpha-quality support for JIT compilation on ARMv6 and ARMv7 JIT, and included CFFI in the standard library.[23][24]
- On 20 June 2014, PyPy3 was declared stable[14] and introduced compatibility with the more modern Python 3. It was released alongside PyPy 2.3.1 and bears the same version number.
- On 21 March 2017, the PyPy project released version 5.7 of both PyPy and PyPy3, with the latter introducing beta-quality support for Python 3.5.[25]
- On 26 April 2018, version 6.0 was released, with support for Python 2.7 and 3.5 (still beta-quality on Windows).[26]

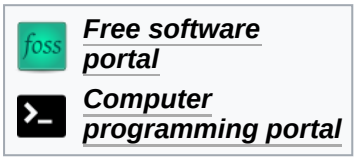
- On 11 February 2019, version 7.0 was released, with support for Python 2.7 and 3.5.[27]
- On 14 October 2019, version 7.2 was released, with support for Python 3.6.9.[28]
- On 24 December 2019, version 7.3 was released, with support for Python 3.6.9.[29]
- On 16 February 2020, the PyPy team announced the move of the source code hosting from Bitbucket to heptapod.net with the repositories of the CFFI (C Foreign Function Interface) project. A new logo and website design are also published. However, the author and the license of the new logo are unknown.[30]
- On 29 December 2023, PyPy announced hosting has moved to GitHub and development will now be tracked with git.[31]

Funding

PyPy was funded by the European Union being a Specific Targeted Research Project[32] between December 2004 and March 2007. In June 2008, PyPy announced funding being part of the Google Open Source programs and has agreed to focus on making PyPy more compatible with CPython. In 2009 Eurostars, a European Union funding agency specially focused on SMEs,[33] accepted a proposal from PyPy project members titled "PYJIT – a fast and flexible toolkit for dynamic programming languages based on PyPy". Eurostars funding lasted until August 2011.[34] At PyCon US 2011, the Python Software Foundation provided a $10,000 grant for PyPy to continue work on performance and compatibility with newer versions of the language.[35] The port to ARM architecture was sponsored in part by the Raspberry Pi Foundation.[23]

The PyPy project also accepts donations through its status blog pages.[36] As of 2013, a variety of sub-projects had funding: Python 3 version compatibility, built-in optimized NumPy support for numerical calculations and software transactional memory support to allow better parallelism.[23]

See also



- Bootstrapping (compilers)
- Cython
- GraalVM
- Partial evaluation
- Psyco
- Self-hosting
- Self-interpreter
- Unladen Swallow

Notes

1. "PyPy v7.3.19 release" (https://pypy.org/posts/2025/02/pypy-v7319-release.html). Retrieved 12 March 2025.

2. "Interview Maciej Fijalkowski PyPy" (https://decisionstats.com/2015/11/29/interview-maciej-fijalkowski-pypy/). 29 November 2015.
3. "PyPy Speed" (https://speed.pypy.org/). *speed.pypy.org*. Retrieved 2019-12-01.
4. "PyPy" (https://pypy.org/index.html). 30 December 2024.
5. "Differences between PyPy and CPython — PyPy documentation" (https://doc.pypy.org/en/latest/cpython_differences.html). 30 December 2024.
6. Samuele Pedroni (March 2007). "PyPy – Goals and Architecture Overview" (https://web.archive.org/web/20120614024217/http://doc.pypy.org/en/latest/architecture.html#mission-statement). Archived from the original (http://codespeak.net/pypy/dist/pypy/doc/architecture.html#mission-statement) on 2012-06-14.
7. "PyPy – Goals and Architecture Overview – Mission Statement" (http://doc.pypy.org/en/latest/architecture.html#mission-statement). Retrieved 11 October 2013.
8. Our runtime interpreter is "RPython" (https://doc.pypy.org/en/latest/coding-guide.html#our-runtime-interpreter-is-rpython), Coding Guide – PyPy documentation
9. "It is a proper subset of Python, restricted in a way that enables easy analysis and efficient code generation", Ancona et al., 2007.
10. Bolz, Carl; Cuni, Antonio; Fijalkowski, Maciej; Rigo, Armin. *Tracing the Meta-Level: PyPy's Tracing JIT Compiler*. ICOOOLPS '09. doi:10.1145/1565824.1565827 (https://doi.org/10.1145%2F1565824.1565827).
11. Timothy Balridge interview (https://medium.com/this-is-not-a-monad-tutorial/indie-languages-interview-pixie-and-timothy-baldridge-cadbc36418dc).
12. "PyPy – Python compatibility" (http://pypy.org/compat.html). *pypy.org*. 28 December 2019. Retrieved 2020-12-15.
13. "PyPy v7.3.7: bug-fix release of 3.7, 3.8" (https://www.pypy.org/posts/2021/10/pypy-v737-release.html). *pypy.org*. 25 October 2021. Retrieved 2021-11-10.
14. the PyPy team (20 June 2014). "PyPy3 2.3.1 – Fulcrum" (http://morepypy.blogspot.it/2014/06/pypy3-231-fulcrum.html). PyPy blog.
15. "PyPy v7.2.0: release of 2.7, and 3.6" (https://pypy.readthedocs.io/en/latest/release-v7.2.0.html). pypy.org. 16 October 2019.
16. "PyPy – Python compatibility" (http://pypy.org/compat.html). 28 December 2019.
17. "Running pylons on top of PyPy" (http://morepypy.blogspot.com/2008/06/running-pylons-on-top-of-pypy.html). 10 June 2008.
18. "Running Pyglet on top of PyPy" (http://morepypy.blogspot.com/2008/02/running-pyglet-on-pypy.html). 20 February 2008.
19. "Running Nevow on top of PyPy" (http://morepypy.blogspot.com/2008/06/running-nevow-on-top-of-pypy.html). 20 June 2008.
20. "PyPy runs unmodified django 1.0 beta" (http://morepypy.blogspot.com/2008/08/pypy-runs-unmodified-django-10-beta.html). 19 August 2008.
21. "Introducing the PyPy 1.2 release" (https://morepypy.blogspot.dk/2011/04/pypy-15-released-catching-up.html). 30 April 2011.
22. "PyPy 1.5 Released: Catching Up" (https://morepypy.blogspot.dk/2010/03/introducing-pypy-12-release.html). 12 March 2010.
23. Jake Edge (15 May 2013). "A look at the PyPy 2.0 release" (https://lwn.net/Articles/550427/). LWN.net.
24. "PyPy 2.0 – Einstein Sandwich" (https://morepypy.blogspot.dk/2013/05/pypy-20-einstein-sandwich.html). 9 May 2013.
25. "PyPy2.7 and PyPy3.5 v5.7 – two in one release" (https://morepypy.blogspot.dk/2017/03/pypy27-and-pypy35-v57-two-in-one-release.html). 21 March 2017.
26. "PyPy2.7 and PyPy3.5 v6.0 dual release" (https://morepypy.blogspot.de/2018/04/pypy27-and-pypy35-v60-dual-release.html). 26 April 2018.

27. Cuni, Antonio (2019-02-11). "PyPy Status Blog: PyPy v7.0.0: triple release of 2.7, 3.5 and 3.6-alpha" (https://morepypy.blogspot.com/2019/02/pypy-v700-triple-release-of-27-35-and.html). *PyPy Status Blog*. Retrieved 2020-08-17.
28. Mattip (2019-10-14). "PyPy Status Blog: PyPy v7.2 released" (https://morepypy.blogspot.com/2019/10/pypy-v72-released.html). *PyPy Status Blog*. Retrieved 2020-08-17.
29. Mattip (2019-12-24). "PyPy Status Blog: PyPy v7.3.0 released" (https://morepypy.blogspot.com/2019/12/pypy-730-released.html). *PyPy Status Blog*.
30. "PyPy and CFFI have moved to Heptapod" (https://www.pypy.org/posts/2020/02/pypy-and-cffi-have-moved-to-heptapod-5791595152472747032.html). 16 February 2020.
31. "PyPy has moved to Git, GitHub" (https://www.pypy.org/posts/2023/12/pypy-moved-to-git-github.html). 29 December 2023.
32. "EU Community Research and Development Information Service Entry" (http://cordis.europa.eu/projects/rcn/74481_en.html).
33. "Eurostars – Aim Higher" (http://www.eurostars-eureka.eu/).
34. "Project Page on Eureka Network" (https://web.archive.org/web/20120403075231/http://www.eurekanetwork.org/project/-/id/4791). Archived from the original (http://www.eurekanetwork.org/project/-/id/4791) on 2012-04-03. Retrieved 2011-10-17.
35. "A thank you to the PSF" (http://morepypy.blogspot.com/2011/03/thank-you-to-psf.html). 22 March 2011.
36. "PyPy Status Blog: Oh, and btw: PyPy gets funding through "Eurostars" " (http://morepypy.blogspot.com/2010/12/oh-and-btw-pypy-gets-funding-through.html). 10 December 2010.

References

- Davide Ancona, Massimo Ancona, Antonio Cuni, Nicholas D. Matsakis, 2007. RPython: a Step Towards Reconciling Dynamically and Statically Typed OO Languages (https://web.archive.org/web/20170706104906/ftp://ftp.disi.unige.it/pub/person/AnconaD/DLS08.pdf). In Proc. Dynamic Language Symposium (DLS), 2007. ACM Press.
- Carl Friedrich Bolz, Antonio Cuni, Maciej Fijalkowski, 2009. Tracing the meta-level: PyPy's Tracing JIT Compiler (https://web.archive.org/web/20110716123213/http://codespeak.net/svn/pypy/extradoc/talk/icooolps2009/bolz-tracing-jit-final.pdf). In Proc. ICOOOLPS, 2009. ACM Press.
- Corbet, Jonathan (11 May 2011). "A brief experiment with PyPy" (https://lwn.net/Articles/442268/). LWN.net.
- von Eitzen, Chris (21 November 2011). "PyPy 1.7 widens the performance "sweet spot" " (http://www.h-online.com/open/news/item/PyPy-1-7-widens-the-performance-sweet-spot-1382249.html). *The H*. Heinz Heise.
- Rose, John (2 December 2011). "A Day with PyPy" (http://blogs.oracle.com/jrose/entry/a_day_with_pypy). Oracle developer blog.
- "Interview Maciej Fijalkowski pypy" (http://decisionstats.com/2015/11/29/interview-maciej-fijalkowski-pypy/). Decisionstats blog. 29 November 2015.

External links

- Official website (https://pypy.org/)

Python for S60

Python for S60, also called **PyS60**—a term reminiscent of the Unix naming convention—is a port of the Python programming language developed by Nokia for its S60 software platform, originally based on Python 2.2.2 from 2002.[1] The final version, PyS60-2.0.0, was released on 11 February 2010. It came with multiple improvements, the most notable of which was an update to a new core based on Python 2.5.4.[2]

Python for S60

Paradigm	Multi-paradigm: Object-oriented, Imperative, Functional
Designed by	Guido van Rossum
Developer	Python Software Foundation
First appeared	2006
Stable release	2.0.0 / 11 February 2010
Implementation language	C++, Python
OS	Symbian OS, S60 platform
License	Apache license, Python Software Foundation License
Website	garage.maemo.org/projects/pys60/ (https://garage.maemo.org/projects/pys60/)

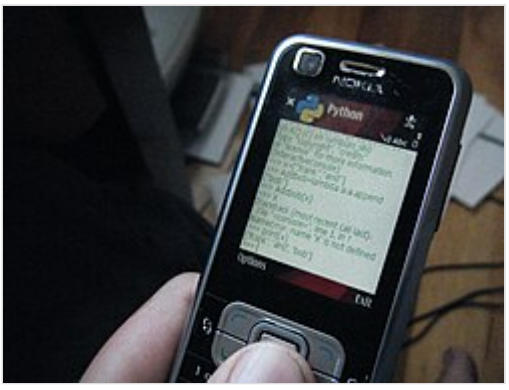
Release history

First released in 2005, PyS60 featured a relatively small set of modules and functions. Version 1.2, the last closed-source release and the second version of PyS60, brought many improvements and was made available on 21 October 2005 on the Nokia Forums.

After becoming open-source, PyS60 had the advantage of a strong and dedicated community that actively contributed to improving it. The milestone release was version 1.3.11.

The final version that supported the S60 2nd Edition platform, 1.4.5, was released on 3 December 2008. On 24 December 2008, a developer version, 1.9.0, was released. It featured several improvements, the most notable of which was a new core based on Python 2.5.1.

The final version, 2.0.0, was released on 11 February 2010. Which core is based on Python 2.5.4.[2]



Python Shell on Nokia 6120 Classic

See also

Free and open-source software portal

- List of Python software
- List of integrated development environments for Python

- Open Programming Language for older Symbian devices

References

1. "Nokia - Nokia to Release Python for S60 Source Code to Open-Source Software Developer Community" (https://web.archive.org/web/20070518052247/http://press.nokia.com/PR/200601/1032017_5.html). Archived from the original (http://press.nokia.com/PR/200601/1032017_5.html) on 18 May 2007. Retrieved 17 November 2008.
2. "Download:Ensymble v0.29 - Now a distutils package" (https://code.google.com/p/ensymble/downloads/detail?name=ensymble-0.29.tar.gz). 17 May 2010. Retrieved 28 January 2014.

