PYTHON FOR DATA SCIENCE

UNIT 12

BOOLEANS
&
COMPARISON OPERATORS

MAIN TOPICS TO BE COVERED

- What is boolean data … i.e., what is the boolean data type
- Comparison operators
- Logical operators
- How to create simple logical expressions
- How to create more complex logical expressions

BOOLEAN BASICS

WHAT IS BOOLEAN DATA?

- **bool** is a data type
- The **Boolean data type** is a data type that has one of two possible values … usually denoted **True** and **False**
- The Boolean data type is named after George Boole, who first defined an algebraic system of logic in the mid 19th century.
- The Boolean data type is primarily associated with conditional statements, which allow different actions by changing control flow depending on whether a programmer-specified Boolean *condition* evaluates to true or false.

WHAT IS BOOLEAN DATA?

```
bool_true = True
print(bool_true)
print(type(bool_true))
```

```
True
<class 'bool'>
```

```
bool_false = False
print(bool_false)
print(type(bool_false))
```

```
False
<class 'bool'>
```

True AND False ARE SPECIAL WORDS IN PYTHON

- True and False are explicitly boolean data
- True and False are reserved words in Python, so we must use them properly

```
print(type(True))
print(type(False))
```

```
<class 'bool'>
<class 'bool'>
```

True AND False ARE SPECIAL WORDS IN PYTHON

- The Python Boolean type has only two possible values:

  1. True
  2. False

- No other value will have bool as its type.

- The type bool is built in, meaning it's always available in Python and doesn't need to be imported. However, the name bool itself isn't a keyword in the language.

- In contrast, the names True and False are not built-ins. They are keywords. Unlike many other Python keywords, True and False are Python expressions. Since they're expressions, they can be used wherever other expressions, like 1 + 1, can be used.

BASIC LOGIC

AND

COMPARISON

BOOLEAN EXPRESSIONS EVALUATE AS True AND False

- In Python, you can create expressions that produce Boolean outputs …
    - comparisons
    - logical statements
    - membership statements
- Boolean expressions are used frequently in software and in data science scripts

COMPARISONS WILL PRODUCE BOOLEAN OUTPUTS

Here, we are comparing two integers, which produces a boolean output

```
12 > 21
```

```
False
```

Here, we are comparing two variables, which also produces a boolean output

```
x = 35
y = 10 + 25
x == y
```

```
True
```

These comparisons are types of “boolean expressions”

COMPARISON / RELATIONAL OPERATORS IN PYTHON

- Comparison operators can be used to create boolean expressions
- Comparison operators are also called “relational operators”
- These operators all return a boolean value … i.e., they return either True or False

COMPARISON / RELATIONAL OPERATORS IN PYTHON

Operator	Meaning	Example
>	Greater than - True if left operand is greater than the right	x > y
<	Less than - True if left operand is less than the right	x < y
==	Equal to - True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y
>=	Greater than or equal to - True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to - True if left operand is less than or equal to the right	x <= y

MEMBERSHIP OPERATORS IN PYTHON

`in` and `not in` are the membership operators in Python. They are used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary).

In a dictionary we can only test for presence of key, not the value.

Operator	Meaning	Example
in	True if value/variable is found in the sequence	5 in x
not in	True if value/variable is not found in the sequence	5 not in x

IDENTITY OPERATORS IN PYTHON

Identity operators

`is` and `is not` are the identity operators in Python. They are used to check if two values (or variables) are located on the same part of the memory. Two variables that are equal does not imply that they are identical.

Operator	Meaning	Example
is	True if the operands are identical (refer to the same object)	x is True
is not	True if the operands are not identical (do not refer to the same object)	x is not True

COMPARISONS ARE VERY SIMPLE EXAMPLES OF “LOGICAL EXPRESSIONS”

- In Python, we can create much more complicated logical expressions

- Complex logical expressions combine
  - Comparison operators (Membership Operators, Identity Operators)
  - Logic operators

LOGICAL OPERATORS

AND

COMPOUND BOOLEAN EXPRESSIONS

LOGICAL / BOOLEAN OPERATORS CAN BE USED TO CREATE COMPLEX (COMPOUND) LOGICAL STATEMENTS

- Example:
    - The number is greater than 3 and the number is less than 10
    - The number is less than -2 or the number is greater than 5
- You need to use logical operators (also called boolean operators) to combine simple comparison (membership, identity) expressions
- The logical / boolean operators are …
    - and
    - or
    - not

LOGICAL / BOOLEAN OPERATORS IN PYTHON

- Logical operators can be used to create complex logical expressions
    - i.e., these logical / boolean operators help us create “compound” boolean expressions

Logical operators are the `and`, `or`, `not` operators.

Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x

EXAMPLES: LOGICAL / BOOLEAN EXPRESSIONS

- These simple examples demonstrate using logical / boolean operators to combine boolean expressions into more complex expressions
  - i.e., simple comparisons are being combined with **and**, **or**, and **not**

```
(50 > 12) and (10 == 10)
```

```
True
```

```
(50 > 12) and not (10 == 10)
```

```
False
```

```
(35 > 15) or (16 < 20)
```

```
True
```

```
(18 > 25) or (10 == 10)
```

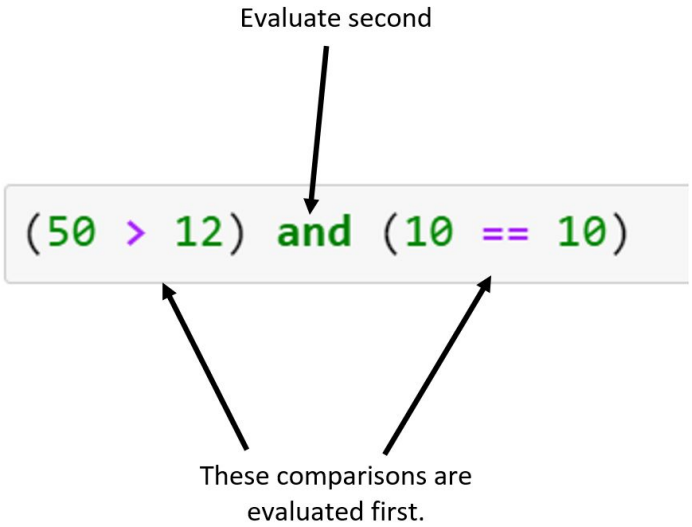
```
True
```

```
(18 > 25) or not (10 == 10)
```

```
False
```

COMPARISON OPERATORS ARE EVALUATED BEFORE THE LOGICAL OPERATORS and, or, AND not

- Comparison operators have higher precedence than the boolean operators
- Be careful when creating logical expressions!



USING PARENTHESES IN LOGICAL STATEMENTS IS A “BEST PRACTICE”

```
(50 > 12) and (10 == 10)

50 > 12 and (10 == 10)
```

- These two statements are essentially the same
- But the one with parentheses is easier to read and process!
- Therefore, use parentheses whenever possible when writing logical statements