CSCI 141

Lecture 6:
The bool data type
Boolean Expressions
Boolean Operators
Happenings

Tuesday, 4/16 – Artificial Intelligence Association Presents: Tableau and Power BI – 6 pm in PH 228

Wednesday, 4/17 – Cybersecurity Lecture Series: Information Security with Austin Tipton – 5 pm in CF 105

Wednesday, 4/17 – Peer Lecture Series: Vim Workshop – 5 pm in CF 162

Thursday, 4/18 – Group Advising to Declare the CS Major – 3 pm in CF 420

Thursday, 4/18 – Western Information Systems Connection – 5 pm in the WWU Library
Announcements

• A2 is out.
  • Due next Monday night

• Assignments will get more involved as the quarter progresses - start early.
Goals

• Know how to apply operator precedence rules to determine the order in which pieces of an expression are evaluated.

• Understand the use and values of the type `bool` and the meaning of a boolean expression.

• Understand the behavior of the arithmetic comparison operators (`<, >, <=, >=, ==, !=`).

• Understand the behavior of the boolean logical operators `and`, `or`, and `not`
Today’s Quiz

• 2 minutes
Today’s Quiz

• 2 minutes

• Working with a neighbor: do your answers agree? (2 minutes)
Practice Problem: Operators

Suppose we run the following program, and the user types 6 and presses enter.

What value gets stored in result?

```python
user_num = input("Enter a number: ")
result = 5 % (3 ** (user_num // 4))
```

A: 1
B: 2
C: 3
D: None of the above
Practice Problem: Operators

Suppose we run the following program, and the user types 6 and presses enter.

What value gets stored in result?

```python
user_num = input("Enter a number: ")
result = 5 % (3 ** (user_num // 4))
```

Let’s try it out...
Practice Problem: Operators

Suppose we run the following program, and the user types 6 and presses enter.

What value gets stored in result?

```python
user_num = input("Enter a number: ")
result = 5 % (3 ** (user_num // 4))
```

A: 1  
B: 2  
C: 3  
D: None of the above
Bugs

• We had a bug in our code!

• Why are they called bugs? An anecdote from the history of computing:

At 3:45 p.m., Grace Murray Hopper records 'the first computer bug' in the Harvard Mark II computer's log book. The problem was traced to a moth stuck between relay contacts in the computer, which Hopper duly taped into the Mark II's log book with the explanation: “First actual case of bug being found.” The bug was actually found by others but Hopper made the logbook entry.

Source: https://www.computerhistory.org/tdih/september/9/
“First actual case of a bug being found”
Suppose we run the following program, and the user types 6 and presses enter.

What value gets stored in result?

```python
user_num = int(input("Enter a number: "))
result = 5 % (3 ** (user_num // 4))
```
Suppose we run the following program, and the user types 6 and presses enter.

What value gets stored in result?

```python
user_num = int(input("Enter a number: "))
result = 5 % (3 ** (user_num // 4))
```

Practice Problem: Operators

A: 1
B: 2
C: 3
D: None of the above
Practice Problem: Operators

Suppose we run the following program, and the user types 6 and presses enter.

What value gets stored in `result`?

```python
user_num = int(input("Enter a number: "))
result = 5 % (3 ** (user_num // 4))
```
Suppose we run the following program, and the user types 6 and presses enter.

What value gets stored in result?

```
user_num = int(input("Enter a number: "))
result = 5 % (3 ** (user_num // 4))
result = 5 % (3 ** (6 // 4))
```
Practice Problem: Operators

Suppose we run the following program, and the user types 6 and presses enter.

What value gets stored in result?

```python
user_num = int(input("Enter a number: "))
result = 5 % (3 ** (user_num // 4))
result = 5 % (3 ** (6 // 4))
result = 5 % (3 ** 1)
```
Suppose we run the following program, and the user types 6 and presses enter.

What value gets stored in `result`?

```python
user_num = int(input("Enter a number: "))
result = 5 % (3 ** (user_num // 4))
result = 5 % (3 ** ( 6    // 4))
result = 5 % (3 **       1)
result = 5 % (      3   )
```

Practice Problem: Operators
Suppose we run the following program, and the user types 6 and presses enter.

What value gets stored in result?

```python
user_num = int(input("Enter a number: "))
result = 5 % (3 ** (user_num // 4))
result = 5 % (3 ** (6 // 4))
result = 5 % (3 ** 1)
result = 5 % 3
result = 2
```
Order of Operations

We know parenthesized expressions get evaluated from inside to out. Are there any other rules?

What if we took the parentheses out:

\[
\text{result} = 5 \% (3 \** (6 \// 4))
\]

\[
\text{result} = 5 \% 3 \** 6 \// 4
\]
Order of Operations

We know parenthesized expressions get evaluated from inside to out. Are there any other rules? Yes: operator precedence.

Remember PEMDAS? BIDMAS? BODMAS?

- Parentheses
- Exponentiation
- Multiplication and Division
- Addition and Subtraction

precedence
Order of Operations

We know parenthesized expressions get evaluated from inside to out. Are there any other rules? Yes: **operator precedence**.

