What's a computer's favourite snack?

Microchips!
Last Lecture

• We introduced **branching**
  • A way of controlling the **execution flow** when the program is executed
Today’s Menu

• We’ll continue introducing conditional statements and using them in our Python programs
  • What if there are many conditions (many branches)?
  • Can these conditional statements be nested/chained?
• Also, what if we are dealing with integers?

• We shall also play around Boolean values and Boolean expressions
From last lecture: Review Questions

1. What is wrong with this code?
   
   ```python
   3_colours = ["blue", "green", "white", "pink"]
   ```

2. Why do we want to put an `import` statement at the beginning (top) of a program?

3. When a program executes, what is first executed?

4. Is the `if` part of a conditional statement mandatory?
From last lecture: Review Questions

5. Is the **else** part of a conditional statement mandatory?

6. What is wrong with this code fragment?
   ```python
   if color = "purple":
       print("Cool!")
   ```

7. How could this code be improved?
   ```python
   favBand = input()
   print("Oh, I like this band!")
   ```
Your turn!

• **Step 1 - Problem Statement**
  • Write a login program, which allows a user to login in with a password
Possible Solution

```python
# login.py
#
# Description: Write a login program that allows a user
to login with a password.
#
# Author: Anne Lavergne
# Date: T Jan. 16 2024
#
# Set the actual password
thePassword = "123456"

# Ask user to enter a password
# Get user's password
password = input("Please, enter a password? ")

# Let the user know if the login was successful ...
if password == thePassword:
    print("Successful login!")

# ... or not
else:
    print("Wrong password! Try again!!")
print("----------")
```
Review: Conditional statement

```python
if <boolean expression>:
    ...
else:
    ...
```

- The only thing required in a conditional stmt
- Optional
  Can only have one else per if
Behind the scene!

- What happens when this conditional statement is executed?

```python
if password == thePassword:
```