External links

- Nokia Research Center - Python for S60 (https://web.archive.org/web/20080810154656/http://opensource.nokia.com/projects/pythonfors60/)
- Maemo Garage - Python for S60 (https://garage.maemo.org/projects/pys60/) Archived (https://web.archive.org/web/20110718043620/https://garage.maemo.org/projects/pys60/) 18 July 2011 at the Wayback Machine
- SourceForge.net - Python for S60

Shed Skin

Shed Skin is an experimental restricted-Python (3.8+) to C++ programming language compiler. It can translate pure, but implicitly statically typed Python programs into optimized C++. It can generate stand-alone programs or extension modules that can be imported and used in larger Python programs.

Shed Skin is an open source project with contributions from many people, however the main author is Mark Dufour. Work has been going into Shed Skin since 2005.[1]

Shed Skin

Original author(s)	Mark Dufour, others
Initial release	2005
Stable release	0.9.9 / June 22, 2024
Repository	github.com/shedskin/shedskin (https://github.com/shedskin/shedskin)
Written in	Python, C++
Operating system	Cross-platform: Linux, macOS, Windows
Available in	Python
Type	Source-to-source compiler
License	GPLv3, BSD, MIT
Website	shedskin.github.io (http://shedskin.github.io)

Features

Besides the typing restriction,[2] programs cannot freely use the Python standard library, although about 20 common modules, such as `random`, `itertools` and `re` (regular expressions), are supported as of 2011. Also, not all Python features, such as nested functions and variable numbers of arguments, are supported. Many introspective dynamic parts of the language are unsupported. For example, functions like `getattr`, and `hasattr` are unsupported.

As of May 2011, Unicode is not supported.[3]

As of June 2016 for a set of 75 non-trivial test programs (at over 25,000 lines of code in total),[4] measurements show a typical speedup of 2-20 times over Psyco, and 2-200 times over CPython.[5][6][7][8] Shed Skin is still in an early stage of development, so many other programs will not compile unmodified.[9]

Shed Skin can be used to generate standalone executables which need only the C++ runtime libraries. It can also be used to generate CPython modules. This allows compiling parts of larger programs with Shed Skin, while running the other parts using regular CPython.

Another use has been to wrap C++ classes using Shed Skin to allow C++ classes to be used as Python classes.[10]

The license of the Shed Skin source code is under two parts. The main compiler code is under the GNU General Public License (GPL). The supporting code that it uses as a run time library is under a BSD or MIT license depending on the module. This allows compiling both GPL and non-GPL programs.

Type inference

Shed Skin combines Ole Agesen's Cartesian Product Algorithm (CPA) with the data-polymorphic part of John Plevyak's Iterative Flow Analysis (IFA).[11] Version 0.6 introduced scalability improvements which repeatedly analyze larger versions of a program (in addition to the mentioned techniques), until it is fully analyzed. This allows Shed Skin to do type inference on larger programs than previously. It starts with an empty callgraph, essentially, and slowly adds to it, until the whole call graph has been added. A graph has been published by the author, showing analysis times for 50 example programs, at a total of around 15,000 lines.[12]

Modules

For version 0.9 the following 25 modules are largely supported.[13] Several of these, such as os.path, were compiled to C++ using Shed Skin.

- array
- binascii
- bisect
- collections (defaultdict, deque)
- ConfigParser (no SafeConfigParser)
- copy
- colorsys
- csv (no Dialect, Sniffer)
- datetime
- fnmatch
- getopt
- glob
- heapq
- itertools (no starmap)
- math
- mmap
- os
- os.path
- random
- re
- socket
- string
- struct (no Struct, pack_into, unpack_from)
- sys
- time

Note that any other module, such as pygame, pyqt or pickle, may be used in combination with a Shed Skin generated extension module (https://shedskin.readthedocs.io/en/latest/documentation.html#generating-an-extension-module). For examples of this, see the Shed Skin examples (https://github.com/shedskin/shedskin/tree/master/examples).

See also

- Cython
- PyPy

References

1. first Shed Skin release (http://boost.2283326.n4.nabble.com/First-release-of-Shed-Skin-a-Python-to-C-compiler-td2696874.html)
2. Learning Python Book section on Shed Skin (https://books.google.com/books?id=1HxWGezDZcgC&dq=learning%20python%20shedskin&pg=PA31)
3. Shed Skin tutorial ("Python Subset Restrictions" section) (http://shedskin.googlecode.com/files/shedskin-tutorial-0.7.html#Python%32Subset%32Restrictions)
4. "Shedskin/Shedskin" (https://github.com/shedskin/shedskin). *GitHub*. 17 May 2022.
5. Speed up your Python: Unladen vs. Shed Skin vs. PyPy vs. Cython vs. C (http://geetduggal.wordpress.com/2010/11/25/speed-up-your-python-unladen-vs-shedskin-vs-pypy-vs-c/)
6. Taking on Shed-Skin (http://www.philhassey.com/blog/2007/11/29/taking-on-shed-skin/)
7. Speeding up Python code with Shed Skin (http://www.korokithakis.net/node/117)
8. MiniLight, minimal global illumination renderer benchmark (http://www.hxa.name/minilight/)
9. Shed Skin webpage (https://shedskin.github.io/)
10. Wrapping C++ classes using Shed Skin (http://chatter.recreclabs.com/2011/01/wrapping-c-classes-using-shedskin/)
11. Master Thesis Mark Dufour, "Shed Skin. An Optimizing Python-to-C++ Compiler" (http://mark.dufour.googlepages.com/shedskin.pdf), April 19, 2006
12. Type inference scalability (http://shed-skin.blogspot.com/2010/12/shed-skin-07-type-inference-scalability.html), 2010-12
13. Shedskin 0.9 release notes (https://shedskin.readthedocs.io/en/latest/documentation.html#library-limitations)

External links

- Official website (http://shedskin.github.io)
- Shed Skin Blog (http://shed-skin.blogspot.com/)
- shedskin source code repository (https://github.com/shedskin/shedskin/)

Stackless Python

Stackless Python, or **Stackless**, was a Python programming language interpreter. Its Github repository has been archived since February 2025, and the project has been officially discontinued.

It was so named because it avoids depending on the C call stack for its own stack. In practice, Stackless Python uses the C stack, but the stack is cleared between function calls.[2] The most prominent feature of Stackless is microthreads, which avoid much of the overhead associated with usual operating system threads. In addition to Python features, Stackless also adds support for coroutines, communication channels, and task serialization.

Stackless Python

Original author(s)	Christian Tismer
Developer(s)	Anselm Kruis
Initial release	1998
Stable release	3.8.1-slp[1] / January 22, 2020
Preview release	3.9.0 alpha 0
Repository	github.com/stackless-dev/stackless (https://github.com/stackless-dev/stackless)
Written in	C, Python
Operating system	Linux, macOS, Windows
Type	Interpreter
License	Python Software Foundation License
Website	github.com/stackless-dev (https://github.com/stackless-dev/stackless/wiki/)

Design

With Stackless Python, a running program is split into microthreads that are managed by the language interpreter itself, not the operating system kernel—context switching and task scheduling is done purely in the interpreter (these are thus also regarded as a form of green thread). Microthreads manage the execution of different subtasks in a program on the same CPU core. Thus, they are an alternative to event-based asynchronous programming and also avoid the overhead of using separate threads for single-core programs (because no mode switching between user mode and kernel mode needs to be done, so CPU usage can be reduced).

Although microthreads make it easier to deal with running subtasks on a single core, Stackless Python does not remove CPython's global interpreter lock (GIL), nor does it use multiple threads and/or processes. So it allows only cooperative multitasking on a shared CPU and not parallelism (preemption was originally not available but is now in some form[3]). To use multiple CPU cores, one would still need to build an interprocess communication system on top of Stackless Python processes.

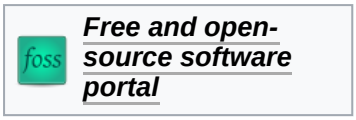
Due to the considerable number of changes in the source, Stackless Python cannot be installed on a preexisting Python installation as an extension or library. It is instead a complete Python distribution in itself. The majority of Stackless's features have also been implemented in PyPy, a self-hosting Python interpreter and JIT compiler.[4]

Use

Although the whole Stackless is a separate distribution, its switching functionality has been successfully packaged as a CPython extension called greenlet.[5] It is used by a number of libraries (e.g. gevent[6]) to provide a green threading solution for CPython. Python since has received a native solution for green threads: await/async.

Stackless is used extensively in the implementation of the *Eve Online* massively multiplayer online game as well as in IronPort's mail platform.

See also



- Erlang (programming language)
- Limbo (programming language)
- Go (programming language)
- SCOOP (software)

References

1. "Release v3.8.1-slp" (https://github.com/stackless-dev/stackless/releases/tag/v3.8.1-slp). 12 August 2021. Retrieved 8 March 2022.
2. Archived at Ghostarchive (https://ghostarchive.org/varchive/youtube/20211211/pDkrkP0yf70) and the Wayback Machine (https://web.archive.org/web/20140103062231/http://www.youtube.com/watch?v=pDkrkP0yf70): *The story of stackless Python* (https://www.youtube.com/watch?v=pDkrkP0yf70). *YouTube*.
3. "About Stackless" (https://web.archive.org/web/20200623233250/https://bitbucket.org/stackless-dev/stackless/wiki/Home). Archived from the original (https://bitbucket.org/stackless-dev/stackless/wiki/Home) on 23 June 2020. Retrieved 26 August 2016. "a round robin scheduler is built in. It can be used to schedule tasklets either cooperatively or preemptively."
4. "Application-level Stackless features — PyPy documentation" (https://pypy.readthedocs.org/en/latest/stackless.html). *pypy.readthedocs.org*.
5. "greenlet: Lightweight concurrent programming — greenlet 0.4.0 documentation" (https://greenlet.readthedocs.org/en/latest/). *greenlet.readthedocs.org*.
6. "What is gevent? — gevent 1.3.0.dev0 documentation" (https://www.gevent.org). *www.gevent.org*.

External links

- Official website (https://github.com/stackless-dev/stackless/wiki)
- Stackless Python Documentation for: 3.7-slp (https://stackless.readthedocs.io/en/3.7-slp/stackless-python.html), 3.6-slp (https://stackless.readthedocs.io/en/3.6-slp/stackless-python.ht

ml), 3.5-slp (https://stackless.readthedocs.io/en/3.5-slp/stackless-python.html), 3.4-slp (https://stackless.readthedocs.io/en/3.4-slp/stackless-python.html), 2.7-slp (https://stackless.readthedocs.io/en/2.7-slp/stackless-python.html)

- stackless (https://github.com/stackless-dev/stackless) on GitHub
- Multithreaded Game Scripting with Stackless Python (https://kalogirou.net/2005/08/10/multithreaded-game-scripting-with-stackless-python/) by Harry Kalogirou
- Continuations and Stackless Python (https://web.archive.org/web/20121108011701/http://zope.stackless.com/spcpaper.htm) by Christian Tismer

---

Unladen Swallow

Unladen Swallow was an optimization branch of CPython, the reference implementation of the Python programming language, which incorporated a just-in-time compiler built using LLVM into CPython's virtual machine. Like many things regarding Python (and the name "Python" itself), "Unladen Swallow" is a Monty Python reference, specifically to the joke about the airspeed velocity of unladen swallows in *Monty Python and the Holy Grail*. The project's stated goals were to provide full compatibility with CPython specific code while quintupling its performance, and for the project to eventually be merged into CPython.[1][2] Although it fell short of all its published goals, some Unladen Swallow code was added into the main Python implementation, such as improvements to the cPickle module.[3][1]

Unladen Swallow was sponsored by Google, and the project owners, Thomas Wouters, Jeffrey Yasskin, and Collin Winter, were Google employees, though most project contributors were not.[4] Unladen Swallow was hosted on Google Code.[5]

In March 2010, a Python Enhancement Proposal (PEP) which proposed merging Unladen Swallow into a special py3k-jit branch of Python's official repository was accepted.[1] However, its implementation was made difficult by Unladen being based on Python 2.6, with which Python 3 broke compatibility, and the PEP was subsequently withdrawn.[6]

In July 2010, speculation began on whether the project was dead or dying since the 2009 Q4 milestone had not yet been released,[7] and the traffic on Unladen's mailing list had decreased from 500 messages in January 2010 to fewer than 10 in September 2010.[8] It had also been reported that Unladen had lost Google's funding,[9] and in November 2010, Collin announced that "Jeffrey and I have been pulled on to other projects of higher importance to Google".[10] By early 2011, it was clear that the project was stopped.[6]

Release history

- 2009 Q1[11]
- 2009 Q2[12]
- 2009 Q3: reduce memory use, improve speed[13]

A 2009 Q4 development branch was created in January 2010,[14] but was not advertised on the website, and its milestone was unmet.[15]

