Engineer

Python

språket som det mesta av webbplatsens verktyg är skrivet i

Vad det är

Python är ett allmänt programspråk vars källkod körs av en tolk i stället för att kompileras i förväg till en separat binär. Den egna dokumentationen kallar språket tolkat, interaktivt och objektorienterat, med moduler, undantag, dynamisk typning och datatyper på hög nivå inbyggda. Det drag som de flesta lägger märke till först är att indrag är syntax: ett block är det som står indraget under raden som öppnar det, så den layout läsaren ser är den struktur tolken kör. Guido van Rossum började på språket under julhelgerna 1989, med erfarenheter från arbetet på språket ABC vid CWI i Amsterdam, publicerade det i februari 1991 och döpte det efter manusen till Monty Pythons Flying Circus, som han läste vid den tiden.

Typningen är dynamisk. Ett namn binds till ett objekt medan programmet körs, och typen hör till objektet, inte till namnet, så samma variabel kan rymma ett tal på en rad och en sträng på nästa. Frivilliga annoteringar, standardiserade i PEP 484, låter en funktion ange vad den väntar sig och vad den returnerar, men den PEP:en säger uttryckligen att ingen typkontroll sker under körning och att Python förblir ett dynamiskt typat språk. Annoteringarna läses av separata verktyg, som typkontrollprogram och editorer, medan tolken sparar dem utan att upprätthålla dem.

Språket levereras med ett stort standardbibliotek, som handledningen beskriver med orden ”batteries included”: filer, JSON, SQLite, reguljära uttryck, underprocesser, tolkning av argument och mer, allt utan att installera något. Allt annat kommer som programpaket från tredje part, oftast installerade med pip från Python Package Index. Eftersom en tolk bara kan hålla en version av varje programpaket ingår också `venv` i standardbiblioteket. Det skapar en fristående miljö per projekt, så att ett program kan använda version 1.0 av ett bibliotek medan ett annat använder 2.0.

Nya versioner kommer enligt en kalender. Sedan Python 3.9 har PEP 602 gett en funktionsversion varje oktober efter ungefär sjutton månaders utveckling, och var och en underhålls i fem år. Från 3.13 och framåt betyder det två år med vanliga felrättningsversioner, ungefär varannan månad, och sedan tre år med enbart säkerhetsrättningar. Python 3.15.0 kom ut den 9 oktober 2026. Python 3.10 nådde slutet av sin livstid den 1 oktober 2026, och Python 2.7, den sista i den andra serien, slutade få stöd den 1 januari 2020. Referenstolken, CPython, har länge haft ett globalt tolklås, GIL, som bara låter en tråd i taget köra Python-kod. PEP 703 lade till ett bygge utan låset, och PEP 779 pekade ut 3.14 som tidsramen för den fas där bygget utan GIL har officiellt stöd men fortfarande är valfritt.

Använd begreppet i ett beslut

Använd ”Vad är Python?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Den del användarna aldrig ser: där data lagras, vilka reglerna är och hur andra system får kommunicera med den.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Utformade och drev 6 ETL/ELT-pipelines — 6 ETL-pipelines över tre databasmotorer

Utformade och drev 6 ETL/ELT-pipelines Hör av er →

Varför det spelar roll

Python är ofta den kortaste vägen från en frågeställning till ett fungerande program. Ett skript som läser en fil, ställer en databasfråga och skriver en rapport kan skrivas med enbart standardbiblioteket, köras direkt och ändras utan något byggsteg. Därför hamnar språket så ofta som kittet i en datapipeline, i byggverktyg och i testmiljöer.

Läsbarheten är ett medvetet designmål och ingen bieffekt. The Zen of Python, PEP 20, slår fast att ”Readability counts”, och PEP 8, stilguiden för standardbiblioteket, avgör det som ett team annars skulle tvista om, från namngivning till raden på 79 tecken.

Flexibiliteten har ett pris, och det kommer sent. Ett felstavat attribut, ett `None` där en sträng väntades eller fel typ på ett argument blir ett fel först när just den raden körs, och det kan vara i produktion. Team som bygger något långlivat i Python lägger därför till de kontroller språket lämnar därhän: en linter, annoteringar med typkontroll i kvalitetsgrinden och tester som faktiskt kör kodvägarna.

Kalendern väger lika tungt som koden. Varje tolkversion har ett fast slutdatum, så en installation som ingen rör glider ut ur säkerhetsstödet enligt ett publicerat schema. Ett projekt som anger sin lägsta version och flyttar framåt inom de två åren med felrättningar byter en nödsituation mot en planerad uppgradering om året.

Så ser det ut i praktiken

I portfolion drev Python i ett dokumenterat utvecklingsskede en databasdriven generator för CV och portfolio: 41 moduler och 10 580 rader som renderade sex utdataformat, från HTML och PDF till Markdown och JSON, ur en enda SQLite-databas med 23 tabeller. En kedja av 19 steg byggde databasen, och varje steg gick att köra om utan risk. Beroendena vid körning hölls till exakt två, en mallmotor och en PDF-renderare, med argumentet att en dokumentgenerator som inte går att installera om fem år inte har bevarat någonting. I gengäld var varje dokument då beroende av den Python-koden, utan något annat sätt att framställa ett. Dessa siffror beskriver implementationen i det dokumenterade utvecklingsskedet.

