Decision Structures & Boolean Logic

CSCI-UA.002
Sequence Structures

- What we have been programming so far is known as a “sequence structure”

- Sequence structures are sets of statements that execute in the order in which they appear.

- Unfortunately not all programs can be written this way, as there are certain times when we need to deviate from a linear structure and adapt our program based on information provided.
Example: Calculating Overtime Pay

- If a worker works more than 40 hours in a week they are entitled to overtime pay.
- Overtime pay is calculated at the rate of 1.5 times the worker’s hourly rate.
- This additional rate is only applied to hours worked above the 40 hour limit.
Example: Calculating Overtime Pay

- **Input:** Hourly rate of pay
- **Input:** Number of hours worked in 1 week

- **Process:** If the hours worked is less than 40, simply multiply hourly rate by hours worked

- **Process:** If the hours worked is greater than 40:
  - Multiply hourly rate by hours worked for 40 hours.
  - Subtract 40 from the total hours to obtain the overtime hours
  - Multiply overtime hours by 1.5 times the rate of pay
  - Add overtime pay to base pay

- **Output:** Total Pay
Example: Calculating Overtime Pay

- Our current Python toolset doesn’t give us the ability to deviate from a linear sequence structure.
The Selection Statement

- A type of nesting structure

- Allows your program to “ask a question” and respond accordingly.

- Simplest form – perform an action only if a certain condition exists

- If the condition is not met, then the action is not performed
The Selection Statement

- In this program we begin by asking a question – “is it cold outside?”
- If the answer to this question is yes (aka “True”) then we can execute an alternate set of commands.
- Otherwise we can continue with the program as-is.
The Selection Statement

Start

Cold Outside?

- True
  - Put on a coat
  - Wear a hat
  - Put on gloves

- False
  - Finish
Programming Challenge: Calculating Overtime Pay

- If a worker works more than 40 hours in a week he or she is entitled to overtime pay.

- Overtime pay is calculated at the rate of 1.5 times the worker’s hourly rate.

- This additional rate is only applied to hours worked above the 40 hour limit.
hourlyRate = float(input("Hourly rate of pay?"))
numberOfHours = float(input("Number of Hours worked this week?"))

if numberOfHours < 40:
    print("your pay is", numberOfHours * hourlyRate)
if numberOfHours > 40:
    print("your pay is", (hourlyRate * 1.5) * (numberOfHours - 40) + hourlyRate)
Boolean Expressions
Writing a condition

- The trick to writing a selection statement is in constructing a condition that matches the question you are trying to ask the computer.

- All selection statements must have a condition to "test".

- Think of conditions as "yes or no" questions. They can only be answered by one of two options – "True" or "False".
Boolean Expressions

**True or False**

```python
if condition:
    statement
    statement
    statement
```
hourlyRate = float(input("Hourly rate of pay?"))
numberOfHours = float(input("Number of Hours worked this week?"))

if numberOfHours < 40:
    print("your pay is", numberOfHours * hourlyRate)
if numberOfHours > 40:
    print("your pay is", (hourlyRate * 1.5) * (numberOfHours - 40) + hourlyRate)
Boolean Expressions

- Named after George Boole, a 19th century English philosopher and mathematician
- Boole developed a system of mathematics that allows us to work with the abstract concepts of “true” and “false”
- Boole is considered one of the founders of modern computer science, as his work underpins the way in which modern computers process binary data
Writing a Boolean Expression

- Boolean expressions can be used as the condition in an “if” statement.
- They are generally formed using “relational operators” which allow you to test to see whether a specific relationship exists between two (or more) values.
Relational Operators

a > b  # is a greater than b ?

a < b  # is a less than b ?

a == b  # is a equal to b ?

a <= b  # is a less than OR
        # equal to b ?

a >= b  # is a greater than OR
        # equal to b ?
Writing a Boolean Expression

- ALL Boolean expressions boil down to “True” or “False”

- Programmers often say that the expression “evaluates” to “True” or “False”
Writing a Boolean Expression

```python
ten = 10
sword = 7

if pen > sword:
    print ('the pen is mightier than the sword!'
```
Let’s Evaluate!

# given these variables

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>99</td>
<td>7</td>
<td>-5</td>
<td>92</td>
</tr>
</tbody>
</table>

# evaluate these expressions

<table>
<thead>
<tr>
<th>a &gt; b</th>
<th>b &lt; c</th>
<th>b &gt;= c</th>
<th>c &lt;= d</th>
<th>a == b + d</th>
<th>d &lt;= a + c</th>
<th>c != b</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Boolean Operator Tips

- Don’t confuse “==” with “=”
  - “=” is used for assigning values to variables
  - “==” is used for testing to see if two values are identical
- Use “!” if you want to test if two values are different
- The “<“ and “>” operators test for more than one relationship
  - “<“ tests to see if a value is less than OR equal to another
  - “>“ tests to see if a value is greater than OR equal to another
Let’s write some programs!
Programming Challenge: Fish Tank Cooling System

- Guppies are hardy fish, but they can’t live in all water temperatures.
- The acceptable range for guppies is between 72 and 86 degrees Fahrenheit.
- Write a program that asks the user for a temperature. Then display one of two messages based on the information provided:
  - You’re going to freeze your guppies!
  - You’re going to boil your guppies!
Programming Challenge: Number Guessing Game (part 1)

- Ask the user to guess a number between 1 and 10. Assume they will enter an Integer.

