# پایتون (زبان برنامه‌نویسی)

**پایتون** (به انگلیسی: **Python**) یک زبان برنامه‌نویسی همه منظوره،[3] سطح بالا،[3] شیءگرا و مفسر است که توسط خودو فان روسوم (به هلندی: Guido van Rossum) در سال ۱۹۹۱ در کشور هلند طراحی شد.

فلسفهٔ ایجاد آن تأکید بر دو هدف اصلی خوانایی بالای برنامه‌های نوشته شده[4] و کوتاهی و بازدهی نسبی بالای آن است.[5] کلمات کلیدی و اصلی این زبان به صورت حداقلی تهیه شده‌اند و در مقابل کتابخانه‌هایی که در اختیار کاربر است بسیار وسیع هستند.

بر خلاف برخی زبان‌های برنامه‌نویسی رایج دیگر که بلاک‌های کد در آکولاد تعریف می‌شوند (به‌ویژه زبان‌هایی که از گرامر زبان سی پیروی می‌کنند) در زبان پایتون از نویسه فاصله و جلوبردن متن برنامه برای مشخص کردن بلاک‌های کد استفاده می‌شود. به این معنی که تعدادی یکسان از نویسه فاصله در ابتدای سطرهای هر بلاک قرار می‌گیرند، و این تعداد در بلاک‌های کد درونی‌تر افزایش می‌یابد. بدین ترتیب بلاک‌های کد به صورت خودکار ظاهری مرتب دارند.

پایتون مدل‌های مختلف برنامه‌نویسی (از جمله شیء گرا و برنامه‌نویسی دستوری و تابع محور) را پشتیبانی می‌کند و برای مشخص کردن نوع متغییرها از یک سامانهٔ پویا استفاده می‌کند.

این زبان از زبان‌های برنامه‌نویسی مفسر بوده و به صورت کامل یک زبان شی‌گرا است که در ویژگی‌ها با زبانهای تفسیری پرل، روبی، اسکیم، اسمال‌تاک و تی‌سی‌ال مشابهت دارد و از مدیریت خودکار حافظه استفاده می‌کند.[6][7][8]

پایتون پروژه‌ای آزاد و متن‌باز توسعه‌یافته‌است و توسط بنیاد نرم‌افزار پایتون مدیریت می‌گردد.[9]

مخترع زبان پایتون خودو فان روسوم

## ۱ تاریخچه

پایتون اواخر دهه ۱۹۸۰ (میلادی) توسط خودو فان روسوم در مؤسسه ملی تحقیقات ریاضی و رایانه (CWI) در کشور هلند ایجاد شد. هدف خَودو ایجاد جانشینی برای زبان برنامه‌نویسی ای‌بی‌سی بود[10] که قابلیت پردازش استثناها را داشته باشد.[11] خَودو طراح اصلی پایتون است و نقش مداوم او در تصمیم‌گیری پیرامون اهداف پایتون، باعث شد که انجمن پایتون به او لقب دیکتاتور خیرخواه جاویدان (به انگلیسی: Benevolent Dictator For Life) را بدهد.[12]

### ۱.۱ نسخه‌های اولیه

در سال ۱۹۹۱ فان روسوم کدی با برچسب نسخه ۰٫۹٫۰ را منتشر کرد. البته در این مرحله از پیشرفت کلاس‌هایی با خاصیت ارث بری، پردازش استثنا، توابع و انواع داده list, dict, str وجود داشت. همچنین در این نسخه ابتدایی یک سیستم ماژول با اقتباس از ماژول-۳ وجود داشت، که فان روسوم این ماژول را به عنوان «یکی از واحدهای عمده برنامه‌نویسی پایتون» توصیف کرد. مدل استثناء پایتون نیز شباهت‌هایی به ماژول-۳ داشت، که به آن شرط else افزوده شده بود. در سال ۱۹۹۴ اولین مجمع مباحثه پیرامون پایتون شکل گرفت که مرحله برجسته‌ای در پیشرفت کاربری پایتون بود.

### ۲.۱ نسخه ۱٫۰

پایتون در ژانویه ۱۹۹۴ به نسخه ۱٫۰ رسید. خصوصیات عمده جدید این نسخه شامل ابزارهای برنامه‌نویسی تابعی lambda, map, filter, reduce بود.

پایتون نسخه ۱٫۲ در سال ۱۹۹۵، زمانیکه خَودو در CVVI بود، ارائه شد. خَودو به فعالیت‌های خود روی پایتون در مؤسسه ملی تحقیقات و ابتکارات (CNRI) در رستون ادامه داد و در همان‌جا چندین نسخه جدید انتشار یافت.

در نسخه ۱٫۴ به پایتون چندین ویژگی جدید اضافه شد. از ویژگی‌های جالب توجه در میان این اضافات می‌توان به الهام گرفتن از آرگومان‌های کلیدی ماژول-۳ (که خود از آرگومان‌های کلیدی لیسپ الهام گرفته بود) و همچنین پشتیبانی اعداد مختلط اشاره کرد.

در مدت فعالیت فان روسوم در CNRI، او پروژه «برنامه‌نویسی کامپیوتر برای هر کس» (CP4E) را ایجاد کرد تا برنامه‌نویسی را در دسترس افراد بیشتری که اطلاعات پایه‌ای برای برنامه‌نویسی (برای مثال توانایی در زبان انگلیسی و مهارتهای اولیهٔ ریاضی) را دارند، قرار دهد. زبان برنامه‌نویسی پایتون به دلیل تمرکزش بر روی پاکسازی فرم دستوراتش، نقش مرکزی را ایفا کرد. این پروژه توسط دارپا سرمایه‌گذاری شد واز سال ۲۰۰۷ غیرفعال

شد.

