RRC POLYTECH

Module 6
Variables, constants, literals and PEP8 guidelines

Introduction

In the world of programming, variables, constants, literals, and coding style guidelines play a vital role in writing clean, organized, and efficient code.

In this chapter, we'll dive into these concepts using the Python programming language and follow the **PEP 8 style guidelines to enhance our coding practices**.

Variables

Variables **are like containers that store data for later use**.

In Python, we create a variable by giving it a name and assigning a value to it.

Variable names should follow these rules:

a) Start with a letter (a-z, A-Z).
b) Followed by letters or digits (0-9), then underscore
c) We use the term snake_case for this.
e) Variables' names are case-sensitive.

Examples:

```
my_name = "Alice"
my_age = 30
height = 5.8
```

Constants

Constants **are values that remain the same throughout the program's execution**.

Although **Python doesn't have true constants**, <u>we use uppercase letters for constants names to indicate that these values should not be changed</u>.

Examples:

```
PI = 3.14159
GRAVITY = 9.8
```

Literals

Literals are unchanging values embedded directly within code, representing themselves without any transformation

When the literals come from the concatenation of string literals, they are considered good coding practices. However, if they are randomly **used as variables substitutes** within lines of code, they represent bad coding practices.

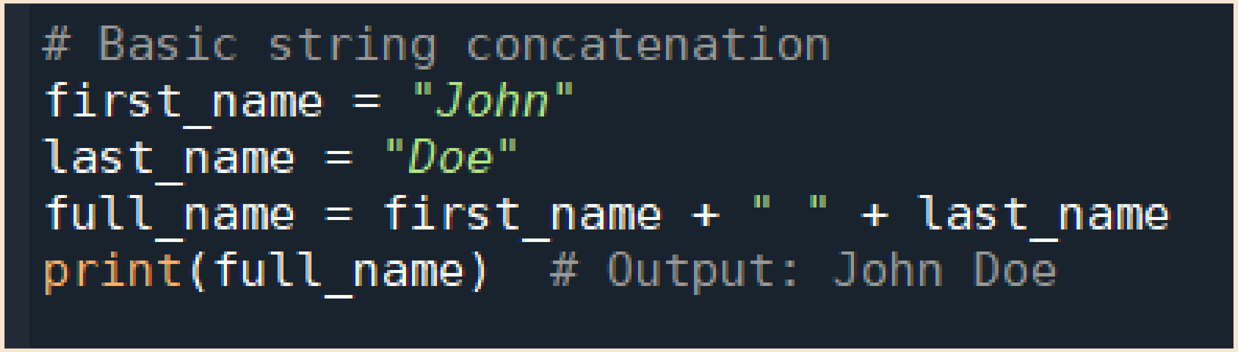
Examples:

"Hello" is a string literal
42 is an integer literal
pi_value = 3.14 is a float literal

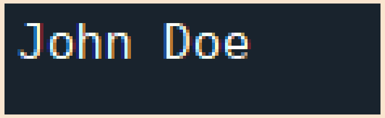
Literals – concatenation

String literals are useful when concatenating as we mentioned before.

Basic string concatenation is achieved using the + operator between two strings of literals, as follows:



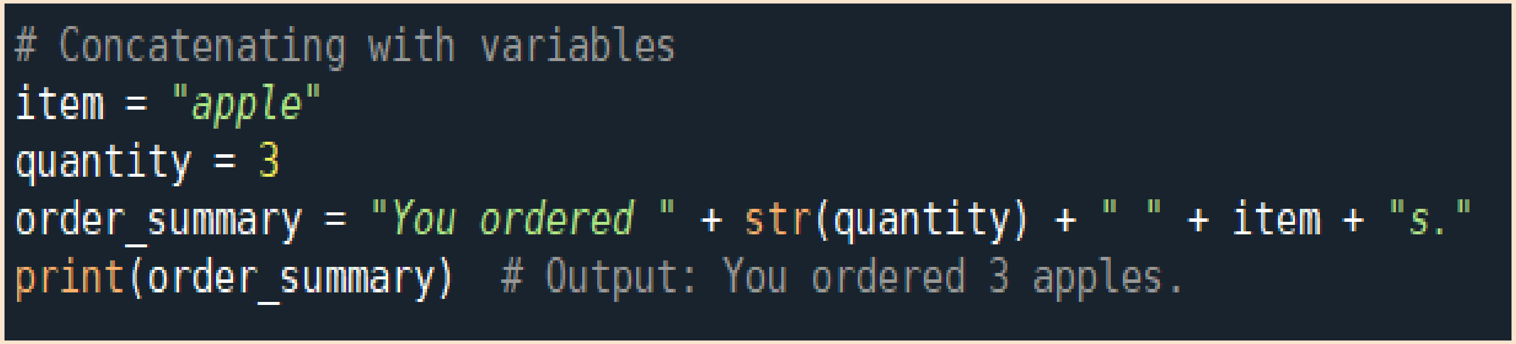
```python
# Basic string concatenation
first_name = "John"
last_name = "Doe"
full_name = first_name + " " + last_name
print(full_name)  # Output: John Doe
```



