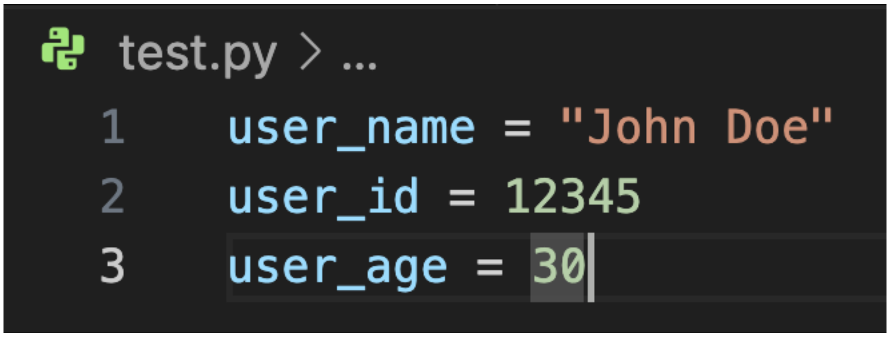
# Python coding standards

## 1. Naming Conventions

### 1.1.Variables

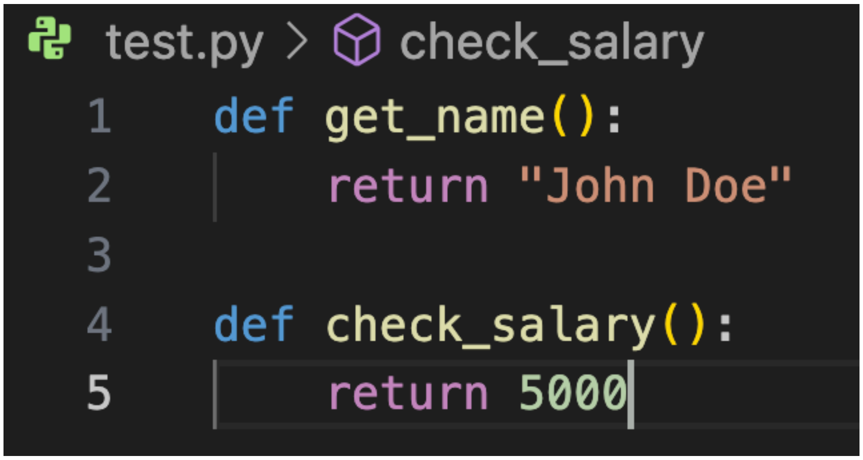
- We will be using **snake_case conventions**, and the first letter will be lowercase.
- Example : user_name , user_id , user_age .



```python
user_name = "John Doe"
user_id = 12345
user_age = 30
```

### 1.2.Functions and Methods

- We will be using **snake_case conventions** for function and method names.
- Example : get_name(), check_salary().



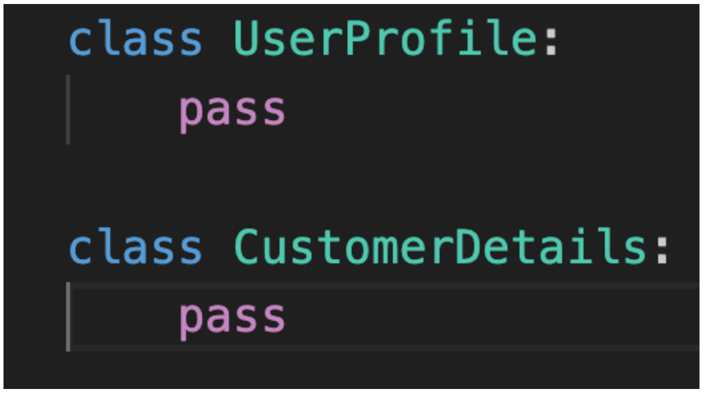
```python
def get_name():
    return "John Doe"

def check_salary():
    return 5000
```

### 1.3.Classes

- We will be using the **PascalCase** convention.

- Example : UserProfile, CustomerDetails.