### ۳.۱ متن باز و آزاد بودن

در سال ۲۰۰۰ تیم توسعه دهنده پایتون به BeOpen.com منتقل شد و بدین صورت تیم کتابخانه باز پایتون شکل گرفت. به پیشنهاد CNRI ورژن ۱٫۶ساخته شد، بدین ترتیب برنامه‌های تولید شده برای ۱٫۶و ۲٫۰اشتراک‌های قابل توجهی داشتند. فقط پایتون ۲٫۰ توسط BeOpen.com طراحی شده بود. بعد از تولید پایتون ۲٫۰ توسط BeOpen.com، خَودو و دیگر توسعه دهندگان کتابخانه پایتون به ایجاد دیجیتال روی آوردند. تولید پایتون ۱٫۶ شامل جواز جدید CNRI بود که به طور قابل توجهی طولانی‌تر از جواز CWI (که برای تولیدات قبلی استفاده شده بود) بود. بنیاد نرم‌افزار آزاد توضیح داد که انتخاب شرط قانون با GNU GPL ناسازگار بود. , BeOpen CNRI و FSF تغییراتی را در جواز پایتون ایجاد کردند که با GPL سازگار باشد. پایتون ۱٫۶٫۱ عمدتاً مشابه پایتون ۱٫۶ است فقط با کمی اشکال، و با جواز سازگار با GPL.

### ۴.۱ نسخه ۲٫۰

پایتون ۲٫۰ فهرستی از ویژگی‌هایی را که از زبان‌های برنامه‌نویسی تابعی ستل و همسکل اقتباس شده بود، معرفی کرد.[13] نحو پایتون برای این ساختار (جدا از برتری همسکل برای کاراکترهای نقطه گذاری و کلمات الفبا) بسیار مشابه همسکل بود. پایتون ۲٫۰ همچنین یک سیستم بازیافت حافظه با قابلیت جمع‌آوری منابع معرفی کرد. پایتون ۲٫۱ به پایتون ۱٫۶٫۱ و ۲٫۰ نزدیک بود. جواز آن به جواز مؤسسه نرم‌افزار پایتون تغییر نام یافت. همه کدها، اسناد و مشخصات اضافه شده را از زمان تولید الفبای پایتون ۲٫۱ توسط مؤسسه نرم‌افزار پایتون (PFS) دارا شد. یک سازمان غیرانتفاعی در سال ۲۰۰۱ تشکیل شد که از مؤسسه نرم‌افزار آپاچی مدل گرفته بود. تولیدات شامل تغییراتی در خصوصیات زبان در پوشش حوزه‌های تو در تو بود، مشابه دیگر زبان‌های حوزه‌ای ایستا. (این خصوصیات دوباره از بین رفتند و به پایتون ۲٫۲ منتقل نشدند) یک تغییر بزرگ در پایتون ۲٫۲ یکسان‌سازی انواع داده‌ای پایتون و کلاس‌ها به یک سلسله مراتب بود. این یکسان‌سازی اشیاء پایتون را کاملاً شیء گرا کرد.

### ۵.۱ میراث جاوا

انتخاب نحو و ضمایم کتابخانه استاندارد پایتون شدیداً وابسته به بعضی موارد در جاوا بود: بسته logging در ورژن ۲٫۳، تجزیه کننده SAX در ورژن ۲٫۰ و ساختمان‌های نحو که در ورژن ۲٫۴ اضافه شد.

## ۲ توسعه خصوصیات

یک طرح افزایش (PEP) در پایتون یکنواخت کردن اسنادی است که اطلاعات عمومی ای را که پایتون را شرح می‌دهند تولید می‌کنند؛ شامل پیشنهادها، توصیف‌ها و توضیح‌ها برای خصوصیات زبان. PEP در نظر داشت همانند روش‌های اولیه، برای پیشنهاد خصوصیات جدید و نیز برای مستندسازی طرح‌های اساسی، هر عامل بزرگ در پایتون را توضیح دهد. طرح‌های برجسته توسط van Rossum تجدید نظر شده و توضیح داده شدند.

### ۱.۲ پایتون ۳

پایتون ۳٫۰ (که پایتون ۳۰۰۰ ویا Py3k نیز خوانده می‌شود) به منظور شکستن سازگاری عقبرو (به انگلیسی: backward compatibility) یا به عبارتی قطع سازگاری با گذشته پایتون ۲ و بهبود خطاها و رخنه‌ها در نسخهٔ ۲ و همچنین حذف روشهای قدیمی طراحی شد.[14] به عبارتی دیگر لازم نیست که پایتون ۳ بتواند کدی که با پایتون ۲ نوشته شده را تفسیر کند که البته این مشکل توسط نرم‌افزار 2to3 حل می‌شود.[15]

### ۲.۲ فلسفه

پایتون ۳٫۰ با همان فلسفه ورژن‌های قبل، در حال توسعه یافتن است، بنابراین هر منبعی در فلسفه پایتون، در پایتون ۳٫۰ به خوبی ظاهر خواهد شد. اگر چه، همان‌طور که پایتون روش‌های جدید در برنامه‌ریزی را جمع‌آوری کرده، پایتون ۳٫۰ تأکید زیادی بر از بین بردن ساختارها و ماژول‌های تکراری دارد: «باید یک  و ترجیحاً فقط یک  روش بدیهی برای انجام آن وجود داشته باشد.» با این وجود پایتون ۳٫۰ به ساختار زبان چند نمونه‌ای ادامه خواهد داد. کد نویس‌ها همچنان اختیارات شیء گرایی، برنامه‌نویسی ساخت یافته، برنامه‌نویسی تابعی و دیگر نمونه‌ها را دارند، اما در انتخاب‌های وسیع، جزئیات در پایتون ۳٫۰ آشکارتر از پایتون سری x.۲ هستند.