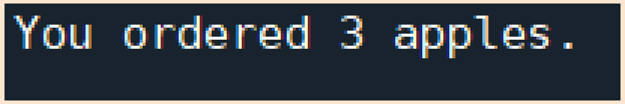
```
John Doe
```

Literals – concatenation

Variable string concatenation, you can also concatenate strings literals with variables to create dynamic strings, as follows:



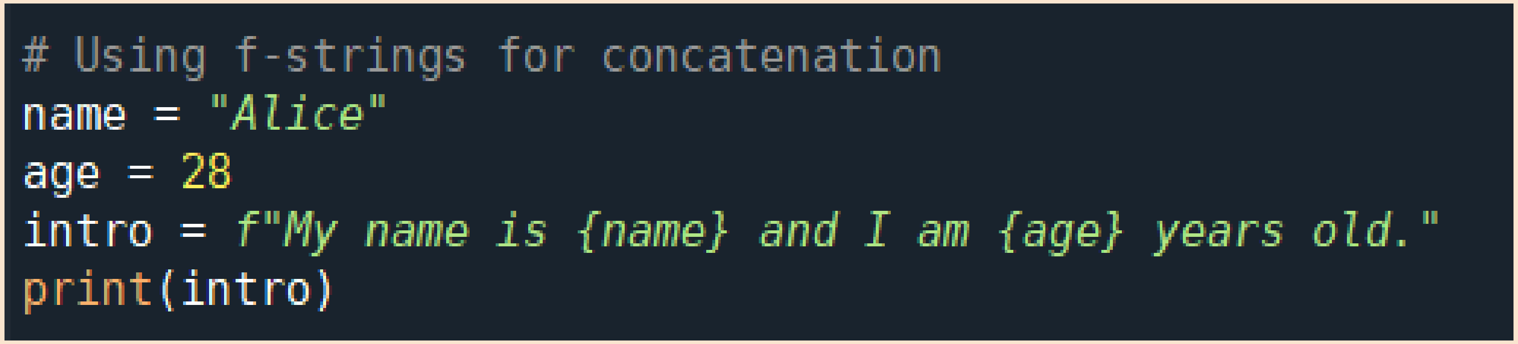
```
# Concatenating with variables
item = "apple"
quantity = 3
order_summary = "You ordered " + str(quantity) + " " + item + "s."
print(order_summary)  # Output: You ordered 3 apples.
```



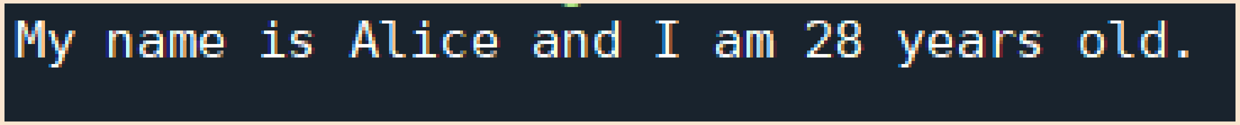
```
You ordered 3 apples.
```

Literals – concatenation

Formatted Strings concatenation also known as f-strings, provide a concise and readable way to concatenate string literals and variables. You can embed variables directly into the string using curly braces {}, there is not need to use the + operator, as follows:



