CSCI 141

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

• A1 is due tonight!
Announcements

• A1 is due tonight!
  • Please name your files as described. There are 200 of you.
Announcements

• A1 is due tonight!

  • Please name your files as described. There are 200 of you.

  • arithmetic.py will break our scripts and require manual handling
Announcements

• A1 is due tonight!

• Please name your files as described. There are 200 of you.

  • arithmetic.py will break our scripts and require manual handling

  • arithmetic-2.py will not break our scripts - you can resubmit and the latest version will be graded.
Announcements

• A1 is due tonight!
  
  • Please name your files as described. There are 200 of you.
    
      • arithmetic.py will break our scripts and require manual handling
    
      • arithmetic-2.py will not break our scripts - you can resubmit and the latest version will be graded.
  
• Canvas Submission comments: I don't read them. Your TA may not either.
Announcements

• A1 is due tonight!
  
  • Please name your files as described. There are 200 of you.
    
    • arithmetic.py will break our scripts and require manual handling
    
    • arithmetic-2.py will not break our scripts - you can resubmit and the latest version will be graded.

• Canvas Submission comments: I don't read them. Your TA may not either.
  
  • If you need to communicate something to me or your TA, send it by email or Canvas message.
Announcements

• A1 is due tonight!
  
  • Please name your files as described. There are 200 of you.
    
    • arithmetic.py **will** break our scripts and require manual handling
    
    • arithmetic-2.py **will not** break our scripts - you can resubmit and the latest version will be graded.

• Canvas Submission comments: I don't read them. Your TA may not either.
  
  • If you need to communicate something to me or your TA, send it by email or Canvas message.

• A2 will be out tomorrow. Due next Monday night.
Goals

• 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`.
What does the following program print? Be sure to write the result exactly as Python would print it out.

```python
a = 31
b = a // 4
c = (5 % b) - 1.0
print(a, b, c, sep=" ", end="!")
```
QOTD

The table lists the first few powers of two, their values, and the binary digits in the representation of the decimal number 19.

- In the second column, fill in the missing powers of two.
- In the third column, fill in the remaining digits of the binary representation of the decimal number 19.

<table>
<thead>
<tr>
<th>Power</th>
<th>Value</th>
<th>Binary Digit:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^5$</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>$2^4$</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>$2^3$</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>$2^2$</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>$2^1$</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$2^0$</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
The table lists the first few powers of two, their values, and the binary digits in the representation of the decimal number 19.

- In the second column, fill in the missing powers of two.
- In the third column, fill in the remaining digits of the binary representation of the decimal number 19.
The table lists the first few powers of two, their values, and the binary digits in the representation of the decimal number 19.

- In the second column, fill in the missing powers of two.
- In the third column, fill in the remaining digits of the binary representation of the decimal number 19.

<table>
<thead>
<tr>
<th>Power</th>
<th>Value</th>
<th>Binary Digit:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^5$</td>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>$2^4$</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>$2^3$</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>$2^2$</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>$2^1$</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$2^0$</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
The table lists the first few powers of two, their values, and the binary digits in the representation of the decimal number 19.

- In the second column, fill in the missing powers of two.
- In the third column, fill in the remaining digits of the binary representation of the decimal number 19.

<table>
<thead>
<tr>
<th>Power</th>
<th>Value</th>
<th>Binary Digit:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^5$</td>
<td>32</td>
<td>*</td>
</tr>
<tr>
<td>$2^4$</td>
<td>16</td>
<td>*</td>
</tr>
<tr>
<td>$2^3$</td>
<td>8</td>
<td>*</td>
</tr>
<tr>
<td>$2^2$</td>
<td>4</td>
<td>*</td>
</tr>
<tr>
<td>$2^1$</td>
<td>2</td>
<td>*</td>
</tr>
<tr>
<td>$2^0$</td>
<td>+1</td>
<td>*</td>
</tr>
</tbody>
</table>
The table lists the first few powers of two, their values, and the binary digits in the representation of the decimal number 19.

- In the second column, fill in the missing powers of two.
- In the third column, fill in the remaining digits of the binary representation of the decimal number 19.

<table>
<thead>
<tr>
<th>Power</th>
<th>Value</th>
<th>Binary Digit:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^5$</td>
<td>32</td>
<td>*</td>
</tr>
<tr>
<td>$2^4$</td>
<td>16</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>$2^3$</td>
<td>8</td>
<td>*</td>
</tr>
<tr>
<td>$2^2$</td>
<td>4</td>
<td>*</td>
</tr>
<tr>
<td>$2^1$</td>
<td>2</td>
<td>*</td>
</tr>
<tr>
<td>$2^0$</td>
<td>+1</td>
<td>*</td>
</tr>
</tbody>
</table>

\[ 16 + 4 + 2 + 1 = 19 \]
The table lists the first few powers of two, their values, and the binary digits in the representation of the decimal number 19.

