Python

Python (найчастіше вживане прочитання — «**Пáйтон**», запозичено назву[7] з британського шоу Монті Пайтон) — інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією.[8] Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Зміст

Python

Парадигма	кілька парадигм: імперативна, функціональна, об'єктно-орієнтована
Дата появи	20 лютого 1991[1]
Творці	Ґвідо ван Россум[2]
Розробник	Гвідо ван Россум
Останній реліз	3.11.1 / 6 грудня, 2022[3], 2.7.18 / 20 квітня, 2020[4]
Тестова версія	3.11.0a3 / 8 грудня, 2021[5]
Система типізації	качина типізація, динамічна типізація і gradual typing[d]
Діалекти	Python 3.10, Python 2.7 (підтримка закінчена 1 січня 2020 року.[6])
Під впливом від	ABC, C, Haskell, Icon, Lisp, Modula-3, Perl, Smalltalk, Tcl
Вплинула на	Boo, Groovy, D
Платформа	Microsoft Windows
Операційна система	Багатоплатформова
Ліцензія	Python Software Foundation License
Звичайні розширення файлів	`.py, .pyw, .pyc, .pyo, .pyd`
Репозиторій вихідного коду	github.com/python/cpython (https://github.com/python/cpython)
Вебсайт	www.python.org (https://www.python.org)

Python у Вікісховищі

Переваги

Серед основних її переваг можна назвати такі:

- чистий синтаксис (для виділення блоків слід використовувати відступи);
- переносність програм (що властиве більшості інтерпретованих мов);
- стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
- можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);
- стандартний дистрибутив має просте, але разом із тим досить потужне середовище

розробки, яке зветься IDLE і яке написане мовою Python;
- зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор);
- відкритий код (можливість редагувати його іншими користувачами).

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

Інтерпретатор мови Python і багата Стандартна бібліотека (як вихідні тексти, так і бінарні дистрибутиви для всіх основних операційних систем) можуть бути отримані з сайту Python www.python.org (https://www.python.org/) [Архівовано (https://web.archive.org/web/20180417023815/https://www.python.org/) 17 квітня 2018 у Wayback Machine.], і можуть вільно розповсюджуватися. Цей самий сайт має дистрибутиви та посилання на численні модулі, програми, утиліти та додаткову документацію.

Інтерпретатор мови Python може бути розширений функціями та типами даних, розробленими на C чи C++ (або на іншій мові, яку можна викликати із C). Python також зручна як мова розширення для прикладних програм, що потребують подальшого налагодження.

Історія

Розробка мови Python була розпочата в кінці 1980-х років[9] співробітником голландського інституту CWI Гвідо ван Россумом. Для розподіленої ОС Amoeba потрібна була розширювана скриптова мова, і Гвідо почав писати Python на дозвіллі, запозичивши деякі напрацювання для мови ABC (Гвідо брав участь у розробці цієї мови, орієнтованої на навчання програмування). У лютому 1991 року Гвідо опублікував вихідний текст в групі новин alt.sources[10]. Мова почала вільно поширюватися через Інтернет і сподобалася іншим програмістам. З 1991 року Python є цілком об'єктно-орієнтованим. Python також запозичив багато рис таких мов, як C, C++, Modula-3 і Icon, й окремі риси функціонального програмування з Ліспу.

.py

Назва мови виникла зовсім не від виду плазунів. Автор назвав мову на честь популярного британського комедійного серіалу 70-х років «Повітряний цирк Монті Пайтона». Втім, все одно назву мови частіше асоціюють саме зі змією, ніж з фільмом — піктограми файлів в KDE або в Windows, і навіть емблема на сайті python.org зображують зміїну голову.

Гвідо ван Россум, автор Python

Наявність дружньої спільноти користувачів, поряд з дизайнерською інтуїцією Гвідо, вважається одним з головних факторів успіху Python. Розвиток мови відбувається згідно з чітко регламентованими процесами створення, обговорення, відбору та реалізації документів PEP (Python Enhancement Proposal) — пропозицій щодо розвитку Python.[11]

3 грудня 2008 року[12], після тривалого тестування, вийшла перша версія Python 3000 (або Python 3.0, також використовується скорочена Py3k). У Python 3000 усунено багато недоліків архітектури з максимально можливим (але не повним) збереженням сумісності зі старішими версіями. На сьогодні підтримуються Python версії 3.

Філософія

Розробники мови Python є прихильниками певної філософії програмування, яку називають «The Zen of Python» («Дзен Пайтона»)[13]. Її текст можна отримати в інтерпретаторі Python за допомогою команди `import this` (лише один раз за сесію). Автором цієї філософії вважається Тім Пейтерс.

Текст філософії:

- Гарне краще за потворне.
- Явне краще за неявне.
- Просте краще за складне.
- Складне краще за заплутане.
- Плоске краще за вкладене.
- Розріджене краще за щільне.
- Легкість читання має значення.
- Особливі випадки не є настільки особливими, щоб порушувати правила.
- Хоча практичність є важливішою за бездоганність.
- Помилки ніколи не повинні проходити непомітно.
- Якщо їх приховування не прописано явно.
- Зустрівши неоднозначність, опирайтесь спокусі вгадати.
- Має бути один — і, бажано, *тільки* один — очевидний спосіб зробити це.
- Хоча спочатку він може бути й не очевидним, якщо ви не голландець[14].
- Зараз — краще, ніж ніколи.
- Хоча ніколи, найчастіше, — краще, ніж *просто* зараз.
- Якщо реалізацію важко пояснити — задум поганий.
- Якщо реалізацію легко пояснити — *можливо*, задум добрий.
- Простори імен — чудова річ, тож робімо їх більше!

Оригінальний текст (англ.)

- *Beautiful is better than ugly.*
- *Explicit is better than implicit.*
- *Simple is better than complex.*
- *Complex is better than complicated.*
- *Flat is better than nested.*
- *Sparse is better than dense.*
- *Readability counts.*
- *Special cases aren't special enough to break the rules.*
- *Although practicality beats purity.*

- *Errors should never pass silently.*
- *Unless explicitly silenced.*
- *In the face of ambiguity, refuse the temptation to guess.*
- *There should be one — and preferably only one — obvious way to do it.*
- *Although that way may not be obvious at first unless you're Dutch.*
- *Now is better than never.*
- *Although never is often better than 'right now'.*
- *If the implementation is hard to explain, it's a bad idea.*
- *If the implementation is easy to explain, it may be a good idea.*
- *Namespaces are one honking great idea — let's do more of those!*

Вплив інших мов на Python

З'явившись порівняно пізно, Python створювався під впливом багатьох мов програмування:

- ABC — відступи (поля) для групування операторів, високорівневі структури даних (map)[15][16] (фактично, Python створювався як спроба виправити помилки, допущені при проєктуванні ABC);
- Modula-3 — пакети, модулі, використання `else` спільно з `try` та `except`, іменовані аргументи функцій (на це також вплинув Common Lisp);
- C, C++ — деякі синтаксичні конструкції (як пише сам Гвідо ван Россум — він використовував найбільш несуперечливі конструкції з C, щоб не викликати неприязнь у Сі-програмістів до Python[15]);
- Smalltalk — об'єктно-орієнтоване програмування;
- Lisp — окремі риси функціонального програмування (`lambda, map, reduce, filter` та інші);
- Fortran — зрізи масивів, комплексна арифметика;
- Miranda — спискові вирази;
- Java — модулі logging, unittest, threading (частина можливостей оригінального модуля не реалізована), xml.sax стандартної бібліотеки, спільне використання finally та except при обробці винятків, використання @ для декораторів;
- Icon — генератори.
- TypeScript — ідея синтаксису підказок типів (type hinting (https://peps.python.org/pep-0484/)). З версії Python 3.10 система підказок типів стала унікальною, «злившись» з мовою в одне ціле. До того ж у Python підказки типів ніяк не впливають на роботу інтерпретатора.

Більша частина інших можливостей Python (наприклад, байт-компіляція вихідного коду) також була реалізована раніше в інших мовах.

Портованість

Python портованна і працює майже на всіх відомих платформах — від КПК до мейнфреймів. Існують порти під Microsoft Windows, всі варіанти UNIX (включаючи FreeBSD та GNU/Linux), Plan 9, Mac OS та Mac OS X, iPhone OS 2.0 і вище, Palm OS, OS/2, Amiga, AS/400 та навіть OS/390, Symbian та Android[17].

У міру старіння платформи її підтримка в основній гілці мови припиняється. Наприклад, з версії 2.6 припинена підтримка Windows 95, Windows 98 та Windows ME.[18] Однак на цих платформах можна використовувати попередні версії Python — спільнота активно підтримує версії Python починаючи від 2.3 (для них виходять виправлення).