See also

- History of Python
- PyPy, another Python implementation with a JIT compiler

References

1. Winter, Collin; Yasskin, Jeffrey; Kleckner, Reid (17 March 2010). "PEP 3146 - Merging Unladen Swallow into CPython" (https://peps.python.org/pep-3146/). Python.org.
2. Paul, Ryan (26 March 2009). "Ars Technica report on Unladen Swallow goals" (https://arstechnica.com/information-technology/2009/03/google-launches-project-to-boost-python-performance-by-5x/). Arstechnica.com. Retrieved 19 August 2011.
3. "Issue 9410: Add Unladen Swallow's optimizations to Python 3's pickle. - Python tracker" (https://bugs.python.org/issue9410). *bugs.python.org*. Retrieved 8 August 2019.
4. "People working on Unladen Swallow" (https://web.archive.org/web/20151029092746/https://code.google.com/p/unladen-swallow/people/list). Archived from the original (https://code.google.com/p/unladen-swallow/people/list) on 29 October 2015. Retrieved 8 August 2019.
5. "Unladen Swallow project page" (https://code.google.com/p/unladen-swallow/). Retrieved 19 August 2011.
6. Kleckner, Reid (26 March 2011). "Unladen Swallow Retrospective" (https://qinsb.blogspot.com/2011/03/unladen-swallow-retrospective.html). *QINSB is not a Software Blog (qinsb.blogspot.com)*.
7. "Message on comp.lang.python" (https://groups.google.com/g/comp.lang.python/c/afhcrbQkodQ?pli=1#bdf73a28f3f770cb). Retrieved 19 August 2011.
8. "Unladen Swallow | Google Groups" (https://groups.google.com/g/unladen-swallow/about). Retrieved 19 August 2011.
9. "reddit post by an Unladen committer" (https://www.reddit.com/r/Python/comments/cilk0/comment/c0su1wt/). Reddit.com. 24 June 2010. Retrieved 19 August 2011.
10. Winter, Collin (8 November 2010). "Current status of Unladen-Swallow" (https://groups.google.com/g/unladen-swallow/c/8gERKcRBTQQ/m/KiUjXJNvTr0J).
11. "Unladen Swallow 2009Q1" (https://code.google.com/p/unladen-swallow/wiki/Release2009Q1). unladen-swallow, A faster implementation of Python. Retrieved 19 October 2012.
12. "Unladen Swallow 2009Q2" (https://code.google.com/p/unladen-swallow/wiki/Release2009Q2). unladen-swallow, A faster implementation of Python. Retrieved 19 October 2012.
13. "Unladen Swallow 2009Q3" (https://code.google.com/p/unladen-swallow/wiki/Release2009Q3). unladen-swallow, A faster implementation of Python. Retrieved 19 October 2012.
14. "2009 Q4 release branch creation" (https://code.google.com/p/unladen-swallow/source/detail?spec=svn1164&r=1042). 26 January 2010. Retrieved 19 August 2011.
15. "Message on comp.lang.python" (https://groups.google.com/g/comp.lang.python/c/afhcrbQkodQ?pli=1#bdf73a28f3f770cb). Retrieved 19 August 2011.

eric (software)

eric is a free integrated development environment (IDE) used for computer programming. Since it is a full featured IDE, it provides by default all necessary tools needed for the writing of code and for the professional management of a software project.

eric is written in the programming language Python and its primary use is for developing software written in Python. It is usable for development of any combination of Python 3 or Python 2, Qt 5 or Qt 4 and PyQt 5 or PyQt 4 projects, on Linux, macOS and Microsoft Windows platforms.

License, price and distribution

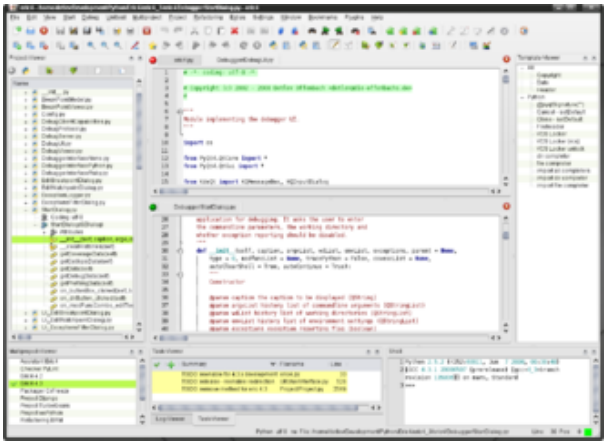
eric is licensed under the GNU General Public License version 3 or later and is thereby Free Software. This means in general terms that the source code of eric can be studied, changed and improved by anyone, that eric can be run for any purpose by anyone and that eric - and any changes or improvements that may have been made to it - can be redistributed by anyone to anyone as long as the license is not changed (copyleft).

eric can be downloaded at SourceForge and installed manually with a python installer script.[2] Most major Linux distributions include eric in their software repositories, so when using such Linux distributions eric can be obtained and installed automatically by using the package manager of the particular distribution.[3] Additionally, the author offers access to the source code via a public Mercurial repository.[4]

Characteristics

eric is written in Python and uses the PyQt Python bindings for the Qt GUI toolkit.[5] By design, eric acts as a front end for several programs, for example the QScintilla editor widget.[6]

eric



eric4 running in KDE SC 4

Original author(s)	Detlev Offenbach
Developer(s)	Detlev Offenbach
Initial release	2002
Stable release	25.3[1] / 1 March 2025
Repository	hg.die-offenbachs.homelinux.org/eric (https://hg.die-offenbachs.homelinux.org/eric)
Written in	Python
Operating system	Linux, macOS, Microsoft Windows
Platform	Python, Qt, PyQt
Available in	English, German, French, Russian, Czech, Spanish, Italian, Turkish, Chinese

Features

Type	Integrated Development Environment
License	GPL version 3 or later
Website	eric-ide.python-projects.org (https://eric-ide.python-projects.org)

The key features of eric 6 are:[7]

- Source code editing:
  - Unlimited number of editors
  - Configurable window layout
  - Configurable syntax highlighting
  - Sourcecode autocompletion
  - Sourcecode calltips
  - Sourcecode folding
  - Brace matching
  - Error highlighting
  - Advanced search functionality including project wide search and replace
  - Integrated class browser
  - Integrated profiling and code coverage support
- GUI designing:
  - Integration of Qt Designer, a Graphical user interface builder for the creation of Qt-based Graphical user interfaces
- Debugging, checking, testing and documenting:
  - Integrated graphical python debugger which supports both interactive probing while suspended and auto breaking on exceptions as well as debugging multi-threaded and multiprocessing applications
  - Integrated automatic code checkers (syntax, errors and style, PEP-8) for static program analysis as well as support of Pylint via plug-in
  - Integrated source code documentation system
  - Integrated unit testing support by having the option to run python code with command-line parameters
  - Integrated interface to the enchant spell checking library
  - Application diagrams
- Version control:
  - Integrated version control support for Mercurial and Subversion repositories (as core plug-ins) and git (as optional plug-in)
- Project management and collaboration:
  - Advanced project management facilities
  - Integrated task management with a self-updating To-do list
  - Integrated cooperation functions (chat, shared editor)
- Other:
  - Integrated web browser
  - Integrated support for Django (as optional plug-in)
  - Running external applications from within the IDE
  - Interactive Python shell including syntax highlighting and autocompletion
  - Integrated CORBA support based on omniORB
  - Integrated rope refactoring tool (as optional plug-in)

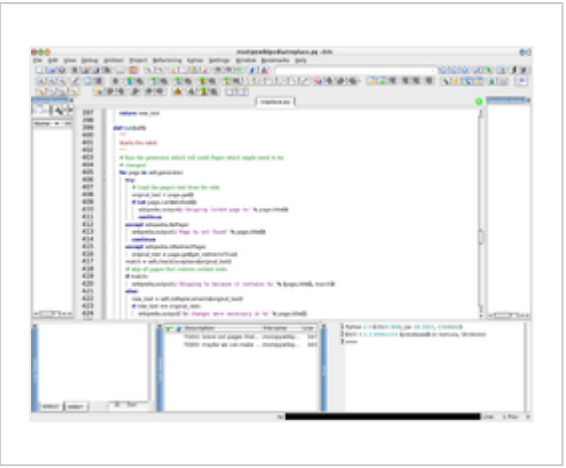
- Integrated interface to cx_freeze (as optional plug-in)
- Many integrated wizards for regex and Qt dialogs (as core plug-ins)
- Tools for previewing Qt forms and translations

Support for Python 2 and 3

Prior to the release of eric version 5.5.0, eric version 4 and eric version 5 coexisted and were maintained simultaneously, while eric 4 was the variant for writing software in Python version 2 and eric version 5 was the variant for writing software in Python version 3.

With the release of eric version 5.5.0 both variants had been merged into one, so that all versions as of eric version 5.5.0 support writing software in Python 2 as well as in Python 3, making the separate development lanes of eric version 4 and 5 obsolete. Those two separate development lanes are no longer maintained, and the last versions prior to merging them both to 5.5.0 were versions 4.5.25 and 5.4.7.[8]

Gallery



eric 3 on Arch Linux with Xfce 4.4

eric 4

Releases

Versioning scheme

Until 2016, eric used a software versioning scheme with a three-sequence identifier, e.g. 5.0.1. The first sequence represents the major version number which is increased when there are significant jumps in functionality, the second sequence represents the minor number, which is incremented when only some features or significant fixes have been added, and the third sequence is the revision number, which is incremented when minor bugs are fixed or minor features have been added.

From late 2016, the version numbers show the year and month of release, e.g. 16.11 for November 2016.[9]

Release strategy

eric follows the development philosophy of *Release early, release often*, following loosely a time-based release schedule. Currently a revision version is released around the first weekend of every month, a minor version is released annually, in most cases approximately between December and February.

Version history

The following table shows the version history of eric, starting from version 4.0.0. Only major (e.g. 6.0.0) and minor (e.g. 6.1.0) releases are listed; revision releases (e.g. 6.0.1) are omitted.

Old version, not maintained | Old version, still maintained | **Latest version**
Future version

Branch	Version	Release date	Major changes
4	4.0.0	2007-06-03	
	4.1.0	2008-02-03	▪ Added plug-in system[10] ▪ Added support for **Django** and TurboGears (later transformed to plug-ins)[10]
	4.2.0	2008-08-09	▪ Added a toolbar manager
	4.3.0	2009-02-08	▪ Changed license to GPL "v3 or later"
	4.4.0	2010-01-09	▪ Changed the help viewer to a full blown web browser (based on QtWebKit)[11]
	4.5.0	2012-02-04	▪ Added Mac OS X to the officially supported platforms[12]
5	5.0.0	2010-07-04	▪ The eric 5 branch represents the new **Python 3** variant of eric. It is *not* supporting **Python 2** (yet). For Python 2 support the development of the eric 4 branch is continued.[11]
	5.1.0	2011-02-27	
	5.2.0	2012-02-18	▪ Added Mac OS X to the officially supported platforms[12]
	5.3.0	2013-02-03	▪ Added support for **Qt5**[13]
	5.4.0	2014-01-07	▪ Added support for **PyQt5** projects[8]
	5.5.0	2014-10-27	▪ Added **Python 2** support to the eric 5 branch, so that from now on both, **Python 3** and **Python 2**, are supported by one single version of eric, making the eric 4 branch for Python 2 obsolete, which is not further continued[8]
6	6.0.0	2014-12-28	▪ eric 6 replaces the eric 5.5.x line of development. It is usable with any combination of **Python 2** or **Python 3**, **Qt5** or **Qt4** and **PyQt5** or **PyQt4**, on Linux, Mac OS X and Windows platforms
	6.1.0	2015-12-05	▪ Added multithreading support for checkers to make use of multiple CPUs/CPU-Cores[14]
16	16.11	2016-11-12	▪ Switching the release scheme[9]
	16.12	2016-12-03	
17	17.01	2017-01-01	
	17.02	2017-02-04	

	17.03	2017-03-03	
	17.04	2017-04-07	▪ Minimum required Python versions increased: Python 2 - 2.7.10; Python 3 - 3.4.0
	17.05	2017-05-06	
	17.06	2017-06-03	
	17.07	2017-07-02	
	17.08	2017-08-03	
	17.09	2017-09-01	
	17.10	2017-10-07	
	17.11	2017-11-03	
	17.12	2017-12-02	
18	**18.01**	2018-01-06	
	18.02	2018-02-03	▪ Added support for attributes introduced with Qt 5.9 and Qt 5.10[15] ▪ New session file format
	18.03	2018-03-04	
	18.04	2018-04-02	
	18.05	2018-05-01	
	18.06	2018-06-02	
	18.07	2018-07-07	
	18.08	2018-08-02	
	18.09	2018-09-02	
	18.10	2018-10-03	
	18.11	2018-11-01	
	18.12	2018-12-01	
19	**19.01**	2019-01-10	

	19.02	2019-02-02
	19.03	2019-03-02
	19.04	2019-04-06
	19.05	2019-05-04
	19.06	2019-06-02
	19.07	2019-07-07
	19.08	2019-08-03
	19.09	2019-09-07
	19.10	2019-10-03
	19.11	2019-11-01
	19.12	2019-12-07
20	**20.01**	2020-01-01
	20.02	2020-02-02

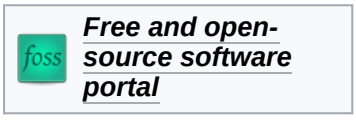
Name

Several allusions are made to the British comedy group Monty Python, which the Python programming language is named after. Eric alludes to Eric Idle, a member of the group, as does IDLE, the standard python IDE shipped with most distributions.[16]

Criticism

The Eric Python IDE does not feature an integrated toolchain for now.