Boolean expression
## ASCII Table

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Char</th>
<th>Hx</th>
<th>Oct</th>
<th>Char</th>
<th>Hx</th>
<th>Oct</th>
<th>Char</th>
<th>Hx</th>
<th>Oct</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>000</td>
<td>MUL (null)</td>
<td>32</td>
<td>20</td>
<td>040</td>
<td>#32</td>
<td>Space</td>
<td>64</td>
<td>40</td>
<td>100</td>
<td>#64</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>001</td>
<td>SOH (start of heading)</td>
<td>33</td>
<td>21</td>
<td>041</td>
<td>#33</td>
<td>!</td>
<td>65</td>
<td>41</td>
<td>101</td>
<td>#65</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>002</td>
<td>STX (start of text)</td>
<td>34</td>
<td>22</td>
<td>042</td>
<td>#34</td>
<td>&quot;</td>
<td>66</td>
<td>42</td>
<td>102</td>
<td>#66</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>003</td>
<td>ETX (end of text)</td>
<td>35</td>
<td>23</td>
<td>043</td>
<td>#35</td>
<td>#</td>
<td>67</td>
<td>43</td>
<td>103</td>
<td>#67</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>004</td>
<td>EOT (end of transmission)</td>
<td>36</td>
<td>24</td>
<td>044</td>
<td>#36</td>
<td>$</td>
<td>68</td>
<td>44</td>
<td>104</td>
<td>#68</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>005</td>
<td>ENQ (enquiry)</td>
<td>37</td>
<td>25</td>
<td>045</td>
<td>#37</td>
<td>%</td>
<td>69</td>
<td>45</td>
<td>105</td>
<td>#69</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>006</td>
<td>ACK (acknowledge)</td>
<td>38</td>
<td>26</td>
<td>046</td>
<td>#38</td>
<td>&amp;</td>
<td>70</td>
<td>46</td>
<td>106</td>
<td>#70</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>007</td>
<td>BEL (bell)</td>
<td>39</td>
<td>27</td>
<td>047</td>
<td>#39</td>
<td>'</td>
<td>71</td>
<td>47</td>
<td>107</td>
<td>#71</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>010</td>
<td>BS (backspace)</td>
<td>40</td>
<td>28</td>
<td>050</td>
<td>#40</td>
<td>(</td>
<td>72</td>
<td>48</td>
<td>110</td>
<td>#72</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>011</td>
<td>TAB (horizontal tab)</td>
<td>41</td>
<td>29</td>
<td>051</td>
<td>#41</td>
<td>)</td>
<td>73</td>
<td>49</td>
<td>111</td>
<td>#73</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>012</td>
<td>LF (NL line feed, new line)</td>
<td>42</td>
<td>2A</td>
<td>052</td>
<td>#42</td>
<td>\</td>
<td>74</td>
<td>4A</td>
<td>112</td>
<td>#74</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>013</td>
<td>VT (vertical tab)</td>
<td>43</td>
<td>2B</td>
<td>053</td>
<td>#43</td>
<td>+</td>
<td>75</td>
<td>4B</td>
<td>113</td>
<td>#75</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>014</td>
<td>FF (NP form feed, new page)</td>
<td>44</td>
<td>2C</td>
<td>054</td>
<td>#44</td>
<td>,</td>
<td>76</td>
<td>4C</td>
<td>114</td>
<td>#76</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>015</td>
<td>CR (carriage return)</td>
<td>45</td>
<td>2D</td>
<td>055</td>
<td>#45</td>
<td>-</td>
<td>77</td>
<td>4D</td>
<td>115</td>
<td>#77</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>016</td>
<td>SO (shift out)</td>
<td>46</td>
<td>2E</td>
<td>056</td>
<td>#46</td>
<td>.</td>
<td>78</td>
<td>4E</td>
<td>116</td>
<td>#78</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>017</td>
<td>SI (shift in)</td>
<td>47</td>
<td>2F</td>
<td>057</td>
<td>#47</td>
<td>/</td>
<td>79</td>
<td>4F</td>
<td>117</td>
<td>#79</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>020</td>
<td>DLE (data link escape)</td>
<td>48</td>
<td>30</td>
<td>060</td>
<td>#48</td>
<td>0</td>
<td>80</td>
<td>50</td>
<td>120</td>
<td>#80</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>021</td>
<td>DC1 (device control 1)</td>
<td>49</td>
<td>31</td>
<td>061</td>
<td>#49</td>
<td>1</td>
<td>81</td>
<td>51</td>
<td>121</td>
<td>#81</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>022</td>
<td>DC2 (device control 2)</td>
<td>50</td>
<td>32</td>
<td>062</td>
<td>#50</td>
<td>2</td>
<td>82</td>
<td>52</td>
<td>122</td>
<td>#82</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>023</td>
<td>DC3 (device control 3)</td>
<td>51</td>
<td>33</td>
<td>063</td>
<td>#51</td>
<td>3</td>
<td>83</td>
<td>53</td>
<td>123</td>
<td>#83</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>024</td>
<td>DC4 (device control 4)</td>
<td>52</td>
<td>34</td>
<td>064</td>
<td>#52</td>
<td>4</td>
<td>84</td>
<td>54</td>
<td>124</td>
<td>#84</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>025</td>
<td>NAK (negative acknowledge)</td>
<td>53</td>
<td>35</td>
<td>065</td>
<td>#53</td>
<td>5</td>
<td>85</td>
<td>55</td>
<td>125</td>
<td>#85</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>026</td>
<td>SYN (synchronous idle)</td>
<td>54</td>
<td>36</td>
<td>066</td>
<td>#54</td>
<td>6</td>
<td>86</td>
<td>56</td>
<td>126</td>
<td>#86</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>027</td>
<td>ETB (end of trans. block)</td>
<td>55</td>
<td>37</td>
<td>067</td>
<td>#55</td>
<td>7</td>
<td>87</td>
<td>57</td>
<td>127</td>
<td>#87</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>030</td>
<td>CAN (cancel)</td>
<td>56</td>
<td>38</td>
<td>068</td>
<td>#56</td>
<td>8</td>
<td>88</td>
<td>58</td>
<td>130</td>
<td>#88</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>031</td>
<td>EM (end of medium)</td>
<td>57</td>
<td>39</td>
<td>071</td>
<td>#57</td>
<td>9</td>
<td>89</td>
<td>59</td>
<td>131</td>
<td>#89</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>032</td>
<td>SUB (substitute)</td>
<td>58</td>
<td>3A</td>
<td>072</td>
<td>#58</td>
<td>:</td>
<td>90</td>
<td>5A</td>
<td>132</td>
<td>#90</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>033</td>
<td>ESC (escape)</td>
<td>59</td>
<td>3B</td>
<td>073</td>
<td>#59</td>
<td>&lt;</td>
<td>91</td>
<td>5B</td>
<td>133</td>
<td>#91</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>034</td>
<td>FS (file separator)</td>
<td>60</td>
<td>3C</td>
<td>074</td>
<td>#60</td>
<td>&gt;</td>
<td>92</td>
<td>5C</td>
<td>134</td>
<td>#92</td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>035</td>
<td>GS (group separator)</td>
<td>61</td>
<td>3D</td>
<td>075</td>
<td>#61</td>
<td>=</td>
<td>93</td>
<td>5D</td>
<td>135</td>
<td>#93</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>036</td>
<td>RS (record separator)</td>
<td>62</td>
<td>3E</td>
<td>076</td>
<td>#62</td>
<td>?</td>
<td>94</td>
<td>5E</td>
<td>136</td>
<td>#94</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>037</td>
<td>US (unit separator)</td>
<td>63</td>
<td>3F</td>
<td>077</td>
<td>#63</td>
<td>?</td>
<td>95</td>
<td>5F</td>
<td>137</td>
<td>#95</td>
</tr>
</tbody>
</table>

Source: www.LookupTables.com
Let’s try this one!

