Basic python

DAY 5

Control Flow Statements with Strings

Python Control Flow Statements

A program’s control flow is the order in which statements or blocks of code are executed at runtime based on a condition. The control flow of a Python program is regulated by conditional statements, loops, and function calls

Control Flow Statements with Strings

Control flow statement

The control flow statements are divided into three categories

1. Conditional statements
2. Transfer statements
3. Iterative statements

Conditional statements

conditional statements act depending on whether a given condition is true or false. You can execute different blocks of codes depending on the outcome of a condition. Condition statements always evaluate to either True or False.
Three types of conditional statements.

1. if statement
2. if-else
3. if-elif-else
4. nested if-else

Iterative statements

In Python, iterative statements allow us to execute a block of code repeatedly as long as the condition is True. We also call it a loop statements. Python provides us the following two loop statement to perform some actions repeatedly

1. for loop
2. while loop

Transfer statements

In Python, transfer statements are used to alter the program's way of execution in a certain manner. For this purpose, we use three types of transfer statements.

1. break statement
2. continue statement
3. pass statements

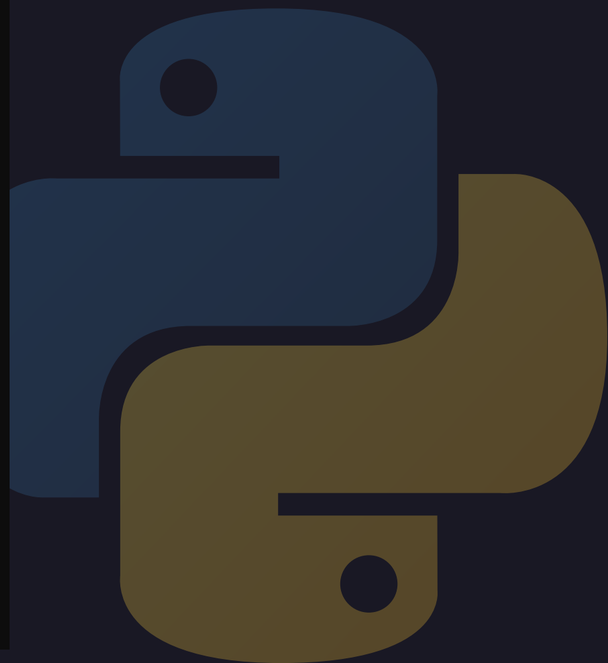
Conditional statements

1. if Statement

python

```python
# Example 1: Check if two strings are equal
string1 = "hello"
string2 = "hello"
if string1 == string2:
    print("The strings are equal.")

# Example 2: Check if a character exists in a string
text = "Python"
if "y" in text:
    print("'y' is present in the text.")
```