See also



- Comparison of integrated development environments for Python

References

1. https://eric-ide.python-projects.org/eric-news.html (https://eric-ide.python-projects.org/eric-news.html). `{{cite web}}`: Missing or empty `|title=` (help)
2. Sourceforge: Eric Integrated Development Environment (https://sourceforge.net/projects/eric-ide)
3. Ubuntu package search: eric (http://packages.ubuntu.com/search?keywords=eric&searchon=names&suite=all&section=all)
4. Official website: Access information for mercurial repository (https://eric-ide.python-projects.org/eric-code.html)
5. Reitz, Kenneth; Schlusser, Tanya (August 30, 2016). *The Hitchhiker's Guide to Python: Best Practices for Development* (https://books.google.com/books?id=nHDtDAAAQBAJ&q=Eric+ide&pg=PA31). O'Reilly Media, Inc. p. 31. ISBN 9781491933237. Retrieved January 18, 2019.
6. Charney, Reg (August 30, 2004). "Programming Tools: Eric3" (https://www.linuxjournal.com/article/7739). Linux Journal. Retrieved January 18, 2019.
7. eric-ide.python-projects.org: (https://eric-ide.python-projects.org) Features
8. eric news 2014 (https://eric-ide.python-projects.org/eric-news-2014.html)
9. eric news 2016 (https://eric-ide.python-projects.org/eric-news-2016.html)
10. eric news 2007 (https://eric-ide.python-projects.org/eric-news-2007.html)
11. eric news 2010 (https://eric-ide.python-projects.org/eric-news-2010.html)
12. eric news 2012 (https://eric-ide.python-projects.org/eric-news-2012.html)
13. eric news 2013 (https://eric-ide.python-projects.org/eric-news-2013.html)
14. eric news 2015 (https://eric-ide.python-projects.org/eric-news-2015.html)
15. eric news 2018 (https://eric-ide.python-projects.org/eric-news.html)
16. Bidwell, Jonni (April 14, 2018). "Best IDE for Python in 2018" (https://www.techradar.com/news/best-ide-for-python). TechRadar. Retrieved January 18, 2019.

External links

- Official website (https://eric-ide.python-projects.org)

IDLE

IDLE (short for **Integrated Development and Learning Environment**)[2][3] is an integrated development environment for Python, which has been bundled with the default implementation of the language since 1.5.2b1.[4][5] It is packaged as an optional part of the Python packaging with many Linux distributions. It is completely written in Python and the Tkinter GUI toolkit (wrapper functions for Tcl/Tk).

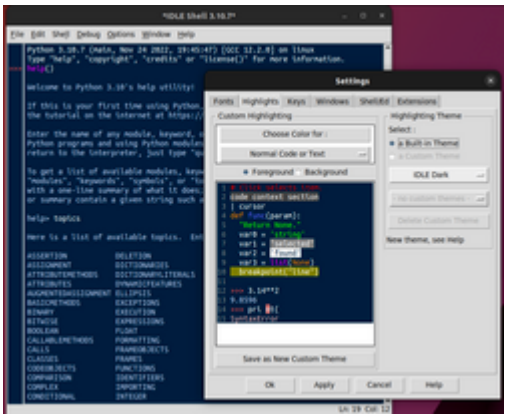
IDLE is intended to be a simple IDE and suitable for beginners, especially in an educational environment. To that end, it is cross-platform, and avoids feature clutter.

According to the included README, its main features are:

- Multi-window text editor with syntax highlighting, autocompletion, smart indent and other features.
- Python shell with syntax highlighting.
- Integrated debugger with stepping, persistent breakpoints, and call stack visibility.

Author Guido van Rossum says IDLE stands for "Integrated Development and Learning Environment",[6] and since Van Rossum named the language Python after the British comedy group Monty Python, the name IDLE was probably also chosen partly to honor Eric Idle, one of Monty Python's founding members.[7][8]

IDLE



IDLE in action under Ubuntu: shell with highlights settings

Original author(s)	Guido van Rossum
Initial release	December 22, 1998
Stable release	3.12.9[1] / 4 February 2025
Repository	github.com/python/cpython/tree/master/Lib/idlelib (https://github.com/python/cpython/tree/master/Lib/idlelib)
Written in	Python
Type	Integrated development environment
Website	docs.python.org/library/idle.html (https://docs.python.org/library/idle.html)

See also

Free and open-source software portal

- List of integrated development environments for Python

References

1. "Changelog" (https://docs.python.org/release/3.12.9/whatsnew/changelog.html). Retrieved 1 April 2025.
2. From the Help > About screen
3. "IDLE — Python 3.9.5 documentation" (https://web.archive.org/web/20200604195234/https://docs.python.org/3/library/idle.html). Archived from the original (https://docs.python.org/3/library/idle.html) on 2020-06-04. Retrieved 2020-06-04.
4. Subject: IDLE 0.1 -- a Python IDE (https://lwn.net/1998/1119/idle.html) Archived (https://web.archive.org/web/20180925142146/https://lwn.net/1998/1119/idle.html) 2018-09-25 at the Wayback Machine, By Guido van Rossum - 16 Nov 1998 - comp.lang.python, *At the conference I mentioned a few times that I was working on a Tkinter-based IDE for Python. I've decided to use the paradigm "release early and often" for this piece of software (especially since I don't expect I'll have much time to work on it), so version 0.1 (essentially a dump of my directory) is now sitting in the contrib directory ftp.python.org.*
5. IDLE 0.1 was distributed with the Python 1.5.2b1 release on 12/22/98. (https://web.archive.org/web/20191101181226/https://hg.python.org/cpython/file/tip/Lib/idlelib/HISTORY.txt), From: *\Python-1.5.2\Tools\idle\NEWS.txt*
6. "IDLE — Python 3.9.2 documentation" (https://docs.python.org/3/library/idle.html). *docs.python.org*. Retrieved 2021-02-26.
7. Lutz, Mark & Ascher, David (2004). *Learning Python*, p. 40. O'Reilly Media, Inc. ISBN 978-0-596-00281-7.
8. Hammond, Mark; Robinson, Andy (2000). *Python programming on Win32* (https://books.google.com/books?id=fzUCGtyg0MMC&pg=PA59) (1. ed.). O'Reilly Media, Inc. p. 59. ISBN 978-1-56592-621-9.

External links

- Media related to IDLE at Wikimedia Commons
- IDLE home page in the Python documentation (https://docs.python.org/library/idle.html)
- IDLE page in the Python wiki (https://wiki.python.org/moin/IDLE)
- A guide to using IDLE (https://web.archive.org/web/20150901103436/http://www.annedawson.net/Python_Editor_IDLE.htm)

Ninja-IDE

NINJA-IDE (from the recursive acronym: "Ninja-IDE Is Not Just Another IDE"), is a cross-platform integrated development environment (IDE) designed to build Python applications.

It provides tools to simplify Python software development and handles many kinds of situations thanks to its rich extensibility.

Features

Some of the current features of the IDE are:

- Light weight IDE
- Common functions such as: file handling, find in files code locator, go to line, tabs, automatic indentation, editor zoom, etc.
- Multi-platform: Linux, Windows, FreeBSD.
- Syntax highlighting for a wide variety of languages. Even though it is intended to be mainly a Python IDE, it can also handle several other languages.
- Static and PEP 8 error highlighting.
- Show tips to help migrate code from Python2 to Python3.
- Embedded Python console.
- Project management, allowing to add, modify and delete files and folders to projects, creating automatically the "__init__.py" files inside each module, etc.
- Allows showing/hiding the panels of the interface in a very simple way to fit each programmer's preferences.
- Completely configurable UI.
- Allows using more than one editor at once.
- An extensible plug-in system, which creation the IDE supports.
- Session handling: remembers opened files and projects after closing the IDE.
- Code Auto-completion.
- Code Locator: Lets you jump to any code in your project with just a few keystrokes.

NINJA-Compiler

Screenshot of NINJA-IDE 2.3

Developer(s)	Diego Sarmentero, Horacio Durán, Gabriel Acosta, Pedro Mourelle, Jose Rostagno
Stable release	2.4[1] / 23 June 2019[2]
Repository	github.com/ninja-ide/ninja-ide (https://github.com/ninja-ide/ninja-ide)
Written in	Python
Operating system	Cross-platform: Linux, Mac OS X, Windows, FreeBSD
Platform	CPython, PyQt
Available in	Multilingual
Type	Software development
License	GNU GPL 3
Website	ninja-ide.org (http://ninja-ide.org)

Versions names

NINJA-IDE always takes its version name based on the name of a weapon.

Previous Versions:

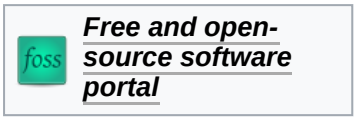
- 2.x: **Shuriken**
- 1.x: **Kunai**

Some available plugins

Many plugins are being developed, mostly with the help of the Community. A complete list of Plugins can be found here:

- NINJA-IDE Plugins (http://ninja-ide.org/plugins/)

See also



- Comparison of integrated development environments for Python

References

1. ninja-ide.org Downloads (http://ninja-ide.org/downloads/)
2. GitHub (2019-06-23), *Ninja-IDE 2.4 released* (https://github.com/ninja-ide/ninja-ide/releases/tag/v2.4)

External links

- Official website (http://ninja-ide.org)
- Python IDE Wiki (https://wiki.python.org/moin/IntegratedDevelopmentEnvironments)
- Python IDE (PyAr) (http://python.org.ar/IDEs)
- Linux Magazine: Do Python coding with NINJA-IDE (http://www.linux-magazine.com/Online/Blogs/Productivity-Sauce/Do-Python-Coding-with-NINJA-IDE)
- NINJA-IDE a powerful IDE for developing Python Apps (https://web.archive.org/web/20151123134310/http://webresourcesdepot.com/ninja-ide-a-powerful-ide-for-developing-python-apps/)
- NINJA-IDE, el ide que me atrapo (in Spanish) (http://braindumpve.wordpress.com/2013/05/11/ninja-ide-el-ide-que-me-atrapo/)
- NINJA-IDE, un IDE pensado para Python (in Spanish) (http://www.diegosarmentero.com/2010/12/ninja-ide-un-ide-pensado-para-python.html)

PyCharm

PyCharm is an integrated development environment (IDE) used for programming in Python. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems, and supports web development with Django. PyCharm is developed by the Czech company JetBrains and built on their IntelliJ platform.[4]

It is cross-platform, working on Microsoft Windows, macOS, and Linux. PyCharm has a Professional Edition, released under a proprietary license and a Community Edition released under the Apache License.[5] PyCharm Community Edition is less extensive than the Professional Edition.[6]

Features

In both versions

- Python coding assistance and analysis, with code completion, syntax and error highlighting, linter integration, and quick fixes
- Project and code navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages
- Python code refactoring: including rename, update function signature, extract method, introduce variable, introduce constant, pull up, push down and others
- Integrated Python debugger
- Integrated unit testing, with line-by-line coverage
- Virtual environment, build tool and package management
- Embedded terminal and Python console
- Docker support
- HTML,[7] XML, JSON, YAML, Markdown support
- Spell- and grammar-checking[8]

PyCharm

PyCharm 2023.2 Community Edition

Developer(s)	JetBrains
Initial release	3 February 2010
Stable release	2024.3.2[1] / 28 January 2025
Written in	Java, Python
Operating system	Windows, macOS, Linux
Size	174–555 MB
Type	Python IDE
License	Community edition: Apache License 2.0 Professional edition: Trialware
Website	www.jetbrains.com/pycharm/ (https://www.jetbrains.com/pycharm/)

PyCharm Edu

Developer(s)	JetBrains
Initial release	30 October 2014[2]
Final release	2022.2.5 (Build: 222.4554.11) / 16 March

- Version control integration: unified user interface for Mercurial, Git, Subversion, Perforce and CVS with changelists and merge, integration with GitHub and GitLab hosting services

Only in the Professional version

- Scientific tools integration: integrates with Jupyter Notebook, supports Anaconda as well as multiple scientific packages including Matplotlib and NumPy.
- Front-end and back-end web development: special support for Django,[9] Flask,[10] FastAPI[11] and Pyramid,[12] CSS[13] and JavaScript[14] assistance, Npm, Webpack and other JavaScript tools
- SQL and database utilities[15]
- Cython support[16][17][18][6]

	2023[3]
Written in	Java, Python
Operating system	Windows, macOS, Linux
Size	320–430 MB
Type	IDE
License	Apache License 2.0
Website	www.jetbrains.com/pycharm-edu/ (https://www.jetbrains.com/pycharm-edu/)

History

PyCharm was released to the market of the Python-focused IDEs to compete with PyDev (for Eclipse) or the more broadly focused Komodo IDE by ActiveState.

The beta version of the product was released in July 2010, with the 1.0 arriving 3 months later. Version 2.0 was released on December 13, 2011, version 3.0 was released on September 24, 2013, and version 4.0 was released on November 19, 2014.[19]

PyCharm became open source on October 22, 2013. The open source variant is released under the name *Community Edition* while the commercial variant, *Professional Edition*, contains closed-source modules.[5]

As of December 2022, JetBrains has discontinued PyCharm Edu and IntelliJ IDEA Edu. The educational functionality is now bundled with the Community and Professional editions of IntelliJ IDEA and PyCharm.[3] Users are encouraged to install the Community or Professional editions and enable educational features through the IDE settings.

Licensing

- PyCharm *Professional Edition* is commercial, proprietary software and is gratis for open-source projects and for some educational uses.[20] It is paid on a subscription basis, though after paying for one year a "Perpetual Fallback License" will be granted for the version which was available one year before ending the subscription.[21]
- PyCharm *Community Edition* is distributed under Apache 2 license. The source code is available on GitHub.[22]

Limitations

The PyCharm Python IDE does not feature a GUI builder for now. While there is no native GUI builder provided within PyCharm, by using PySide6/PyQt6 (the Python bindings to Qt V6), one can gain access to the Qt Widget Designer graphical UI builder. This is currently available with the PyCharm community edition and provides an advantage over the use of Tkinter which is bundled natively with Python and does not offer a GUI designer tool.

