Python Type Hints

Thaynã Moretti & Daniel Bastos

MAGRATHEA LABS

PEPs

- PEP 3107 - Function Annotations
  - https://www.python.org/dev/peps/pep-3107
- PEP 484 - Type Hints
  - https://www.python.org/dev/peps/pep-0484
- PEP 526 - Syntax for Variable Annotations
  - https://www.python.org/dev/peps/pep-0526

PEP 3107 - Function Annotations

- Implementado no **Python 3.0**;
- Sintaxe para adicionar anotações arbitrárias de metadados às funções;
- A versão 2.x do Python não possuía uma maneira padrão de anotar os parâmetros de uma função e retornar valores.
- Cenários: parâmetros, valores de retorno.
- Não é possível: funções lambdas

PEP 3107 - Function Annotations

```
def compile(source: "something compilable",
            filename: "where the compilable thing comes from",
            mode: "is this a single statement or a suite?") -> None:
```

```
>>> compile.__annotations__
{'source': 'something compilable', 'filename': 'where the compilable
thing comes from', 'mode': 'is this a single statement or a suite?',
'return': None}
```

PEP 484 - Type Hints

- Implementado no **Python 3.5**;
- Não exclui outros usos de anotações (PEP 3107), nem exige qualquer outras anotações. **Simplesmente permite uma melhor coordenação**;
- Fornece uma sintaxe padrão para anotações de tipo.

PEP 484 - Type Hints

```
def greeting(name: str) -> str:
    return 'Hello ' + name

>>> greeting.__annotations__
{'name': <class 'str'>, 'return': <class 'str'>}
```

Type hints aceitáveis

- Classes internas “primitivas” (biblioteca padrão)
- Tipos disponíveis no módulo types (“typing”)
- Classes de tipos definidas pelo usuário

“None”

```
None == type(None)
```

Type aliases

```
Url = str
def retry(url: Url, retry_count: int) -> None: ...
```

Callable

```
# Callable[[Arg1Type, Arg2Type], ReturnType]
from typing import Callable
def feeder(get_next_item: Callable[[], str]) -> None:
    # Body
```

Generics

```
from typing import Mapping, Set

def notify_by_email(employees: Set[Employee], overrides:
Mapping[str, str]) -> None:
```

PEP 484 - Type Hints

```
# stars is a list of integers
stars = []  # type: List[int]

# 'captain' is a string (Note: initial value is a problem)
captain = ...  # type: str
planet = ‘magrathea’ # type: str

class Starship:
    # 'stats' is a class variable
    stats = {}  # type: Dict[str, int]
```