### ۳.۲ سازگاری و همزمانی

اولین کاندید پایتون ۳٫۰ در ۱۷ سپتامبر ۲۰۰۸منتشر شد. پایتون سری x.۲ و x.۳ به طور موازی با هم وجود خواهند داشت، جایی که سری x.۲ سازگاری بیشتری دارد، به جای سری x.۳ مورد استفاده قرار خواهد گرفت. PEP ۳۰۰۰ اطلاعات بیشتری را در مورد فهرست نشریات دارا ست. پایتون ۳٫۰ سازگاری قبل را نقض خواهد کرد. الزامی ندارد که کدهایی که با پایتون x.۲ اجرا می‌شوند، برای پایتون ۳٫۰ بدون تغییر اجرا شوند. چون تغییرات اساسی بین این دو ورژن وجود دارد مثل اختلاف در حالت پرینت (بنابراین هر استفاده از پرینت به عنوان توضیح باعث شکست برنامه می‌شود) نوع پویای پایتون با طرح‌های تغییر معنای روش‌های خاص دیکشنری ترکیب می‌شود، به عنوان مثال، انتقال مکانیکی بی نقص از پایتون x.۲ به پایتون ۳٫۰ را بسیار دشوار می‌کند. اگرچه ابزاری به نام «۲to۳» بسیاری از این وظایف انتقال را انجام می‌دهد، اما باید توجه داشت که استفاده از توضیحات یا اخطارها با ابهام همراه است. البته در یک مرحله از الفبا، ۲to۳ انتقال را حقیقتاً کامل انجام می‌دهد. PEP ۳۰۰۰ پیشنهاد می‌کند که یک منبع نگه داشته شود (برای سری x.۲)، و نسخه‌ای بر مبنای پایتون ۳٫۰ با استفاده از ۲to۳ تولید شود. کدهای نتیجه شده نباید تصحیح شوند، مگر اینکه کدی طولانی‌تر از محدوده سری x.۲ باشد. پایتون ۲٫۶ شامل خصوصیات سازگاری مستقیم است، به طوری که یک روش اخطار (warning) به صورت خودکار به مسائل انتقال هشدار می‌دهد. هشدارها باید برای تشخیص خطا گزارش داده شوند، مشابه خصوصیات ورژن‌های قبلی پایتون. (برای اطلاعات بیشتر به PEP ۳۶۱ رجوع کنید)

### ۴.۲ خصوصیات

فهرست برخی از تغییرات عمده پایتون ۳٫۰:

- تغییر پرینت چون یک تابع غیرقابل انتقال است نه یک توضیح. این باعث می‌شود که تغییر یک ماژول برای استفاده از یک تابع پرینت متفاوت، آسان باشد و بنابراین ایجاد نحو منظم تر می‌شود. در پایتون ۲٫۶ این امکان با تایپ کردن from – future – import print – function فراهم شد.
- اضافه شدن حمایت از یادآوری تابع انتخابی که می‌تواند برای معرفی تایپ خصوصی یا اهداف دیگر استفاده شود.
- یکسان کردن تایپ str/Unicode، به نمایندگی از یک متن، و معرفی یک تایپ byte تغییرناپذیر؛ با یک تایپ مطابق با bytearray تغییرپذیر، که هر دو آرایه از بایت را ارائه می‌کنند.

- از بین بردن خصوصیات سازگاری معکوس، شامل کلاس‌های به فرم قبل، قسمت کردن اعداد صحیح، استثناءهای رشته‌ای، و گزارش‌های نسبتاً نا آشکار.

# ۳ کاربرد

سازمان‌های بزرگی که از پایتون استفاده می‌کنند، شامل گوگل، یاهو، سرن و ناسا هستند. ITA نیز از پایتون برای بعضی از اجزای خود استفاده می‌کند.

## ۱.۳ امنیت اطلاعات

پایتون همچنین استفاده وسیعی از صنعت ایمنی اطلاعات می‌کند. مثلاً در چندین ابزار پیشنهاد شده توسط تأمین امنیت و امنیت مرکزی و اسکنر امنیت کاربردی وب واپیتی. پایتون معمولاً در توسعه کاربرد مورد استفاده قرار می‌گیرد.

## ۲.۳ جاسازی

پایتون با موفقیت در تعدادی از تولیدات نرم‌افزاری مثل زبان فایل آغاز گر تعبیه شده‌است. پایتون معمولاً در بسته‌های انیمیشن ۳D استفاده می‌شود، مانند Houdini, Maya, Softimage XSI, TrueSpace, Poser, GIMP, Modo, Nuke, Blender, کریتا، Inkcape و Scribus, Paint Shop Pro.

شرکت ازری (ESRI) هم اکنون در حال ترقی دادن پایتون به عنوان بهترین انتخاب برای نوشتن فایل آغازگر در آرک جی‌آی‌اس (ArcGIS) است. همچنین در بازی‌ها استفاده می‌شود، مانند Civilization IV و Mount&Blade به عنوان زبان کنترل برای نمایش و عکس العمل حوادث.

