Booleans and Strings

Key Concepts:

- String Formats and Operators
- Boolean Expressions

String Syntax in Python

Strings can be defined using three different syntaxes:

- Single quotes:
  `print('example')`
- Double quotes:
  `print("example")`
- Triple quotes:
  `print("""example""")`

Advantages:

- Double quotes within single-quoted strings:
  `print('He said "hello"!')`
- Single quotes within double-quoted strings:
  `print("It's a beautiful day!")`
- Both single and double quotes within triple-quoted strings:
  `print("""She said, "It's a beautiful day!" """)`
- Triple quotes also allow multi-line strings.

String Operations

Length:

```
s = "abcde"
print(len(s))  # Output: 5
```

Concatenation:

```
print("abc" + "def")  # Output: 'abcdef'
```

Repetition:

```
print("ta " * 4)  # Output: 'ta ta ta ta'
```

Boolean Expressions

Logical OR: `or`

- `expr1 or expr2` is true if at least one of the expressions is true.
- Python uses short-circuit evaluation: if the first expression is true, the second is not evaluated.

Example:

```
(2 == 1 + 1) or (a >= 5)  # True, no error even if a is not defined
(3 == 1 + 1) or (a >= 5)  # Error if a is not defined
```

Logical AND: `and`

- `expr1 and expr2` is true only if both expressions are true.
- Python uses short-circuit evaluation: if the first expression is false, the second is not evaluated.

Example:

```
(2 > 8) and (a >= 5)  # False, no error even if a is not defined
(2 < 8) and (a >= 5)  # Error if a is not defined
```

De Morgan's Laws

De Morgan's Laws are transformation rules that relate the logical operators "and" (`and`) and "or" (`or`) with the negation operator (`not`). These laws are very useful in simplifying logical expressions.

Law 1:

not (expr1 or expr2) is equivalent to (not expr1) and (not expr2)

This law states that the negation of a disjunction (an `or` statement) is the same as the conjunction (an `and` statement) of the negations.

Law 2:

not (expr1 and expr2) is equivalent to (not expr1) or (not expr2)

This law states that the negation of a conjunction (an `and` statement) is the same as the disjunction (an `or` statement) of the negations.

- `not (expr1 or expr2)` is equivalent to `not (expr1) and not (expr2)`
- `not (expr1 and expr2)` is equivalent to `not (expr1) or not (expr2)`

Examples:

```
not (a > 2 or b <= 4)  # Equivalent to: (a <= 2) and (b > 4)
not (a > 2 and b <= 4)  # Equivalent to: (a <= 2) or (b > 4)
```