- In the second column, fill in the missing powers of two.
- In the third column, fill in the remaining digits of the binary representation of the decimal number 19.

<table>
<thead>
<tr>
<th>Power</th>
<th>Value</th>
<th>Binary Digit:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^5$</td>
<td>32</td>
<td>*</td>
</tr>
<tr>
<td>$2^4$</td>
<td>16</td>
<td>* 1 (3 left)</td>
</tr>
<tr>
<td>$2^3$</td>
<td>8</td>
<td>*</td>
</tr>
<tr>
<td>$2^2$</td>
<td>4</td>
<td>*</td>
</tr>
<tr>
<td>$2^1$</td>
<td>2</td>
<td>*</td>
</tr>
<tr>
<td>$2^0$</td>
<td>+ 1</td>
<td>*</td>
</tr>
</tbody>
</table>

= 19
The table lists the first few powers of two, their values, and the binary digits in the representation of the decimal number 19.

- In the second column, fill in the missing powers of two.
- In the third column, fill in the remaining digits of the binary representation of the decimal number 19.
The table lists the first few powers of two, their values, and the binary digits in the representation of the decimal number 19.

- In the second column, fill in the missing powers of two.
- In the third column, fill in the remaining digits of the binary representation of the decimal number 19.

\[
\begin{array}{|c|c|c|}
\hline
\text{Power} & \text{Value} & \text{Binary Digit:} \\
\hline
2^5 & 32 & * \\
\hline
2^4 & 16 & * & 1 \\
\hline
2^3 & 8 & * & 0 \\
\hline
2^2 & 4 & * & 0 \\
\hline
2^1 & 2 & * \\
\hline
2^0 & 1 & * \\
\hline
\end{array}
\]
The table lists the first few powers of two, their values, and the binary digits in the representation of the decimal number 19.

- In the second column, fill in the missing powers of two.
- In the third column, fill in the remaining digits of the binary representation of the decimal number 19.
The table lists the first few powers of two, their values, and the binary digits in the representation of the decimal number 19.

- In the second column, fill in the missing powers of two.
- In the third column, fill in the remaining digits of the binary representation of the decimal number 19.

<table>
<thead>
<tr>
<th>Power</th>
<th>Value</th>
<th>Binary Digit:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^5$</td>
<td>32</td>
<td>*</td>
</tr>
<tr>
<td>$2^4$</td>
<td>16</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(3 left)</td>
</tr>
<tr>
<td>$2^3$</td>
<td>8</td>
<td>*</td>
</tr>
<tr>
<td>$2^2$</td>
<td>4</td>
<td>*</td>
</tr>
<tr>
<td>$2^1$</td>
<td>2</td>
<td>*</td>
</tr>
<tr>
<td>$2^0$</td>
<td>+ 1</td>
<td>*</td>
</tr>
</tbody>
</table>

\[\text{\(=\) 19}\]
The table lists the first few powers of two, their values, and the binary digits in the representation of the decimal number 19.

- In the second column, fill in the missing powers of two.
- In the third column, fill in the remaining digits of the binary representation of the decimal number 19.

<table>
<thead>
<tr>
<th>Power</th>
<th>Value</th>
<th>Binary Digit:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^5$</td>
<td>32</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>$2^4$</td>
<td>16</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>(3 left)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$2^3$</td>
<td>8</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(3 left)</td>
</tr>
<tr>
<td>$2^2$</td>
<td>4</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(1 left)</td>
</tr>
<tr>
<td>$2^1$</td>
<td>2</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>(1 left)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$2^0$</td>
<td>+1</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

= 19
The table lists the first few powers of two, their values, and the binary digits in the representation of the decimal number 19.

- In the second column, fill in the missing powers of two.
- In the third column, fill in the remaining digits of the binary representation of the decimal number 19.

<table>
<thead>
<tr>
<th>Power</th>
<th>Value</th>
<th>Binary Digit:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^5$</td>
<td>32</td>
<td>*</td>
</tr>
<tr>
<td>$2^4$</td>
<td>16</td>
<td>* 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(3 left)</td>
</tr>
<tr>
<td>$2^3$</td>
<td>8</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>$2^2$</td>
<td>4</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>$2^1$</td>
<td>2</td>
<td>* 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(1 left)</td>
</tr>
<tr>
<td>$2^0$</td>
<td>+1</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0 left)</td>
</tr>
</tbody>
</table>

= 19
The third column of the following table contains the binary representation for what decimal number?