PEP 526 - Syntax for Variable Annotations

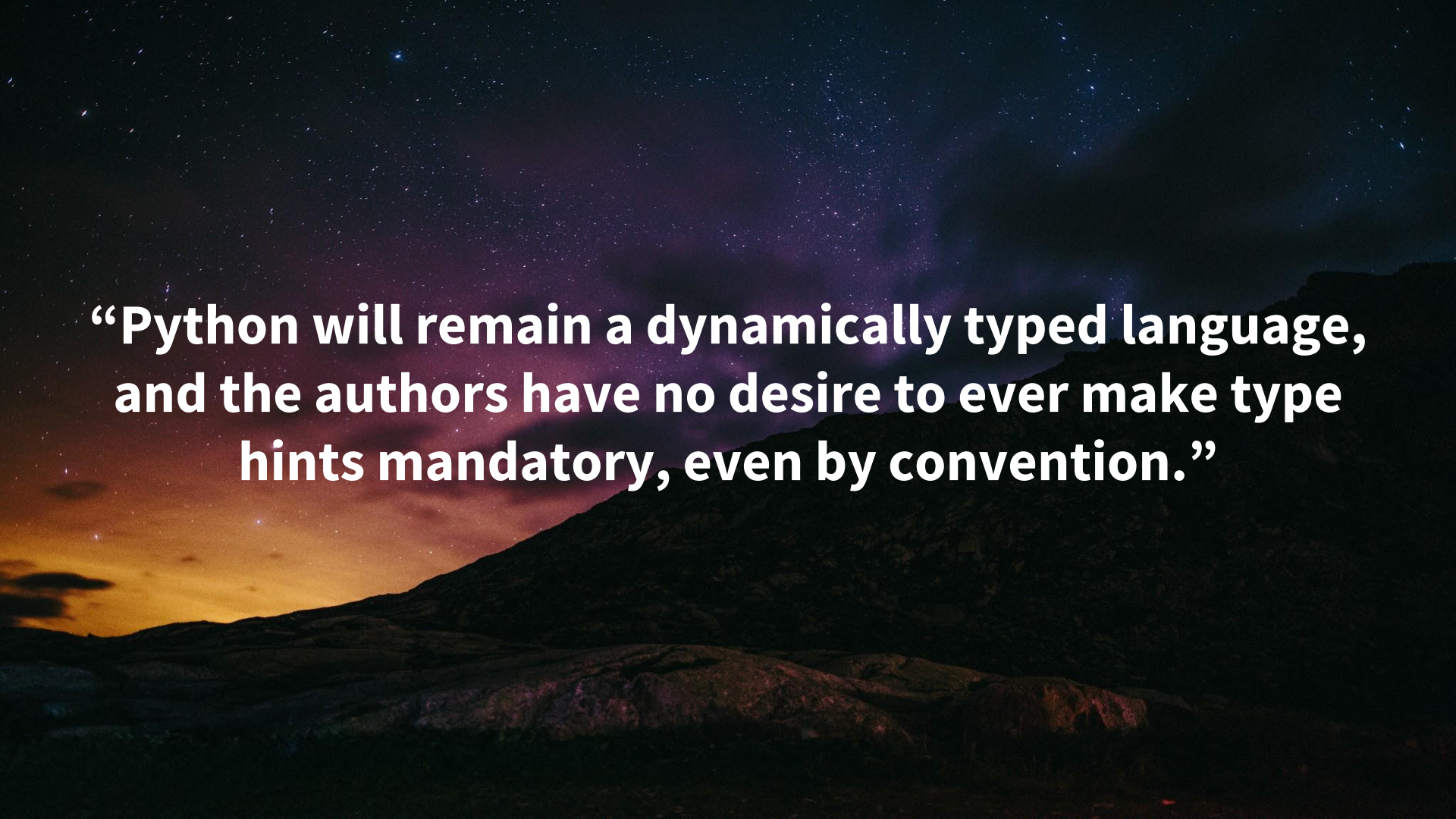
- Implementado no **Python 3.6**;
- Visa adicionar sintaxe para anotar os tipos de variáveis (incluindo variáveis de classe e variáveis de instância), em vez de expressá-las através de comentários;

PEP 526 - Syntax for Variable Annotations

```
from typing import ClassVar, Dict

stars: List[int] = []
captain: str
planet: str = ‘magrathea’

class Starship:
    stats: ClassVar[Dict[str, int]] = {}
```


“Python will remain a dynamically typed language, and the authors have no desire to ever make type hints mandatory, even by convention.”

mypy

Linter **experimental** de tipo estático **opcional** que visa combinar os benefícios de digitação dinâmica e estática. O Mypy combina o poder expressivo e a conveniência do Python com um sistema de tipo e verificação em tempo de compilação.

```
$ pip install mypy

$ mypy program.py
```

mypy

```python
class BankAccount:
    def __init__(self, initial_balance: int = 0) -> None:
        self.balance = initial_balance
    def deposit(self, amount: int) -> None:
        self.balance += amount
    def withdraw(self, amount: int) -> None:
        self.balance -= amount
    def overdrawn(self) -> bool:
        return self.balance < 0

my_account = BankAccount(15)
my_account.withdraw(5)
print(my_account.balance)
```

```
$ python program.py
10

$ mypy program.py
```

mypy + pytests

```
$ pip install pytest-mypy

$ pytest --mypy
```

>> https://github.com/dbader/pytest-mypy

mypy + IDE

Vi/Vim

- Syntastic -> https://github.com/vim-syntastic/syntastic
- Neomake -> https://github.com/neomake/neomake
- Ale -> https://github.com/w0rp/ale

Sublime Text

- SublimeLinter-contrib-mypy -> https://github.com/fredcallaway/SublimeLinter-contrib-mypy

Atom

- linter-mypy -> https://github.com/elarivie/linter-mypy
- atom-mypy -> https://github.com/viktor25/atom-mypy

Live coding!

Referências

- Python e Tipagem Estática - Carlos Henrique Coêlho -> https://youtu.be/oeacyCi3u_o
- How to Use Static Type Checking in Python 3.6 -> https://medium.com/@ageitgey/learn-how-to-use-static-type-checking-in-python-3-6-in-10-minutes-12c86d72677b
- Gradual Typing in Python -> http://2014.es.pycon.org/static/talks/gradual-typing-in-python/index.html#16
- mypy-lang -> mypy-lang.org