## ۳.۳ مقبولیت

در بسیاری از سیستم‌های عملیاتی، پایتون یک جزء استاندارد است؛ چون با بیشتر بخش‌های لینوکس انتقال داده می‌شود و روی NetBSD و OpenBSD و Mac OS X هم قابل نصب است. ردهت لینوکس و فدورا هر دو از نصب کننده پایتونی آناکوندا استفاده می‌کنند. لینوکس Gentoo از پایتون در سیستم مدیریت بسته، حمل و ابزارهای دستیابی خود استفاده می‌کند. Pardus از آن برای مدیریت و در طول راه‌اندازی سیستم استفاده می‌کند.

# ۴ صرف و نحو

پایتون در نظر دارد که زبانی بسیار قابل خواندن باشد؛ بنابراین به سمت یک طرح بندی ویژوال بدون پارازیت می‌رود، و اغلب از کلمات کلیدی انگلیسی استفاده می‌کند، در صورتیکه دیگر زبان‌ها از نقطه گذاری استفاده می‌کنند. پایتون نسبت به زبان‌های ساخت یافته سنتی، مثل C و پاسکال، نیاز به تکیه کلام‌های کمتر و همچنین استثناءهای نحوی و موارد خاص کمتری دارد.

نوشتار اصلی: Python syntax and semantics

## ۱.۴ ایجاد فضای خالی

پایتون از فضاهای خالی بیشتر از آکولاد یا کلمات کلیدی برای تعیین بلوک‌های حالت استفاده می‌کند (ویژگی ای که به نام قانون off-side نیز

```
def add5(x):
   return x+5

def dotwrite(ast):
   nodename = getNodename()
   label=symbol.sym_name.get(int(ast[0]),ast[0])
   print '    %s [label="%s' % (nodename, label),
   if isinstance(ast[1], str):
      if ast[1].strip():
         print '= %s"];' % ast[1]
      else:
         print '"]'
   else:
      print '"];'
      children = []
      for n, child in enumerate(ast[1:]):
         children.append(dotwrite(child))
      print '    %s -> {' % nodename,
      for name in children:
         print '%s' % name,
```

*Syntax-highlighted Python 2.x code.*

شناخته می‌شود)

## ۲.۴ شرط‌ها و روند کنترل

شرط‌های پایتون شامل:

- شرط if، که یک بلوک کد، تا else و elif را اجرا می‌کند. (یک اختصار از else-if)
- شرط for، که روی یک شیء تکرار شدنی تکرار می‌شود، به هر متغیر محلی مقدار داده می‌شود برای استفاده توسط بلوک مربوطه.
- شرط class، که یک بلوک کد را اجرا می‌کند و فضاهای محلی آن را به یک کلاس ملحق می‌کند، برای استفاده در برنامه‌نویسی شیء گرا.
- شرط def، که یک تابع را تعریف می‌کند.
- شرط with، که یک بلوک کد را به یک مدیر متن ضمیمه می‌کند. (به عنوان مثال، اندوختن یک قفل قبل از اجرای بلوک کد و آزاد کردن قفل بعد از اجرا)

هر شرطی برای خود قواعد معنایی خاصی دارد: به عنوان مثال، شرط def، بر خلاف دیگر شرط‌ها بلوک خود را فوراً اجرا نمی‌کند.

سی پایتون استمرار را پشتیبانی نمی‌کند، و مطابق نظر خَودو فان روسوم هرگز نخواهد کرد. در ورژن‌های قبلی مولد تکرار کند بود چون اطلاعات تنها در یک جهت از مولد عبور می‌کردند.

## ۳.۴ روش‌ها

روش‌ها در اشیاء پایتون، ملحق کردن توابع به اشیاء کلاس است؛ با نحو instance.method(argument) برای روش‌ها و توابع نرمال، و Class.method (instance,argument). روش‌های پایتون، یک پارامتر self آشکار برای دستیابی به داده‌های instance دارند، در برابر پارامتر self غیر آشکار در برخی زبان‌های برنامه‌نویسی شیء گرا (مانند جاوا، C++، یا روبی)

## ۴.۴ نوع دهی

پایتون از اشیاء تایپ شده و در مقابل نام متغیرهای غیر تایپ استفاده می‌کند. محدودیت تایپ در زمان کامپایل چک نمی‌شود؛ بنابراین عمل گرهای روی یک شیء ممکن است شکست بخورند، به این مفهوم که شیء داده شده از

یک تایپ مناسب نیست. با اینکه اجباری در تایپ دهی ایستا نیست، پایتون شدیداً تایپ دهی شده، و عمل گرهای نامناسب را نهی می‌کند. (مثل مقدار دهی یک رشته با یک عدد)

پایتون همچنین به برنامه‌نویس‌ها اجازه می‌دهد که تایپ دلخواه خود را تعریف کنند. این کار با استفاده از کلاس‌ها امکان‌پذیر است، و اغلب برای شیء گرایی در برنامه‌نویسی استفاده می‌شود. نمونه‌های جدید از کلاس‌ها با صدا زدن کلاس ساخته می‌شوند، و کلاس‌ها خودشان نمونه‌هایی از کلاس type هستند (خودش یک نمونه از خودش است).