Python vann också en direkt jämförelse i en datapipeline som summerar elförbrukning timme för timme. Den byggdes på tre sätt, i Python, i JQ och i SQL på PostgreSQL, för att göra om oregelbundna avläsningar till en rad per timme och zon, med nettoförbrukningen som produktion plus import minus export. Varianten som skrevs som rena Python-funktioner var snabbast och hanterade ett dygns data på 34 ms och ett dataset på 30 dagar på 180 ms.

Den här webbplatsen byggs av Hugo, en statisk webbplatsgenerator. I den ögonblicksbild av verktygen som dokumenterades i oktober 2026 var dess 22 hjälpskript, 8 408 rader sammanlagt, skrivna i Python. Samtliga klarade sig med standardbiblioteket utom skriptet som importerade teckningar, som behövde Pillow för att beskära dem. Filen `.tool-versions` angav Python 3.14.7 med regeln att tolken fick vara nyare men aldrig äldre, och `task doctor` prövade det mot den installerade binären i stället för att lita på filen. Ruff var lintern, och basedpyright granskade typer på nivån `standard`, eftersom den förvalda nivån rapporterade 358 fynd som nästan alla gällde saknade annoteringar i kod som var enkel med avsikt. Vid första körningen hittade typkontrollen ett värde som kunde vara `None` och som användes som nyckel i en dict, en defekt som Ruff hade släppt igenom utan anmärkning.

Var det går fel

Det mesta besväret med Python kommer från kod som tolken godtar utan protester och som beter sig annorlunda än den läses.

Ett standardargument minns tidigare anrop. En funktion returnerar en lista som fortfarande innehåller element från tidigare anrop, så resultatet beror på i vilken ordning anropen sker. Ett standardvärde beräknas en gång, när `def`-satsen körs, så ett föränderligt standardvärde som en lista eller en dict är ett och samma objekt för varje anrop som utelämnar argumentet. Använd `None` som standardvärde och skapa listan inne i funktionen. Ruffs regel B006 rapporterar mönstret.

```
def add_tag(tag, tags=None):
    if tags is None:
        tags = []  # en ny lista vid varje anrop
    tags.append(tag)
    return tags
```

Projekt förstör varandras beroenden. En uppgradering av ett bibliotek för ett projekt får ett annat att gå sönder, eller så stannar `pip install` med ett fel om en externt hanterad miljö. Båda beror på programpaket som installerats direkt i en delad tolk: projekten konkurrerar om en enda kopia av varje bibliotek, och enligt PEP 668 kan den som distribuerar ett operativsystem märka sin egen Python som förbjuden för pip. Ge varje projekt en egen miljö med `python -m venv .venv`, installera i den och skriv upp beroendena i en fil så att miljön går att bygga upp på nytt.

Typannoteringar behandlas som om de vore kontroller. En funktion som är annoterad för att ta emot en `int` får en sträng från en JSON-fil, och felet visar sig tre anrop senare. Annoteringar beskriver avsikt, och tolken upprätthåller dem inte. Kör typkontroll som en del av kvalitetsgrinden så att avvikelser mellan annoterad kod stoppar bygget, och validera data där de kommer in i programmet, eftersom en annotering aldrig kan gå i god för indata.

En tolk åldras ur sitt stöd. En server behåller den version den installerades med tills säkerhetsrättningarna för den upphör och programpaketen stryker den ur sina stödda versioner. Varje funktionsversion får fem år, och de fem åren för 3.10

tog slut den 1 oktober 2026. Ange den lägsta versionen i en fil som en kontroll läser, flytta upp inom de två åren med felrättningar för varje version och kör testsviten på den nya versionen innan du byter.

Läs mer

- Python documentation — handledningen, biblioteksreferensen och språkreferensen för den aktuella versionen.
- The Python Tutorial — språket från grunden, inklusive varningen för föränderliga standardargument.
- PEP 8 – Style Guide for Python Code — de konventioner för layout och namngivning som det mesta av all Python-kod följer.
- PEP 602 – Annual Release Cycle for Python — varför en funktionsversion kommer varje oktober och hur länge var och en får stöd.
- PEP 484 – Type Hints — syntaxen för annoteringar och varför tolken överlåter granskningen åt andra verktyg.
- Virtual Environments and Packages — hur man skapar en miljö per projekt med `venv` och installerar i den med pip.

Ordlista

Välj utifrån era faktiska förutsättningar

Beskriv arbetsflödet, befintliga gränssnitt och användarna som behöver förändringen. Vi kan definiera en testbar leverans och en överlämning som gör att teamet kan underhålla lösningen.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är Python?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

welcome@engineer.company +45 50 10 61 84 Köpenhamn, Danmark

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/python/