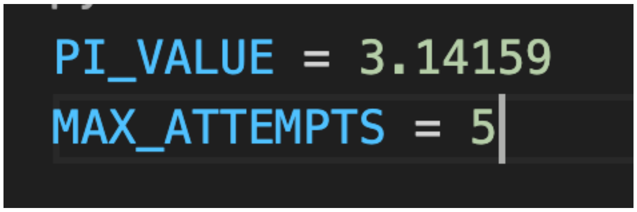


```python
class UserProfile:
    pass

class CustomerDetails:
    pass
```

### 1.4.**Constants**

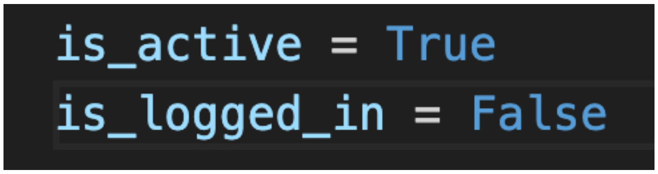
- We will be **ALL_UPPERCASE** for constants.
- Example : PI_VALUE, MAX_ATTEMPTS



```python
PI_VALUE = 3.14159
MAX_ATTEMPTS = 5
```

### 1.5.**Additional_Naming Guidelines**

- Avoid names that are long .
- Packages and module names should be lower case . When multiple words are needed separate them with underscores but single-word names are preferable .



```python
is_active = True
is_logged_in = False
```

- Use the **is_** prefix for all Boolean variables or functions .

## 2. Code Layout

### 2.1. **Indentation**

- We will be using 4 spaces per indentation, and we have to avoid tabs.

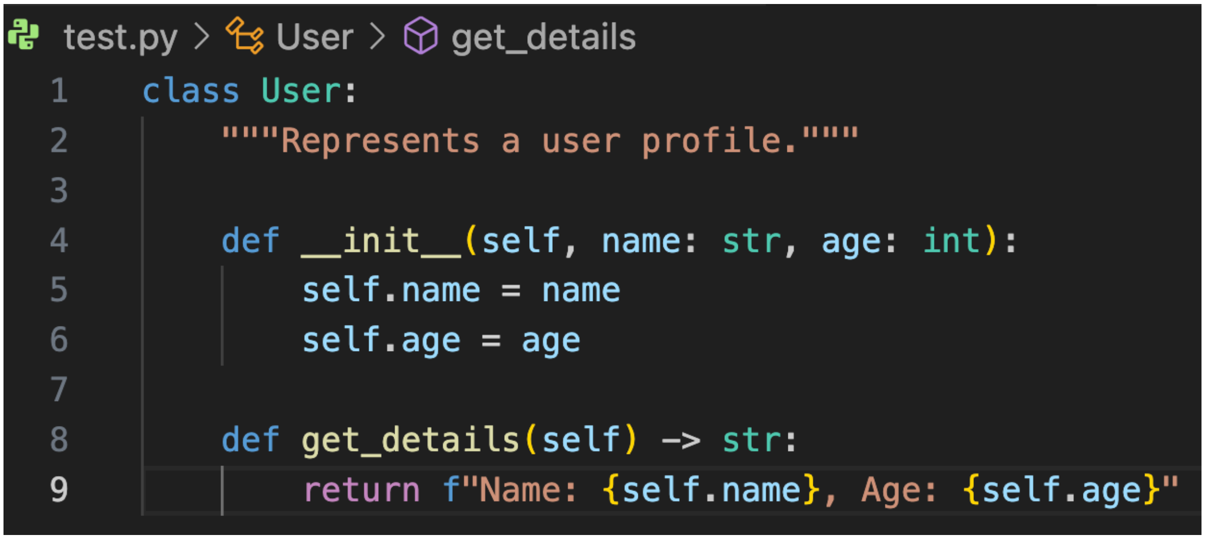
## 2.2. **Line length**

- We will limit all lines to a maximum of 79 characters.

## 2.3.**Spacing Rules**

- We will use two blank lines before top-level functions and classes.
- We will surround method definitions inside a class with a single blank line.
- We will avoid extra spaces inside parentheses, brackets or braces.
- We will avoid extra spaces between a trailing comma and a closing parenthesis.
- We will avoid extra spaces immediately before a comma, semicolon, or colon.

## 2.4.Example of Proper Formatting :



```python
class User:
    """Represents a user profile."""

    def __init__(self, name: str, age: int):
        self.name = name
        self.age = age

    def get_details(self) -> str:
        return f"Name: {self.name}, Age: {self.age}"
```