При цьому, на відміну від багатьох портованих систем, для всіх основних платформ Python має підтримку характерних для даної платформи технологій (наприклад, Microsoft COM/DCOM). Навіть більше, існує спеціальна версія Python для віртуальної машини Java — Jython, що дозволяє інтерпретатору виконуватися на будь-якій системі, яка підтримує Java, при цьому класи Java можуть безпосередньо використовуватися з Python й навіть бути написаними на ньому. Також кілька проєктів забезпечують інтеграцію з платформою Microsoft.NET, основні з яких — IronPython та Python.Net.

Типи й структури даних

Python підтримує динамічну типізацію, тобто, тип змінної визначається лише під час виконання. З базових типів слід зазначити підтримку цілих чисел довільної довжини і комплексних чисел. Python має багату бібліотеку для роботи з рядками, зокрема, кодованими в юнікоді.

З колекцій Python підтримує кортежі (*tuples*), списки (масиви), словники (асоціативні масиви) і від версії 2.4, множини.

Система класів підтримує множинне успадкування і метапрограмування. Будь-який тип, включаючи базові, входить до системи класів, й за необхідності можливе успадкування навіть від базових типів.

Можливості

Інтерактивний режим

Подібно Ліспу та Прологу в режимі відлагодження, інтерпретатор Python має інтерактивний режим роботи, при якому введені з клавіатури вирази відразу ж виконуються, а результат виводиться на екран. Цей режим цікавий не тільки новачкам, але й досвідченим програмістам, які можуть протестувати в інтерактивному режимі будь-який фрагмент коду, перш ніж використовувати його в основній програмі, або просто використовувати як калькулятор з великим набором функцій.

Діалог роботи з Python в інтерактивному режимі має такий вигляд:

```
>>> 2 ** 100 # піднесення 2 до 100-го степеня
1267650600228229401496703205376L
>>> from math import * # імпорт математичних функцій
>>> sin (pi * 0.5) # обчислення синуса від половини пі
1.0
>>> help (sorted) # допомогу по функції sorted
Help on built-in function sorted in module __builtin__:
sorted (…)
   sorted (iterable, cmp=none, key=none, reverse=false) -> new sorted list
```

В інтерактивному режимі доступний дебагер `pdb` та система довідки (викликається за `help()`). Система допомоги працює для модулів, класів і функцій, тільки якщо ті були забезпечені рядками документації.

Крім вбудованої, існує й покращена інтерактивна оболонка IPython[19].

Об'єктно-орієнтоване програмування

Дизайн мови Python побудований навколо об'єктно-орієнтованої моделі програмування. Реалізація ООП в Python є елегантною, потужною та добре продуманою, але разом з тим, достатньо специфічною в порівнянні з іншими об'єктно-орієнтованими мовами.

Можливості та особливості:

1. Класи є одночасно об'єктами з усіма нижче наведеними можливостями.
2. Успадкування, в тому числі множинне.
3. Поліморфізм (всі функції віртуальні).
4. Інкапсуляція (два рівні — загальнодоступні та приховані методи і поля). Особливість — приховані члени доступні для використання та помічені як приховані лише особливими іменами.
5. Спеціальні методи, що керують життєвим циклом об'єкта: конструктори, деструктори, розподільники пам'яті.
6. Перевантаження операторів (усіх, крім `is, '.', '='` і символьних логічних).
7. Властивості (імітація поля за допомогою функцій).
8. Управління доступу до полів (емуляція полів і методів, частковий доступ тощо).
9. Методи для управління найпоширенішими операціями (істиннісне значення, `len()`, глибоке копіювання, серіалізація, ітерація по об'єкту, …)
10. Метапрограмування (управління створенням класів, тригери на створення класів, та ін)
11. Повна інтроспекція.
12. Класові та статичні методи, класові поля.
13. Класи, вкладені у функції та інші класи.

Функціональне програмування

Python підтримує парадигму функціонального програмування, зокрема:

- функція є об'єктом;
- функції вищих порядків;
- рекурсія;
- розвинена обробка списків (спискові вирази, операції над послідовностями, ітератори);
- аналог замикань (closures);
- часткове застосування функції;

- можливість реалізації інших засобів на самій мові (наприклад, каррінг).

Модулі та пакети

Програмне забезпечення (застосунок або бібліотека) на Python оформлюється у вигляді модулів, які у свою чергу можуть бути зібрані в *пакунки*. Модулі можуть розташовуватися як у каталогах, так і в ZIP-архівах. Модулі можуть бути двох типів за своїм походженням: модулі, написані на «чистому» Python, і модулі розширення (extension modules), написані на інших мовах програмування. Наприклад, в стандартній бібліотеці є «чистий» модуль pickle і його аналог на Сі: cPickle. Модуль оформляється у вигляді окремого файлу, а пакет — у вигляді окремого каталогу. Підключення модуля до програми здійснюється оператором `import`. Після імпорту модуль представлений окремим об'єктом, що дає доступ до простору імен модуля. У ході виконання програми модуль можна перезавантажити функцією `reload()`.

Інтроспекція

Python підтримує повну інтроспекцію часу виконання. Це означає, що для будь-якого об'єкта можна отримати всю інформацію про його внутрішню структуру.

Застосування інтроспекції (метапрограмування) є важливою частиною того, що називають «pythonic style», і широко застосовується в бібліотеках і фреймворках Python, таких як PyRO, Pyro, PLY, CherryPy, Django та інших, заощаджуючи час програміста, що ними користується.

Обробка винятків

Обробка винятків підтримується в Python допомогою операторів `try`, `except`, `else`, `finally`, `raise`, що утворюють блок обробки винятків. У загальному випадку блок має такий вигляд:

```
try:
    # Тут код, в якому може виникнути виняткова ситуація
    raise ExceptionType ("message")
except (Тип винятку1, Тип винятку2, …), Змінна:
    # Код в блоці виконується, якщо тип винятку збігається з одним з типів
    # (Тип винятку1, Тип винятку2, ...) або є спадкоємцем одного
    # з цих типів.
    # Отриманий виняток доступний в необов'язковій Змінній.
except (Тип винятку3, Тип винятку4, …), Змінна:
    # Кількість блоків except не обмежено
    raise # Згенерувати виняток "поверх" отриманого; без параметрів – повторно згенерувати
отримане
except:
    # Буде виконано за будь-якого винятку, не обробленого типізованими блоками except
else:
    # Код блоку виконується, якщо не було отримано винятків.
finally:
    # Буде виконано в будь-якому випадку, можливо після відповідного
    # блоку except або else
```

Спільне використання `else`, `except` і `finally` стало можливо тільки починаючи з Python 2.5. Інформація про поточний виняток завжди доступна через `sys.exc_info()`. Крім значення винятку, Python також зберігає стан стеку аж до точки збудження винятку — так званий traceback.

На відміну від мов програмування, що компілюються, в Python використання винятку не призводить до значних накладних витрат (а часто навіть дозволяє прискорити виконання програм) і дуже широко використовується. Винятки узгоджуються з філософією Python (10-й пункт «дзену Python» — «Помилки ніколи не повинні ігноруватися») та є одним із засобів підтримки «качиної типізації».

Іноді, замість явної обробки винятків, зручніше використовувати блок `with` (доступний, починаючи з Python 2.5).

Ітератори

У програмах на Python широко використовуються ітератори. Цикл `for` може працювати як з послідовністю, так і з ітераторами. Усі колекції, як правило, надають ітератор. Об'єкти визначеного користувачем класу теж можуть бути ітераторами. Модуль `itertools` стандартної бібліотеки містить багато корисних функцій для роботи з ітераторами. На відміну від звичайних послідовностей, всі елементи яких зберігаються в пам'яті, отримання наступного елемента забезпечує генератор — спеціальна функція, звернення до якої обчислює і повертає наступний елемент генератора.

Генератори

Однією з цікавих можливостей мови є **генератори** — функції, що між викликами зберігають внутрішній стан: значення локальних змінних і поточну інструкцію (див. також: співпрограма). Генератори можуть використовуватися як ітератори для структур даних і для лінивих обчислень. Наприклад, генератор чисел Фібоначчі:

```
def fib():
    a, b = 1, 1
    while True:
        yield a
        a, b = b, a + b

while x in fib():
     print(x)
```

При виклику генератора функція негайно повертає об'єкт-ітератор, який зберігає поточну точку виконання та стан локальних змінних функції. При запиті наступного значення (за допомогою методу `next()`, який неявно викликається в циклі `for`) генератор продовжує виконання функції від попередньої точки зупину до наступного оператора `yield` або `return`.