# ۵ پیاده‌سازی

مسیر اصلی پیاده‌سازی پایتون، که با عنوان سی پایتون نیز شناخته می‌شود، در نشست C در استاندارد C۸۹ نوشته شد. سی پایتون برنامه‌های پایتون را به کد بایت تبدیل می‌کند، که سپس توسط ماشین مجازی اجرا می‌شود. سی پایتون با یک کتابخانه استاندارد بزرگ که به صورت مخلوطی از C و پایتون نوشته شده، توزیع شده‌است. سی پایتون در ورژن‌های مختلف برای پایگاه‌های زیادی کار می‌کند، شامل مایکروسافت ویندوز و بیشتر سیستم‌های پیشرفته یونیکس. استفاده و توسعه آن روی پایگاه‌های محرمانه مانند Amoeba، در کنار پایگاه‌های متداول مانند یونیکس یا مکینتاش، به طور عمده در این نظر کمک شده‌است.

پایتون بدون پشته، انشعابی از سی پایتون است که ریز برنامه‌ها را اجرا می‌کند؛ و از پشته حافظه استفاده نمی‌کند. سی پایتون از GIL استفاده می‌کند تا وقتی برنامه پایتون بدون پشته، وابسته به OS است و می‌تواند به صورت همزمان اجرا شود، در هر لحظه فقط به یک زیر برنامه اجازه اجرا داده شود. پایتون برای استفاده در میکرو کنترلرها یا وظایف محدود دیگر پایگاه‌های مرجع، متناسب تر است. پیش بینی می‌شود که پایتون بدون پشته بتواند تقریباً روی همان پایگاهی که سی پایتون اجرا می‌شود، اجرا شود.

جایتون (به انگلیسی: Jython) برنامه‌های پایتون را به کد بایت جاوا کامپایل می‌کند، که بدین ترتیب می‌تواند با هر ماشین مجازی جاوا اجرا شود؛ و همچنین این امکان فراهم می‌شود که توابع کتابخانه‌ای کلاس جاوا از برنامه پایتون به کار گرفته شود. آی رون پایتون از همین شیوه برای اجرای برنامه‌های پایتون روی چارچوب دات‌نت استفاده می‌کند.

پای پای (به انگلیسی: PyPy) یک پیاده‌سازی تجربی از پایتون است که می‌تواند چندین تایپ از کد بایت را تولید نماید.

چندین برنامه در بسته مفسر پایتون با برنامه‌های کاربردی (یا آغازگر) وجود دارد که مستقلاً اجرا می‌شوند مانند یونیکس، لینوکس، ویندوز، AmigaOS ۴ یا Mac OS X. بسیاری از کتابخانه‌های سه قسمتی (و حتی بعضی از یک قسمتی‌ها) فقط روی ویندوز، لینوکس، BSD و Mac OS X موجود هستند.

در نوکیا ۲۰۰۵ یک مفسر پایتون برای موبایل‌های سری ۶۰ با نام PyS۶۰ تولید شد که شامل بسیاری از ماژولها از سی پایتون بود، و همچنین برخی ماژول‌های اضافه شده برای یکپارچه شدن با سیستم‌عامل. این پروژه به منظور اجرا روی همه پایگاه‌های مختلف S۶۰، به روز نگاه داشته می‌شود.

همچنین مفسری به نام پایتون سی ای برای ابزار ویندوز CE (شامل بسته PC) وجود دارد؛ که در آن ابزارهایی برای اجرای آسان و توسعه GUI اضافه شده‌است. اطلاعات بیشتر را می‌توانید روی وب‌گاه PythonCE بیابید.

پایتون چینی زبان برنامه‌نویسی پایتونی است که از لغت نامه زبان چینی استفاده می‌کند. در کنار کلمات رزرو شده و نام متغیرها، بیشتر عمل گرهای تایپ داده، در چینی می‌توانند به خوبی کد دهی شوند.



## ۱.۵ تفسیر معنایی

بیشتر پیاده‌سازی‌های پایتون (شامل سی پایتون، اولین پیاده‌سازی) می‌توانند به عنوان یک مفسر خط فرمان عمل کنند، برای زمانی که کاربر رشته شرط را وارد می‌کند و فوراً نتیجه را می‌پذیرد. خلاصه پایتون به عنوان یک برنامه واسط عمل می‌کند. وقتی صرف دیگر شیوه‌های اجرا (کامپایل کد بایت یا کامپایل کد محلی) به صورت یک رشته صرف ذخیره می‌شود، یک افزایش سرعت در هزینه‌های متقابل به وجود می‌آید، بنابراین آنها معمولاً فقط خارج از مفسر خط فرمان استفاده می‌شوند. (وقتی یک ماژول وارد می‌شود)

برنامه‌های واسط دیگر، امکانات تحت آن را در مفسر بیسیک افزایش می‌دهند، شامل IDLE و I Python. وقتی عموماً از برنامه واسط پایتون پیروی می‌شود، خصوصیاتی مشابه تکمیل خودکار، نگه داشتن زمان اجرای برنامه، و نشان دادن صرف پیاده‌سازی می‌شود.

برخی از پیاده‌سازی‌ها نه تنها به صورت کد بایت می‌توانند کامپایل شوند، بلکه می‌توانند کد پایتون را به کد ماشین تبدیل کنند. تا کنون، این عمل فقط برای زیر مجموعه‌های انحصاری پایتون انجام شده‌است. پای پای این روش را پذیرفته و ورژن‌های قابل کامپایل انحصاری پایتون را RPython نامید.

Psycoیک کامپایلر اختصاصی در زمان است که کد بایت را به کد ماشین، در زمان اجرا تبدیل می‌کند. کد تولید شده، اختصاصی برای تایپ‌های داده خاصی است و از کدهای استاندارد پایتون سریع تر است Psyco .با همه کدهای پایتون سازگار است، نه فقط یک زیر مجموعه.