- **Step 1 - Problem Statement**
  - Write a program that asks the user to enter a positive number and verifies that the user has done so.

- **Step 2 – Design**

- **Step 3 – Implementation**
  - See `positiveNumberFromUser.py`
  - **Step 4 - Testing**
Review: Conditional statement

```python
if <boolean expression>:
...
```

The only thing required in a conditional stmt
Review: Conversion functions

Introducing a few functions:

- `type( ... )`
- `int( ... )`
How about this one?

- **Step 1 - Problem Statement**
  - Write a chatbot that asks a user how their day is going, and makes a comment that changes depending on how the user answered.
Review: Conditional statements

**nested if statement**

```python
if condition 1:
    some statement(s)
else:
    if condition 2:
        some statement(s)
    else:
        some statement(s)
```

**chained if statement**

```python
if condition 1:
    some statement(s)
elif condition 2:
    some statement(s)
else:
    some statement(s)
```
Let’s try this one now!

• **Step 1 - Problem Statement**
  • Write a *grade-to-letter grade converter* that converts a grade into letter grade.
Review: Conditional statement

if <boolean expression>:
    ...

elif <boolean expression>:
    ...

The only thing required in a conditional stmt

Optional
Can be repeated
Review: Conditional statement

```python
if <boolean expression>:
    ...
else:
    ...
```

- **if** is the only required component in a conditional statement.
- **elif** is optional and can be repeated.
- **else** is optional and can only have one per if.

The only thing required in a conditional stmt

Optional
Can be repeated

Optional
Can only have one else per if
How to construct a condition?

**SYNTAX:** `<operand> <operator> <operand>`

**Relational operators** (or comparison operators)

- `<` less than
- `>` greater than
- `<=` less than or equal to
- `>=` greater than or equal to
- `==` equal to
- `!=` not equal to

- Variable
- Literal value
- `->` both of type string, integer or float

**Result of conditional expression:** **True** or **False**

Relational operators (`<`, `>=`, `==`, `!=`) may connect expressions evaluating to different types of values.
How to construct a logical expression (compound condition)?

SYNTAX for

**and** & **or**: <operand> <logical_operator> <operand>

**not**: not <operand>

Logical operators

```
operand

and

or
```

Boolean expression

```
not

operand
```

Result of compound conditional expression: True or False

**Boolean operators** and, **or** must connect two Boolean expressions

**Boolean operator** **not** must be applied to one Boolean expression
How to evaluate a logical expression (compound condition)?

• Boolean truth table:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A and B</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A or B</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>not A</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>
How nested if works

- Conditional Statement – nested if statement

```python
if weekday == 0:
    print("That's Monday!")
else:
    if weekday == 1:
        print("That's Tuesday!")
    else:
        if weekday == 2:
            print("That's Saturday!")
        else:
            if weekday == 3:
                print("That's Sunday!")
            else:
                print("Your number was not between 0 and 3 :( !")
```

"Green" if/else is nested in the "blue" if/else

"Red" if/else is nested in the "green" if/else

"Black" if/else is nested in the "red" if/else
How chained if works

- Conditional Statement – chained if statement

```python
if weekday == 0:
    print("That's Monday!")
elif weekday == 1:
    print("That's Tuesday!")
elif weekday == 2:
    print("That's Saturday!")
elif weekday == 3:
    print("That's Sunday!")
else:
    print("Your number was not between 0 and 3 :( !")
```

"Green" if/elif is chained to the "blue" if/elif

"Red" if/elif is chained to the "green" if/elif

"Black" if/else is chained to the "red" if/elif
Your turn!

• **Step 1 - Problem Statement**
  • Write a guessing game, which allows a user to guess a number between 1 and 10.
Summary

✓ We continued introducing conditional statements and using them in our Python programs
  ✓ When there were many conditions (many branches) to deal with
  ✓ We introduced nested and chained conditional statements
✓ We also introduced conversion functions

✓ We also played around Boolean values and Boolean expressions
Next Lecture

- Feedback from Assignment 0
- How to improving our grade-to-letter grade converter and our guessing game
  - Robustness -> User Validation
  - Efficiency
- Step 4 - Testing and various kinds of errors