<table>
<thead>
<tr>
<th>Power</th>
<th>Value</th>
<th>Binary Digit:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^3$</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>$2^2$</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>$2^1$</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>$2^0$</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
The third column of the following table contains the binary representation for what decimal number?

<table>
<thead>
<tr>
<th>Power</th>
<th>Value</th>
<th>Binary Digit:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^3$</td>
<td>8</td>
<td>* 1</td>
</tr>
<tr>
<td>$2^2$</td>
<td>4</td>
<td>* 0</td>
</tr>
<tr>
<td>$2^1$</td>
<td>2</td>
<td>* 1</td>
</tr>
<tr>
<td>$2^0$</td>
<td>+1</td>
<td>* 0</td>
</tr>
</tbody>
</table>

=
The third column of the following table contains the binary representation for what decimal number?

<table>
<thead>
<tr>
<th>Power</th>
<th>Value</th>
<th>Binary Digit:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^3$</td>
<td>8</td>
<td>* 1</td>
</tr>
<tr>
<td>$2^2$</td>
<td>4</td>
<td>* 0</td>
</tr>
<tr>
<td>$2^1$</td>
<td>2</td>
<td>* 1</td>
</tr>
<tr>
<td>$2^0$</td>
<td>+1</td>
<td>* 0</td>
</tr>
</tbody>
</table>

= 10
QOTD

• Suppose the variable a contains a positive integer. Write a single call to the print function that produces the binary representation of $2^a - 1$. For example, if a is 3, the program should print the binary representation of $2^3 - 1 = 7$. You may print the binary representation without any leading zeros.

• Hint: the binary representation of $2^a - 1$ has a special property - try out a few examples of a to get a feel for it.
Last time...
Last time...

- `str + str` performs string **concatenation**
Last time...

• `str + str` performs string concatenation

  "Bat" + "man" => "Batman"
Last time...

- `str + str` performs string concatenation
  "Bat" + "man" => "Batman"
- `str * int` performs string repetition:
Last time...

- `str + str` performs string **concatenation**
  "Bat" + "man" => "Batman"
- `str * int` performs string repetition:
  " toy boat " * 5
Last time...

- `str + str` performs string concatenation:
  
  "Bat" + "man" => "Batman"

- `str * int` performs string repetition:
  
  "toy boat" * 5
  => "toy boat toy boat toy boat toy boat toy boat toy boat toy boat toy boat toy boat"
Last time...

• `str + str` performs string concatenation
  
  "Bat" + "man" => "Batman"

• `str * int` performs string repetition:
  
  " toy boat " * 5

  => "toy boat toy boat toy boat toy boat toy boat"

• Operator precedence (PEMDAS)
Last time...

• `str + str` performs string concatenation:
  
  "Bat" + "man" => "Batman"

• `str * int` performs string repetition:

  " toy boat " * 5

 => "toy boat toy boat toy boat toy boat toy boat"

• Operator precedence (PEMDAS)

• How integers are represented on a computer: Converting between binary and decimal.
I showed you how an int is stored.

• What about str and float?
How do you store strings?

A *str* is a sequence of letters (or characters).

1. Agree by convention on a number that represents each character.
2. Convert that number to binary.
3. Store a sequence of those numbers to form a string.
How do you store strings?

Various conventions exist:
- ASCII, Unicode

A `str` is a sequence of letters (or characters).

1. Agree by convention on a number that represents each character.
2. Convert that number to binary.
3. Store a sequence of those numbers to form a string.
How do you store strings?

