

BOOLEANS, COMPARISONS, AND LOGICAL EXPRESSIONS

SHARP SIGHT

WHAT YOU'LL LEARN

- What is boolean data
 - i.e., 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
 - Boolean data is logical data: **True** or **False**
- **True** and **False** are special words in Python

```
true_bool = True  
print(true_bool)
```

True

```
type(true_bool)
```

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 use them properly

```
type(True)  
bool
```

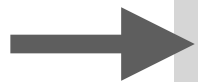
BASIC LOGIC AND COMPARISON

BOOLEAN EXPRESSIONS EVALUATE AS True OR False

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

COMPARISONS WILL PRODUCE BOOLEAN OUTPUTS

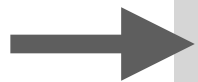
Here, we're comparing two integers, which produces a boolean output



```
10 > 1
```

```
True
```

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



```
x = 42
```

```
y = 11
```

```
x == y
```

```
False
```

These comparisons are types of "boolean expressions"

COMPARISON OPERATORS IN PYTHON

- We can use comparison operators to create boolean expressions
 - note: these are sometimes called “relational operators”
- These operators all return a boolean (i.e., **True/False**)

Operator	Meaning
<code>==</code>	equals
<code>!=</code>	not equal
<code>></code>	greater than
<code><</code>	less than
<code>>=</code>	greater than or equal
<code><=</code>	less than or equal
<code>in</code>	membership

COMPARISONS ARE VERY SIMPLE

EXAMPLES OF "LOGICAL EXPRESSIONS"

- In Python, we can create much more complicated logical expressions
- Complex logical expressions combine
 - comparison operators
 - logic operators

LOGICAL OPERATORS AND COMPOUND BOOLEAN EXPRESSIONS

YOU CAN USE LOGIC OPERATORS TO CREATE COMPLEX LOGICAL STATEMENTS

- Example:
 - the number is greater than 3
 - *and*, the number is less than 10
- You need to use logical operators (AKA: boolean operators) to combine simple comparison expressions
 - i.e., **and**, **or**, and **not**

LOGICAL OPERATORS IN PYTHON

- We can use logical operators to create complex logical expressions
 - these help us create "compound" boolean expressions

Logical operators in Python

Logic operator	What it does
and	returns True if both are True
or	returns True if either one is True
not	returns the opposite truth value e.g., returns False if True , and True if False

EXAMPLES: LOGICAL EXPRESSIONS

- Here are 3 simple examples
 - These demonstrate using logical operators to combine boolean expressions into more complex expressions

Notice that we are combining simple comparisons with and, or, and not →

```
(10 > 1) and (9 == 9)
```

True

```
(10 > 1) and not (9 == 9)
```

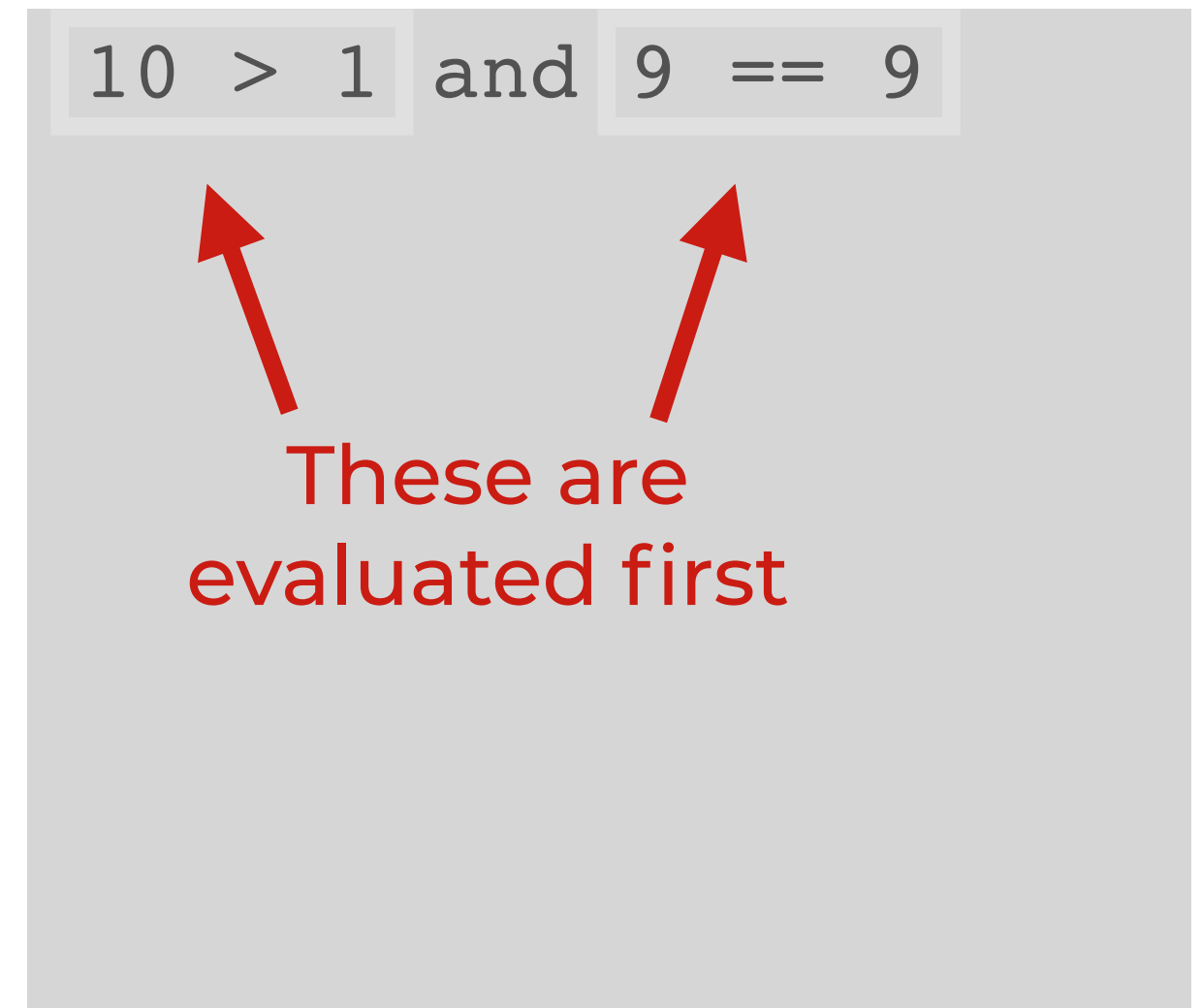
False

```
(42 == 7) or (2 == 2)
```

True

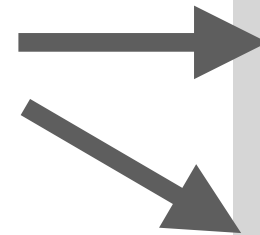
COMPARISON OPERATORS ARE EVALUATED BEFORE *and*, *or*, AND *not*

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



USING PARENTHESIS IN LOGICAL STATEMENTS IS A "BEST PRACTICE"

These two statements are essentially the same



```
10 > 1 and 9 == 9
```

```
(10 > 1) and (9 == 9)
```

But the one with parenthesis is easier to read!

Use parenthesis whenever possible

RECAP

RECAP OF WHAT WE LEARNED

- Boolean data
- Comparison operators
 - greater than, less than, etc
 - How to create simple boolean expressions
- How to use logical operators to combine simple boolean expressions
 - and, or, and not
- Note: boolean expressions and logic are important
 - You need them for higher-order program structures
 - Learn them well!