У Python 2.4 з'явилися **генераторні вирази** — вирази, що дають у результаті генератор. Генераторні вирази дозволяють заощадити пам'ять там, де інакше потрібно було б використовувати список із проміжними результатами:

```
>>> sum(i for i in xrange (1, 100) if i % 2 != 0)
2500
```

У цьому прикладі підсумовуються всі непарні числа від 1 до 99.

Починаючи з версії 2.5, Python підтримує повноцінні співпроцедури: тепер в генератор можна передавати значення за допомогою методу `send()` та збуджувати в його контексті виняток за допомогою методу `throw()`.

Керування контекстом виконання

У Python 2.5 з'явилися засоби для керування контекстом виконання блоку коду — оператор `with` та модуль `contextlib`.

Оператор може застосовуватися в тих випадках, коли до та після деяких дій обов'язково мають виконуватися якісь інші дії, незалежно від створених у блоці винятків або операторів `return`: файли має бути закрито, ресурси звільнено, перенаправлення стандартного введення/виведення завершено, тощо. Оператор полегшує читання коду, отже, допомагає уникати помилок.

Декоратори

Починаючи з версії 2.4, Python дозволяє використовувати, так звані, *Декоратори*[20] (не слід плутати з однойменним шаблоном проєктування) для підтримки існуючої практики перетворення функцій та методів у місці визначення (декораторів може бути декілька). Після довгих дебатів для декораторів став використовуватися символ @ у рядках, що передують визначенню функції або методу. Наступний приклад містить опис статичного методу без застосування декоратора:

```
def my_wonderful_method ():
    return "Деякий метод"
my_wonderful_method = staticmethod(my_wonderful_method)
```

і за допомогою декоратора:

```
@staticmethod
def my_wonderful_method ():
    return "Деякий метод"
```

Декоратор — це функція, першим аргументом якої є декорована функція або метод. Декоратори можна вважати елементом аспектно-орієнтованого програмування.

З версії 2.6 декоратори можна використовувати з класами, аналогічно функціям.

Інші можливості

У Python є ще кілька можливостей, що відрізняють його від багатьох інших мов високою гнучкістю та динамічністю.

Наприклад, клас є об'єктом, а в операторі визначення класу можна використовувати вирази в списку батьківських класів.

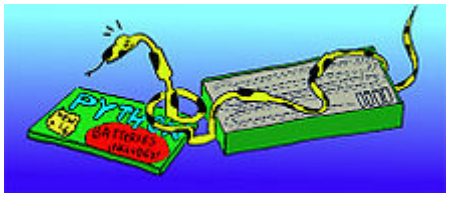
```
def getClass():
    return dict
class D(getClass()):
    pass
d = D()
```

Можна модифікувати багато об'єктів під час виконання, наприклад класи:

```
>>> class X(object): pass
…
>>> y = X()
>>> y.wrongMethod() # такого методу поки немає
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'X' object has no attribute 'wrongMethod'
>>> X.wrongMethod = lambda self : 'im here' # додамо його
>>> y.wrongMethod() # так як доступ до методу призводить до пошуку по __dict__ класу,
'im here' # то wrongMethod стає доступним всім екземплярам
```

Бібліотеки

Стандартна бібліотека



Python поставляється «з батарейками в комплекті».

Багата стандартна бібліотека є однією з привабливостей мови Python. Тут є засоби для роботи з багатьма мережевими протоколами та форматами Інтернету, наприклад, модулі для написання HTTP-серверів та клієнтів, для розбору та створення поштових повідомлень, для роботи з XML, тощо. Набір модулів для роботи з операційною системою дозволяє писати крос-платформні застосунки. Існують модулі для роботи з регулярними виразами, текстовими кодуваннями, мультимедійними форматами, криптографічними протоколами, архівами, серіалізацією даних, юніт-тестуванням та ін.

Модулі розширення та програмні інтерфейси

Крім стандартної бібліотеки існує багато інших, що надають інтерфейс до всіх системних викликів на різних платформах; зокрема, на платформі Win32 підтримуються всі виклики Win32 API, а також COM в обсязі не меншому, ніж у Visual Basic або Delphi. Існує велика кількість прикладних бібліотек для Python у різноманітних галузях: веброзробка, бази даних, обробка зображень, обробка тексту, чисельні методи, програми операційної системи тощо.

Для Python прийнята специфікація програмного інтерфейсу до баз даних DB-API 2 та розроблено відповідні цій специфікації пакети для доступу до різних СУБД: PostgreSQL, Oracle, Sybase, Firebird (Interbase), Informix, Microsoft SQL Server, MySQL та sqlite. На платформі Microsoft Windows доступ до БД можливий через ADO (ADOdb). Комерційний пакет mxODBC для доступу до СУБД через ODBC для платформ Windows і UNIX розроблений eGenix[21]. Для Python написано багато ORM: (SQLObject, SQLAlchemy, Dejavu, Django), виконані програмні каркаси для розробки вебзастосунків (Django, Pylons).

Бібліотека NumPy для роботи з багатовимірними масивами дозволяє досягти продуктивності наукових розрахунків, порівнянної зі спеціалізованими пакетами. SciPy використовує NumPy і надає доступ до великого спектра математичних алгоритмів (матрична алгебра — BLAS, level 1-3 і LAPACK; ШПФ).

Бібліотека WSGI[22] — інтерфейс шлюзу з вебсервером (Python Web Server Gateway Interface).

Python надає простий і зручний програмний інтерфейс C API для написання власних модулів на мовах C та C++. Інструмент SWIG дозволяє майже автоматично отримувати прив'язки для використання C/C++ бібліотек у коді на Python. Можливості цього та інших інструментів варіюються від автоматичної генерації (C/C++/Fortran)-Python інтерфейсів за спеціальними файлами (SWIG, pyste[23], SIP[24], pyfort[25]) до надання зручніших API (boost::python[26], CXX[27] та ін.) Інструмент стандартної бібліотеки ctypes дозволяє програмам Python безпосередньо викликати функції з динамічних бібліотек/DLL, написаних на C. Існують модулі, що дозволяють вбудовувати код на C/C++ прямо у вихідні файли Python, створюючи розширення «на льоту» (pyinline[28], weave[29]). Для підключення математичних функцій, особливо із застосуванням NumPy, наразі офіційно рекомендованим є Cython[30]

Інший підхід полягає у вбудовуванні інтерпретатора Python у застосунки. Python легко вбудовується в програми на Java, C/C++, Ocaml. Взаємодія Python-застосунків з іншими системами можлива також за допомогою CORBA, XML-RPC, SOAP, COM.

За допомогою Pyrex[31] можлива компіляція Python-подібної мови (додано можливість типізації) в еквівалентний Сі-код і зв'язування із зовнішніми модулями.

Експериментальний проєкт shed skin[32][33] передбачає створення компілятора для трансформації неявно типізованих Python програм в оптимізований C++ код. Починаючи з версії 0.22 shed skin дозволяє компілювати окремі функції в модулі розширень. Повна компіляція (станом на 1 липня 2007) далека від завершення.

Python та переважна більшість бібліотек до нього безкоштовні й поставляються у вихідних кодах. Навіть більше, на відміну від багатьох відкритих систем, ліцензія ніяк не обмежує використання Python у комерційних розробках та не накладає ніяких зобов'язань, крім зазначення авторських прав.

Графічні бібліотеки

З Python поставляється бібліотека tkinter на основі Tcl/Tk для створення крос-платформних програм з графічним інтерфейсом.

Для науково-технічної мети найбільшого поширення набуло використання matplotlib — бібліотеки з інтерфейсом, аналогічним MATLAB Plot Tool.

Існують розширення, що дозволяють використовувати всі основні GUI бібліотеки — wxPython[34], засноване на бібліотеці wxWidgets, PyGTK для GTK+, PyQt та PySide для Qt та інші. Деякі з них також надають широкі можливості для роботи з базами даних, графікою та мережами, використовуючи всі можливості бібліотеки, на якій базуються.

Для створення ігор та програм, що вимагають нестандартного інтерфейсу, можна використовувати бібліотеку Pygame. Вона також надає великі засоби роботи з мультимедіа: з її допомогою можна керувати звуком і зображеннями, відтворювати відео. Надаване pygame апаратне прискорення графіки OpenGL має більш високорівневий інтерфейс в порівнянні з PyOpenGL[35], що копіює семантику C-бібліотеки для OpenGL. Є також PyOgr[36], що забезпечує прив'язку до OGRE — високорівневої об'єктно-орієнтованої бібліотеки 3D-графіки. Крім того, існує бібліотека pythonOCC[37], що забезпечує прив'язку до середовища 3D-моделювання та симуляції OpenCascade.[38]

Для роботи з растровою графікою використовується бібліотека Python Imaging Library.

