

Built-in Types

<code>x = 1</code>	Integer with no type
<code>x: int = 1</code>	Integer with explicit type
<code>x: float = 1.0</code>	Float
<code>x: bool = True</code>	Boolean
<code>x: str = "test"</code>	String
<code>s: bytes = b"test"</code>	Bytes

Explicit Collections (from typing import ...)

<code>x: List[int] = [1]</code>	Explicitly typed list
<code>x: Set[int] = { 2, 3 }</code>	Explicitly typed set
<code>x: Dict[str, float] = { "str": 1.0 }</code>	Explicitly typed dict
<code>x: Tuple = (1, "yes", 2.5)</code>	Explicitly typed tuple
<code>x: Tuple = (1, 2, 3)</code>	Explicitly typed open tuple
<code>x: list[int str] = [1, 2, "test", "true"]</code>	Union type (version 3.10+)
<code>x: Optional[str] = x None or Union[X, None]</code>	Optional type (union)
<code>↑ □ "test" if condition() else None</code>	Optional type (conditional)

Collections (from typing import ...)

<code>x: list[int] = [1]</code>	Array
<code>x: set[int] = { 1, 2 }</code>	Set
<code>x: dict[str, float] = { "test", 1.0 }</code>	Dictionary / Hash Table
<code>x: tuple[int, str, float] { 1, "yes", 2.5 }</code>	Tuple with 3 values
<code>x: tuple[int, ...] = { 1, 2, 3 }</code>	Tuple of variable size

Functions (from typing import...)

<code>def function(x: int) -> str</code>	Basic function with a return value
<code>def function(x: int = 1) -> str</code>	Basic function with a default value
<code>def function(x)</code>	Basic function with no argument type
<code>x: Callable[[int, float], float]</code>	Function as a class
<code>def gen(n: int) -> Iterator[int]</code>	Generator function that yields ints
<code>def function(self, *args: str, **kwargs: str) -> int</code>	All position args and keywords are strings



By MrBeverage

cheatography.com/mrbeverage/

Not published yet.

Last updated 25th March, 2024.

Page 1 of 1.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>