



# Modern Python From Alternate Universes



Reece Jones

Python is very cool

...but so are other languages.

How can other languages inspire  
expressive and safe Python?

# Overview

---

- Type hints
  - New minor features
  - Type parameters
  - Parameter specifications
  - Protocols
  - Deduced self type
- Dataclasses
- Return codes & pattern matching

# Type Hints

Annotate code with type information for static analysis



## Key uses

- Everywhere!

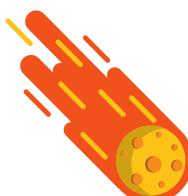


*Static typing*  
C, Java, Haskell, etc.



## Highlights

- Detect bugs earlier
- Confidence in code safety
- Enhanced readability



## Lowlights

- Can't static type everything
- Tooling isn't perfect
- More dev effort

# Type Hints - New Minor Features

---

List[str] → list[str]



Union[int, str] → int | str



# Type Parameters

I



*Generics*

Java, C++, Haskell



Statically parameterise  
functions and classes  
by types



Key uses

- Generic data structures
- Generic algorithms

# Parameter Specifications

Static analysis for manipulating arbitrary function signatures



*Variadic generics\**  
C++



## Key uses

- Function wrappers/decorators



## Lowlights

- Limited parameter manipulation

\*sort of

# Protocols



Statically analysable  
ad-hoc interface  
implementation



## Key uses

- Widely-used interfaces



*Structural subtyping*  
Go, OCaml, C++



## Highlights

- Simple interfaces
- Easily extendable interfaces



## Lowlights

- Less explicit

# Deduced Self Type



*Self type*

TypeScript, Rust, Scala



Statically capture derived  
type in base class

✨ Key uses

- Method chaining / fluent interfaces
- Factory classmethods

# Dataclasses



## Record types

Elixir, C++, C#, Kotlin



Automatically generated  
record types

## Key uses

- Functional programming
- Procedural programming



## Highlights

- Simple & concise
- Immutability
- Flexible & customisable
- Statically analysable



## Lowlights

- Not 100% statically analysable  
in practice

# Return Codes

Return values and pattern matching for indicating errors

## Key uses

- ?

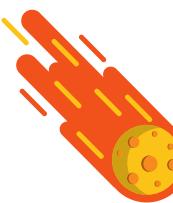


*Return values*  
Erlang, Elixir



## Highlights

- Explicit error modes
- Explicit control flow
- Clean error handling
- Statically analysable



## Lowlights

- Weak pattern matching
- No Elixir "with"
- Tooling isn't great

} currently