# ۶ کتابخانه استاندارد

پایتون یک کتابخانه استاندارد بزرگ دارد، که از آن به عنوان یکی از بزرگ‌ترین توانایی‌های پایتون یاد می‌شود، مشروط به اینکه ابزارهای از پیش نوشته شده، با بسیاری از وظایف سازگار باشد. ماژول‌های کتابخانه استاندارد می‌توانند به شیوه ماژول‌های نوشته شده در سی یا پایتون آرگومان دهی شوند. اخیراً کتابخانه‌های C++ به یک کتابخانه به نام Boost.Python رشد یافته‌است، برای ایجاد قابلیت همکاری بین C++ و پایتون. به دلیل تنوع گسترده در ابزارهای تولید شده توسط کتابخانه استاندارد، این کتابخانه با توانایی استفاده یک زبان سطح پایین ترکیب شده، مثل C و C++، که البته به عنوان واسط بین کتابخانه‌های دیگر است، پایتون می‌تواند یک واسط قوی بین زبان‌ها و ابزارها باشد.

کتابخانه استاندارد با تعداد زیاد فرمت‌ها و پروتکل‌هایی که حمایت می‌کند (مانند MIME و HTTP)، خصوصاً برای نوشتن علائم اینترنت مناسب است. ماژول‌ها برای ایجاد واسط کاربر گرافیکی، به پایگاه داده مربوطه متصل می‌شود، محاسبات را با دقت دلخواه دسیمال انجام می‌دهد، و عبارت‌های منظم را دستکاری می‌کند. پایتون همچنین یک واحد تست مدیریت پایگاه داده برای تست کردن رشته‌ها دارد.

بعضی از قسمت‌های کتابخانه استاندارد با مشخصات پوشانده شده، اما اکثریت ماژول‌ها اینگونه نیستند. آنها از طریق کدها، اسناد داخلی، و دنباله تست شان (اگر موجود باشد) تعیین می‌شوند. اگر چه، به دلیل اینکه اکثر کتابخانه‌های استاندارد، کد پایتون مربوط به پایگاه را دارند، فقط ماژول‌های اندکی هستند که باید تغییر داده شوند یا مجدًّدا با یک پیاده‌سازی دیگر نوشته شوند.

# ۷ فلسفه برنامه‌نویسی

پایتون یک زبان برنامه‌نویسی چند پارادایمی است، شیء گرایی و برنامه‌نویسی ساخت یافته کاملاً تحت پوشش هستند و تعدادی از خصوصیات زبان‌های برنامه‌نویسی هستند که برنامه‌نویسی تابعی و ظاهر سازی را

پشتیبانی می‌کنند. پایتون از تایپ پویا و یک ترکیبی از شمارش مرجع و یک حلقه کشف و بازیافت قسمت‌های هدر رفته حافظه برای مدیریت حافظه، استفاده می‌کند. یک ویژگی مهم پایتون تحلیل نام پویا است، که روش‌ها و نام متغیرها را در طول اجرای برنامه به هم ملحق می‌کند.

هدف دیگر طراحی زبان آسان کردن توسعه پذیری است. ماژول‌هایی که تازه ساخته شده‌اند، به‌سادگی در C و C++ نوشته می‌شوند. پایتون همچنین می‌تواند به عنوان زبان توسعه برای ماژول‌ها و کاربردهای موجود که به برنامه واسط قابل برنامه‌ریزی نیاز دارد، استفاده شود. این طرح که یک زبان هسته کوچک با یک کتابخانه استاندارد بزرگ و یک مفسر آسان توسعه پذیر همراه باشد، توسط Van Rossum بیان شد.

طرح پایتون به پشتیبانی محدود برای برنامه‌نویسی تابعی به شیوه لیسپ، ارائه شد. اگر چه، تشابه‌های عمده‌ای بین پایتون و زبان خانواده لیسپ وجود دارد. این کتابخانه دو ماژول دارد (تکرار و تابعی) که ابزارهای تابعی را با اقتباس از هسکل و اMال استاندارد پیاده‌سازی می‌کند.

وقتی انتخاب‌هایی در روش‌شناسی کدها ارائه شد، پایتون نحوهای فراوان را کنار گذاشت. همچنان‌که با پرل، توسعه دهندگان پایتون فوراً یک فرهنگ یا ایدئولوژی را بر مبنای آنچه از یک زبان می‌خواهند، ترقی دادند، ساختار زبان‌ها زیبا، آشکار و ساده شد. Alex Martelli این مطلب را در کتاب خود قرار داد: برای توضیح برخی مسائل، در فرهنگ پایتون تعریفی مطرح نشده‌است. «پایتون روش پرل را (بیش از یک روش در انجام آن وجود دارد) در طراحی زبان در حمایت از» باید یک راه  و ترجیحاً فقط یک راه  آشکار برای انجام آن وجود دارد. رد کرد.

پایتون از بهینه‌سازی بی موقع اجتناب کرد، و بعلاوه به هم جور کردن قسمت‌های غیر ضروری سی پایتون را که افزایش سرعت نهایی در هزینه را ارائه می‌کرد، رد کرد. آن گاهی اوقات با نام ' slow ' شناخته می‌شود. اگر چه، بیشتر مسائل چندان بحرانی نیستند، و همین‌طور سرعت سخت‌افزار کامپیوتر با سرعت نمائی رو به رشد است. وقتی سرعت یک مسئله باشد، برنامه نویسان پایتون بیشتر تلاش می‌کنند تا عملیات محدود را با بهبود الگوریتم یا تغییر ساختار داده، بهینه کنند.

