
Python 3.7
DATA CLASSES

A bit of history

```python
# PEP 484 - Type hints
# Python3.5
def greeting(name: str) -> str:
    return f"Hello {name}"
```

```python
# PEP 526 - Syntax for variable annotations
# Python3.6

class Vehicle:
    make: str = 'Maruti'
    year_of_manufacture: int = 2018

    def __init__(self, registration_number: str ):
        self.registration_number = registration_number

    def __repr__(self):
        return f"Vehicle({self.registration_number})"
```

Motivation

- Standard Library
  - collections.namedtuple
  - typing.NamedTuple
- The popular attrs project.
- George Sakkis' recordType recipe, a mutable data type inspired by collections.namedtuple.

Dataclasses

- One main design goal of Data Classes is to **support static type checkers**
- **Standard library** called Data Classes, can be thought of as "mutable namedtuples with defaults".
- A **class decorator** is provided which inspects a class definition for variables with type annotations as defined in PEP 526 (fields)
- Data Classes use normal class definition syntax, you are free to use **inheritance,metaclasses, docstrings, user-defined methods, class factories, and other Python class features**.
- Attribute type annotation is completely ignored by Data Classes, No base classes or metaclasses are used by Data Classes.

Dataclasses

```python
# PEP 557 - Dataclasses
# Python3.7

from dataclasses import dataclass

@dataclass
class Vehicle:
    registration_number: str
    make: str = 'Maruti'
    year_of_manufacture: int = 2018
```

- Can be thought of as "mutable namedtuples with defaults".
- Part of standard Library

dataclass parameters

- “dataclass” function is typically used as a class decorator is provided to post-process classes and add generated methods (“dunder”)
- The dataclass decorator examines the class to find “fields.” A field is defined as any variable identified in __annotations__, is guaranteed to be an ordered mapping
- None of the Data Class machinery examines the type specified in the annotation.
- The dataclass decorator is typically used with no parameters
- def dataclass(*, init=True, repr=True, eq=True, order=False, unsafe_hash=False, frozen=False)

Field objects

- Field objects describe each defined field
- These objects are created internally, and are returned by the fields()
- Users should never instantiate a Field object directly
- Its documented attributes are
  - name: The name of the field.
  - type: The type of the field.
  - default, default_factory, init, repr, hash, compare, and metadata

Customization

- Post-init processing
- Class variables
- Init-only variables
- Frozen instances
- Default factory functions
- Mutable default values
- Exceptions

Behaviour

- Inheritance
- More than one constructor
- Abstract classes