Порівняння з іншими мовами

Найчастіше Python порівнюють з Perl та Ruby. Ці мови також є інтерпретованими та мають приблизно однакову швидкість виконання програм. Як і Perl, Python може успішно застосовуватися для написання скриптів (сценаріїв). Як і Ruby, Python є добре продуманою системою для ООП.

Засоби функціонального програмування частково запозичені з Scheme та Icon.

У середовищі комерційних застосунків швидкість виконання програм на Python можуть порівнювати з Java-застосунками.[39]

Попри те, що Python має досить самобутній синтаксис, одним із принципів дизайну цієї мови є принцип найменшого подиву.

Недоліки

Див також списки недоліків мови Python[40].

Низька швидкодія

Python, як і багато інших інтерпретованих мов, які не застосовують, наприклад, JIT-компілятори, мають загальний недолік — порівняно низьку швидкість виконання програм.[41] Однак, у випадку з Python цей недолік компенсується зменшенням часу розробки програми.[41] У середньому, програма, написана на Python, в 2-4 рази компактніша, ніж її аналог на C++ або Java.[41] Збереження байт-коду (файли `.pyc` і `.pyo`) дозволяє інтерпретатору не витрачати зайвий час на перекомпіляцію коду модулів при кожному запуску, на відміну, наприклад, від мови Perl. Крім того, існує спеціальна JIT-бібліотека psyco[42] (проте призводить до збільшення споживання оперативної пам'яті). Ефективність psyco значною мірою залежить від архітектури програми.

Існують проєкти реалізацій мови Python, що вводять високопродуктивні віртуальні машини (ВМ) як компілятора заднього плану. Прикладами таких реалізацій може служити PyPy, що базується на LLVM; більш ранньою ініціативою є проєкт Parrot. Очікується, що використання ВМ типу LLVM призведе до тих самих результатів, що й використання аналогічних підходів для реалізацій мови Java, де низька обчислювальна продуктивність в основному подолана.[43]

Низка програм/бібліотек для інтеграції з іншими мовами програмування (див. вище) надають можливість використовувати іншу мову для написання критичних ділянок.

У найпопулярнішій реалізації мови Python інтерпретатор досить великий і більш вимогливий до ресурсів, ніж в аналогічних популярних реалізаціях Tcl, Forth, LISP або Lua, що обмежує його застосування у вбудованих системах. Тим не менше, Python знайшов застосування в КПК і деяких моделях мобільних телефонів.[44]

Відсутність статичної типізації

Відсутність статичної типізації є не стільки вадою інтерпретатора, скільки вибором розробника мови. Річ у тому, що в Python прийнята так звана «качина типізація». Через це типи переданих значень недоступні на етапі компіляції, та помилки на зразок `AttributeError` можуть виникати під час виконання. Відсутність статичної типізації також є однією з основних причин низької швидкодії.

Існують модулі, які дозволяють контролювати типи параметрів функцій на етапі виконання, наприклад *typecheck*[45] або *method signature checking decorators*[46]. Додавання необов'язковою статичної типізації параметрів функції заплановано для Python3000.[47][48] При цьому, однак, безпосередньо інтерпретатор не буде перевіряти типи, а тільки додавати відповідну інформацію до метаданих функції для її (інформації) подальшого використання модулями розширень.

Відсутність статичної типізації і деякі інші причини не дозволяють реалізувати в Python механізм перевантаження функцій на етапі компіляції. Можливості Python дозволяють реалізувати динамічне перевантаження на етапі виконання, що, звичайно, уповільнює виклик, бо вирішення яку саме функцію викликати проводиться при кожному зверненні і є, в загальному випадку, досить складною процедурою. Відсутність перевантаження в Python компенсують використанням функцій з динамічними параметрами.

Плани з підтримки перевантаження в Python3000.[47][49] Перевантаження функцій реалізована різними сторонніми бібліотеками, в тому числі PEAK[50][51] надає надзвичайно багатий можливостями механізм перевантаження функцій з використанням довільних правил.

Неможливість модифікації вбудованих класів

У порівнянні з Ruby та деякими іншими мовами, в Python відсутня можливість модифікувати вбудовані класи, такі, як `int`, `str`, `float`, `list` та інші, що, однак, дозволяє Python споживати менше оперативної пам'яті і швидше працювати. Ще однією причиною введення такого обмеження є необхідність узгодження з модулями розширення. Багато модулів (з метою оптимізації швидкодії) перетворять Python-об'єкти елементарних типів до відповідних Сі-типів замість маніпуляцій з ними за допомогою Сі-API.

Глобальне блокування інтерпретатора (GIL)

GIL (Global Interpreter Lock) — проблема, притаманна CPython, Stackless та PyPy, але відсутня в Jython та IronPython. При своїй роботі основний інтерпретатор Python постійно використовує велику кількість потіконебезпечних даних. В основному це словники, в яких зберігаються атрибути об'єктів. Для уникнення руйнування цих даних при спільній модифікації з різних потоків перед початком виконання декількох інструкцій (за замовчуванням 100) потік інтерпретатора захоплює GIL, а після закінчення звільняє. Внаслідок цієї особливості в кожен момент часу може виконуватися тільки одна нить Python коду, навіть якщо на комп'ютері є кілька процесорів або процесорних ядер (GIL також звільняється на час виконання блокуючих операцій, таких як введення-виведення, зміни/перевірка стану синхронізуючих примітивів та інших — таким чином, якщо одна нить блокується, інші можуть виконуватися). Була зроблена спроба переходу до більш гранульованої синхронізації, проте через часті захоплення/звільнення блокувань ця реалізація виявилася занадто повільною.[52] У найближчому майбутньому перехід від GIL до інших технік не передбачається, однак є python-safethread[53] — CPython без GIL і з деякими іншими змінами (за твердженнями його авторів, на однонитевих застосунках швидкість відповідає 60-65 % від швидкості оригінальному CPython).

Ця проблема має два основних варіанти вирішення. Перший — відмова від спільного використання змінюваних даних. При цьому дані дублюються в нитях і необхідність забезпечення їхньої синхронізації (якщо така потрібна) лягає на програміста.[54] Цей підхід веде до збільшення споживання оперативної пам'яті (однак не настільки сильно, як при використанні процесів).

Другий підхід — забезпечення більш гранульованої синхронізації — для окремих структур даних. У цьому випадку падає продуктивність внаслідок збільшення числа звільнень/захоплень блокувань.

Якщо необхідно паралельне виконання декількох нитей Python-коду, то можна скористатися процесами, наприклад, модулем processing[55], який імітує семантику стандартного модуля threading, але використовує процеси замість нитей. Є безліч модулів, що спрощують написання паралельних та/або розподілених застосунків на Python, таких як parallelpython[56], Pyparr[57], pympi[58] та інші. GIL звільняється при виконанні коду більшості розширень, наприклад, NumPy/SciPy, дозволяючи на час розрахунків виконуватися іншому Python-ниті. Іншим рішенням може бути використання IronPython або Jython, позбавлених даного недоліку.

Реалізації

Python портований на всі відомі платформи — від КПК до мейнфреймів. Існують порти під Windows[59], всі варіанти UNIX[60] (включно з Linux), Plan 9[61], Mac OS і Mac OS X[62], Palm OS[63], OS/2[64], Amiga, AS/400[64] і навіть OS/390[64] і Symbian[64].

При цьому, на відміну від багатьох портованих систем, на кожній платформі Python підтримує характерні для даної платформи технології (наприклад, Microsoft COM). Крім того, існує спеціальна версія Python для віртуальної машини Java — Jython, що дозволяє інтерпретатору виконуватися на будь-якій системі, що підтримує Java, класи Java можуть безпосередньо використовуватися з Python і навіть бути написаними на Python. Нещодавно почалася розробка системи, спрямованої на повнішу інтеграцію з платформою .NET — Iron Python.

Подальша розробка

Python Enhancement Proposal («PEP») — це документ зі стандартизованим дизайном, що надає загальну інформацію про мову Python, включаючи нові пропозиції, описи та роз'яснення можливостей мови. PEP пропонуються як основне джерело для пропозиції нових можливостей і для роз'яснення вибору того або іншого дизайну для основних елементів мови. Видатні PEP рецензуються і коментуються BDFL.

Графік і сумісність

Серії Python 2.x і Python 3.x протягом кількох випусків існували паралельно, при цьому серія 2.x використовувалася для забезпечення сумісності. PEP 3000 містить більше інформації про випуски.

