

Introduction to Type Hinting with Python 3

North Austin Pythonistas
December 2019

Goals

- * What is type hinting?
- * What are the benefits of type hinting?
- * How do I begin annotating my code?

Wait, What Is a “type” and a “hint”?

- * Type == class
- * Hint == Annotation
- * Common python classes include:
int , float, str, list, dict and set
- * All user-defined classes too.

Type Hinting is for Variables

```
1 n: int = 1
2 f: float = 1.0
3 s: str = "foo"
4 d: dict = {}
5 l: list = []
6 S: set = set()
7 t: tuple = ()
```

```
1 $ mypy variables.py
2 Success: no issues found in 1 source file
```

- * A declaration ignored by the Python compiler
- * Records the programmer's intent
- * Appended to a variable on its first use

Type Hinting is Also for Functions

```
1 def function(foo: str, bar: bool = True) -> None:  
2     pass
```

- * Insert `: TYPE` before the comma
- * Insert `: TYPE = value` before the comma
- * Insert `-> TYPE` before the final colon

Annotating Class Methods

```
1 class Foobar:
2     def __init__(self, name: str) -> None:
3         self.name: str = name
4         self.value: float = 0.0
5
6     def __str__(self) -> str:
7         return f"{self.name}={self.value:05.3f}"
8
9     @property
10    def status(self) -> str:
11        if self.value <= 0.0:
12            return "normal"
13        return "oh no!"
14
15    def acknowledge(self, ack: bool = True) -> bool:
16        if ack:
17            return self.value == 0.0
18        return False
19
```

- * Never annotate “self”

- * `__init__` returns None

- * Business as Usual

A Simple Example with an Insidious Bug

```
1 def reuntokenizer(token):  
2     """re-untokenizes token"""  
3     return token + 1  
4  
5 for arg in [int(), float(), str()]:  
6     result = reuntokenizer(arg)  
7     print(type(result), result)
```


Executing The Insidious Bug

```
1 def reuntokenizer(token):  
2     """re-untokenizes token"""  
3     return token + 1  
4  
5 for arg in [int(), float(), str()]:  
6     result = reuntokenizer(arg)  
7     print(type(result), result)
```

```
1 $ python3 broke.py  
2 <class 'int'> 1
```


Executing The Insidious Bug (continued)

```
1 def reuntokenizer(token):  
2     """re-untokenizes token"""  
3     return token + 1  
4  
5 for arg in [int(), float(), str()]:  
6     result = reuntokenizer(arg)  
7     print(type(result), result)
```

```
1 $ python3 broke.py  
2 <class 'int'> 1  
3 <class 'float'> 1.0
```


Executing The Insidious Bug (oh crap!)

```
1 def reuntokenizer(token):  
2     """re-untokenizes token"""  
3     return token + 1  
4  
5 for arg in [int(), float(), str()]:  
6     result = reuntokenizer(arg)  
7     print(type(result), result)
```

```
1 $ python3 broke.py  
2 <class 'int'> 1  
3 <class 'float'> 1.0  
4 Traceback (most recent call last):  
5   File "broke.py", line 7, in <module>  
6     result = reuntokenizer(arg)  
7   File "broke.py", line 3, in reuntokenizer  
8     return token + 1  
9 TypeError: can only concatenate str (r
```


Welp, That's Broke

```
1 $ python3 broke.py
2 <class 'int'> 1
3 <class 'float'> 1.0
4 Traceback (most recent call last):
5   File "broke.py", line 7, in <module>
6     result = reuntokenizer(arg)
7   File "broke.py", line 3, in reuntokenizer
8     return token + 1
9 TypeError: can only concatenate str (not "int") to str
```


Adding Type Hinting to Reuntokenizer

```
1 def reuntokenizer(token: int) -> int:
2     """re-untokenizes token"""
3     return token + 1
4
5
6 for arg in [int(), float(), str()]:
7     result = reuntokenizer(arg)
8     print(type(result), result)
```


Still Broke, What Gives?

```
1 $ python3 hinted.py
2 <class 'int'> 1
3 <class 'float'> 1.0
4 Traceback (most recent call last):
5   File "hinted.py", line 7, in <module>
6     result = reuntokenizer(arg)
7   File "hinted.py", line 3, in reuntokenizer
8     return token + 1
9 TypeError: can only concatenate str (not "int") to str
```


Enter mypy, the Python 3 type checker

```
1 $ python3 -m pip install mypy
2 $ mypy hinted.py
3 hinted.py:6: error: Argument 1 to "reuntokenizer" has
  incompatible type "object"; expected "int"
4 Found 1 error in 1 file (checked 1 source file)
```

```
1 def reuntokenizer(token):
2     """re-untokenizes token"""
3     return token + 1
4
5 for arg in [int(), float(), str()]:
6     result = reuntokenizer(arg)
7     print(type(result), result)
```


Resources & Links

- * <https://docs.python.org/3.7/library/typing.html>
- * <https://mypy.readthedocs.io/en/latest>
- * https://mypy.readthedocs.io/en/latest/cheat_sheet_py3.html
- * <https://realpython.com/python-type-checking/>
- * <https://pre-commit.com>
- * <https://black.readthedocs.io/en/stable/>
- * <https://carbon.now.sh> – used to format code examples

One Way to Fix the Insidious Bug

```
1 from typing import Any
2
3
4 def reuntokenizer(token: Any) -> Any:
5     """re-untokenizes token"""
6     return token + type(token)(1)
7
8
9 for arg in [int(), float(), str()]:
10     result = reuntokenizer(arg)
11     print(type(result), result)
```

```
1 $ python3 fixed.py
2 <class 'int'> 1
3 <class 'float'> 1.0
4 <class 'str'> 1
```


What it Looks Like When Things Are Wrong

```
1 n: int = "nope"
2 f: float = 1
3 s: str = float()
4 d: dict = None
5 l: list = {}
6 S: set = tuple()
7 t: tuple = []
```

```
1 $ mypy broke-variables.py
2 broke-variables.py:1: error: Incompatible types in assignment
  (expression has type "str", variable has type "int")
3 broke-variables.py:3: error: Incompatible types in assignment
  (expression has type "float", variable has type "str")
4 broke-variables.py:4: error: Incompatible types in assignment
  (expression has type "None", variable has type "Dict[Any, Any]")
5 broke-variables.py:5: error: Incompatible types in assignment
  (expression has type "Dict[<nothing>, <nothing>]", variable has type
  "List[Any]")
6 broke-variables.py:6: error: Incompatible types in assignment
  (expression has type "Tuple[<nothing>, ...]", variable has type
  "Set[Any]")
7 broke-variables.py:7: error: Incompatible types in assignment
  (expression has type "List[<nothing>]", variable has type "Tuple[Any,
  ...]")
8 Found 6 errors in 1 file (checked 1 source file)
```