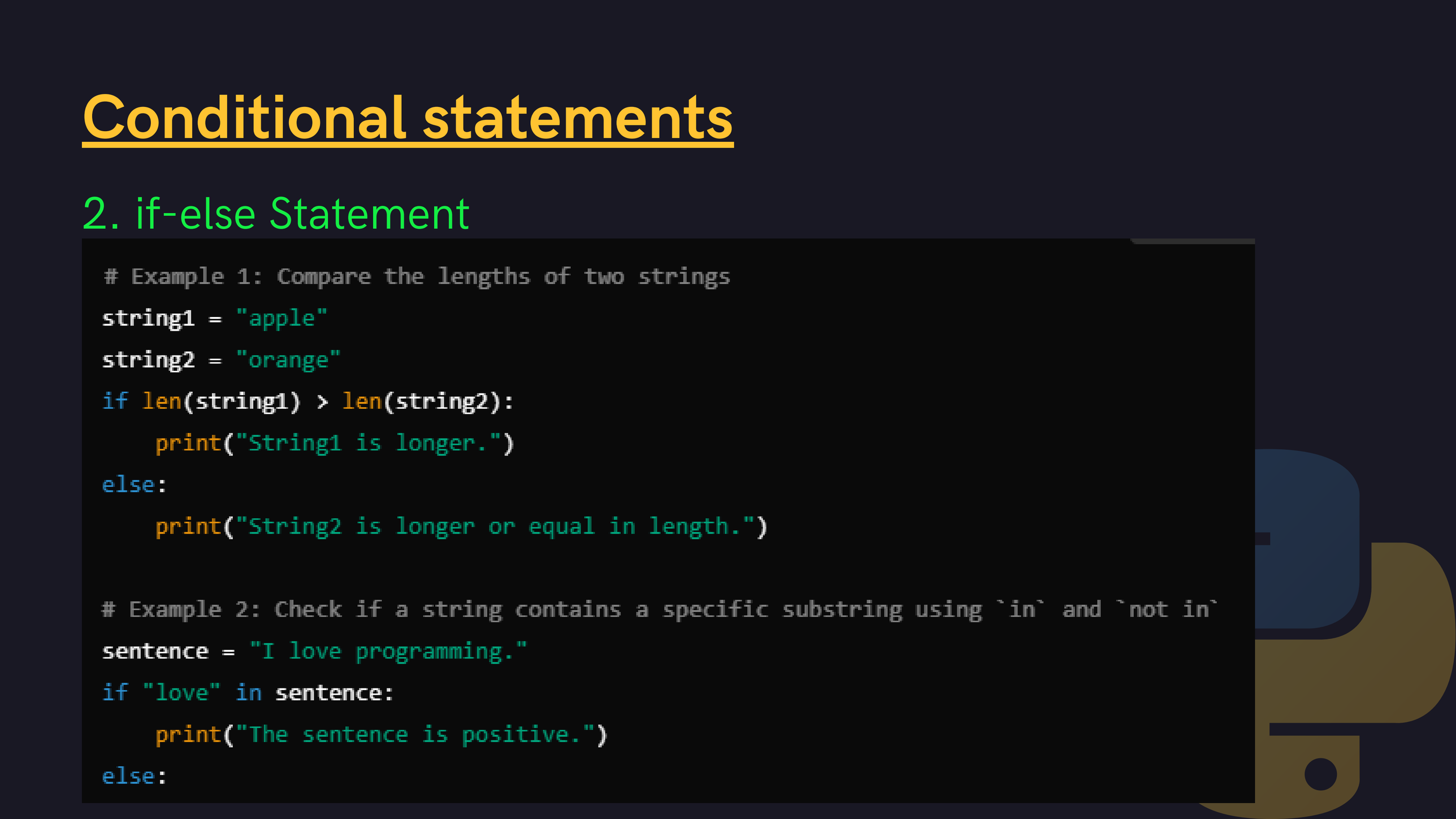


Conditional statements

2. if-else Statement

```
# Example 1: Compare the lengths of two strings
string1 = "apple"
string2 = "orange"
if len(string1) > len(string2):
    print("String1 is longer.")
else:
    print("String2 is longer or equal in length.")

# Example 2: Check if a string contains a specific substring using `in` and `not in`
sentence = "I love programming."
if "love" in sentence:
    print("The sentence is positive.")
else:
```



Conditional statements

3. if-elif-else Statement

```
# Example 1: Categorize strings based on their size
word = "Python"
if len(word) < 4:
    print("The string is small.")
elif len(word) == 6:
    print("The string is medium.")
else:
    print("The string is long.")
```

Example 2

```
# Example 2: Compare two strings alphabetically
string1 = "apple"
string2 = "banana"
if string1 < string2:
    print(f"'{string1}' comes before '{string2}' alphabetically.")
elif string1 > string2:
    print(f"'{string1}' comes after '{string2}' alphabetically.")
else:
    print("The strings are the same.")
```

Conditional statements

4. Nested if-else

python

```python
# Example 1: Check if a string starts and ends with specific characters
text = "hello"
if text[0] == "h":  # First character
    if text[-1] == "o":  # Last character
        print("The string starts with 'h' and ends with 'o'.")
    else:
        print("The string starts with 'h' but doesn't end with 'o'.")
else:
    print("The string does not start with 'h'.")
```

Example 2

python

Copy code

```python
# Example 2: Check if one string contains another and compare lengths
string1 = "hello world"
string2 = "hello"
if string2 in string1:
    if len(string1) > len(string2):
        print(f"'{string2}' is a substring of '{string1}', and '{string1}' is longer.")
    else:
        print(f"'{string2}' is a substring, but lengths are equal.")
else:
    print(f"'{string2}' is not a substring of '{string1}'.")
```

THANK YOU