### ۱.۷ واژه تراشی

یک واژه تراشی رایج در انجمن پایتون، در pythonic است، که می‌تواند محدوده وسیع معنایی وابسته به استیل برنامه داشته باشد. در مقابل یک کد unpythonic تلاش می‌کند تا یک کد C++ را در پایتون بنویسد.

## ۸ IDEمحیط‌های ویرایشگر کد پایتون

## ۹ واسط گرافیکی

برای پایتون واسط گرافیکی کاربر بسیاری نوشته شده است پرکاربردترین آنها به شرح زیر می‌باشد:[16][17]

- تکینتر (به صورت پیش‌فرض همراه با نسخه‌های استاندارد پایتون ارائه می‌شود و یک رابط شی‌گرا برای ابزار Tcl/Tk در محیط پایتون فراهم می‌کند)
- پای‌کیوت
- PyGTK
- wxPython
- pyFLTK
- FXpy
- PyOpenGL

## ۱۰ برنامه‌هایی که کاملاً یا بخشی از آن‌ها با پایتون نوشته شده است

- اینستاگرام (Insagram): نرم‌افزار اشتراک گذاری تصاویر و ویدیوها
- بیت‌تورنت (نرم‌افزار) (BitTorrent): نرم‌افزار کلاینت برای فایل‌های به اشتراک گذاشته شدهٔ (p2p) توسط پروتکل بیت‌تورنت
- بلندر (Blender): یک نرم‌افزار ۳ بعدی و اپن سورس بسیار معروف
- چندلر (Chandler): مدیر اطلاعات شخصی شامل تقویم، میل، کارهای روزانه، یادداشت‌ها و...
- Civilization IV:یک گیم کامپیوتری بر مبنای پایتون که از boost.pythonاستفاده می‌کند
- میلمن (Mailman): یکی از معروفترین نرم‌افزارهای مرتبط با ایمیل
- کمبیلو (Kombilo): مدیر پایگاه داده و مرورگرگیم‌های go
- موین‌موین (MoinMoin): یکی از قدرتمندترین و معروفترین ویکی‌های موجود
- پلون (Plone): یک ابزار مدیریتی محتوایی اپن سورس، قدرتمند و کاربر پسند
- پورتاژ (Portage): قلب توزیع جنتو. یک مدیر بستهای سیستم لینوکس
- زوپ (zope): یک پلتفورم شیء گرای مبتنی بر وب. زوپ شامل یک سرور نرم‌افزار به همراه پایگاه داده شیء گرا و یک رابط مدیریتی درونی مبتنی بر وب می‌باشد
- اس‌پی‌ای (SPE): یک IDE رایگان، اپن سورس برای سیستم‌عامل‌های ویندوز، لینوکس، مک که از wxGlade(طراحی رابط کاربر),PyChecker (دکتر کد) Blender(3D) پشتیبانی می‌کند.
- یام (Yum): یک برنامه مدیریت بسته متن‌باز برای توزیع‌های سازگار با آرپی‌ام.
- آباکوس(Abaqus):نرم‌افزار شبیه‌سازی با روش المان محدود که امکان اسکریپت نویسی به زبان پایتون را به کاربر می‌دهد.

## ۱۱ جستارهای وابسته

- داده‌کاوی با پایتون
- پای‌کیوت

## ۱۲ منابع


[1] Hastings, Larry (2015-12-07). "Python 3.5.1 and Python 3.4.4rc1 are now available". *Python Insider*. The Python Core Developers. Retrieved 2015-12-08.

[2] "Python Release Python 2.7.11". Python Software Foundation. Retrieved 16 December 2015.

[3] مستندات پایتون - پایتون برای چه مواردی مناسب است؟

[4] “What is Python? Executive Summary”. *Python documentation*. Python Software Foundation. Retrieved 2007-03-21.

[5] “General Python FAQ”. *python.org*. Python Software Foundation. Retrieved 2009-06-27.

[6] Python Garbage Collection

[7] مستندات پایتون ۲ - زباله‌روب

[8] مستندات پایتون ۲ - زباله‌روب

[9] «Python Programming Language – Official Website». Python Software Foundation.بازبینی‌شده در ۵ فروردین ۱۳۹۰.

[10] «History and License». The Python Software Foundation، Mar 24, 2011.بازبینی‌شده در ۵ فروردین ۱۳۹۰.

[11] مستندات پایتون - پایتون برای چه ایجاد شد؟

[12] مشارکت‌کنندگان ویکی‌پدیا، «Benevolent Dictator For Life»، ویکی‌پدیای انگلیسی، دانشنامهٔ آزاد (بازیابی در ۵ فروردین ۱۳۹۰).

[13] Functional Programming HOWTO

[14] «What’s New In Python 3.0». Python Software Foundation، Feb 14, 2009.بازبینی‌شده در ۵ فروردین ۱۳۹۰.

[15] «Automated Python 2 to 3 code translation». Python Software Foundation، Feb 14, 2009.بازبینی‌شده در ۵ فروردین ۱۳۹۰.

[16] «GUI Programming in Python». The Python Wiki، 2010-10-10.بازبینی‌شده در ۵ فروردین ۱۳۹۰.

[17] «Graphic User Interface FAQ». Python Software Foundation، Mar 13, 2010.بازبینی‌شده در ۵ فروردین ۱۳۹۰.