### ASCII TABLE

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Hex</th>
<th>Char</th>
<th>Decimal</th>
<th>Hex</th>
<th>Char</th>
<th>Decimal</th>
<th>Hex</th>
<th>Char</th>
<th>Decimal</th>
<th>Hex</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>[NULL]</td>
<td>32</td>
<td>20</td>
<td>[SPACE]</td>
<td>64</td>
<td>40</td>
<td>@</td>
<td>96</td>
<td>60</td>
<td>`</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>[START OF HEADING]</td>
<td>33</td>
<td>21</td>
<td>!</td>
<td>65</td>
<td>41</td>
<td>A</td>
<td>97</td>
<td>61</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>[START OF TEXT]</td>
<td>34</td>
<td>22</td>
<td>&quot;</td>
<td>66</td>
<td>42</td>
<td>B</td>
<td>98</td>
<td>62</td>
<td>b</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>[END OF TEXT]</td>
<td>35</td>
<td>23</td>
<td>#</td>
<td>67</td>
<td>43</td>
<td>C</td>
<td>99</td>
<td>63</td>
<td>c</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>[END OF TRANSMISSION]</td>
<td>36</td>
<td>24</td>
<td>$</td>
<td>68</td>
<td>44</td>
<td>D</td>
<td>100</td>
<td>64</td>
<td>d</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>[ENQUIRY]</td>
<td>37</td>
<td>25</td>
<td>%</td>
<td>69</td>
<td>45</td>
<td>E</td>
<td>101</td>
<td>65</td>
<td>e</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>[ACKNOWLEDGE]</td>
<td>38</td>
<td>26</td>
<td>&amp;</td>
<td>70</td>
<td>46</td>
<td>F</td>
<td>102</td>
<td>66</td>
<td>f</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>[BEL]</td>
<td>39</td>
<td>27</td>
<td>'</td>
<td>71</td>
<td>47</td>
<td>G</td>
<td>103</td>
<td>67</td>
<td>g</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>[BACKSPACE]</td>
<td>40</td>
<td>28</td>
<td>(</td>
<td>72</td>
<td>48</td>
<td>H</td>
<td>104</td>
<td>68</td>
<td>h</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>[HORIZONTAL TAB]</td>
<td>41</td>
<td>29</td>
<td>)</td>
<td>73</td>
<td>49</td>
<td>I</td>
<td>105</td>
<td>69</td>
<td>i</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>[VERTICAL TAB]</td>
<td>43</td>
<td>2B</td>
<td>+</td>
<td>75</td>
<td>4B</td>
<td>K</td>
<td>107</td>
<td>6B</td>
<td>k</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>[FORM FEED]</td>
<td>44</td>
<td>2C</td>
<td>,</td>
<td>76</td>
<td>4C</td>
<td>L</td>
<td>108</td>
<td>6C</td>
<td>l</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>[CARRIAGE RETURN]</td>
<td>45</td>
<td>2D</td>
<td>-</td>
<td>77</td>
<td>4D</td>
<td>M</td>
<td>109</td>
<td>6D</td>
<td>m</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>[SHIFT OUT]</td>
<td>46</td>
<td>2E</td>
<td>.</td>
<td>78</td>
<td>4E</td>
<td>N</td>
<td>110</td>
<td>6E</td>
<td>n</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>[SHIFT IN]</td>
<td>47</td>
<td>2F</td>
<td>/</td>
<td>79</td>
<td>4F</td>
<td>O</td>
<td>111</td>
<td>6F</td>
<td>o</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>[DATA LINK ESCAPE]</td>
<td>48</td>
<td>30</td>
<td>0</td>
<td>80</td>
<td>50</td>
<td>P</td>
<td>112</td>
<td>70</td>
<td>p</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>[DEVICE CONTROL 1]</td>
<td>49</td>
<td>31</td>
<td>1</td>
<td>81</td>
<td>51</td>
<td>Q</td>
<td>113</td>
<td>71</td>
<td>q</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>[DEVICE CONTROL 2]</td>
<td>50</td>
<td>32</td>
<td>2</td>
<td>82</td>
<td>52</td>
<td>R</td>
<td>114</td>
<td>72</td>
<td>r</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>[DEVICE CONTROL 3]</td>
<td>51</td>
<td>33</td>
<td>3</td>
<td>83</td>
<td>53</td>
<td>S</td>
<td>115</td>
<td>73</td>
<td>s</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>[DEVICE CONTROL 4]</td>
<td>52</td>
<td>34</td>
<td>4</td>
<td>84</td>
<td>54</td>
<td>T</td>
<td>116</td>
<td>74</td>
<td>t</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>[NEGATIVE ACKNOWLEDGE]</td>
<td>53</td>
<td>35</td>
<td>5</td>
<td>85</td>
<td>55</td>
<td>U</td>
<td>117</td>
<td>75</td>
<td>u</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>[SYNCHRONOUS IDLE]</td>
<td>54</td>
<td>36</td>
<td>6</td>
<td>86</td>
<td>56</td>
<td>V</td>
<td>118</td>
<td>76</td>
<td>v</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>[ENG. OF TRANS. BLOCK]</td>
<td>55</td>
<td>37</td>
<td>7</td>
<td>87</td>
<td>57</td>
<td>W</td>
<td>119</td>
<td>77</td>
<td>w</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>[CANCEL]</td>
<td>56</td>
<td>38</td>
<td>8</td>
<td>88</td>
<td>58</td>
<td>X</td>
<td>120</td>
<td>78</td>
<td>x</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>[END OF MEDIUM]</td>
<td>57</td>