See also

- Comparison of Python integrated development environments
- List of Python software

References

1. "PyCharm 2024.3.2: uv Package Management Support and More! | The PyCharm Blog" (https://blog.jetbrains.com/pycharm/2025/01/pycharm-2024-3-2/). 28 January 2025. Retrieved 1 April 2025.
2. "JetBrains Debuts PyCharm Educational Edition" (https://blog.jetbrains.com/pycharm/2014/10/jetbrains-debuts-pycharm-educational-edition/). *JetBrains Blog*. 10 October 2014. Archived (https://web.archive.org/web/20221127122450/https://blog.jetbrains.com/pycharm/2014/10/jetbrains-debuts-pycharm-educational-edition/) from the original on 27 November 2022. Retrieved 5 March 2019.
3. "Sunsetting Educational IDEs" (https://blog.jetbrains.com/education/2022/12/01/sunsetting-educational-ides/). *JetBrains Blog*. December 1, 2022. Archived (https://web.archive.org/web/20240608060029/https://blog.jetbrains.com/education/2022/12/01/sunsetting-educational-ides/) from the original on 8 June 2024. Retrieved 8 June 2024.
4. "JetBrains Strikes Python Developers with PyCharm 1.0 IDE" (https://archive.today/20130122124720/http://www.eweek.com/c/a/Application-Development/JetBrains-Strikes-Python-Developers-with-PyCharm-10-IDE-304127/). eWeek. Archived from the original (http://www.eweek.com/c/a/Application-Development/JetBrains-Strikes-Python-Developers-with-PyCharm-10-IDE-304127/) on January 22, 2013.
5. PyCharm 3.0 community edition source code now available (http://blog.jetbrains.com/pycharm/2013/10/pycharm-3-0-community-edition-source-code-now-available/) Archived (https://web.archive.org/web/20131022144923/http://blog.jetbrains.com/pycharm/2013/10/pycharm-3-0-community-edition-source-code-now-available/) 2013-10-22 at the Wayback Machine Jet Brains. October 2013.
6. "JetBrains Products Comparison" (https://www.jetbrains.com/products/compare/?product=pycharm&product=pycharm-ce). *JetBrains*. Retrieved 2024-09-04.
7. "Working with HTML files | PyCharm" (https://www.jetbrains.com/help/pycharm/editing-html-files.html#ws_html_generate_reference). *PyCharm Help*. Retrieved 2024-09-04.
8. "Grazie Lite - IntelliJ IDEs Plugin | Marketplace" (https://plugins.jetbrains.com/plugin/12175-grazie-lite). *JetBrains Marketplace*. Retrieved 2024-09-04.
9. "Create and run your first Django project | PyCharm" (https://www.jetbrains.com/help/pycharm/creating-and-running-your-first-django-project.html). *PyCharm Help*. Retrieved 2024-09-04.

10. "Creating a Flask Project | PyCharm" (https://www.jetbrains.com/help/pycharm/creating-flask-project.html). *PyCharm Help*. Retrieved 2024-09-04.
11. "FastAPI | PyCharm" (https://www.jetbrains.com/help/pycharm/fastapi-project.html). *PyCharm Help*. Retrieved 2024-09-04.
12. "Pyramid | PyCharm" (https://www.jetbrains.com/help/pycharm/pyramid.html). *PyCharm Help*. Retrieved 2024-09-04.
13. "Style Sheets | PyCharm" (https://www.jetbrains.com/help/pycharm/style-sheets.html). *PyCharm Help*. Retrieved 2024-09-04.
14. "JavaScript | PyCharm" (https://www.jetbrains.com/help/pycharm/javascript-specific-guidelines.html). *PyCharm Help*. Retrieved 2024-09-04.
15. "Database Tools and SQL | PyCharm" (https://www.jetbrains.com/help/pycharm/relational-databases.html). *PyCharm Help*. Retrieved 2024-09-04.
16. "Cython support | PyCharm" (https://www.jetbrains.com/help/pycharm/cython.html). *PyCharm Help*. Retrieved 2024-09-04.
17. "What is PyCharm | Where do we Use PyCharm? | Features" (https://www.educba.com/what-is-pycharm/). *EDUCBA*. 2021-11-04. Archived (https://web.archive.org/web/20230828154027/https://www.educba.com/what-is-pycharm/) from the original on 2023-08-28. Retrieved 2023-08-28.
18. "Explore PyCharm Features - JetBrains' Leading Python IDE" (https://www.jetbrains.com/pycharm/features/). Archived (https://web.archive.org/web/20170509025940/https://www.jetbrains.com/pycharm/features/) from the original on 2017-05-09. Retrieved 2016-09-20.
19. Filippov, Dmitry (November 19, 2014). "Announcing General Availability of PyCharm 4" (http://blog.jetbrains.com/pycharm/2014/11/announcing-general-availability-of-pycharm-4/). *PyCharm Blog*. Archived (https://web.archive.org/web/20150224224651/http://blog.jetbrains.com/pycharm/2014/11/announcing-general-availability-of-pycharm-4/) from the original on 24 February 2015. Retrieved 24 February 2015.
20. PyCharm Students & Teachers Pricing (https://www.jetbrains.com/pycharm/buy/?section=students), Jet Brains website.
21. "What is a perpetual fallback license?" (https://sales.jetbrains.com/hc/en-gb/articles/207240845-What-is-a-perpetual-fallback-license). *Licensing and Purchasing FAQ*. Retrieved 2024-09-04.
22. PyCharm Community Edition (https://github.com/JetBrains/intellij-community/tree/master/python#readme) Archived (https://web.archive.org/web/20161208173648/https://github.com/JetBrains/intellij-community/tree/master/python#readme) 2016-12-08 at the Wayback Machine on GitHub.

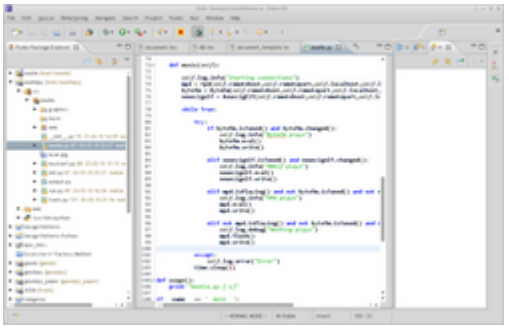
External links

- Official website (http://www.jetbrains.com/pycharm/)

PyDev

PyDev is a third-party plug-in for Eclipse. It is an Integrated Development Environment (IDE) used for programming in Python supporting code refactoring, graphical debugging, code analysis among other features.

PyDev



Original author(s)	Aleks Totic
Developer(s)	Appcelerator
Initial release	July 2003
Stable release	13.0.2[1] / 2 March 2025
Repository	github.com/fabioz/Pydev (https://github.com/fabioz/Pydev)
Written in	Java, Python
Operating system	Cross-platform
Type	Integrated development environment
License	Eclipse Public License
Website	pydev.org (http://pydev.org)

History

PyDev was originally created by Aleks Totic in July 2003, but Fabio Zadrozny became the project's main developer in January 2005. In September of that same year, PyDev Extensions was started as a commercial counterpart of PyDev, offering features such as code analysis and remote debugging.

In July 2008, Aptana acquired PyDev, retaining Zadrozny as the project head.[2] They open sourced PyDev Extensions in September 2009, and merged it with PyDev.[3]

When Appcelerator acquired Aptana in January 2011, they acquired PyDev by extension. Zadrozny was kept as head of the project.[4] Since then, development of PyDev has accelerated.

In March 2011, PyDev 2.0 was released with TDD actions support, and in April of the following year, version 2.5 was released with Django support. May 2013 saw a major milestone as PyDev raised more than its target in a successful crowd sourcing round to continue development, and version 2.7.5 was released. The campaign also funded Zadrozny's creation of LiClipse, a paid closed source fork of Eclipse which bundles PyDev by default.[5]

PyDev received improvements to type inference and a notable increase in contributions to code base when version 2.8 was released in July 2013.[6] Since then, numerous additional improvements have been made to PyDev and it has gained many positive reviews.[7][8]

Version 5.4.0 was released on November 30, 2016. The main new feature of this release is support for Python 3.6.[9]

Features

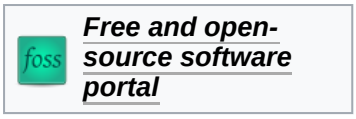
Below there are some of the features available (version 2.7.5):

- **CPython**, **Jython** and **IronPython** support
- **Code completion**
- **Code completion with auto-import**
- **Code analysis** (with quick-fix for problems found in code analysis—Ctrl+1)
- **Debugger**
- **Django**
- **Remote Debugger** (allows debugging scripts not launched from within Eclipse)
- **Debug console** (allows interactive probing in suspended mode)
- **Interactive console**
- Python 2.x and 3.x syntax
- Basic syntax highlighting
- Parser errors
- Outline view
- Tabs or spaces preferences
- Smart indent / dedent
- Comment / uncomment / comment blocks
- Code folding
- Go to definition
- Code coverage
- Mark occurrences
- Pylint integration
- TODO tasks
- Content Assistants (Ctrl+1)
  - Assign result to attribute or local
  - Surround code with try..catch / finally
  - Create docstring
  - Move import to global scope
- Keywords presented as auto-completions as you type
- Quick-outline

PyDev extensions

Until September 2009, two versions of PyDev existed: an open-source version, and a shareware version called PyDev Extensions. Certain advanced features such as code analysis, quick-fixes, and remote debugging were reserved for the non-free version. On September 3, 2009, Aptana announced PyDev version 1.5, a combined version of PyDev and PyDev Extensions, all available under the Eclipse Public License.

See also



- Eclipse
- Comparison of Python integrated development environments
- Komodo Edit
- PyCharm

References

1. "Release 13.0.2" (https://github.com/fabioz/Pydev/releases/tag/pydev_13_0_2). 2 March 2025. Retrieved 31 March 2025.
2. Aptana. "Aptana Acquires PyDev" (https://web.archive.org/web/20120420132905/http://aptana.ulitzer.com/node/654422/mobile). Archived from the original (http://aptana.ulitzer.com/node/654422/mobile) on 2012-04-20. Retrieved 2012-08-06.
3. Zadrozny, Fabio (3 September 2009). "PyDev Extensions Open Sourced" (http://pydev.blogspot.com.br/2009/09/pydev-150-pydev-extensions-open-sourced.html).
4. Appcelerator. "Appcelerator Acquires Aptana" (http://developer.appcelerator.com/blog/2011/01/appcelerator-acquires-aptana.html).
5. "PyDev and LiClipse for a Fast, Sexy -- and Dark Eclipse" (http://www.indiegogo.com/projects/352570/fblk). *Indiegogo*.
6. Zadrozny, Fabio (July 25, 2013). "PyDev adventures: PyDev 2.8.0 released" (http://pydev.blogspot.com/2013/07/pydev-280-released.html).
7. Fruit, Jason (14 January 2013). "Comparison of Python IDEs for Development" (http://www.pythoncentral.io/comparison-of-python-ides-development/#pydev).
8. "Evaluating IDEs for scientific Python" (https://xcorr.net/2013/04/17/evaluating-ides-for-scientific-python/). April 18, 2013.
9. "PyDev 5.4.0 (Python 3.6, Patreon crowdfunding)" (https://pydev.blogspot.nl/2016/11/pydev-540-python-36-patreon-crowdfunding.html). *pydev.blogspot.nl*. 30 November 2016. Retrieved 2016-12-10.

External links

- Official website (http://pydev.org)

Spyder (software)

Spyder is an open-source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates with a number of prominent packages in the scientific Python stack, as well as other open-source software.[4][5] Created by Pierre Raybaut[6] and released in 2009[1][2] under the MIT license,[7] since 2012 Spyder has been maintained and continuously improved by Python developers and the community.