- Pick a number between 1 and 10 that is your “secret” number (for example, 5)

- If the user types in your secret number, tell them that they win!

- If the user types in a number less than or greater than your secret number, tell them that they’re either above or below the number and to try again
secretNumber = 6
number = int(input("number?"))
if number < secretNumber:
    print("too low")
if number > secretNumber:
    print("too high")
if number == secretNumber:
    print("you win!")
Selection Statements in the Wild!

- How are selection statements used in ATM machines?

- How many selection statements can you count from your last ATM transaction?
The IF – ELSE structure
Simple Selection Statements

- The selection statements we have been writing so far have only allowed us to create a single alternate branch of execution.

- There are many times when we need to create multiple branches of execution based on the value of a Boolean expression.
The IF-ELSE structure

- The IF-ELSE structure allows you to perform one set of statements if a condition is true, and another if it is false.
The IF-ELSE structure

Start

Water Temp

T > 76
T < 82

False
Dead Fishies

True
Happy Fishies

Finish
The IF-ELSE structure

```python
if temperature < 72:
    print ("the water is too cold")

if temperature > 86:
    print ("the water is too hot")

else:
    print ("the water is perfect")
```
String Comparison
String Comparison

- So far we have been writing Boolean expressions that evaluate based on numeric data
  - Example: $x > 5; y < 10; z == 100$

- We can also construct Boolean expressions that can test relationships between strings

- When we compare strings we are essentially reducing them to their zeros and ones and comparing them numerically
<table>
<thead>
<tr>
<th>Dec</th>
<th>ASCII</th>
<th>Description</th>
<th>Dec</th>
<th>ASCII</th>
<th>Description</th>
<th>Dec</th>
<th>ASCII</th>
<th>Description</th>
<th>Dec</th>
<th>ASCII</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NUL</td>
<td>Control</td>
<td>16</td>
<td>DLE</td>
<td>Control</td>
<td>32</td>
<td>SP</td>
<td>Space</td>
<td>48</td>
<td>0</td>
<td>Number</td>
</tr>
<tr>
<td>1</td>
<td>SOH</td>
<td>Start of Header</td>
<td>17</td>
<td>DC1</td>
<td>Data Carriage</td>
<td>33</td>
<td>!</td>
<td>Exclamation</td>
<td>49</td>
<td>1</td>
<td>Digit</td>
</tr>
<tr>
<td>2</td>
<td>STX</td>
<td>Start of Text</td>
<td>18</td>
<td>DC2</td>
<td>Data Carriage</td>
<td>34</td>
<td>&quot;</td>
<td>Quotation Mark</td>
<td>50</td>
<td>2</td>
<td>Digit</td>
</tr>
<tr>
<td>3</td>
<td>ETX</td>
<td>End of Text</td>
<td>19</td>
<td>DC3</td>
<td>Data Carriage</td>
<td>35</td>
<td>#</td>
<td>Hash Mark</td>
<td>51</td>
<td>3</td>
<td>Digit</td>
</tr>
<tr>
<td>4</td>
<td>EOT</td>
<td>End of Transmission</td>
<td>20</td>
<td>DC4</td>
<td>Data Carriage</td>
<td>36</td>
<td>$</td>
<td>Dollar Sign</td>
<td>52</td>
<td>4</td>
<td>Digit</td>
</tr>
<tr>
<td>5</td>
<td>ENQ</td>
<td>Enquiry</td>
<td>21</td>
<td>NAK</td>
<td>Negative Ack</td>
<td>37</td>
<td>%</td>
<td>Percent</td>
<td>53</td>
<td>5</td>
<td>Digit</td>
</tr>
<tr>
<td>6</td>
<td>ACK</td>
<td>Acknowledge</td>
<td>22</td>
<td>SYN</td>
<td>Synch</td>
<td>38</td>
<td>&amp;</td>
<td>Ampersand</td>
<td>54</td>
<td>6</td>
<td>Digit</td>
</tr>
<tr>
<td>7</td>
<td>BEL</td>
<td>Bell</td>
<td>23</td>
<td>ETB</td>
<td>Echo</td>
<td>39</td>
<td>'</td>
<td>Single Quote</td>
<td>55</td>
<td>7</td>
<td>Digit</td>
</tr>
<tr>
<td>8</td>
<td>BS</td>
<td>Back Space</td>
<td>24</td>
<td>CAN</td>
<td>Cancel</td>
<td>40</td>
<td>(</td>
<td>Left Parenthesis</td>
<td>56</td>
<td>8</td>
<td>Digit</td>
</tr>
<tr>
<td>9</td>
<td>HT</td>
<td>Vertical Tab</td>
<td>25</td>
<td>EM</td>
<td>End of Medium</td>
<td>41</td>
<td>)</td>
<td>Right Parenthesis</td>
<td>57</td>
<td>9</td>
<td>Digit</td>
</tr>
<tr>
<td>10</td>
<td>LF</td>
<td>Line Feed</td>
<td>26</td>
<td>SUB</td>
<td>Substitute</td>
<td>42</td>
<td>*</td>
<td>Asterisk</td>
<td>58</td>
<td>:</td>
<td>Colon</td>
</tr>
<tr>
<td>11</td>
<td>VT</td>
<td>Vertical Tab</td>
<td>27</td>
<td>ESC</td>
<td>Escape</td>
<td>43</td>
<td>+</td>
<td>Plus Sign</td>
<td>59</td>
<td>;</td>
<td>Semicolon</td>
</tr>
<tr>
<td>12</td>
<td>FF</td>
<td>Form Feed</td>
<td>28</td>
<td>FS</td>
<td>Form Stop</td>
<td>44</td>
<td>,</td>
<td>Comma</td>
<td>60</td>
<td>&lt;</td>
<td>Less Than</td>
</tr>
<tr>
<td>14</td>
<td>SO</td>
<td>Shift Out</td>
<td>30</td>
<td>RS</td>
<td>Reverse Substitue</td>
<td>46</td>
<td>.</td>
<td>Period</td>
<td>62</td>
<td>&gt;</td>
<td>Greater Than</td>
</tr>
<tr>
<td>15</td>
<td>SI</td>
<td>Shift In</td>
<td>31</td>
<td>US</td>
<td>Unit Separator</td>
<td>47</td>
<td>/</td>
<td>Slash</td>
<td>63</td>
<td>?</td>
<td>Question Mark</td>
</tr>
</tbody>
</table>

