
**Python Type Definitions and Mypy**

**Article # 22** – Recently a friend introduced me to Mypy – an optional static type checker for Python that combines the benefits of static and dynamic typing. That is when I got introduced to explicit type checking in Python. I felt it very interesting and hence decided to learn and implement this in my code. While learning, I came across an interesting historical timeline that led towards the reality of type checking in Python through Python Type Definitions and Mypy. In this article, I am sharing a basic understanding of Type Safety in programming languages, historical timelines, and links to informative articles and videos that can help in learning and practicing the subject.

**What is Type-Safe code? (A quick revision)**

The term 'Type Safety' evolves as follows:


**Data Type**: In computer science, a *Data Type* is an attribute of data that tells the compiler or interpreter how the program intends to use the data. Examples: Integer, Floating point, Character, Boolean, and user-defined data types.

Detailed reading on Wiki: https://en.wikipedia.org/wiki/Data_type

**Type Systems**: While writing computer programs, we use different constructs where the data type of those constructs is an important attribute towards the correctness of the code, e.g. variables, parameters, return variables, etc. A *Type System* is a logical system that comprises a set of rules that governs the assignment of the data type to these constructs.

Detailed reading on Wiki: https://en.wikipedia.org/wiki/Type_system

**Type Checking**: *Type Checking* is the process of verifying the constraints associated with the data types when the constructs are used in processing. Type checking would ensure that the constructs adhere to the rules defined by type systems for an underlying data type.

Detailed reading on Wiki: https://en.wikipedia.org/wiki/Type_system#Type_checking

**Type Errors:** *Type Errors* are those errors that pop up in a computer program that does not adhere to the underlying type system of the compiler or interpreter.

Detailed reading on Wiki: https://en.wikipedia.org/wiki/Type_system#Type_errors

**Type Safety:** In a software programming language, *Type Safety* is the extent up to which a programming language discourages or prevents Type Errors.

Detailed reading on Wiki: https://en.wikipedia.org/wiki/Type_safety

**Statically typed languages:** *Static Type Checking* is the process of verifying the type safety of a program based on analysis of a program's text (source code). If a program passes a static type checker, then the program is guaranteed to satisfy some set of type safety properties for all possible inputs.

Detailed reading on Wiki: https://en.wikipedia.org/wiki/Type_system#Type_checking

**Dynamically typed languages:** *Dynamic Type Checking* is the process of verifying the type safety of a program at runtime.

Detailed reading on Wiki:
https://en.wikipedia.org/wiki/Type_system#Dynamic_type_checking_and_runtime_type_information

**Why Type Safety is important?**

Every software developer feels comfortable if his code is confirmed as 'type safe' before it is deployed for testing. This ensures that the code would not break due to incompatible data types. Also, it is more comfortable when someone else works on our code with a complete understanding of the valid data types of variables, function parameters, and return types.

**A type-safe code:**

- Helps to eliminate errors that can crop up due to incompatible data types and this can happen much ahead in the development cycle rather than stumbling upon in the production environment. Cost of fixing an error increase as you move further in the development cycle from development to production execution
- Takes advantage of the compiler's power to give you one round of check on the compatibility of your code blocks
- Helps in giving some form of readability to your code and makes it easy for others to understand your code
- Helps in giving some form of readability and understandability of the business logic that your code implements

**Dropbox example**

As a testimony towards the benefits of writing type-safe code, there can't be an example bigger than the Dropbox team which invested efforts in type checking 4 million lines of code

Their experience is beautifully documented at: "*DropBox story of typechecking 4 million lines of code*" :
https://dropbox.tech/application/our-journey-to-type-checking-4-million-lines-of-python

**History and timelines of Python Type Definitions and Mypy**

Python is a dynamically type-checked language and it intends to remain like that due to the benefits that it offers. But the huge popularity of Python gradually made it important to have type definitions and static type checking in some form.

There are two groups that worked in parallel to make this a reality:

**Python Type Definitions:**

Python core group worked on this and came up with enhancements in form of PEP 483, PEP 484, and PEP 526.