Spyder is extensible with first-party and third-party plugins,[8] and includes support for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments, such as Pyflakes, Pylint[9] and Rope.[10][11] Spyder uses Qt for its GUI and is designed to use either of the PyQt or PySide Python bindings.[12] QtPy, a thin abstraction layer developed by the Spyder project and later adopted by multiple other packages, provides the flexibility to use either backend.[13]

History

Initially created and developed by Pierre Raybaut,[6] it was published on October 18, 2009[1][2] under the MIT license.[7]

Since 2012 Spyder has been maintained and continuously improved by a team of scientific Python developers and the community. As of 2024, the Spyder website lists the Chan Zuckerberg Initiative and NumFOCUS as their two major sponsors, also noting donations received from users through Open Collective.[14] Carlos Cordoba was listed as the lead maintainer of the software, with Daniel Althiz as co-maintainer.[6]

Spyder

Screenshot of Spyder on Windows

Original author(s)	Pierre Raybaut
Developer(s)	Spyder project contributors (https://github.com/spyder-ide/spyder/graphs/contributors)
Initial release	18 October 2009[1][2]
Stable release	6.0.5[3] / 27 March 2025
Repository	github.com/spyder-ide/spyder (https://github.com/spyder-ide/spyder)
Written in	Python
Operating system	Cross-platform
Platform	Qt, Windows, macOS, Linux
Type	Integrated development environment
License	MIT
Website	www.spyder-ide.org (https://www.spyder-ide.org/)

Software

It is an open-source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates with a number of prominent packages in the scientific Python stack, including NumPy, SciPy, Matplotlib, pandas, IPython, SymPy and Cython, as well as other open-source software.[4][5]

Spyder is extensible with first-party and third-party plugins,[8] includes support for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments, such as Pyflakes, Pylint[9] and Rope. It is available cross-platform through Anaconda, on Windows, on macOS through MacPorts, and on major Linux distributions such as Arch Linux, Debian, Fedora, Gentoo Linux, openSUSE and Ubuntu.[10][11]

Spyder uses Qt for its GUI and is designed to use either of the PyQt or PySide Python bindings.[12] QtPy, a thin abstraction layer developed by the Spyder project and later adopted by multiple other packages, provides the flexibility to use either backend.[13]

Features

Features include:[15]

- An editor with syntax highlighting, introspection, code completion
- Support for multiple IPython consoles
- The ability to explore and edit variables from a GUI
- A Help pane able to retrieve and render rich text documentation on functions, classes and methods automatically or on-demand
- A debugger linked to IPdb, for step-by-step execution
- Static code analysis, powered by Pylint
- A run-time Profiler, to benchmark code
- Project support, allowing work on multiple development efforts simultaneously
- A built-in file explorer, for interacting with the filesystem and managing projects
- A "Find in Files" feature, allowing full regular expression search over a specified scope
- An online help browser, allowing users to search and view Python and package documentation inside the IDE
- A history log, recording every user command entered in each console
- An internal console, allowing for introspection and control over Spyder's own operation

Plugins

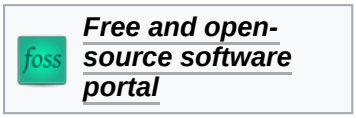
Available plugins include:[16]

- Spyder-Unittest, which integrates the popular unit testing frameworks Pytest, Unittest and Nose with Spyder
- Spyder-Notebook, allowing the viewing and editing of Jupyter Notebooks within the IDE
    - Download Spyder Notebook

  - Using conda: *conda install spyder-notebook -c spyder-ide*
  - Using pip: *pip install spyder-notebook*

- Spyder-Reports, enabling use of literate programming techniques in Python
- Spyder-Terminal, adding the ability to open, control and manage cross-platform system shells within Spyder
  - Download Spyder Terminal
  - Using conda: *conda install spyder-terminal -c spyder-ide*
  - Using pip: *pip install spyder-terminal*
- Spyder-Vim, containing commands and shortcuts emulating the Vim text editor
- Spyder-AutoPEP8, which can automatically conform code to the standard PEP 8 code style
- Spyder-Line-Profiler and Spyder-Memory-Profiler, extending the built-in profiling functionality to include testing an individual line, and measuring memory usage

See also



- Comparison of integrated development environments for Python

References

1. "spyder-ide/spyder at v1.0.0" (https://github.com/spyder-ide/spyder/tree/v1.0.0). GitHub. Retrieved 3 April 2017.
2. "(Python)(ANN) Spyder v1.0.0 released" (https://www.riverbankcomputing.com/pipermail/pyqt/2009-October/024764.html). 18 October 2009.
3. "Release 6.0.5" (https://github.com/spyder-ide/spyder/releases/tag/v6.0.5). 27 March 2025. Retrieved 31 March 2025.
4. "Migrating from MATLAB to Python" (https://web.archive.org/web/20141010221143/http://web.ics.purdue.edu:80/~smit1447/blog/?p=24). *Greener Engineering*. et.byu.edu. Archived from the original (http://web.ics.purdue.edu/~smit1447/blog/?p=24) on 2014-10-10. Retrieved 9 February 2014.
5. "Spyder review" (https://web.archive.org/web/20131203014000/http://review.techworld.com/applications/3238833/spyder-review/). review.techworld.com. Archived from the original (http://review.techworld.com/applications/3238833/spyder-review/) on 3 December 2013. Retrieved 9 February 2014.
6. "About" (https://www.spyder-ide.org/about/). *spyder-ide.org*. 2024. Retrieved December 2, 2024.
7. "Spyder license" (https://github.com/spyder-ide/spyder/blob/master/LICENSE.txt). *GitHub*.
8. "SpyderPlugins – spyderlib – Plugin development – Spyder is the Scientific PYthon Development EnviRonment" (https://web.archive.org/web/20131024165518/http://code.google.com/p/spyderlib/wiki/SpyderPlugins). Archived from the original (https://code.google.com/p/spyderlib/wiki/SpyderPlugins) on 24 October 2013. Retrieved 9 February 2014.
9. "Pylint extension – Spyder 2.2 documentation" (https://web.archive.org/web/20140201022109/http://packages.python.org/spyder/pylint.html). packages.python.org. Archived from the original (http://packages.python.org/spyder/pylint.html) on 1 February 2014. Retrieved 9 February 2014.

10. "Reviews for spyder" (https://apps.ubuntu.com/cat/applications/oneiric/spyder/reviews/). apps.ubuntu.com. Retrieved 9 February 2014.
11. "Seznámení s Python IDE Spyder" (https://web.archive.org/web/20130820121204/http://fedora.cz/seznameni-s-python-ide-spyder/). fedora.cz. Archived from the original (http://fedora.cz/seznameni-s-python-ide-spyder/) on 20 August 2013. Retrieved 9 February 2014.
12. "Spyder runtime dependencies" (https://github.com/spyder-ide/spyder/blob/master/README.md#runtime-dependencies). github.com. 21 February 2015.
13. "QtPy: Abstraction layer for PySide/PyQt4/PyQt5" (https://github.com/spyder-ide/qtpy/blob/master/README.md). github.com. 23 October 2015. Retrieved 28 December 2015.
14. "Spyder website main page" (https://www.spyder-ide.org/). *spyder-ide.org*. 2024. Retrieved December 2, 2024.
15. "Spyder Documentation – Features Overview" (https://web.archive.org/web/20190123054617/http://docs.spyder-ide.org/overview.html). Spyder Project. Archived from the original (https://docs.spyder-ide.org/overview.html) on 2019-01-23. Retrieved 2018-07-30.
16. "Spyder Plugins List" (https://github.com/spyder-ide). Spyder Project. Retrieved 2018-07-30.

External links

- Official website (https://www.spyder-ide.org/)
- Documentation (https://docs.spyder-ide.org/)
- spyder (https://github.com/spyder-ide/spyder) on GitHub

Web Server Gateway Interface

The **Web Server Gateway Interface** (**WSGI**, pronounced *whiskey*[1][2] or *WIZ-ghee*[3]) is a simple calling convention for web servers to forward requests to web applications or frameworks written in the Python programming language. The current version of WSGI, version 1.0.1, is specified in Python Enhancement Proposal (PEP) 3333.[4]

WSGI was originally specified as PEP-333 in 2003.[5] PEP-3333, published in 2010, updates the specification for Python 3.

Background

In 2003, Python web frameworks were typically written against only CGI, FastCGI, mod_python, or some other custom API of a specific web server.[6] To quote PEP 333:

> Python currently boasts a wide variety of web application frameworks, such as Zope, Quixote, Webware, SkunkWeb, PSO, and Twisted Web -- to name just a few. This wide variety of choices can be a problem for new Python users, because generally speaking, their choice of web framework will limit their choice of usable web servers, and vice versa... By contrast, although Java has just as many web application frameworks available, Java's "servlet" API makes it possible for applications written with any Java web application framework to run in any web server that supports the servlet API.

WSGI was thus created as an implementation-neutral interface between web servers and web applications or frameworks to promote common ground for portable web application development.[4]

Specification overview

The WSGI has two sides:

- the server/gateway side. This is often running full web server software such as Apache or Nginx, or is a lightweight application server that can communicate with a webserver, such as flup (https://www.saddi.com/software/flup/).
- the application/framework side. This is a Python callable, supplied by the Python program or framework.

Between the server and the application, there may be one or more *WSGI middleware components*, which implement both sides of the API, typically in Python code.

WSGI does not specify how the Python interpreter should be started, nor how the application object should be loaded or configured, and different frameworks and webservers achieve this in different ways.

WSGI middleware

A WSGI middleware component is a Python callable that is itself a WSGI application, but may handle requests by delegating to other WSGI applications. These applications can themselves be WSGI middleware components.[7]

A middleware component can perform such functions as:[7]

- Routing a request to different application objects based on the target URL, after changing the environment variables accordingly.
- Allowing multiple applications or frameworks to run side-by-side in the same process
- Load balancing and remote processing, by forwarding requests and responses over a network
- Performing content post-processing, such as applying XSLT stylesheets

Examples

Example application

A WSGI-compatible "Hello, World!" application written in Python:

```
1  def application(environ, start_response):
2      start_response('200 OK', [('Content-Type', 'text/plain')])
3      yield b'Hello, World!\n'
```

Where:

- Line 1 defines a function[8] named `application`, which takes two parameters, `environ` and `start_response`. `environ` is a dictionary containing CGI environment variables as well as other request parameters and metadata under well-defined keys.[9] `start_response` is a callable itself, taking two positional parameters, `status` and `response_headers`.
- Line 2 calls `start_response`, specifying "200 OK" as the HTTP status and a "Content-Type" response header.
- Line 3 makes the function into a generator. The body of the response is returned as an iterable of byte strings.

Example of calling an application

A full example of a WSGI network server is outside the scope of this article. Below is a sketch of how one would call a WSGI application and retrieve its HTTP status line, response headers, and response body, as Python objects.[10] Details of how to construct the `environ` dict have been omitted.

```
from io import BytesIO

def call_application(app, environ):
    status = None
```

```
    headers = None
    body = BytesIO()

    def start_response(rstatus, rheaders):
        nonlocal status, headers
        status, headers = rstatus, rheaders

    app_iter = app(environ, start_response)
    try:
        for data in app_iter:
            assert status is not None and headers is not None, \
                "start_response() was not called"
            body.write(data)
    finally:
        if hasattr(app_iter, 'close'):
            app_iter.close()
    return status, headers, body.getvalue()

environ = {...}  # "environ" dict
status, headers, body = call_application(app, environ)
```

WSGI-compatible applications and frameworks

Numerous web frameworks support WSGI:

- bjoern (https://github.com/jonashaag/bjoern)
- BlueBream
- bobo[11]
- Bottle
- CherryPy
- Django[12]
- Eventlet[13]
- FastWSGI (https://github.com/jamesroberts/fastwsgi)
- Flask
- Falcon (web framework) [14]
- Gevent-FastCGI[15]
- Google App Engine's webapp2
- Gunicorn
- prestans[16]
- mod_wsgi for use with Apache[17]
- netius
- pycnic[18]
- Paste component WebOb is specifically a WSGI extension. It was adopted by the Pylons project.
- Pylons
- Pyramid
- restlite[19]
- Tornado
- Trac
- TurboGears
- Uliweb[20]

- uWSGI
- Waitress[21]
- web.py[22]
- web2py
- weblayer[23]
- Werkzeug[24]
- Radicale[25]

Currently wrappers are available for FastCGI, CGI, SCGI, AJP (using flup), twisted.web, Apache (using mod_wsgi or mod_python), Nginx (using ngx_http_uwsgi_module),[26] Nginx Unit (using the Python language module),[27] and Microsoft IIS (using WFastCGI,[28] isapi-wsgi,[29] PyISAPIe,[30] or an ASP gateway).

See also

- Asynchronous Server Gateway Interface (ASGI) – The spiritual successor to WSGI, adding support for asynchronous applications
- Rack – Ruby web server interface
- PSGI – Perl Web Server Gateway Interface
- SCGI – Simple Common Gateway Interface
- JSGI – JavaScript web server gateway interface

References

1. Simionato, Michele (June 11, 2007). "An Introduction to Web Programming with WSGI" (http://www.phyast.pitt.edu/~micheles/python/europython07/talk.html).
2. Edge, Jake (July 9, 2019). "Mucking about with microframeworks" (https://lwn.net/Articles/792882/). *LWN*.
3. Goldberg, Kevin (2016-05-09). "An Introduction to Python WSGI Servers for Performance | AppDynamics" (https://www.appdynamics.com/blog/engineering/an-introduction-to-python-wsgi-servers-part-1/). *Application Performance Monitoring Blog | AppDynamics*. Retrieved 2020-08-20.
4. "PEP 3333 - Python Web Server Gateway Interface v1.0.1" (https://www.python.org/dev/peps/pep-3333/). *Python.org*. Retrieved 2018-04-04.
5. "PEP 333 -- Python Web Server Gateway Interface v1.0" (https://www.python.org/dev/peps/pep-0333/). *Python.org*. Retrieved 2018-04-04.
6. "FrontPage - Python Wiki" (https://www.python.org/cgi-bin/moinmoin/WebProgramming). *Python.org*. Retrieved 2017-01-27.
7. "PEP 3333 -- Python Web Server Gateway Interface v1.0.1" (https://www.python.org/dev/peps/pep-3333/#middleware-components-that-play-both-sides). *Python.org*. Retrieved 2018-04-04.
8. i.e. "a function, method, class, or an instance with a `__call__` method"
9. "PEP 3333 -- Python Web Server Gateway Interface v1.0.1" (https://www.python.org/dev/peps/pep-3333/#environ-variables). *Python.org*. Retrieved 2018-04-04.

