mypy – Python’s Gradual Typing Implementation

By Itzhak Kasovitch
Nice to meet you!

I am Itzhak Kasovitch
I am a Software Developer with Dropbox
I cook code (and food) for a living
And I am here to talk with you about mypy
Agenda

- Motivation
- Gradual Typing and Type Hints
- Basic mypy usage
- Special use cases
Hey Python, how would you describe yourself?
Dynamic typing vs Static typing

**Duck typing advantages**
- Less verbose 📝
- Faster development ⏩
- Runtime evaluation 🧑‍🚀

**Static typing advantages**
- Easier to understand and maintain 🏝
- Find bugs earlier 🐜
- Dev tools friendlier ⚒
- Better performance 🚀
Can you guess the arguments types?

def fv(rate, nper, pmt, pv, when='end'): pass
What about this one?

def magic(major, minor):
    pass
What can we do? #1 attempt
What can we do? #2 attempt

PYTEST

WHY WOULDN'T YOU?
What can we do? #3 attempt
Dude - what is Gradual Typing?
What is Gradual Typing?

The most important stuff:
- A special type `Any` that is:
  - Down cast from `Any` to all other types
  - Up cast from all other types to `Any`
Type hints

- Formulated by PEP-484 and PEP-3107
- Those PEPs lay the ground for using Gradual Typing in Python
- mypy is an implementation of these PEPs
Docstrings + pytest + type hints == 💪

KEEP CALM AND BITE THE BULLET
Installing mypy

- mypy requires python 3.3 and later
- To install mypy simply
  $ pip3 install mypy
- And you are good to go
Using Type Hints in python 3

File greeting_py3.py

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

greeting(5)
```
Using Type Hints in Python 2

File greeting_py2.py

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

greeting(5)
```
Invoking mypy

```bash
mypy [--py2] file_name
```

Both files produce the same error:

```text
error: Argument 1 to "greeting" has incompatible type "int"; expected "str"
```
Things to note

- Static types have no effect when the program runs
- You should import the types you use in your Type hints
- Docstrings should appear after the type comment
This is what Static types
def send_email(address, sender, cc, bcc, subject='', body=None):
    # type: (Dict[str, str], str, str, str, str, List[str]) -> bool
    
    """Send an email message. Return True if successful."""
    pass
Built-in types – typing module

int, float, bool, str, object

List[T], Dict[K,V], Iterable[I], Sequence[I], Mapping[K, V]

Any
Type aliases

```
Url = str
def retry(url: Url, retry_count: int) -> None: ...
```
from typing import TypeVar, Iterable, Tuple

T = TypeVar('T', int, float, complex)
Vector = Iterable[Tuple[T, T]]

def inproduct(v):
    # type: (Vector[T]) -> T
    return sum(x*y for x, y in v)

inproduct([[1,1], [2,2]])  # T = int
# type: ignore

```python
import frobnicate  # Error: No module "frobnicate"
frobnicate.start()
```

```python
import frobnicate  # type: ignore
frobnicate.start()  # Okay!
```
Covfefe Typing
Type Inference

```python
from typing import List

i = 5  # inferred type is int
...
i = '123'  # error: Incompatible types in assignment

def f(l, k):
  # type: (List[object], List[int]) -> None
  l = k  # incompatible types in assignment
```
Explicit Types

```python
x = 1  # type: Union[int, str]
l = []  # type: List[int]
d = {}  # type: Dict[str, int]
```
if False:
    from typing import List, cast

reveal_type(1)  # Revealed type is 'builtins.int'
a = [4]
reveal_type(a)  # Revealed type is 'builtins.list[builtins.int*]'
b = cast(List[str], a)
reveal_type(b)  # Revealed type is 'builtins.list[builtins.str]'
from typing import Optional

def greeting(name):
    # type: (Optional[str]) -> str
    if name:
        return 'Hello, {}'.format(name)
    else:
        return 'Hello, stranger'

print(greeting('Python'))  # Okay!
print(greeting(None))     # Also okay!
class FancyString(str):
    ...

    def __eq__(self, value):
        # type: (object) -> bool
        if isinstance(value, str):
            return self._string in value
        return False
References

- What is Gradual Typing by Jeremy Siek
- Mypy playground
- PEPs
  - PEP-483
  - PEP-484
Thank you!

Dropbox has open positions
Check our booth for more information
That’s it for today

SO LITTLE TIME

SO MUCH TO DO!