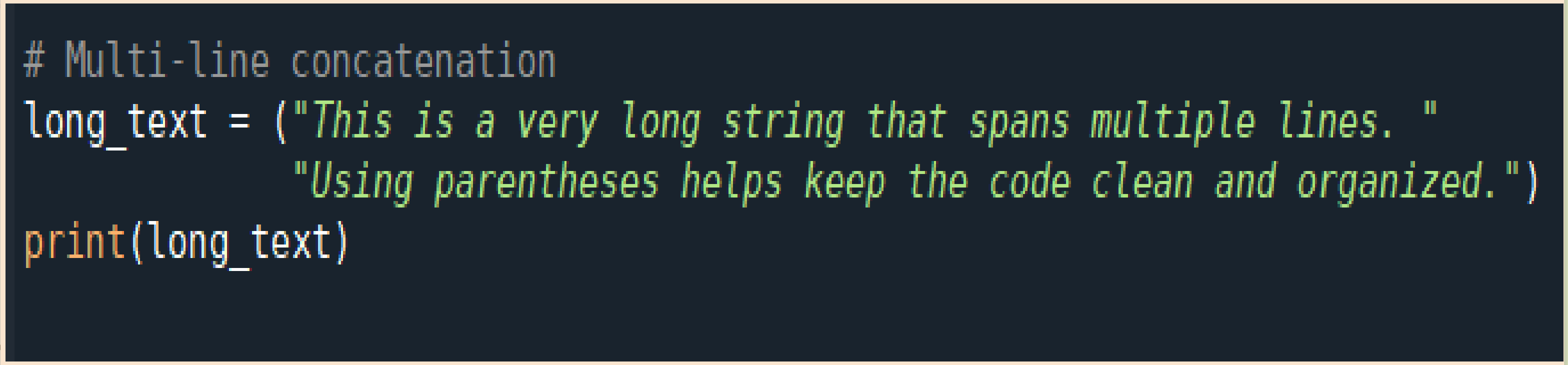
```python
# Using f-strings for concatenation
name = "Alice"
age = 28
intro = f"My name is {name} and I am {age} years old."
print(intro)
```



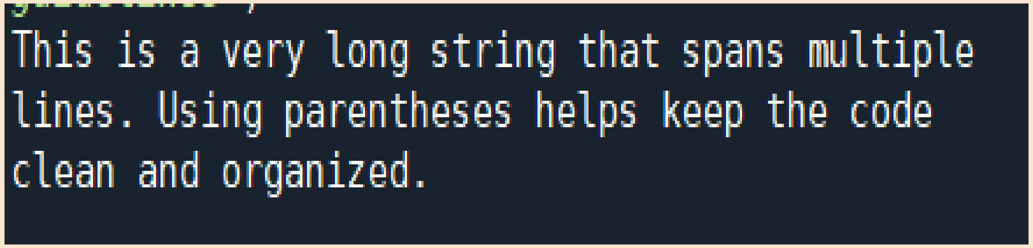
```
My name is Alice and I am 28 years old.
```

Literals – concatenation

Multi-line or long string concatenation allows you concatenate long strings. This really helps keeping your code clean and organized, as follows:



```python
# Multi-line concatenation
long_text = ("This is a very long string that spans multiple lines. "
             "Using parentheses helps keep the code clean and organized.")
print(long_text)
```



```
This is a very long string that spans multiple
lines. Using parentheses helps keep the code
clean and organized.
```

PEP 8 Guidelines

PEP 8 is the official style guide for Python code. Following these guidelines makes your code more consistent and easier to understand for both you and other developers.

Let's explore some key PEP 8 recommendations:

1) Indentation: Use 4 spaces per indentation level. Python uses indentation to define all of its structures as we will discuss in few weeks.

2) Naming Conventions: Use **snake_case for variable, py files** and function names, CamelCase for class names, and **UPPER_CASE for constants**.

3) Whitespace: Use **spaces around operators and after commas**. Avoid extraneous white-space.