10. "Creating WSGI Middleware - Alan Christopher Thomas - Minted - PythonKC" (https://www.youtube.com/watch?v=afnDANxsaYo). YouTube. 2015-08-28. Archived (https://ghostarchive.org/varchive/youtube/20211212/afnDANxsaYo) from the original on 2021-12-12. Retrieved 2017-01-27.
11. "プエラリアジェルの効果は?" (http://bobo.digicool.com). *Bobo.digicool.com*. Retrieved 2017-01-27.
12. "Django without mod_python, and WSGI support | Weblog | Django" (https://www.djangoproject.com/weblog/2005/jul/18/local_server/). *Djangoproject.com*. 2005-07-18. Retrieved 2017-01-27.
13. "wsgi – WSGI server — Eventlet 0.20.1 documentation" (http://eventlet.net/doc/modules/wsgi.html). *Eventlet.net*. Retrieved 2017-01-27.
14. "Falcon - Bare-metal web API framework for Python" (https://falconframework.org). Retrieved 2017-10-22.
15. "gevent-fastcgi 1.0.2.1 : Python Package Index" (https://pypi.python.org/pypi/gevent-fastcgi). *Pypi.python.org*. 2015-12-06. Retrieved 2017-01-27.
16. "anomaly/prestans: A WSGI compliant REST micro-framework" (https://github.com/prestans/prestans/). *GitHub.com*. Retrieved 2017-01-27.
17. "Google Code Archive - Long-term storage for Google Code Project Hosting" (https://code.google.com/p/modwsgi/). *Code.google.com*. Retrieved 2017-01-27.
18. "Pycnic Framework" (http://pycnic.nullism.com). *Pycnic.nullism.com*. Retrieved 2017-01-27.
19. "theintencity/restlite: Light-weight RESTful server tools in Python" (https://github.com/theintencity/restlite). *GitHub.com*. Retrieved 2017-01-27.
20. "limodou/uliweb: Simple and easy use python web framework" (https://github.com/limodou/uliweb). *GitHub.com*. Retrieved 2017-01-27.
21. "waitress documentation" (https://docs.pylonsproject.org/projects/waitress/en/latest/). *docs.pylonsproject.org*. Retrieved 2018-09-26.
22. "Welcome to" (http://webpy.org/). *Web.py*. 2009-09-11. Retrieved 2017-01-27.
23. "weblayer — weblayer v0.4.3 documentation" (http://packages.python.org/weblayer). *Packages.python.org*. Retrieved 2017-01-27.
24. "Welcome | Werkzeug (The Python WSGI Utility Library)" (http://werkzeug.pocoo.org/). *Werkzeug.pocoo.org*. Retrieved 2017-01-27.
25. "CalDAV and CardDAV Server - A Simple Calendar and Contact Server" (http://radicale.org/). *Radicale.org*. Retrieved 2017-01-27.
26. "Module ngx_http_uwsgi_module" (http://nginx.org/en/docs/http/ngx_http_uwsgi_module.html). *Nginx.org*. Retrieved 2017-01-27.
27. "Configuration — NGINX Unit" (https://unit.nginx.org/configuration/#python). *Unit.nginx.org*. Retrieved 2023-05-04.
28. "Python Tools for Visual Studio - Documentation" (https://pytools.codeplex.com/wikipage?title=wfastcgi). *Pytools.codeplex.com*. Retrieved 2017-01-27.
29. "Google Code Archive - Long-term storage for Google Code Project Hosting" (https://code.google.com/p/isapi-wsgi/). *Code.google.com*. Retrieved 2017-01-27.
30. "Python ISAPI Extension for IIS download | SourceForge.net" (http://pyisapie.sourceforge.net/). *Pyisapie.sourceforge.net*. 2012-04-24. Retrieved 2017-01-27.

External links

- PEP 333 – Python Web Server Gateway Interface (https://www.python.org/dev/peps/pep-0333/)

- PEP 3333 – Python Web Server Gateway Interface v1.0.1 (https://www.python.org/dev/peps/pep-3333/)
- WSGI metaframework (http://www.pythonpaste.org/)
- Comprehensive wiki about everything WSGI (http://www.wsgi.org/)
- WSGI Tutorial (http://wsgi.tutorial.codepoint.net/)
- Python standard library module wsgiref (https://docs.python.org/library/wsgiref.html)
- Getting Started with WSGI (http://lucumr.pocoo.org/2007/5/21/getting-started-with-wsgi/)
- NWSGI (http://nwsgi.codeplex.com/) – .NET implementation of the Python WSGI specification for IronPython and IIS
- Gevent-FastCGI server implemented using gevent coroutine-based networking library (https://pypi.python.org/pypi/gevent-fastcgi)

Asynchronous Server Gateway Interface

The **Asynchronous Server Gateway Interface** (**ASGI**) is a calling convention for web servers to forward requests to asynchronous-capable Python frameworks, and applications. It is built as a successor to the Web Server Gateway Interface (WSGI).

Where WSGI provided a standard for synchronous Python application, ASGI provides one for both asynchronous and synchronous applications, with a WSGI backwards-compatibility implementation and multiple servers and application frameworks.

ASGI Specification

Version	3.0
Developer	ASGI Team
Release date	2019-03-04[1]
Website	asgi.readthedocs.io/en/latest/specs/index.html (https://asgi.readthedocs.io/en/latest/specs/index.html)
License	public domain[2]
Status	Draft

Example

An ASGI-compatible "Hello, World!" application written in Python:

```
1  async def application(scope, receive, send):
2      event = await receive()
3      ...
4      await send({"type": "websocket.send", ...})
```

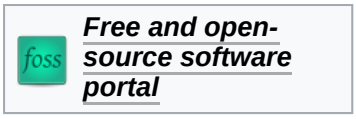
Where:

- Line 1 defines an asynchronous function named `application`, which takes three parameters (unlike in WSGI which takes only two), `scope`, `receive` and `send`.
  - `scope` is a `dict` containing details about current connection, like the protocol, headers, etc.
  - `receive` and `send` are asynchronous callables which let the application receive and send messages from/to the client.
- Line 2 receives an incoming event, for example, HTTP request or WebSocket message. The **`await`** keyword is used because the operation is asynchronous.
- Line 4 asynchronously sends a response back to the client. In this case, it is a WebSocket communication.

Web Server Gateway Interface (WSGI) compatibility

ASGI is also designed to be a superset of WSGI, and there's a defined way of translating between the two, allowing WSGI applications to be run inside ASGI servers through a translation wrapper (provided in the asgiref library). A threadpool can be used to run the synchronous WSGI applications away from the async event loop.

See also



- Comparison of web frameworks
- FastCGI
- Python (programming language)
- Web Server Gateway Interface (WSGI)

References

1. "Version History" (https://asgi.readthedocs.io/en/latest/specs/main.html#version-history).
2. "Copyright" (https://github.com/django/asgiref/blob/main/specs/asgi.rst#copyright). *GitHub*. Retrieved 2022-09-14.

External links

- Asynchronous Server Gateway Interface Documentation (https://asgi.readthedocs.io)
- Asynchronous Server Gateway Interface Specification (https://asgi.readthedocs.io/en/latest/specs/index.html)

Guido van Rossum

Guido van Rossum (Dutch: [ˈxidoː vɑn ˈrɔsʏm]; born 31 January 1956) is a Dutch programmer. He is the creator of the Python programming language, for which he was the "benevolent dictator for life" (BDFL) until he stepped down from the position on 12 July 2018.[4][5] He remained a member of the Python Steering Council through 2019, and withdrew from nominations for the 2020 election.[6]

Guido van Rossum

Van Rossum in 2024

Born	31 January 1956[1] The Hague, Netherlands[2]
Alma mater	University of Amsterdam
Occupation(s)	Computer programmer, author
Employer	Microsoft
Known for	Creating the Python programming language
Spouse	Kim Knapp (m. 2000)
Children	1[3]
Awards	Award for the Advancement of Free Software (2001)
Website	gvanrossum.github.io (https://gvanrossum.github.io/)

Life and education

Van Rossum was born and raised in the Netherlands, where he received a master's degree in mathematics and computer science from the University of Amsterdam in 1982. He received a bronze medal in 1974 in the International Mathematical Olympiad.[7] He has a brother, Just van Rossum, who is a type designer and programmer who designed the typeface used in the "Python Powered" logo.[8]

Van Rossum lives in Belmont, California, with his wife, Kim Knapp,[9] and their son.[10][11][12]

Work

Centrum Wiskunde & Informatica

While working at the Centrum Wiskunde & Informatica (CWI), Van Rossum wrote and contributed a glob() routine to BSD Unix in 1986[13][14] and helped develop the ABC programming language. He once stated, "I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it."[15] He also created Grail, an early web browser written in Python, and engaged in discussions about the HTML standard.[16]

He has worked for various research institutes, including the Centrum Wiskunde & Informatica (CWI) in the Netherlands, the U.S. National Institute of Standards and Technology (NIST), and the Corporation for National Research Initiatives (CNRI). In May 2000, he left CNRI along with three other Python core

developers to work for tech startup BeOpen.com, which subsequently collapsed by October of the same year.[17][18] From late 2000 until 2003 he worked for Zope Corporation. In 2003 Van Rossum left Zope for Elemental Security. While there he worked on a custom programming language for the organization.[19]

Google

From 2005 to December 2012, Van Rossum worked at Google, where he spent half of his time developing the Python language. At Google, he developed Mondrian, a web-based code review system written in Python and used within the company. He named the software after the Dutch painter Piet Mondrian.[20] He named Rietveld, another related software project, after Gerrit Rietveld, a Dutch designer.[21] On 7 December 2012, Van Rossum left Google.[22]

Van Rossum at the 2008 Google I/O Developer's Conference

Dropbox

In January 2013, Van Rossum started working at the cloud file storage company Dropbox.[23][24]

In October 2019, Van Rossum left Dropbox and officially retired.[25][26][27]

Microsoft

On 12 November 2020 Van Rossum announced that he was coming out of retirement to join the Developer Division at Microsoft. He currently holds the title Distinguished Engineer at Microsoft.[28][29][30]

Python

In December 1989, Van Rossum had been looking for a "'hobby' programming project that would keep [him] occupied during the week around Christmas" as his office was closed when he decided to write an interpreter for a "new scripting language [he] had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers". He chose the name Python for his programming language not because of the snake type but the comedy series *Monty Python's Flying Circus*.


Van Rossum at the 2006 O'Reilly Open Source Convention (OSCON)

He has explained that Python's predecessor, ABC, was inspired by SETL, noting that ABC co-developer Lambert Meertens had "spent a year with the SETL group at NYU before coming up with the final ABC design".[31]

On 12 July 2018, Van Rossum announced that he would be stepping down from the position of benevolent dictator for life of the Python programming language.[32]

"Computer Programming for Everybody" proposal

In 1999, Van Rossum submitted a funding proposal to the Defense Advanced Research Projects Agency (DARPA) called "Computer Programming for Everybody", in which he further defined his goals for Python:

- An easy and intuitive language just as powerful as major competitors
- Open source, so anyone can contribute to its development
- Code that is as understandable as plain English
- Suitability for everyday tasks, allowing for short development times

In 2019, Python became the second most popular language on GitHub, the largest source code management website on the internet, after JavaScript.[33] In 2024 Python became the most used language on GitHub, overtaking JavaScript after a 10-year run as the most used language.[34] According to a programming language popularity survey[35] it is consistently among the top 10 most mentioned languages in job postings. Furthermore, Python has been among the 10 most popular programming languages every year since 2004 according to the TIOBE Programming Community Index and got the number one spot on the index in October 2021.[36]

Awards

- At the 2002 FOSDEM conference in Brussels, Van Rossum received the 2001 Award for the Advancement of Free Software from the Free Software Foundation (FSF) for his work on Python.
- In May 2003, he received a NLUUG Award.[37]
- In 2006, he was recognized as a Distinguished Engineer by the Association for Computing Machinery.
- In 2018, he was made a Fellow of the Computer History museum.[38]
- In 2019, he was awarded the honorary title of Dijkstra Fellow by CWI.[39]
- In 2023, he was awarded the C&C Prize by NEC Corporation for developing the Python programming language.[40]

References

1. van Rossum, Guido (31 January 2007). "(Python-Dev) Happy Birthday, Guido!" (http://mail.python.org/pipermail/python-dev/2007-January/070849.html). Python-Dev mailing list. Archived (https://web.archive.org/web/20090908131440/http://mail.python.org/pipermail/python-dev/2007-January/070849.html) from the original on 8 September 2009.

