Python 3.11

Kicsi és gyors lépésekkel a paradigma váltás és a gyorsaság felé

Dr. Guta Gábor

∀xonmatics

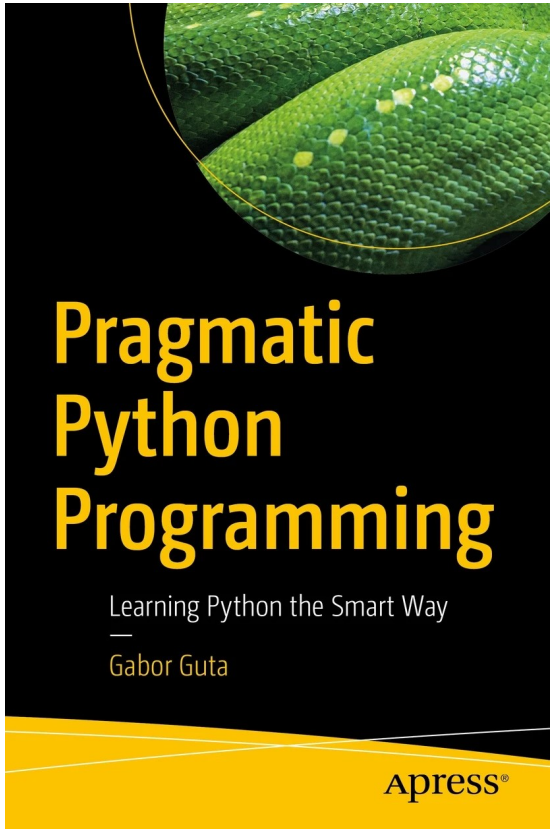
SCIENCE MEETS SOFTWARE DEVELOPMENT

Rólam

- kb. 15 éve fejlesztek Pythonban

- a Pragmatic Python Programming (Apress) és a Szoftverfejlestés okosan Pythonnal (Pánem) szerzője vagyok

- foglalkoztam programozási nyelvek specifikálással, programok helyesség bizonyításával, és fordítókkal



∀xonmatics
SCIENCE MEETS SOFTWARE DEVELOPMENT

Tematika

- Miért érdekes ez?
- Milyen gyakran kerül kiadásra új változat?
- Mi van általában egy új Python kiadásban?
- Hogyan fejlődik a Python?
- Főbb trendek

Miért érdekes ez?

- Kell-e „újra” tanulni a Pythont?
  → Vannak teljesen új részei:
-- Új szintaxis elemek
-- Típus jelölések
-- Async

- Kit érdekel a sebesség?
  →Sok projekt foglalkozik vele (mypyc, cinder, numba, etc.)
  „Nagyban” számít, hogy mekkora kapacitást kell mögé tenni

Milyen gyakran érkeznek az új változatok?

A PEP602-ben meghatározott módon évente jönnek ki új változatok

- 1,5 évig van teljes támogatás és 5 évig biztonsági támogatás
- Személyes tapasztalat az, hogy kb. fél-egy év még mindenhol támogatva lesz az új kiadás (pl. Anaconda, AWS lambda, stb.)

Verzió	Kiadási dátum
3.0	2008/12
3.1	2009/06
2.7	2010/07
3.2	2011/02
...	...
3.7	2018/06
3.8	2019/10
3.9	2020/10
3.10	2021/10
3.11	2022/10

∀xonmatics
SCIENCE MEETS SOFTWARE DEVELOPMENT

Miből áll egy új kiadás?

- Néhány nagy újdonság:
  - Új szintaxis (async, match-case, expect*)
  - Új csomagok (TOML)
  - Fontos belső változások (Faster Python projekt)

Rengeteg apró javítás/módosítás

Miből áll egy új kiadás?

General changes

- PEP 657 -- Include Fine-Grained Error Locations in Tracebacks
- PEP 654 -- Exception Groups and `except*`
- PEP 680 -- tomllib: Support for Parsing TOML in the Standard Library
- gh-90908 -- Introduce task groups to asyncio
- gh-34627 -- Atomic grouping (`(?>...)`) and possessive quantifiers (`*+, ++, ?+, {m,n}+`
- The Faster CPython Project is already yielding some exciting results. Python 3.11 is up to 10-60% benchmark suite. See Faster CPython for details.