## ۱۳ پیوند به بیرون

- وب‌گاه رسمی پایتون
- نسخهٔ برخط کتاب شیرجه داخل پایتون (Dive into Python) آموزش پایتون برای برنامه‌نویس‌ها (انگلیسی)

# ۱۴ منابع متن و تصویر، مشارکت‌کنندگان و مجوزها

## ۱.۱۴ متن

- **پایتون (زبان برنامه‌نویسی)**منبع: https://fa.wikipedia.org/wiki/%D9%BE%D8%A7%DB%8C%D8%AA%D9%88%D9%86_(%D8%B2%D8%A8%D8%A7%D9%86_%D8%A8%D8%B1%D9%86%D8%A7%D9%85%D9%87%E2%80%8C%D9%86%D9%88%DB%8C%D8%B3%DB%8C)?oldid=18824183 مشارکت‌کنندگان:ماني، حسام، دانيل، Mohsens، Zeerak، Aliparsa، Robbot، Chobot، Shervinafshar، سروش رادپور، Mehran، Soroush، R0stam، Thijs!bot، علي نادعليزاده، Elessar، Farzad ghanei، Bayazee، JhsBot، محمدرضا، Qoqnous، Raamin، Behaafarid، Meisam Irnavash، Sicaspi، CommonsDelinker، Wayiran، Ebrahim، Pj.attar، TXiKiBoT، VolkovBot، Pykello، Behtis، KiaThr~fawiki، MOSIOR، AlnoktaBOT، Amirrad، SieBot، Ariyayi، Bersam، Synthebot، PixelBot، Strong46202، Tanhabot، MelancholieBot، Amirobot، Smorteza، Jotterbot، ArthurBot، Zahedeh-noori، Saeed.afshari، Reza1615، Xqbot، Rubinbot، SassoBot، Majidvp، ICEAGE، Adlerbot، TobeBot، Dalba، Leyth، RedBot، Arash.pt، Persian knight shiraz، MastiBot، HaDi، Sefid par، ZxxZxxZ، Example123789، EmausBot، Elph، Ebrambot، AliBot، Efazati، ChuispastonBot، WikitanvirBot، Kanaan، Rezabot، MerlIwBot، Hpaalm، MahdiBot، MohammadtheEditor، User-000، علیرضا، Dexbot، Zatron، FawikiPatroller، M.ollivander، Yamaha5، Rayeshman، Oyakmohsen، Wiki rasam، Fatemibot، Editor-1، SerendiPity، Behruz، وحيدنا، Alihajikhaluie، Saeiddrv و ناشناس: 29

## ۲.۱۴ تصاویر

- **پرونده:Ambox_wikify.svg**منبع: https://upload.wikimedia.org/wikipedia/commons/e/e1/Ambox_wikify.svg مجوز: Public domain مشارکت‌کنندگان: اثر شخصی هنرمند اصلی:penubag
- **پرونده:Commons-logo.svg**منبع: https://upload.wikimedia.org/wikipedia/commons/4/4a/Commons-logo.svg مجوز: Public domain مشارکت‌کنندگان: This version created by Pumbaa, using a proper partial circle and SVG geometry features. (Former versions used to be slightly warped.) هنرمند اصلی:SVG version was created by User:Grunt and cleaned up by 3247, based on the earlier PNG version, created by Reidab.
- **پرونده:Folder_Hexagonal_Icon.svg**منبع: https://upload.wikimedia.org/wikipedia/commons/4/48/Folder_Hexagonal_Icon.svg مجوز: CC-BY-SA-3.0 مشارکت‌کنندگان: اثر شخصی بر پایهٔ Folder.gif. هنرمند اصلی:**اصلی:** John Cross **برداری‌سازی:** Shazz
- **پرونده:Guido_van_Rossum_OSCON_2006.jpg**منبع: https://upload.wikimedia.org/wikipedia/commons/6/66/Guido_van_Rossum_OSCON_2006.jpg مجوز: CC BY-SA 2.0 مشارکت‌کنندگان: 2006oscon_203.JPG هنرمند اصلی:Doc Searls
- **پرونده:Opengraph-icon-200x200.png**منبع: https://upload.wikimedia.org/wikipedia/fa/e/e6/Opengraph-icon-200x200.png مجوز: استفاده منصفانه مشارکت‌کنندگان: https://www.python.org/static/opengraph-icon-200x200.pngهنرمند اصلی?:
- **پرونده:Python_add5_syntax.svg**منبع: https://upload.wikimedia.org/wikipedia/commons/e/e1/Python_add5_syntax.svg مجوز: Copyrighted free use مشارکت‌کنندگان: http://en.wikipedia.org/wiki/Image:Python_add5_syntax.png هنرمند اصلی:Xander89
- **پرونده:Question_book.svg**منبع: https://upload.wikimedia.org/wikipedia/commons/9/97/Question_book.svg مجوز: CC-BY-SA-3.0 مشارکت‌کنندگان: ? هنرمند اصلی:?
- **پرونده:Symbol_list_class.svg**منبع: https://upload.wikimedia.org/wikipedia/commons/d/db/Symbol_list_class.svg مجوز: Public domain مشارکت‌کنندگان: Self-made in Inkscape, similar to Image:Symbol support vote.svg. هنرمند اصلی:Mysid
- **پرونده:Symbol_neutral_vote.svg**منبع: https://upload.wikimedia.org/wikipedia/commons/8/89/Symbol_neutral_vote.svg مجوز: Public domain مشارکت‌کنندگان: ? هنرمند اصلی:?

## ۳.۱۴ محتوای مجوز

- Creative Commons Attribution-Share Alike 3.0