

Python typing

Hong-Phuc Bui

HTW Saar

18. April 2024

Primitiven Typen

Typen Überprüfen

Zusammengesetzten Typen

Union

TypeAlias

Referenz

Einfache Nutzung

```
# filename: developer.py
```

```
# integer
```

```
staff: int = 123_456
```

```
# float
```

```
developer_quote: float = 0.7
```

```
# string
```

```
team: str = "DevOp"
```

```
developer: int = developer_quote * staff
```

```
# Bug!
```

```
print(f"Es gibt {developer} Entwickler in Team „{team}“!")
```

```
# → Es gib 86419.2 Entwickler in Team „DevOp“!
```

Wir wollen keine ein-fünftel Entwickler!

Eine Variante ist explizite Konvertierung.

Bugfix:

```
developer: int = int(developer_quote * staff)
```

Bugfixed

```
print(f"Es gibt etwa {developer} Entwickler in Team „{team}“!")
```

Konstante

```
# filename: light-speed.py
```

```
from typing import Final
```

```
LIGHT_SPEED: Final[int] = 299_792_458 # Exact value [m/s]
```

```
LIGHT_SPEED = 299_792.458 # [km/s]
```

Keine Fehlermeldung!

Run Type-check

```
$ python3 -m pip install mypy
$ mypy light-speed.py
light-speed.py:5: error: Cannot assign to final name "LIGHT_SPEED"  [misc]
light-speed.py:5: error: Incompatible types in assignment
      (expression has type "float", variable has type "int")  [assignment]
Found 2 errors in 1 file (checked 1 source file)
```

Zusammengesetzten Typen

list of prof

```
prof: list[str] = ["D. Knuth", "R. Sedgwick", "E. Gamma"]
```

grade is a tuple of student and his note

```
grade: tuple[str, float] = ("Tommy", 12.5)
```

set of all possible notes

```
notes: set[str] = {"A", "B", "C", "D", "E"}
```

dictionary

```
german_articles: dict[str, str] = {  
    "der": "the", "des": "the", "dem": "the", "den": "the",  
    "die": "the",  
    "das": "the",  
}
```

Union

```
from typing import Sequence
```

```
measurement: Sequence[int | float] = [12.5, 11, 12.3, 11.6, 11.1, 12]
```

Wozu ist das gut?

Funktionsparameter

```
def avg(values: Sequence[int | float]) -> float:  
    return sum(values) / len(values)
```

```
from typing import Sequence, Callable
```

```
avg: Callable[[Sequence[int | float]], float] = \  
    lambda values: sum(values) / len(values)
```

TypeAlias

```
from typing import Sequence, Callable, TypeAlias
```

```
Scale: TypeAlias = Sequence[int | float]
```

```
avg: Callable[[Scale], float] = \
    lambda values: sum(values) / len(values)
```

```
# noch besser:
```

```
avg: Callable[[Scale], float] = \
    lambda v: sum(v) / len(v) if len(v) > 0 else 0.0
```

Referenz

1. <https://mypy.readthedocs.io/>
2. https://mypy.readthedocs.io/en/stable/cheat_sheet_py3.html

Weiter lesen: Was ist der Unterschied zwischen Iterable und Sequence?