Python Annotations & Docstrings

Type Annotations

Type annotations in Python provide hints about variable and function return types, improving code clarity and assisting with static analysis.

Function Annotations

```python
def add(x: int, y: int) -> int:
    return x + y
```

✅ `x: int` and `y: int` specify expected parameter types
✅ `-> int` indicates the return type

Variable Annotations

```python
count: int = 0
name: str = "Python"
```

✅ Improves readability and documentation
✅ No enforcement at runtime (only hints)

Docstrings

Docstrings provide **documentation** for modules, functions, and classes.

Function Docstrings

```python
def greet(name: str) -> str:
    """Returns a greeting message."""
    return f"Hello, {name}!"
```

✅ First statement inside a function as a `"""string"""`
✅ Helps with `help(greet)`

Class Docstrings

```
class Dog:
    """A class representing a dog."""

    def __init__(self, name: str, age: int):
        """Initialize a new dog with name and age."""
        self.name = name
        self.age = age
```

✅ Documents the purpose of a class
✅ Helps maintain clean, understandable code

Best Practices

✔ Always use type hints for complex functions
✔ Write **clear and concise** docstrings
✔ Use **Google-style** or **NumPy-style** docstrings for large projects
✔ Keep docstrings up-to-date with function changes