**Mypy:**

A group of enthusiasts under the vision and leadership of Jukka Lehtosalo created Mypy - an optional static type checker for Python that combines the benefits of static and dynamic typing.

The efforts of the two groups gave a big thrust in changing Python from “Dynamically typed programming language” to “Dynamically typed programming language with optional static type checking facility”.

I found it very interesting to know how the two visionary groups worked closely in achieving the milestones. Based on the information available from the Internet, I created the following table that takes us through the timelines of both the groups

| ⇩ | Python | Mypy |
|---|---|---|
| 2003 | • PEP 318 -- Decorators for Functions and Methods | |
| 2006 | • PEP 3107 -- Function Annotations | |
| … | … | … |
| 2011 | | • Jukka Lehtosalo published Phd thesis - “*Language with Pluggable Type System and Optional Runtime Monitoring of Type Errors*” |
| 2012 | | • Jukka Lehtosalo gave a talk on Mypy at PyCon Finland<br>• Mypy Source Code released |
| 2013 | | • Mypy at PyCon US<br>• Jukka Lehtosalo collected Inputs from Guido and Python community<br>• Mypy switches to Python compatible syntax |
| 2014 | • PEP 483 – Theory of Type Hints<br>• PEP 484 – Type Hints | |
| 2015 | • Python 3.5 – PEP 484 support and a Typing module | • Mypy 0.2 released with PEP 484 compatibility |
| 2016 | • PEP 526 – Inline variable annotations<br>• Python 3.7 with PEP 526 support | • Release of various enhanced versions (2015 to 2021) |
| … | … | … |
| 2021 | | • Mypy 0.910 released |
| ∞ | • Keep on amazing the IT Industry | • Keeps on amazing the software engineers |

There are some very good sources on the Internet that help to understand and practice Type Definitions in Python and Static Type checking with Mypy through easy-to-understand examples and explanations. Sharing below what I collated as 'great sources' on this topic.

**Blogs/Articles:**

| |
|---|
| **MyPy documentation:** https://mypy.readthedocs.io/en/stable/introduction.html |
| **RealPython tutorial:** https://realpython.com/lessons/type-checking-mypy/ |
| **MyPy brief introduction :** https://gcallah.github.io/DevOps/coding/mypy_brief_introduction.html |
| **MyPy News - with MyPy timelines:** http://mypy-lang.org/news.html |

**Vlogs:**

| |
|---|
| **Static Typing in Python - PyGotham 2019:** https://www.youtube.com/watch?v=2gBP1qN5T7I |
| **How to use python type hinting - Kie Codes:** https://www.youtube.com/watch?v=yScuF1UgGU0 |
| **Mypy Getting to Four Million Lines of Typed Python - Michael Sullivan:** **https://www.youtube.com/watch?v=FT_WHV4-QcU** |
| **Why you should use Type Hints in Python - Are type hints worth it? - Kie Codes:** https://www.youtube.com/watch?v=5y7pQaP-5Qw |
| **Carl Meyer - Type-checked Python in the real world : PyCon 2018:** https://www.youtube.com/watch?v=pMgmKJyWKn8 |

A PDF Version of this article can also be downloaded from my GitHub: Link

# About Me:

My Name is **Hiral Amodia.** I am based in Bangalore, India. I am a Software Engineer working in Indian IT Industry for over 16 years now. Currently, I am employed as a Software Engineering Manager at a leading Indian IT services company. I am passionate about learning new technologies and concepts. I strongly believe that teaching is the best way of learning and that caring is the true way of sharing. With this philosophy in mind, I keep on writing articles on technology, concepts, etc. that I learn.

Feel free to buzz me on my below coordinates if you want to share any feedback or improvement areas that you encounter in my articles.

**My Coordinates are as below:**

**Email: amodia.hiral@gmail.com**

**LinkedIn: https://www.linkedin.com/in/hiral-amodia/**

**GitHub**: **https://github.com/amodiahs**