Python

type hints cheat sheet (python 3.9+)

```
from typing import Optional, Callable, Match, MutableMapping, Mapping


x: int = 1
x: float = 1.0
x: bool = True
x: str = "test"
x: bytes = b"test"
x: list[int] = [1]
x: set[int] = {6, 7}
x: dict[str, float] = {"field": 2.0}
x: tuple[int, str, float] = (3, "yes", 7.5)
x: tuple[int, ...] = (1, 2, 3)
x: Optional[str] = some_function()
x: Callable[[int, float], float] = f

# User-defined classes are valid as types in annotations
x: MyClass = MyClass()


# An argument can be declared positional-only by giving it a name
# starting with two underscores:
def quux(__x: int) -> None:
    pass

quux(3)  # Fine
quux(__x=3)  # Error


# If you initialize a variable with an empty container or "None"
# you may have to help mypy a bit by providing a type annotation
x: list[str] = []
x: Optional[str] = None


# "typing.Match" describes regex matches from the re module
x: Match[str] = re.match(r'[0-9]+', "15")


# Mapping describes a dict-like object (with "__getitem__") that we won't
# mutate, and MutableMapping one (with "__setitem__") that we might
def f(my_mapping: Mapping[int, str]) -> list[int]:
    my_mapping[5] = 'maybe'  # if we try this, mypy will throw an error...
    return list(my_mapping.keys())
```

```
f({3: 'yes', 4: 'no'})

def f(my_mapping: MutableMapping[int, str]) -> set[str]:
    my_mapping[5] = 'maybe'  # ...but mypy is OK with this.
    return set(my_mapping.values())
```