PEP 8 Guidelines

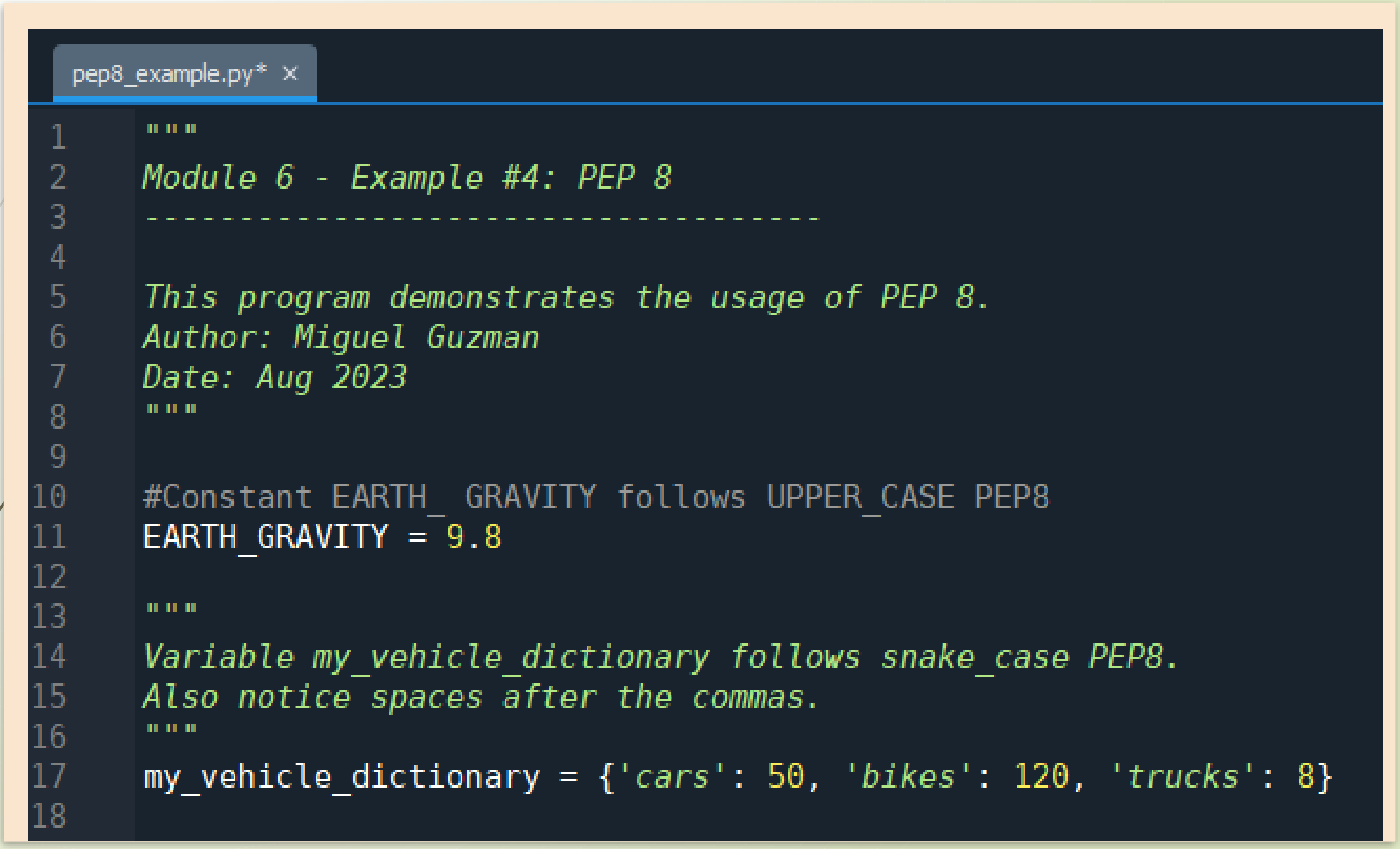
4) Comments: Write **clear and concise line comments to explain complex code**. Use **docstrings at the beginning of your program** or modules; and after you present a function.

5) Line Length: **Limit lines to 79 characters**. For longer lines, break them using appropriate indentation.

6) Imports: Place import statements at the top of the file. Group standard library imports, third-party imports, and local imports separately.

7) Function and Class Definitions: Use two blank lines before class and function definitions.

PEP 8 Guidelines - Example



```python
"""
Module 6 - Example #4: PEP 8
------------------------------------

This program demonstrates the usage of PEP 8.
Author: Miguel Guzman
Date: Aug 2023
"""

#Constant EARTH_ GRAVITY follows UPPER_CASE PEP8
EARTH_GRAVITY = 9.8

"""
Variable my_vehicle_dictionary follows snake_case PEP8.
Also notice spaces after the commas.
"""
my_vehicle_dictionary = {'cars': 50, 'bikes': 120, 'trucks': 8}

```

PEP 8 Guidelines – Links

You should review PEP 8 official Style Guide for Python visiting:
https://peps.python.org/pep-0008/

An easy approach is to visit PYLEECAN website, this project presents a concise adaptation of PEP8 focus to mathematics contexts:
https://www.pyleecan.org/coding.convention.html

The Real Python website provides additional information:
https://realpython.com/python-variables/