Python 3.0 зворотно не сумісний з попередньою серією 2.x. Код Python 2.x швидше за все буде видавати помилки при виконанні в Python 3.0. Динамічна типізація Python, разом зі змінами декількох методів словників, робить механічний переклад з Python 2.x в Python 3.0 дуже складним. Однак, утиліта «2to3» здатна зробити більшість роботи з перекладу коду, вказуючи на підозрілі їй частини за допомогою коментарів і попереджень. PEP 3000 рекомендує тримати вихідний код для серії 2.x, і робити випуски для Python 3.x за допомогою «2to3». Отриманий код не слід редагувати, поки програма повинна бути працездатною в Python 2.x.

Розробники припинили підтримувати гілку Python 2.x у січні 2020 року. Остання випущена версія гілки Python 2.x: Python 2.7. Далі розробка ведеться лише у гілці Python 3.x[65].

Можливості

Основні зміни, внесені до версії 3.0:[66][67]

- Синтаксична можливість для анотації параметрів і результату функцій (наприклад, для передачі інформації про тип або документування).
- Повний перехід на unicode для рядків.
- Введення нового типу «незмінні байти» і типу «змінюваний буфер». Обидва необхідні для подання двійкових даних.
- Нова підсистема вводу-виводу (модуль `io`), що має окремі вигляди для бінарних і текстових даних.
- Абстрактні класи, абстрактні методи (є вже в 2.6).
- Ієрархія типів для чисел.
- Вирази для словників і множин `{k: v for k, v in a_dict}` і `{el1, el2, el3}` (за аналогією зі списковими виразами).
- Зміни `print` з вбудованого виразу у вбудовану функцію. Це дозволить модулям робити зміни, підлаштовуючись під різне використання функції, а також спростить код. У Python 2.6 ця можливість активується введенням `from __future__ import print_function`.
- Переміщення `reduce` (але не `map` або `filter`) з вбудованого простору в модуль `functools` (використання `reduce` істотно менш читабельне в порівнянні з циклом).
- Видалення деяких застарілих можливостей, які підтримуються у гілці 2.x для сумісності, зокрема: класи старого стилю, цілочисельний поділ з обрізанням результату як поведінка за вмовчанням, рядкові винятки, неявний відносний імпорт, оператор `exec` тощо
- Реорганізація стандартної бібліотеки.
- Новий синтаксис для метакласів.
- Змінений синтаксис присвоєння. Стало можливим, наприклад, надання `(a, * rest, b) = range(5)`. З іншого боку, формальні параметри функцій на зразок `def foo (a, (b, c))` більше неприпустимі.

Основні пропозиції

Перелік всіх пропозицій наведено на офіційному сайті[68] разом із їх статусом. Нижче наведено ті, які здобули найбільше поширення:

- PEP8 Настанова щодо стилю оформлення коду.[69]

Спеціалізовані підмножини/розширення Python

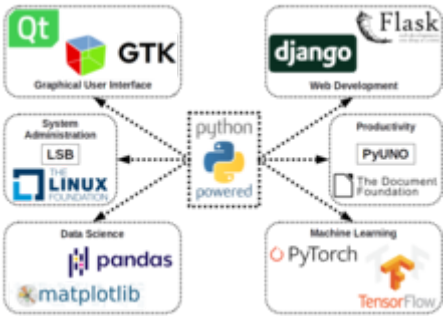
На основі Python було створено кілька спеціалізованих підмножин мови, в основному призначених для статичної компіляції в машинний код. Деякі з них:

- RPython[70] — створена в рамках проєкту PyPy значно обмежена реалізація Python без динамізму часу виконання та деяких інших можливостей. RPython код можна компілювати в багато інших мов/платформ — C, JavaScript, Lisp, .NET[71], LLVM. На RPython написано інтерпретатор PyPy.
- Pyrex[31] — обмежена реалізація Python, але трохи менше, ніж RPython. PyReX розширено можливостями статичної типізації типами з мови C і дозволяє вільно змішувати типізований та не типізований код. Призначений для написання модулів розширень, компілюється в код на мові C.
- Cython[30] — розширена версія Pyrex.
- pyastra[72] — компілятор Python коду в асемблер для PIC архітектури.
- Shed Skin[33] — призначений для компіляції неявно статично типізованого Python коду в оптимізований код на мові C++, проєкт далекий від завершення.
- PyScript — це фронтенд-фреймворк[73] для створення програм на Python, вбудованих у HTML-код, для виконання в браузері. Дозволяє імпортувати більшість модулів Python.

Застосування

Python — стабільна та поширена мова. Вона використовується в багатьох проєктах та в різних якостях: як основна мова програмування або для створення розширень та інтеграції додатків. На Python реалізована велика кількість проєктів, також вона активно використовується для створення прототипів майбутніх програм.

Python використовується в багатьох великих компаніях.[74]



Python Powered

Документація та підручники

Опис стандартних об'єктів та модулів, дивіться Python Library Reference (https://web.archive.org/web/20051003194314/http://python.org/doc/2.4.1/lib/lib.html). Python Reference Manual (https://web.archive.org/web/20050901161316/http://www.python.org/doc/2.4.1/ref/ref.html) містить формальніше визначення мови.

Щоб писати розширення на C та C++, читайте Extending and Embedding the Python Interpreter (https://web.archive.org/web/20050913161653/http://python.org/doc/2.4.1/ext/ext.html) та Python/C API Reference (https://web.archive.org/web/20050915035315/http://python.org/doc/2.4.1/api/api.html).

- Марк Пілігрим. *Пориньте у Python 3*.

Інтегровані середовища розробки

Для Python існує кілька спеціалізованих середовищ розробки, зокрема, IDLE, Thonny, PyCharm,PyScripter, Spyder (останній призначено для наукових розробок), а також розширення для Visual Studio Code і плагін PyDev для Eclipce[75].

Див. також

- Anaconda
- Інтерпретатор
- Інші популярні інтерпретовані мови програмування
  - PHP
  - Perl
  - Ruby

Джерела

1. Historique et licence (https://docs.python.org/3/license.html)
2. https://docs.python.org/3/license.html (https://docs.python.org/3/license.html)
3. Łukasz Langa (06 грудня 2021). Python 3.10.1 is available (https://web.archive.org/web/20220106181415/https://blog.python.org/2021/12/python-3101-is-available.html). *Python Insider*. Архів оригіналу (https://blog.python.org/2021/12/python-3101-is-available.html) за 6 січня 2022. Процитовано 06 грудня 2021.
4. Donovan, Ryan (23 квітня 2020). The final Python 2 release marks the end of an era (https://web.archive.org/web/20201016224330/https://stackoverflow.blog/2020/04/23/the-final-python-2-release-marks-the-end-of-an-era/). *Stackoverflow Blog*. Stack Overflow. Архів оригіналу (https://stackoverflow.blog/2020/04/23/the-final-python-2-release-marks-the-end-of-an-era/) за 16 жовтня 2020. Процитовано 09 жовтня 2020.
5. Python 3.11.0a3 is available (https://web.archive.org/web/20220106181414/https://blog.python.org/2021/12/python-3110a3-is-available.html). Python Software Foundation. Архів оригіналу (https://blog.python.org/2021/12/python-3110a3-is-available.html) за 6 січня 2022. Процитовано 6 січня 2022.
6. Архівована копія (https://web.archive.org/web/20191216072443/https://github.com/python/devguide/pull/344). Архів оригіналу (https://github.com/python/devguide/pull/344) за 16 грудня 2019. Процитовано 1 січня 2020.
7. Архівована копія (https://web.archive.org/web/20180106000102/https://www.python.org/doc/essays/foreword/). Архів оригіналу (https://www.python.org/doc/essays/foreword/) за 6 січня 2018. Процитовано 28 серпня 2008.
8. Guido van Rossum, *Python Reference Manual*, release 2.4.4, 18 October 2006.
9. The Making of Python (https://web.archive.org/web/20160901183332/http://www.artima.com/intv/pythonP.html). Архів оригіналу (http://www.artima.com/intv/pythonP.html) за 1 вересня 2016. Процитовано 4 липня 2010.
10. Архівована копія (https://web.archive.org/web/20160217132249/http://svn.python.org/view/*checkout*/python/trunk/Misc/HISTORY). Архів оригіналу (http://svn.python.org/view/*checkout*/python/trunk/Misc/HISTORY) за 17 лютого 2016. Процитовано 4 липня 2010.
11. Index of Python Enhancement Proposals (PEPs) (https://web.archive.org/web/20070128214511/http://www.python.org/dev/peps/). Архів оригіналу (https://www.python.org/dev/peps/) за 28 січня 2007. Процитовано 4 липня 2010.
12. Python 3.0 Release (https://web.archive.org/web/20090602015124/http://www.python.org/download/releases/3.0/). Архів оригіналу (https://python.org/download/releases/3.0/) за 2 червня 2009. Процитовано 4 липня 2010.

