Python Syntax, Whitespace, and Indentation

1. Using Print Statements

- A simple `print()` statement generates text in the terminal.
  Example:

  ```
  print("Hello")
  ```

  Output:

  ```
  Hello
  ```

- **Common Error**: Placing two `print()` statements on the same line causes a syntax error.
  Example:

  ```
  print("Hello") print("World")  # SyntaxError: invalid syntax
  ```

- **Solutions**:
  - Move the second `print()` statement to a new line.
  - Separate the two statements with a semicolon (`;`).
    Example:

    ```
    print("Hello"); print("World")
    ```

    Output:

    ```
    Hello
    World
    ```

2. Whitespace in Python

- Extra whitespace around operators like `+` is ignored by the interpreter.
  Example:

  ```
  x = 1     +      2  # No error
  print(x)  # Output: 3
  ```

- **Issues with Multiline Expressions**:
  - Adding an operator on a new line without a continuation indicator causes errors.
    Example:

    ```
    x = 1 + 2
    ```

```
+ 3  # Only the first line is executed; output is 3
```

- **Solution**: Use a backslash (`\`) to indicate a continuation.
  Example:

```
x = 1 + 2 \
    + 3
print(x)  # Output: 6
```

3. Indentation in Python

- Indentation is crucial for defining code blocks (e.g., in `if` statements).
  Example:

```
name = "John"
if name == "John":
    print(name)  # Correctly indented; Output: John
```

- **Common Error**: Missing or incorrect indentation.
  Example:

```
name = "John"
if name == "John":
print(name)  # IndentationError: expected an indented block
```

- **Error Handling**:
  - Python's error messages indicate the exact line where the issue occurred.
  - Correct the indentation to resolve the error.

Key Takeaways

- **Print Statements**: Avoid placing multiple statements on the same line without proper separation.
- **Whitespace**: While excessive whitespace is ignored within a single line, multiline expressions require continuation indicators.
- **Indentation**: Proper indentation is mandatory in Python to define code blocks. Errors like `IndentationError` help identify where the issue is.
- **Debugging**: Reading Python's error messages often provides specific details about what went wrong and where to fix it.