<td>39</td>
<td>9</td>
<td>89</td>
<td>59</td>
<td>Y</td>
<td>121</td>
<td>79</td>
<td>y</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>[SUBSTITUTE]</td>
<td>58</td>
<td>3A</td>
<td>:</td>
<td>90</td>
<td>5A</td>
<td>Z</td>
<td>122</td>
<td>7A</td>
<td>z</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>[ESCAPE]</td>
<td>59</td>
<td>3B</td>
<td>;</td>
<td>91</td>
<td>5B</td>
<td>[</td>
<td>123</td>
<td>7B</td>
<td>`{</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>[FILE SEPARATOR]</td>
<td>60</td>
<td>3C</td>
<td>&lt;</td>
<td>92</td>
<td>5C</td>
<td>\</td>
<td>124</td>
<td>7C</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>[GROUP SEPARATOR]</td>
<td>61</td>
<td>3D</td>
<td>=</td>
<td>93</td>
<td>5D</td>
<td>]</td>
<td>125</td>
<td>7D</td>
<td>\</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>[RECORD SEPARATOR]</td>
<td>62</td>
<td>3E</td>
<td>&gt;</td>
<td>94</td>
<td>5E</td>
<td>^</td>
<td>126</td>
<td>7E</td>
<td>~</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>[UNIT SEPARATOR]</td>
<td>63</td>
<td>3F</td>
<td>?</td>
<td>95</td>
<td>5F</td>
<td>_</td>
<td>127</td>
<td>7F</td>
<td>[DEL]</td>
</tr>
<tr>
<td>Decimal</td>
<td>Hex</td>
<td>Char</td>
<td>Decimal</td>
<td>Hex</td>
<td>Char</td>
<td>Decimal</td>
<td>Hex</td>
<td>Char</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td>------</td>
<td>---------</td>
<td>------</td>
<td>------</td>
<td>---------</td>
<td>------</td>
<td>------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>20</td>
<td>SPACE</td>
<td>64</td>
<td>40</td>
<td>@</td>
<td>96</td>
<td>60</td>
<td>`</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>21</td>
<td>!</td>
<td>65</td>
<td>41</td>
<td>A</td>
<td>97</td>
<td>61</td>
<td>a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>22</td>
<td>&quot;</td>
<td>66</td>
<td>42</td>
<td>B</td>
<td>98</td>
<td>62</td>
<td>b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>23</td>
<td>#</td>
<td>67</td>
<td>43</td>
<td>C</td>
<td>99</td>
<td>63</td>
<td>c</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>24</td>
<td>$</td>
<td>68</td>
<td>44</td>
<td>D</td>
<td>100</td>
<td>64</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>25</td>
<td>%</td>
<td>69</td>
<td>45</td>
<td>E</td>
<td>101</td>
<td>65</td>
<td>e</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>26</td>
<td>&amp;</td>
<td>70</td>
<td>46</td>
<td>F</td>
<td>102</td>
<td>66</td>
<td>f</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>27</td>
<td>'</td>
<td>71</td>
<td>47</td>
<td>G</td>
<td>103</td>
<td>67</td>
<td>g</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>28</td>
<td>(</td>
<td>72</td>
<td>48</td>
<td>H</td>
<td>104</td>
<td>68</td>
<td>h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>29</td>
<td>)</td>
<td>73</td>
<td>49</td>
<td>I</td>
<td>105</td>
<td>69</td>
<td>i</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>2A</td>
<td>*</td>
<td>74</td>
<td>4A</td>
<td>J</td>
<td>106</td>
<td>6A</td>
<td>j</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>2B</td>
<td>+</td>
<td>75</td>
<td>4B</td>
<td>K</td>
<td>107</td>
<td>6B</td>
<td>k</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>2C</td>
<td>,</td>
<td>76</td>
<td>4C</td>
<td>L</td>
<td>108</td>
<td>6C</td>
<td>l</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>2D</td>
<td>-</td>
<td>77</td>
<td>4D</td>
<td>M</td>
<td>109</td>
<td>6D</td>
<td>m</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>2E</td>
<td>.</td>
<td>78</td>
<td>4E</td>
<td>N</td>
<td>110</td>
<td>6E</td>
<td>n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>2F</td>
<td>/</td>
<td>79</td>
<td>4F</td>
<td>O</td>
<td>111</td>
<td>6F</td>
<td>o</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>30</td>
<td>0</td>
<td>80</td>
<td>50</td>
<td>P</td>
<td>112</td>
<td>70</td>
<td>p</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>31</td>
<td>1</td>
<td>81</td>
<td>51</td>
<td>Q</td>
<td>113</td>
<td>71</td>
<td>q</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>32</td>
<td>2</td>
<td>82</td>
<td>52</td>
<td>R</td>
<td>114</td>
<td>72</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>33</td>
<td>3</td>
<td>83</td>
<td>53</td>
<td>S</td>
<td>115</td>
<td>73</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>34</td>
<td>4</td>
<td>84</td>
<td>54</td>
<td>T</td>
<td>116</td>
<td>74</td>
<td>t</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>35</td>
<td>5</td>
<td>85</td>
<td>55</td>
<td>U</td>
<td>117</td>
<td>75</td>
<td>u</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>36</td>