Remember PEMDAS? BIDMAS? BODMAS?

<table>
<thead>
<tr>
<th>Precedence</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parentheses</td>
<td>Exponentiation</td>
</tr>
<tr>
<td>Multiplication and Division</td>
<td>Addition and Subtraction</td>
</tr>
</tbody>
</table>

Example: $10 \times 6 \times 2 / 5 // 11 - 4$
Order of Operations

We know parenthesized expressions get evaluated from inside to out. Are there any other rules? Yes: operator precedence.

Remember PEMDAS? BIDMAS? BODMAS?

1. Parentheses
2. Exponentiation
3. Multiplication and Division (left-to-right)
4. Addition and Subtraction (left-to-right)

Example: \(10 \times 6^{2} / 5 \div 11 - 4\)
Order of Operations

We know parenthesized expressions get evaluated from inside to out. Are there any other rules? Yes: operator precedence.

Remember PEMDAS? BIDMAS? BODMAS?

Example:

- Parentheses
- Exponentiation
- Multiplication and Division (left-to-right)
- Addition and Subtraction (left-to-right)
Order of Operations

We know parenthesized expressions get evaluated from inside to out. Are there any other rules? Yes: **operator precedence**.

Remember PEMDAS? BIDMAS? BODMAS?

- Parentheses
- Exponentiation *(right-to-left)*
- Multiplication and Division *(left-to-right)*
- Addition and Subtraction *(left-to-right)*
Questions?
Some more familiar operators

<  Less than
>  Greater than
<=  Less than or equal to
>=  Greater than or equal to
==
!=

These ones do what you think.

3 < 4
4 <= 4
6.7 > 6.3
1000 >= 1000

What does 3 < 4 evaluate to?
What does type(3 < 4) evaluate to?
We need a new data type!

\[ a < b \]

can only be one of two things: a **true** statement or a **false** statement.

**Boolean expressions** are expressions that evaluate to one of two possible values: **True** or **False**

What does \( 3 < 4 \) evaluate to? **True**

What does \( \text{type}(3 < 4) \) evaluate to? **bool**
The bool data type

• Named after 19th century philosopher/mathematician George Boole, who developed Boolean algebra

• A boolean value (bool) represents logical propositions that can be either true or false.

• In Python, these values are reserved keywords: True and False. Note capitalization.

• Can be used for things like 3 < 4 or a < b, but also anything else that can be true or false:

   isRaining = False
Some more familiar operators

<  Less than
>  Greater than
\leq  Less than or equal to
\geq  Greater than or equal to
==  Equal to
!=  Not equal to

What does 3 == 4 evaluate to?

A. False
B. True
C. 7
D. None of the above
Some more familiar operators

Less than  
Greater than  
Less than or equal to  
Greater than or equal to  
Equal to  
Not equal to

What does $5 \neq 4$ evaluate to?

A. False
B. True
C. 7
D. None of the above
Some more familiar operators

<  Less than
>  Greater than
\leq  Less than or equal to
\geq  Greater than or equal to
==  Equal to
!=  Not equal to

What does \(16 = 4 \times 4\) evaluate to?

A. False
B. True
C. 7
D. None of the above

A classic mistake: mixing up = and ==
Some more familiar operators

<  Less than
>  Greater than
<= Less than or equal to
>= Greater than or equal to
== Equal to
!= Not equal to

a and b is true only when both a and b evaluate to True

a or b is true when at least one of a and b evaluates to True

not switches the value:
not True => False
not False => True

and logical conjunction, logical and
or logical disjunction, logical or
not logical negation, logical not
Binary vs Unary Operators

• We have already seen some binary operators and one unary operator.

• **Binary operators** take two operands:
  
  ```
  a + b  
  c // d  
  12 != 4 
  ```

• **Unary operators** take one operand:
  
  ```
  -b  
  not False 
  ```

Notice: minus (—) can behave as a unary **or** binary operator!
Truth Tables for and, or

If $x$ is true and $y$ is true, $x$ and $y$ is true.

If $x$ is true and $y$ is false, $x$ and $y$ is false.

If $x$ is true and $y$ is true, $x$ and $y$ is true.
Truth Tables for and, or

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>x and y</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>x or y</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Operator Precedence, Updated

Parentheses

Exponentiation (right-to-left)

Multiplication and Division

Addition and Subtraction

Numerical comparisons <, >, <=, >=, ==, !=

not

and

or

All are evaluated left to right except for exponentiation.
Examples

print(3 ! = 5 and 4 < 7)
    => True and True => True

print(3 == 5 or 4 < 7)
    => False or True => True

print(not False)
    => True

print(3 == 5 or 4 > 7)
    => False or False => False

print(not 6 < 8)
    => not True => False
Evaluate This

1 == 6 and True or (1.2 < (5 % 3))

A. False
B. True
C. 16
D. None of the above
Evaluate This

1 == 6 and True or (1.2 < (5 % 3))
1 == 6 and True or (1.2 < 2)

False and True or True

False or True

True
Next Time: if statements

Conditionals: making decisions about what code to execute based on the value of a boolean expression