Typing and typing language changes

- PEP 673 -- Self Type
- PEP 646 -- Variadic Generics
- PEP 675 -- Arbitrary Literal String Type
- PEP 655 -- Marking individual TypedDict items as required or potentially-missing
- PEP 681 -- Data Class Transforms

More resources

- Online Documentation
- PEP 664, 3.11 Release Schedule
- Report bugs at https://github.com/python/cpython/issues.
- Help fund Python and its community.

3.11 Kiadási dokumentáció

Forrás: https://docs.python.org/3/whatsnew/3.11.html

PEP492 – példa új szintaxisra

```
import asyncio

async def main():
    print('Hello World!')

asyncio.run(main())
```

PEP634 – példa új szintaxisra

```
egy_szám = int(input("Kérek egy számot 1-10 között"))
match egy_szám:
    case 1:
        print('A legkisebbet választottad!')
    case 2:
        print('Az elsö páros szám')
    case 4|6|8|10:
        print('Ez egy páros szám')
    case _:
        print('Ez szerinted 1-10 között van???')
```

“Faster CPython” project

1. Adaptív, specializáló értelmező,
   a „type stability“ kihasználásával

2. Sok kicsi optimalizáció

3. Egyszerű "JIT" fordító: „kis“ régiók fordítása

4. Fejlett „JIT“ fordító: „Kiterjesztett“ régiók fordítására

Forrás: https://github.com/markshannon/faster-cpython/blob/master/plan.md

A Python fejlődése

- 5-8 év alatt megjelenik vagy teljesen megváltozik egy-egy nyelvi elem:
  - Aszinkron nyelvi elemek változása: generator-based coroutine vs. native coroutine
  - Típus jelölések
- Rendszeres apró változások
  - Rozmár operátor a 3.8-ban (:=)
- Fontos belső változások
  - Új PEG (Parsing Expression Grammar) parser a 3.9-ben

Verzió	Aszinkron nyelvi elemek változása
3.4	• PEP 3156, a new "asyncio" module, a new framework for asynchronous I/O
3.5	• PEP 492, coroutines with async and await syntax
3.6	• PEP 525, Asynchronous Generators (provisional) • PEP 530, Asynchronous Comprehensions
3.7	• Backwards incompatible syntax changes: async and await are now reserved keywords. • The asyncio module has received new features, significant usability and performance improvements.
3.8	• asyncio.run() has graduated from the provisional to stable API. • Running python -m asyncio launches a natively async REPL. • Several other changes in asyncio package
3.9	• Parallel running of aclose() / asend() / athrow() is now prohibited, and ag_running now reflects the actual running status of the async generator. (Contributed by Yury Selivanov in bpo-30773.) • Several other changes in asyncio package
3.10	• Two new builtin functions – aiter() and anext() have been added to provide asynchronous counterparts to iter() and next(), respectively. (Contributed by Joshua Bronson, Daniel Pope, and Justin Wang in bpo-31861.)
3.11	• **Introduce task groups to asyncio and change task cancellation semantics gh-34627**

Verzió	Típusos nyelvi elemek változása
3.0	PEP 3107: Function Annotation
3.5	PEP 484: Type Hints
3.6	PEP 526: Syntax for Variable Annotations
3.7	PEP 561: Distributing and Packaging Type Information
3.8	PEP 544: Protocols: Structural subtyping (static duck typing) PEP 563: Postponed Evaluation of Annotations PEP 586: Literal Types PEP 589: TypedDict: Type Hints for Dictionaries with a Fixed Set of Keys PEP 591: Adding a final qualifier to typing
3.9	PEP 585 Type Hinting Generics In Standard Collections PEP 593 Flexible function and variable annotations
3.10	PEP 604: Allow writing union types as X | Y PEP 612: Parameter Specification Variables PEP 613: Explicit Type Aliases PEP 647: User-Defined Type Guards
3.11	**PEP 646: Variadic Generics** **PEP 655: Marking individual TypedDict items as required or potentially-missing** **PEP 673: Self Type** **PEP 675: Arbitrary Literal String Type** **PEP 681: Data Class Transforms**

Főbb trendek

Alkalmazási területek:

- Tovább erősödik a ML/data science vonal
- Back-end használat tovább erősödik
- Sokaknak első nyelv
- Nem „front-end nyelv” továbbra sem, bár a kísérletezés tovább folyik: Pyodide, Anvil’s client side Python :)
- Tesztelés, System administration

Mi lesz 5 év múlva? Szerintem...