<td>6</td>
<td>86</td>
<td>56</td>
<td>V</td>
<td>118</td>
<td>76</td>
<td>v</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>37</td>
<td>7</td>
<td>87</td>
<td>57</td>
<td>W</td>
<td>119</td>
<td>77</td>
<td>w</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>38</td>
<td>8</td>
<td>88</td>
<td>58</td>
<td>X</td>
<td>120</td>
<td>78</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>39</td>
<td>9</td>
<td>89</td>
<td>59</td>
<td>Y</td>
<td>121</td>
<td>79</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>3A</td>
<td>:</td>
<td>90</td>
<td>5A</td>
<td>Z</td>
<td>122</td>
<td>7A</td>
<td>z</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>3B</td>
<td>;</td>
<td>91</td>
<td>5B</td>
<td>[</td>
<td>123</td>
<td>7B</td>
<td>{</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>3C</td>
<td>&lt;</td>
<td>92</td>
<td>5C</td>
<td>\</td>
<td>124</td>
<td>7C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>3D</td>
<td>=</td>
<td>93</td>
<td>5D</td>
<td>]</td>
<td>125</td>
<td>7D</td>
<td>}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decimal</td>
<td>Hex</td>
<td>Char</td>
<td>Decimal</td>
<td>Hex</td>
<td>Char</td>
<td>Decimal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-----</td>
<td>---------------</td>
<td>---------</td>
<td>-----</td>
<td>---------------</td>
<td>---------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>[NULL]</td>
<td>32</td>
<td>20</td>
<td>[SPACE]</td>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>[START OF HEADING]</td>
<td>33</td>
<td>21</td>
<td>!</td>
<td>65</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>[START OF TEXT]</td>
<td>34</td>
<td>22</td>
<td>&quot;</td>
<td>66</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>[END OF TEXT]</td>
<td>35</td>
<td>23</td>
<td>#</td>
<td>67</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>[END OF TRANSMISSION]</td>
<td>36</td>
<td>24</td>
<td>$</td>
<td>68</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>[ENQUIRY]</td>
<td>37</td>
<td>25</td>
<td>%</td>
<td>69</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>[ACKNOWLEDGE]</td>
<td>38</td>
<td>26</td>
<td>&amp;</td>
<td>70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>[BELL]</td>
<td>39</td>
<td>27</td>
<td>^</td>
<td>71</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>[BACKSPACE]</td>
<td>40</td>
<td>28</td>
<td>(</td>
<td>72</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>[HORIZONTAL TAB]</td>
<td>41</td>
<td>29</td>
<td>)</td>
<td>73</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>[LINE FEED]</td>
<td>42</td>
<td>2A</td>
<td>-</td>
<td>74</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>[VERTICAL TAB]</td>
<td>43</td>
<td>2B</td>
<td>'</td>
<td>75</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>[FORM FEED]</td>
<td>44</td>
<td>2C</td>
<td>,</td>
<td>76</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>[CARRIAGE RETURN]</td>
<td>45</td>
<td>2D</td>
<td>-</td>
<td>77</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>[SHIFT OUT]</td>
<td>46</td>
<td>2E</td>
<td>.</td>
<td>78</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>[SHIFT IN]</td>
<td>47</td>
<td>2F</td>
<td>/</td>
<td>79</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>[DATA LINK ESCAPE]</td>
<td>48</td>
<td>30</td>
<td>0</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>[DEVICE CONTROL 1]</td>
<td>49</td>
<td>31</td>
<td>1</td>
<td>81</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>[DEVICE CONTROL 2]</td>
<td>50</td>
<td>32</td>
<td>2</td>
<td>82</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>[DEVICE CONTROL 3]</td>
<td>51</td>
<td>33</td>
<td>3</td>
<td>83</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>[DEVICE CONTROL 4]</td>
<td>52</td>
<td>34</td>
<td>4</td>
<td>84</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>[NEGATIVE ACKNOWLEDGE]</td>
<td>53</td>
<td>35</td>
<td>5</td>
<td>85</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>[SYNCHRONOUS IDLE]</td>
<td>54</td>
<td>36</td>
<td>6</td>
<td>86</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>[ENG OF TRANS. BLOCK]</td>
<td>55</td>
<td>37</td>
<td>7</td>
<td>87</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>[CANCEL]</td>
<td>56</td>
<td>38</td>
<td>8</td>
<td>88</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>[END OF MEDIUM]</td>
<td>57</td>
<td>39</td>
<td>9</td>
<td>89</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This is `\n`: it's just another character!
That’s how str works.