2. Hsu, Hansen (5 April 2018). "2018 Museum Fellow Guido van Rossum, Python Creator & Benevolent Dictator for Life" (http://www.computerhistory.org/atchm/2018-chm-fellow-guido-van-rossum-python-creator-benevolent-dictator-for-life/). *Computer History Museum*. Archived (https://web.archive.org/web/20180724114116/http://www.computerhistory.org/atchm/2018-chm-fellow-guido-van-rossum-python-creator-benevolent-dictator-for-life/) from the original on 24 July 2018. Retrieved 19 May 2021.
3. "Guido van Rossum" (https://web.archive.org/web/20081031103755/http://wiki.codecall.net/Guido_van_Rossum). CodeCall Programming Wiki. Archived from the original (http://wiki.codecall.net/Guido_van_Rossum) on 31 October 2008.
4. "Benevolent dictator for life" (https://web.archive.org/web/20061001143603/http://www.linuxformat.co.uk/modules.php?op=modload&name=Sections&file=index&req=viewarticle&artid=10). Linux Format. 1 February 2005. Archived from the original (http://www.linuxformat.co.uk/modules.php?op=modload&name=Sections&file=index&req=viewarticle&artid=10) on 1 October 2006. Retrieved 1 November 2007.
5. "[python-committers] Transfer of power" (https://www.mail-archive.com/python-committers@python.org/msg05628.html). *www.mail-archive.com*. Archived (https://web.archive.org/web/20180712225051/https://www.mail-archive.com/python-committers@python.org/msg05628.html) from the original on 12 July 2018. Retrieved 12 July 2018.
6. "Steering Council nomination: Guido van Rossum (2020 term)" (https://discuss.python.org/t/steering-council-nomination-guido-van-rossum-2020-term/2657/11). 27 November 2019. Archived (https://web.archive.org/web/20211214000123/https://discuss.python.org/t/steering-council-nomination-guido-van-rossum-2020-term/2657/11) from the original on 14 December 2021. Retrieved 13 November 2020.
7. "International Mathematical Olympiad" (https://www.imo-official.org/participant_r.aspx?id=10303). *www.imo-official.org*. Archived (https://web.archive.org/web/20230310105202/https://www.imo-official.org/participant_r.aspx?id=10303) from the original on 10 March 2023. Retrieved 23 May 2022.
8. Thomas, Jockin (28 May 2016). "Learning Python Makes You A Better Designer: An Interview with Just van Rossum" (https://medium.com/type-thursday/learning-python-makes-you-a-better-designer-an-interview-with-just-van-rossum-8d4758c192d8/). Medium. Archived (https://web.archive.org/web/20191025180434/https://medium.com/type-thursday/learning-python-makes-you-a-better-designer-an-interview-with-just-van-rossum-8d4758c192d8) from the original on 25 October 2019. Retrieved 25 October 2019.
9. Manheimer, Ken (6 June 2000). "(Python-Dev) Guido and Kim married" (http://mail.python.org/pipermail/python-dev/2000-June/004497.html). Python-Dev -- Python core developers. Archived (https://web.archive.org/web/20100928035232/http://mail.python.org/pipermail/python-dev/2000-June/004497.html) from the original on 28 September 2010.
10. "Guido van Rossum - Brief Bio" (https://www.python.org/~guido/bio.html). Archived (https://web.archive.org/web/20140819142627/https://www.python.org/~guido/bio.html) from the original on 19 August 2014.
11. "(Mailman-Announce) forwarded message from Guido van Rossum" (http://mail.python.org/pipermail/mailman-announce/2000-May/000010.html). 30 May 2000. Archived (https://web.archive.org/web/20080527065145/http://mail.python.org/pipermail/mailman-announce/2000-May/000010.html) from the original on 27 May 2008. "Oh, and to top it all off, I'm going on vacation. I'm getting married and will be relaxing on my honeymoon."
12. van Rossum, Guido. "What's New in Python?" (http://csg.csail.mit.edu/6.893/Handouts/PythonWhatsNew.pdf) (PDF). *"Not your usual list of new features". Stanford CSL Colloquium, 29 October 2003; BayPiggies, 13 November 2003*. Elemental Security. Archived (https://web.archive.org/web/20100627001813/http://csg.csail.mit.edu/6.893/Handouts/PythonWhatsNew.pdf) (PDF) from the original on 27 June 2010.
13. " 'Globbing' library routine" (https://web.archive.org/web/20071219090708/http://www.isc.org/sources/devel/func/glob.txt). Archived from the original (http://www.isc.org/sources/devel/func/glob.txt) on 19 December 2007.

14. "File::Glob - Perl extension for BSD glob routine" (https://metacpan.org/module/File::Glob). metacpan.org. Archived (https://web.archive.org/web/20130807081635/https://metacpan.org/module/File::Glob) from the original on 7 August 2013.
15. Venners, Bill. "The Making of Python" (http://www.artima.com/intv/pythonP.html). *www.artima.com*. Archived (https://web.archive.org/web/20160901183332/http://www.artima.com/intv/pythonP.html) from the original on 1 September 2016. Retrieved 14 September 2016.
16. "Re: xmosaic experience" (http://1997.webhistory.org/www.lists/www-talk.1993q1/0184.html). Archived (https://web.archive.org/web/20160828030223/http://1997.webhistory.org/www.lists/www-talk.1993q1/0184.html) from the original on 28 August 2016.
17. "Oral History of Guido van Rossum, part 2 - Computer History Museum" (https://archive.computerhistory.org/resources/access/text/2018/07/102738761-05-01-acc.pdf) (PDF). Archived (https://web.archive.org/web/20211117145843/https://archive.computerhistory.org/resources/access/text/2018/07/102738761-05-01-acc.pdf) (PDF) from the original on 17 November 2021. Retrieved 17 November 2021.
18. "Python 2.3.2 License A. HISTORY OF THE SOFTWARE" (https://www.python.org/download/releases/2.3.2/license/). Archived (https://web.archive.org/web/20211117145844/https://www.python.org/download/releases/2.3.2/license/) from the original on 17 November 2021. Retrieved 17 November 2020.
19. "2018 Museum Fellow Guido van Rossum, Python Creator & Benevolent Dictator for Life - Computer History Museum" (https://web.archive.org/web/20180724114116/http://www.computerhistory.org/atchm/2018-chm-fellow-guido-van-rossum-python-creator-benevolent-dictator-for-life/). 5 April 2018. Archived from the original (http://www.computerhistory.org/atchm/2018-chm-fellow-guido-van-rossum-python-creator-benevolent-dictator-for-life/) on 24 July 2018. Retrieved 23 August 2018.
20. van Rossum, Guido (May 2008). "An Open Source App: Rietveld Code Review Tool" (https://web.archive.org/web/20151017112923/https://cloud.google.com/appengine/articles/rietveld). Archived from the original (https://developers.google.com/appengine/articles/rietveld) on 17 October 2015. Retrieved 24 August 2012. "... the internal web app, which I code-named Mondrian after one of my favorite Dutch painters"
21. "An Open Source App: Rietveld Code Review Tool" (https://web.archive.org/web/20151017112923/https://cloud.google.com/appengine/articles/rietveld). Archived from the original (https://cloud.google.com/appengine/articles/rietveld) on 17 October 2015.
22. "Guido van Rossum" (https://twitter.com/gvanrossum/status/277126763295944705). *@gvanrossum*. Twitter. Archived (https://web.archive.org/web/20131216140256/https://twitter.com/gvanrossum/status/277126763295944705) from the original on 16 December 2013. Retrieved 15 August 2022. "Today's my last day at Google. In January I start a new job at Dropbox: t.co/JxnfdBM0"
23. Constine, Josh (7 December 2012). "Dropbox Hires Away Google's Guido van Rossum, The Father Of Python" (https://techcrunch.com/2012/12/07/dropbox-guido-van-rossum-python/). Techcrunch. Archived (https://web.archive.org/web/20121209015453/http://techcrunch.com/2012/12/07/dropbox-guido-van-rossum-python/) from the original on 9 December 2012. Retrieved 7 December 2012.
24. "Welcome Guido!" (https://tech.dropbox.com/2012/12/welcome-guido/). Dropbox Tech Blog. 7 December 2012. Archived (https://web.archive.org/web/20130907212330/https://tech.dropbox.com/2012/12/welcome-guido/) from the original on 7 September 2013. Retrieved 6 September 2013.
25. @gvanrossum (30 October 2019). "It's bittersweet: I'm leaving @dropbox, and am now retired. I've learned a lot during my time as an engineer here -- e.g. type annotations came from this experience -- and I'll miss working here" (https://x.com/gvanrossum/status/1189546865114529792) (Tweet). Retrieved 30 October 2019 – via Twitter.

26. "Thank you, Guido" (https://blog.dropbox.com/topics/company/thank-you--guido). *Dropbox Blog*. Dropbox. Archived (https://web.archive.org/web/20210216114908/https://blog.dropbox.com/topics/company/thank-you--guido) from the original on 16 February 2021. Retrieved 1 February 2021.
27. Tung, Liam (31 October 2019). "Python programming language creator retires, saying: 'It's been an amazing ride' " (https://www.zdnet.com/article/python-programming-language-creator-retires-saying-its-been-an-amazing-ride/). *ZDNet*. Archived (https://web.archive.org/web/20210121090335/https://www.zdnet.com/article/python-programming-language-creator-retires-saying-its-been-an-amazing-ride/) from the original on 21 January 2021. Retrieved 1 February 2021.
28. "Guido van Rossum" (https://www.linkedin.com/in/guido-van-rossum-4a0756/). "Python's BDFL-emeritus, Distinguished Engineer at Microsoft, Computer History Fellow."
29. @gvanrossum (12 November 2020). "I decided that retirement was boring and have joined the Developer Division at Microsoft. To do what? Too many options to say! But it'll make using Python better for sure (and not just on Windows :-). There's lots of open source here. Watch this space" (https://x.com/gvanrossum/status/1326932991566700549) (Tweet). Retrieved 12 November 2020 – via Twitter.
30. Lardinois, Frederic (12 November 2020). "Python creator Guido van Rossum joins Microsoft" (https://techcrunch.com/2020/11/12/python-creator-guido-van-rossum-joins-microsoft/). *TechCrunch*. Archived (https://web.archive.org/web/20210124085936/https://techcrunch.com/2020/11/12/python-creator-guido-van-rossum-joins-microsoft/) from the original on 24 January 2021. Retrieved 16 November 2020.
31. "Python-Dev] SETL (was: Lukewarm about range literals)" (http://mail.python.org/pipermail/python-dev/2000-August/008881.html). 29 August 2000. Archived (https://web.archive.org/web/20110514231628/http://mail.python.org/pipermail/python-dev/2000-August/008881.html) from the original on 14 May 2011.
32. Fairchild, Carlie (12 July 2018). "Guido van Rossum Stepping Down from Role as Python's Benevolent Dictator For Life" (https://web.archive.org/web/20180713192427/https://www.linuxjournal.com/content/guido-van-rossum-stepping-down-role-pythons-benevolent-dictator-life). *Linux Journal*. Archived from the original (https://www.linuxjournal.com/content/guido-van-rossum-stepping-down-role-pythons-benevolent-dictator-life) on 13 July 2018. Retrieved 12 July 2018.
33. "The State of the Octoverse" (https://octoverse.github.com/). *The State of the Octoverse*. Archived (https://web.archive.org/web/20170405182156/https://octoverse.github.com/) from the original on 5 April 2017. Retrieved 6 May 2021.
34. "Octoverse: AI leads Python to top language as the number of global developers surges" (https://github.blog/news-insights/octoverse/octoverse-2024/). *GitHub Insights*. Archived (https://web.archive.org/web/20241116052434/https://github.blog/news-insights/octoverse/octoverse-2024/) from the original on 16 November 2024. Retrieved 6 January 2025.
35. "Programming Language Popularity" (https://web.archive.org/web/20150412161127/http://langpop.com/). Archived from the original (http://langpop.com/) on 12 April 2015.
36. "index | TIOBE - The Software Quality Company" (https://www.tiobe.com/tiobe-index/). *www.tiobe.com*. Archived (https://web.archive.org/web/20231015080735/https://www.tiobe.com/tiobe-index/) from the original on 15 October 2023. Retrieved 29 May 2020.
37. "Guido van Rossum Ontvangt NLUUG Award" (https://www.nluug.nl/vereniging/persberichten/009.html). *NLUUG*. 28 May 2003. Archived (https://web.archive.org/web/20210308225336/https://www.nluug.nl/vereniging/persberichten/009.html) from the original on 8 March 2021. Retrieved 22 January 2018.
38. "Guido van Rossum" (http://www.computerhistory.org/fellowawards/hall/guido-van-rossum/). *Computer History Museum*. Archived (https://web.archive.org/web/20190703181839/https://www.computerhistory.org/fellowawards/hall/guido-van-rossum/) from the original on 3 July 2019. Retrieved 22 February 2018.

39. "David Chaum and Guido van Rossum awarded Dijkstra Fellowship" (https://www.cwi.nl/en/news/david-chaum-and-guido-van-rossum-awarded-dijkstra-fellowship/). *www.cwi.nl*. Archived (https://web.archive.org/web/20240321182149/https://www.cwi.nl/en/news/david-chaum-and-guido-van-rossum-awarded-dijkstra-fellowship/) from the original on 21 March 2024. Retrieved 21 March 2024.
40. "NEC C&C Foundation Awards 2023 C&C Prize" (https://www.nec.com/en/press/202310/global_20231010_01.html). *nec.com*. Tokyo. 10 October 2023. Archived (https://web.archive.org/web/20240219055207/https://www.nec.com/en/press/202310/global_20231010_01.html) from the original on 19 February 2024. Retrieved 19 February 2024.

External links

Free and open-source software portal

- Official website (https://gvanrossum.github.io/)
- Guido van Rossum. *The History of Python* (https://python-history.blogspot.com/2009/01/introduction-and-overview.html)
- Guido van Rossum. *Neopythonic: Ramblings* (https://neopythonic.blogspot.com/2008/10/thoughts-after-reading-neal-stephensons.html)
- *Computer Programming for Everybody* (https://www.python.org/doc/essays/everybody/)
- Guido van Rossum Interview (http://www.twit.tv/floss11) on FLOSS Weekly
- *Guido van Rossum* interview (http://workspiration.org/guido-van-rossum) - Workspiration.org
- Guido van Rossum on Python Interview (https://web.archive.org/web/20081229095320/http://www.computerworld.com.au/index.php/id%3B66665771) - Computerworld
- Guido van Rossum Run your web applications on Google's infrastructure (http://www.stanford.edu/class/ee380/Abstracts/081105.html) — Google App Engine technical talk at Stanford University. (video archive (https://web.archive.org/web/20090326060917/http://stanford-online.stanford.edu/courses/ee380/081105-ee380-300.asx))
- Oral History of Guido Van Rossum Part 1 (https://www.youtube.com/watch?v=Pzkdci2HDpU) on YouTube Computer History Museum
- Oral History of Guido Van Rossum Part 2 (https://www.youtube.com/watch?v=y-Yetu20snM) on YouTube Computer History Museum