13. PEP 20 — The Zen of Python (https://web.archive.org/web/20050717021621/http://www.python.org/peps/pep-0020.html). Архів оригіналу (https://www.python.org/peps/pep-0020.html) за 17 липня 2005. Процитовано 22 серпня 2014.
14. Жартівливий натяк на національність Гвідо
15. Foreword for «Programming Python» (1st ed .) (https://web.archive.org/web/20180106000102/https://www.python.org/doc/essays/foreword/). Архів оригіналу (https://www.python.org/doc/essays/foreword/) за 6 січня 2018. Процитовано 28 серпня 2008.
16. The Making of Python (https://web.archive.org/web/20170606003042/http://www.artima.com/intv/python2.html). Архів оригіналу (http://www.artima.com/intv/python2.html) за 6 червня 2017. Процитовано 4 липня 2010.
17. Python on Android (http://www.damonkohler.com/2008/12/python-on-android.html) (англійською). www.damonkohler.com. Архів (https://www.webcitation.org/5w4n0S15k?url=http://www.damonkohler.com/2008/12/python-on-android.html) оригіналу за 28 січня 2011. Процитовано 19 грудня 2009.
18. Port-Specific Changes: Windows (https://docs.python.org/whatsnew/2.6.html#port-specific-changes-windows). *Python v2.6.1 documentation. What's New in Python 2.6* (англійською). Python Software Foundation. Архів (https://www.webcitation.org/5w4n1Glim?url=http://docs.python.org/whatsnew/2.6.html#port-specific-changes-windows#port-specific-changes-windows) оригіналу за 28 січня 2011. Процитовано 11 грудня 2008.
19. IPython (https://web.archive.org/web/20180804135112/http://ipython.scipy.org/). Архів оригіналу (http://ipython.scipy.org/) за 4 серпня 2018. Процитовано 20 червня 2019.
20. PEP318 (https://web.archive.org/web/20131106174805/http://www.python.org/dev/peps/pep-0318/). Архів оригіналу (https://www.python.org/dev/peps/pep-0318/) за 6 листопада 2013. Процитовано 5 липня 2010.
21. — Professional Python Software, Skills and Services (http://egenix.com/eGenix.com) [*недоступне посилання з серпня 2019*]
22. PEP333 (https://web.archive.org/web/20210609072317/https://www.python.org/dev/peps/pep-0333/). Архів оригіналу (https://www.python.org/dev/peps/pep-0333/) за 9 червня 2021. Процитовано 5 липня 2010.
23. Pyste Documentation (https://web.archive.org/web/20080830030353/http://boost.org/libs/python/pyste/index.html). Архів оригіналу (http://www.boost.org/libs/python/pyste/index.html) за 30 серпня 2008. Процитовано 5 липня 2010.
24. Архівована копія (https://web.archive.org/web/20080419094314/http://www.riverbankcomputing.co.uk/sip/). Архів оригіналу (http://www.riverbankcomputing.co.uk/sip/) за 19 квітня 2008. Процитовано 5 липня 2010.
25. Архівована копія (https://web.archive.org/web/20070208104412/http://pyfortran.sourceforge.net/). Архів оригіналу (http://pyfortran.sourceforge.net/) за 8 лютого 2007. Процитовано 5 липня 2010.
26. Boost.Python (https://web.archive.org/web/20070203031936/http://www.boost.org/libs/python/doc/). Архів оригіналу (http://www.boost.org/libs/python/doc/) за 3 лютого 2007. Процитовано 5 липня 2010.
27. PyCXX: Write Python Extensions in C (https://web.archive.org/web/20070203132217/http://cxx.sourceforge.net/). Архів оригіналу (http://cxx.sourceforge.net/) за 3 лютого 2007. Процитовано 5 липня 2010.
28. PyInline: Mix Other Languages directly Inline with your Python (https://web.archive.org/web/20070115141930/http://pyinline.sourceforge.net/). Архів оригіналу (http://pyinline.sourceforge.net/) за 15 січня 2007. Процитовано 5 липня 2010.
29. Weave (https://web.archive.org/web/20100706171347/http://www.scipy.org/Weave). Архів оригіналу (http://www.scipy.org/Weave) за 6 липня 2010. Процитовано 5 липня 2010.
30. Cython: C-Extensions for Python (https://web.archive.org/web/20070811093728/http://www.cython.org/). Архів оригіналу (http://www.cython.org/) за 11 серпня 2007. Процитовано 5

липня 2010.

31. Pyrex (https://web.archive.org/web/20180926130058/http://www.cosc.canterbury.ac.nz/greg.ewing/python/Pyrex/). Архів оригіналу (http://www.cosc.canterbury.ac.nz/greg.ewing/python/Pyrex/) за 26 вересня 2018. Процитовано 5 липня 2010.
32. Shedskin | Download Shedskin software for free at SourceForge.net (https://web.archive.org/web/20100611025122/http://sourceforge.net/projects/shedskin/). Архів оригіналу (http://sourceforge.net/projects/shedskin) за 11 червня 2010. Процитовано 5 липня 2010.
33. Shed Skin — An Optimizing Python-to-C++ Compiler (https://web.archive.org/web/20110811024224/http://shed-skin.blogspot.com/). Архів оригіналу (http://shed-skin.blogspot.com/) за 11 серпня 2011. Процитовано 5 липня 2010.
34. wxPython (https://web.archive.org/web/20210609072244/https://www.wxpython.org/). Архів оригіналу (http://www.wxpython.org/) за 9 червня 2021. Процитовано 5 липня 2010.
35. Архівована копія (https://web.archive.org/web/20110615075139/http://pyopengl.sourceforge.net/). Архів оригіналу (http://pyopengl.sourceforge.net/) за 15 червня 2011. Процитовано 5 липня 2010.
36. PyOgre: Ogre Wiki (https://web.archive.org/web/20090129100116/http://www.ogre3d.org/wiki/index.php/PyOgre). Архів оригіналу (http://www.ogre3d.org/wiki/index.php/PyOgre) за 29 січня 2009. Процитовано 5 липня 2010.
37. pythonOCC, 3D CAD/CAE/PLM development framework for the Python programming language (https://web.archive.org/web/20110808024321/http://www.pythonocc.org/). Архів оригіналу (http://www.pythonocc.org/) за 8 серпня 2011. Процитовано 5 липня 2010.
38. Open CASCADE Technology, 3D modeling & numerical simulation (https://web.archive.org/web/20090318141312/http://www.opencascade.org/). Архів оригіналу (http://www.opencascade.org/) за 18 березня 2009. Процитовано 5 липня 2010.
39. Результати однієї зі спроб порівняння (https://web.archive.org/web/20120831065317/http://shootout.alioth.debian.org/). Архів оригіналу (http://shootout.alioth.debian.org/) за 31 серпня 2012. Процитовано 6 липня 2010.
40. zephyrfalcon.org::labs::10 Python pitfalls (https://web.archive.org/web/20130810230937/http://zephyrfalcon.org/labs/python_pitfalls.html). Архів оригіналу (http://zephyrfalcon.org/labs/python_pitfalls.html) за 10 серпня 2013. Процитовано 6 липня 2010.
41. Python/C++ GNU g++ (https://www.webcitation.org/5w4n30Deb?url=http://shootout.alioth.debian.org/u32/benchmark.php?test=all). *Computer Language Benchmarks Game*. ???. Архів оригіналу (http://shootout.alioth.debian.org/u32/benchmark.php?test=all&lang=python&lang2=gpp&box=1) за 28 січня 2011. Процитовано 1 липня 2009.
42. Psyco (http://psyco.sf.net/) (англ.) — JIT-компілятор для Python, що дозволяє збільшити швидкість роботи програм в 3-10 разів
43. unladen-swallow. A faster implementation of Python (https://code.google.com/p/unladen-swallow/wiki/ProjectPlan). code.google. Архів (https://www.webcitation.org/5w4n3soB7?url=http://code.google.com/p/unladen-swallow/wiki/ProjectPlan) оригіналу за 28 січня 2011. Процитовано 22 червня 2009. «Goals: ... Produce a version of Python at least 5x faster than CPython»
44. Python for S60 — OpenSource (https://web.archive.org/web/20090806081738/http://wiki.opensource.nokia.com/projects/PyS60). Архів оригіналу (http://wiki.opensource.nokia.com/projects/PyS60) за 6 серпня 2009. Процитовано 6 липня 2010.
45. Typechecking module for Python (https://web.archive.org/web/20100817032141/http://oakwinter.com/code/typecheck/). Архів оригіналу (http://oakwinter.com/code/typecheck/) за 17 серпня 2010. Процитовано 6 липня 2010.
46. Method signature checking decorators «Python recipes» ActiveState Code (https://web.archive.org/web/20080213130748/http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/426123). Архів оригіналу (http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/426123) за 13 лютого 2008. Процитовано 6 липня 2010.