• What about float?

• It’s harder to write 4.3752 as a sum of powers of two.
That’s how str works.
That’s how \texttt{str} works.

- Floating-point numbers are stored similarly to scientific notation:
That’s how \texttt{str} works.

• Floating-point numbers are stored similarly to scientific notation: $1399.94 = 1.39994 \times 10^3$
That’s how `str` works.

- Floating-point numbers are stored similarly to scientific notation: \( 1399.94 = 1.39994 \times 10^3 \)

- Need to store the base **and** the exponent. In memory, it looks something like this:
That’s how **str** works.

- Floating-point numbers are stored similarly to scientific notation: \(1399.94 = 1.39994 \times 10^3\)
- Need to store the base **and** the exponent. In memory, it looks something like this:
That’s how \texttt{str} works.

- Floating-point numbers are stored similarly to scientific notation: \(1399.94 = 1.39994 \times 10^3\)

- Need to store the base \textbf{and} the exponent. In memory, it looks something like this:
That’s how \texttt{str} works.

- Floating-point numbers are stored similarly to scientific notation: \( 1399.94 = 1.39994 \times 10^3 \)

- Need to store the base \texttt{and} the exponent. In memory, it looks something like this:
That’s how \textit{std} works.

- Floating-point numbers are stored similarly to scientific notation: \( 1399.94 = 1.39994 \times 10^3 \)

- Need to store the base \textbf{and} the exponent. In memory, it looks something like this:

- Base and exponent are represented as base-2 integers, so the precision is finite: not all numbers can be represented!
What have we covered so far?
What have we covered so far?

• Data is stored in memory.
What have we covered so far?

- Data is stored in memory.

  integers are stored using their *binary* representation
What have we covered so far?

• Data is stored in memory.
  integers are stored using their *binary* representation

• Each piece of data has a type.
What have we covered so far?

- Data is stored in memory.
  
  integers are stored using their *binary* representation

- Each piece of data has a type.
  
  so far we’ve seen: `int`, `float`, `str`
What have we covered so far?

• Data is stored in memory.
  
  *integers are stored using their binary representation*

• Each piece of data has a type.
  
  *so far we’ve seen: int, float, str*

• Variables can assign names to pieces of data.
What have we covered so far?

• Data is stored in memory.
  
  integers are stored using their binary representation

• Each piece of data has a type.
  
  so far we’ve seen: int, float, str

• Variables can assign names to pieces of data.
  
  the assignment operator stores a value in a variable, as in:
  
  my_var = "Hello, world!"
What have we covered so far?

• Data is stored in memory.
  
  integers are stored using their *binary* representation

• Each piece of data has a type.
  
  so far we’ve seen: `int`, `float`, `str`

• Variables can assign names to pieces of data.
  
  the assignment operator stores a value in a variable, as in:
  
  `my_var = “Hello, world!”`

• Operators can do things to the data (these operations are performed by the CPU).
What have we covered so far?

- Data is stored in memory.  
  integers are stored using their **binary representation**

- Each piece of data has a type.  
  so far we’ve seen: int, float, str

- Variables can assign names to pieces of data.  
  the assignment operator stores a value in a variable, as in:  
  ```
  my_var = “Hello, world!”
  ```

- Operators can do things to the data (these operations are performed by the CPU).  
  so far: assignment operator (=)  
  arithmetic operators: (+, -, *, /, **, //, %)
What have we covered so far?
What have we covered so far?

- A function can take inputs (arguments) and can produce an output (return value)
What have we covered so far?

• A function can take inputs (arguments) and can produce an output (return value)

• Statements are instructions that are executed
What have we covered so far?

- A function can take inputs (arguments) and can produce an output (return value).

- Statements are instructions that are executed.

- Expressions are like phrases that can be evaluated to determine what value they represent.
What have we covered so far?

• A function can take inputs (arguments) and can produce an output (return value)

• Statements are instructions that are executed

  so far: assignment statements, such as my_var = 64 + 8

• Expressions are like phrases that can be evaluated to determine what value they represent.
What have we covered so far?

- A function can take inputs (arguments) and can produce an output (return value)

- Statements are instructions that are executed
  
  so far: assignment statements, such as `my_var = 64 + 8`

- Expressions are like phrases that can be evaluated to determine what value they represent.
  
  so far:
  - functions that return values, like `int(42.8)`
  - arithmetic expressions, like `(4 + 2) / 2`
  - and combinations of other expressions, like `(2**3) // int(user_input)`
What have we covered so far?