47. PEP-3107 (https://web.archive.org/web/20070508221139/http://www.python.org/dev/peps/pep-3107/). Архів оригіналу (https://www.python.org/dev/peps/pep-3107/) за 8 травня 2007. Процитовано 6 липня 2010.
48. PEP-3100 (https://web.archive.org/web/20100619030924/http://python.org/dev/peps/pep-3100/). Архів оригіналу (https://www.python.org/dev/peps/pep-3100/) за 19 червня 2010. Процитовано 6 липня 2010.
49. PEP-3124 (https://web.archive.org/web/20070703032817/http://www.python.org/dev/peps/pep-3124/). Архів оригіналу (https://www.python.org/dev/peps/pep-3124/) за 3 липня 2007. Процитовано 6 липня 2010.
50. FrontPage — The PEAK Developers Center (https://web.archive.org/web/20080512011822/http://peak.telecommunity.com/DevCenter/FrontPage). Архів оригіналу (http://peak.telecommunity.com/DevCenter/FrontPage) за 12 травня 2008. Процитовано 6 липня 2010.
51. PEAK-Rules (https://web.archive.org/web/20080723132426/http://peak.telecommunity.com/DevCenter/RulesReadme). Архів оригіналу (http://peak.telecommunity.com/DevCenter/RulesReadme) за 23 липня 2008. Процитовано 6 липня 2010.
52. Python 3000 FAQ (https://web.archive.org/web/20201109015427/https://www.artima.com/weblogs/viewpost.jsp?thread=211200). Архів оригіналу (http://www.artima.com/weblogs/viewpost.jsp?thread=211200) за 9 листопада 2020. Процитовано 6 липня 2010.
53. — Project Hosting on Google Code (https://code.google.com/p/python-safethread/python-safethread)
54. perlthrtut — perldoc.perl.org (https://web.archive.org/web/20080522092213/http://perldoc.perl.org/perlthrtut.html). Архів оригіналу (http://perldoc.perl.org/perlthrtut.html) за 22 травня 2008. Процитовано 6 липня 2010.
55. Python Package Index: processing 0.52 (https://web.archive.org/web/20071013121839/http://pypi.python.org/pypi/processing/). Архів оригіналу (https://pypi.python.org/pypi/processing) за 13 жовтня 2007. Процитовано 6 липня 2010.
56. Parallel Python — Home (https://web.archive.org/web/20100528112950/http://www.parallelpython.com/). Архів оригіналу (https://www.parallelpython.com/) за 28 травня 2010. Процитовано 6 липня 2010.
57. Архівована копія (https://web.archive.org/web/20100106152936/http://datamining.anu.edu.au/~ole/pypar/). Архів оригіналу (http://datamining.anu.edu.au/~ole/pypar/) за 6 січня 2010. Процитовано 6 липня 2010.
58. pyMPI.sourceforge.net: Putting the py in MPI (https://web.archive.org/web/20071018033650/http://pympi.sourceforge.net/). Архів оригіналу (http://pympi.sourceforge.net/) за 18 жовтня 2007. Процитовано 6 липня 2010.
59. Python for Windows [1] (https://www.python.org/download/windows/) [Архівовано (https://web.archive.org/web/20140911051759/https://www.python.org/download/windows/) 11 вересня 2014 у Wayback Machine.]
60. Python for UNIX [2] (https://docs.python.org/2/using/unix.html) [Архівовано (https://web.archive.org/web/20140826113426/https://docs.python.org/2/using/unix.html) 26 серпня 2014 у Wayback Machine.]
61. Python for Plan 9[3] (https://mail.python.org/pipermail/python-dev/2013-June/126806.html) [Архівовано (https://web.archive.org/web/20140826113754/https://mail.python.org/pipermail/python-dev/2013-June/126806.html) 26 серпня 2014 у Wayback Machine.]
62. Python for Mac OS X[4] (https://www.python.org/download/mac) [Архівовано (https://web.archive.org/web/20140826115015/https://www.python.org/download/mac) 26 серпня 2014 у Wayback Machine.]
63. Python for Palm OS[5] (http://isr.uci.edu/projects/sensos/python/) [Архівовано (https://web.archive.org/web/20130622165145/http://www.isr.uci.edu/projects/sensos/python/) 22 червня 2013 у Wayback Machine.]
64. Python for Other Platforms | SourceForge.net[6] (https://www.python.org/download/other/) [Архівовано (https://web.archive.org/web/20201127015815/https://www.python.org/download/other/) 27 листопада 2020 у Wayback Machine.]

65. Python endoflife.date (https://web.archive.org/web/20210918162455/https://endoflife.date/python) (англ.). endoflife.date. Архів оригіналу (https://endoflife.date/python) за 18 вересня 2021. Процитовано 18 вересня 2021.
66. What's New In Python 3.0 — Python v3.0.1 documentation (https://web.archive.org/web/20081004052720/http://docs.python.org/dev/3.0/whatsnew/3.0.html). Архів оригіналу (https://docs.python.org/dev/3.0/whatsnew/3.0.html) за 4 жовтня 2008. Процитовано 5 липня 2010.
67. Overview — Python v3.0.1 documentation (https://web.archive.org/web/20090721170811/http://docs.python.org/3.0/index.html). Архів оригіналу (https://docs.python.org/3.0/index.html) за 21 липня 2009. Процитовано 5 липня 2010.
68. PEP 0 -- Index of Python Enhancement Proposals (PEPs) (https://web.archive.org/web/20180701083334/https://www.python.org/dev/peps/). *python.org*. Архів оригіналу (https://www.python.org/dev/peps) за 1 липня 2018. Процитовано 30 червня 2018. (англ.)
69. PEP 8 -- Style Guide for Python Code (https://web.archive.org/web/20180629052213/https://www.python.org/dev/peps/pep-0008/). *python.org*. Архів оригіналу (https://www.python.org/dev/peps/pep-0008) за 29 червня 2018. Процитовано 30 червня 2018. (англ.)
70. PyPy (coding-guide) (https://web.archive.org/web/20070707005837/http://codespeak.net/pypy/dist/pypy/doc/coding-guide.html#restricted-python#restricted-python). Архів оригіналу (http://codespeak.net/pypy/dist/pypy/doc/coding-guide.html#restricted-python) за 7 липня 2007. Процитовано 5 липня 2010.
71. PyPy (carbonpython) (https://web.archive.org/web/20111012175212/http://codespeak.net/pypy/dist/pypy/doc/carbonpython.html). Архів оригіналу (http://codespeak.net/pypy/dist/pypy/doc/carbonpython.html) за 12 жовтня 2011. Процитовано 5 липня 2010.
72. Pyastra: python assembler translator (https://web.archive.org/web/20201109025501/http://pyastra.sourceforge.net/). Архів оригіналу (http://pyastra.sourceforge.net/) за 9 листопада 2020. Процитовано 12 травня 2022.
73. Serdar Yegulalp (JUN 15, 2022). Intro to PyScript: Run Python in your web browser (https://www.infoworld.com/article/3661628/get-started-with-pyscript-the-in-browser-python-by-anaconda.html). *InfoWorld* Подія сталася у 3:00 AM PDT. Процитовано 26 липня 2022.
74. Python Success Stories (https://web.archive.org/web/20100618234049/http://python.org/about/success/usa/). Архів оригіналу (https://www.python.org/about/success/usa) за 18 червня 2010. Процитовано 5 липня 2010.
75. 9 Best Python IDEs and Code Editors (https://www.programiz.com/python-programming/ide). Процитовано 27 липня 2022.

Помилка цитування: Тег `<ref>`, заданий в `<references>`, має атрибут групи "", який не фігурує в попередньому тексті.

Посилання

- Головний сайт спільноти розробників Python. (https://python.org) [Архівовано (https://web.archive.org/web/20210925183953/https://www.python.org/) 25 вересня 2021 у Wayback Machine.]
- Python documentation (https://docs.python.org/2/index.html) [Архівовано (https://web.archive.org/web/20150706044825/https://docs.python.org/2/index.html) 6 липня 2015 у Wayback Machine.] — документація мови програмування.
- Python 2: Курс Молодого Бійця (http://www.vitaliypodoba.com/tutorials/python2-beginners-course/) [Архівовано (https://web.archive.org/web/20140819083904/http://www.vitaliypodoba.com/tutorials/python2-beginners-course/) 19 серпня 2014 у Wayback Machine.] (укр.)
- «Програмування на мові Python (3.x). Початковий курс» (https://sites.google.com/site/pythonukr/) [Архівовано (https://web.archive.org/web/20130923061918/https://sites.google.com/site/pythonukr/) 23 вересня 2013 у Wayback Machine.] (укр.)
- Підручник з Python українською (http://docs.linux.org.ua/Програмування/Python/Підручник_мови_Python/) [Архівовано (https://web.archive.org/web/20161021001234/http://docs.linux.org.ua/Програ

мування/Python/Підручник_мови_Python/) 21 жовтня 2016 у Wayback Machine.]

- embed python compiler in html (https://www.brmgha.com/ide) [Архівовано (https://web.archive.org/web/20231225033536/https://www.brmgha.com/ide) 2 Лютий 2023 у Wayback Machine.]

Python	
Реалізації	CircuitPython · CLPython · CPython · Cython · IronPython · Jython · MicroPython · Psyco · PyPy · Python for S60 · Stackless Python · Unladen Swallow
Вебфреймворк	BlueBream · CherryPy · Django · Flask · Grok · Nagare · Nevow · Pyjs · Pylons · Pyramid · Quixote · Spyce · TACTIC · Tornado · TurboGears · Twisted · Webware · web2py · Zope
IDE	Boa · Eric · IDLE · Ninja-IDE · PyCharm · PyDev · PyScripter · Visual Studio · Spyder · SPE · Wing IDE · *інші...*
Стандарт	WSGI
Інше	Стандартна бібліотека мови Python
Python Software Foundation · PyCon	

Мови програмування		
Низькорівневі		Мова асемблера (ASSK · FASM · GAS · HLA · MASM · NASM · TASM · WASM) · PLAN · SAS · Sawik · TUZ-3 · Байт-код (*P*-код · Байт-код Java · Байт-код Perl · Керований код) · Машинний код
Високорівневі	**Загального призначення**	ABAP/4 · Action! · ActionScript · Ada · ALGOL · Alice · APL · AWK · B · BASIC · BCPL · C · C-- · C++ (C++11, C++14, C++17, C++20, C++23) · C++/CLI · C# · Clarion · Clascal · Clipper · Clojure · COBOL · CoffeeScript · COMAL · Common Lisp · CPL · Crystal · D · Delphi.NET · Delphi Prism · Eiffel · Erlang · F# · Forth · Fortran · FreeBASIC · Free Pascal · GAUSS · Go · Haskell · Icon · Informix-4GL · Java · JavaScript · Kotlin · Lisp · Logo · Lua · MCPL · Meta Language · Microsoft Small Basic · Modula-2 · MUMPS · Nemerle · Oberon · Object Pascal · Objective-C · OCaml · occam · Oxygene · Opa · Oz · Pascal · PL/I · PL/M · PLEX · Processing · Python · REXX · RPG-II · Racket · Ruby · Rust · SAKO · SAS 4GL · Scala · Scheme · Simula · Smalltalk · Snobol · Standard Meta Language · Swift · Tcl · Turbo Pascal · Vala · Visual Basic · Visual Basic.NET · Visual J Sharp · ЛЯПАС

	Серверні	Perl · PHP
	Запитів до баз даних[суперечливо 1]	PL/SQL · SQL · Transact-SQL
	Розмітки та векторної графіки[суперечливо 1]	Asymptote · HTML · Haml · METAFONT · PostScript · XML
	Синхронні	Lustre
	Символьних та чисельних обчислень	Maple · Mathcad · MATLAB · R · Wolfram (Mathematica)
	Квантових обчислень	Q# · QCL
	Логічні	Mercury · Prolog
Академічні	Agda · CLU · Curry · Haskell · Hope · Lisp · ML · OBJ · P″ · Pascal · Racket · Scheme	
Езотеричні	Befunge · Brainfork · Brainfuck · FALSE · FRACTRAN · HQ9+ · Illgoll · INTERCAL · Legal · LOLCODE · Malbolge · Pandora · Piet · Shakespeare · Unlambda · Velato · Whirl · Whitespace	
Порівняння мов програмування · Список мов програмування · Хронологія мов програмування		
Немає загальноприйнятого рішення, чи вважати усі ці мови саме мовами програмування		

Диференційовні обчислення	
Загальне	Диференційовне програмування · Нейронна машина Тюрінга · Диференційовний нейрокомп'ютер · Автоматичне диференціювання · Нейроморфні обчислення · Кабельна теорія · Розпізнавання образів · Теорія обчислювального навчання · Тензорний аналіз
Поняття	Градієнтний спуск (СГС) · Кластерування · Регресія (Перенавчання) · Змагальність · Увага · Згортка · Функції втрат · Зворотне поширення · Унормовування · Передавальна функція (Нормована експоненційна · Сигмоїда · Випрямляч) · Регуляризація · Набори даних (Нарощування)
Мови програмування	*Python* · *Julia*
Застосування	Машинне навчання · Штучна нейронна мережа (Глибинне навчання) · Наукові обчислення · Штучний інтелект
Апаратне забезпечення	Інтелектний процесор · Тензорний процесор · Зоровий процесор · Мемристор · *SpiNNaker*
Програмні бібліотеки	*TensorFlow* · *PyTorch* · *Keras* · *Theano*
Втілення	

	Аудіовізуальні	*NateNet* · *AlexNet* · *WaveNet* · Синтез людських зображень · Розпізнавання рукописного введення · Оптичне розпізнавання символів · Синтез мовлення · Розпізнавання мовлення · Розпізнавання облич · *AlphaFold* · *DALL-E*
	Словесні	*Word2vec* · Трансформер · *BERT* · Нейронний машинний переклад · *Project Debater* · *Watson* · *GPT-2* · *GPT-3*
	Вирішувальні	*AlphaGo* · *AlphaZero* · *Q*-навчання · *SARSA* · *OpenAI Five* · Самокерований автомобіль · *MuZero* · Обирання дії · Керування роботами
Люди	Алекс Ґрейвс · Ян Ґудфелоу · Йошуа Бенжіо · Джефрі Гінтон · Ян ЛеКун · Ендрю Ин · Деміс Гассабіс · Девід Сілвер · Фей-Фей Лі	
Організації	*DeepMind* · *OpenAI* · *MIT CSAIL* · *Mila* · *Google Brain* · *FAIR*	

Портали: Програмування • Техніка

Категорії: Штучні нейронні мережі • Машинне навчання

Вільне та відкрите програмне забезпечення	
Загальне	Копілефт • Вільне ПЗ • Відкрите програмне забезпечення • Список відкритого і вільного ПЗ • CUPS • X Window System
Історія	Лінукс • **Mozilla** (Application Suite • Firefox • Thunderbird) • Проєкт GNU
ОС на базі ВПЗ	BSD · Apple Darwin · FreeDOS · FreeBSD · OpenBSD · FreeNAS · GNU · Haiku · Hurd · Inferno · Linux · Mach · MINIX · OpenSolaris · Plan 9 · ReactOS
Розробка ВПЗ	Eclipse · FreeBASIC · Free Pascal · GCC · Java · libJIT · LLVM · Lua · Open64 · Perl · PHP · Python · ROSE · Ruby · Tcl
Менеджери вікон XWS	Blackbox • Compiz • EDE • Enlightenment • Fluxbox • GNOME • JWM • IceWM • KDE • Openbox • ROX • Window Maker • Xfce
Організації	Фонд вільного ПЗ (FSF) (європейський • індійський • латиноамериканський) • Apache Software Foundation · Blender Foundation · Eclipse Foundation · freedesktop.org · GNOME Foundation · Проєкт GNU · Google Code · Linux Foundation · Mozilla Foundation · Open Source Initiative · SourceForge · The Document Foundation · Xiph.Org · XMPP Standards Foundation · X.Org Foundation
Ліцензії	Apache · APSL · Artistic · BSD · CPL · GNU GPL · GNU LGPL · MIT · CDDL · MPL · Ms-PL/RL · zlib · FSF approved licenses · Дозвільна ліцензія вільного ПЗ · Вільна ліцензія · Розмаїття ліцензій

Проблеми	Двійковий блоб · Технічні засоби захисту авторських прав · Вільні та відкриті графічні драйвери · Ліцензійна проліферація · Iceweasel · Безпека відкритого ПЗ · Власницьке програмне забезпечення · Конфлікт SCO—Linux · Програмні патенти · Тівоізація · Trusted Computing · Апаратне обмеження · Вірусна ліцензія
Інше	Альтернативні назви · Рух · Вільне та відкрите ПЗ • Microsoft Open Specification Promise · Порівняння відкритих та закритих кодів · *Revolution OS* · *Собор і базар*