• A function can take inputs (arguments) and can produce an output (return value)
  so far: input, print, type, int, float, str

• Statements are instructions that are executed
  so far: assignment statements, such as my_var = 64 + 8

• Expressions are like phrases that can be evaluated to determine what value they represent.
  so far:
  • functions that return values, like int(42.8)
  • arithmetic expressions, like (4 + 2) / 2
  • and combinations of other expressions, like (2**3) // int(user_input)
Some more familiar operators

<  Less than
>  Greater than
<= Less than or equal to
>= Greater than or equal to
==
!=
Some more familiar operators

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

These ones do what you think.
Some more familiar operators

These ones do what you think.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>==</td>
<td>Equal to</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to</td>
</tr>
</tbody>
</table>

- $3 < 4$
- $4 \leq 4$
- $6.7 > 6.3$
- $1000 \geq 1000$
Some more familiar operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>==</td>
<td>Equal to</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to</td>
</tr>
</tbody>
</table>

These ones do what you think.

\[
\begin{align*}
3 & < 4 \\
4 & \leq 4 \\
6.7 & > 6.3 \\
1000 & \geq 1000
\end{align*}
\]

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

\[ a < b \]

can only be one of two things: a \textbf{true} statement or a \textbf{false} statement.

\textbf{Boolean expressions} are expressions that evaluate to one of two possible values: \texttt{True} or \texttt{False}

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

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

```python
is_raining = False
```
Some more familiar operators

<  Less than
>
>= Greater than or equal to
>= Greater than or equal to
== Equal to
!= Not equal to
Some more familiar operators

<  Less than
>  Greater than
<= Less than or equal to
>= Greater than or equal to
== Equal to
!= Not equal to
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
<=  Less than or equal to
>=  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
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 $16 = 4 \times 4$ 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
and logical conjunction, logical and
or  logical disjunction, logical or
not logical negation, logical not
Some more familiar operators

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

and logical conjunction, logical and
or  logical disjunction, logical or
not logical negation, logical not
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`
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

\textbf{and}  \text{logical conjunction, logical and}
\textbf{or}  \text{logical disjunction, logical or}
\textbf{not}  \text{logical negation, logical not}

$ a \text{ and } b $ is true only when \textbf{both} $ a $ and $ b $ evaluate to \textbf{True}

$ a \text{ or } b $ is true when \textbf{at least} \textbf{one} of $ a $ and $ b $ evaluates to \textbf{True}
Some more familiar operators

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

and  logical conjunction, logical and
or   logical disjunction, logical or
not  logical negation, logical not

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
Binary vs Unary Operators

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

• **Binary operators** take two operands:
  
  \[
  a + b \\
  c \, \div \, d \\
  12 \neq 4 \\
  \]

• **Unary operators** take one operand:

  \[
  -b \\
  \text{not False} \\
  \]
Binary vs Unary Operators

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

- **Binary operators** take two operands:
  
  \[ a + b \]
  \[ c \div d \]
  \[ 12 \neq 4 \]

- **Unary operators** take one operand:
  
  \[-b\]
  \[\text{not } False\]

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>x and y</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>y</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td></td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td></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>
Truth Tables for and, or

If $x$ is true and $y$ is true, $x \text{ and } y$ 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>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

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

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</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)

print(3 == 5 or 4 < 7)

print(not False)

print(3 == 5 or 4 > 7)

print(not 6 < 8)
Examples

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

print(3 == 5 or 4 < 7)

print(not False)

print(3 == 5 or 4 > 7)

print(not 6 < 8)
Examples

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

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

print(not False)

print(3 == 5 or 4 > 7)

print(not 6 < 8)
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)

print(not 6 < 8)
Examples

\[
\text{print}(3 \neq 5 \text{ and } 4 < 7) \\
\Rightarrow \text{ True and True } \Rightarrow \text{ True}
\]

\[
\text{print}(3 == 5 \text{ or } 4 < 7) \\
\Rightarrow \text{ False or True } \Rightarrow \text{ True}
\]

\[
\text{print}(\text{not False}) \\
\Rightarrow \text{ True}
\]

\[
\text{print}(3 == 5 \text{ or } 4 > 7) \\
\Rightarrow \text{ False or False } \Rightarrow \text{ False}
\]

\[
\text{print}(\text{not } 6 < 8)
\]
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))
Evaluate This

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

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

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

1 == 6 and True or (1.2 < 2)
Evaluate This

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

1 == 6 and True or (1.2 < 2)

1 == 6 and True or True
Evaluate This

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

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

1 == 6 and True or (1.2 < 2)

1 == 6 and True or True

False and True or True
Evaluate This

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

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

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

1 == 6 and True or (1.2 < (5 % 3))
1 == 6 and True or (1.2 < 2)
1 == 6 and True or True
False and True or True
False or True
True
Preview: if statements
Next Time: if statements

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