The table provides a standard ASCII character set, including control and non-control characters, with their decimal and hexadecimal representations and corresponding descriptions.
Boolean Operators for Strings

'dog' > 'cat'
# is 'dog' greater than 'cat'?

'fish' < 'alligator'
# is 'fish' less than 'alligator'?

'elephant' == 'tiger'
# are 'elephant' and 'tiger' equivalent?

'bat' != 'honey badger'
# are these strings different?

'bat' > 'back'
# is 'bat' greater than 'back'?
Programming Challenge: Password Protection

- Write a program that asks the user for a password
- Check to see if the password that was submitted is equal to the string ‘seekrit’
- If it is, print out a “welcome” message
- Otherwise, tell them to try again
Basic string manipulation

- Python has a huge string manipulation library that allows you to interact with and modify strings. We are going to get more in depth with this package later in the semester.

- For now we will only be exploring two small functions in this package – lower() and upper()

- The lower() function converts the characters in a string to all lowercase, while the upper() function converts the characters in a string to all uppercase

- These functions are not built into the Python library directly, but exist inside the “str” module – as such they must be referred to using “dot syntax”

- Example:

  ```python
  string_lc = str.lower('Harry Potter')  # string_lc = 'harry potter'
  
  string_uc = str.upper('Harry Potter')  # string_uc = 'HARRY POTTER'
  ```
Programming Challenge: Case insensitive password

- Rewrite your password protection program to be case insensitive (i.e. the password “Seekrit” will also let you into your program)
Programming Challenge: Case insensitive password

- Rewrite your password protection program to be case insensitive (i.e. the password “Seekrit” will also let you into your program)

```python
password = input("wassur password?"
password_lc = str.lower(password)
if password_lc == "seekrit":
    print("yerin")
elif password_lc != "seekrit":
    print("tough nookies")
```
String Length

- You can ask Python to count the number of characters contained in a string using the `len()` function.

- `len()` returns an integer that represents the total length of a string.

- Example:

  ```python
  myname = 'harry'
  print (len(mynname))  # 5
  ```
Programming Challenge: Comparing the size of two strings

- Ask the user to input two names
- Sort the names in size order and print them out to the user
Programming Challenge: Comparing the size of two strings

- Ask the user to input two names
- Sort the names in size order and print them out to the user

```python
name1 = input("name 1")
name2 = input("name 2")

if len(name1) < len(name2):
    print (name1, name2)
elif len(name1) > len(name2):
    print (name2, name1)
```
Nested Decision Structures
Nested Decision Structures

- Indentation is key – Python will use the indentation level of a structure to determine its relationship to any previous statements.
IF-ELIF-ELSE Structure
Grade Determination Program

g = float(input('grade '))

if (g > 90):
    print ('A')
else:
    if (g > 80):
        print ('B')
    else:
        if (g > 70):
            print ('C')
        else:
            if (g > 60):
                print ('D')
            else:
                print ('F')
IF-ELIF-ELSE

- You can simplify complex IF statements by using the ELIF structure
- ELIF is an optional structure that can be placed between your IF and ELSE statements
- It allows you to evaluate additional conditions at the same level as the original IF statement
g = float(input('grade '))

if g > 90:
    print ('A')
elif g > 80:
    print ('B')
elif g > 70:
    print ('C')
elif g > 60:
    print ('D')
else:
    print ('F')
Some notes about using ELIFs:

- Conditions are tested in the order in which they are written. Once a condition evaluates to True all future conditions are skipped.

- An ELSE statement at the end of a decision structure is considered the “catch all” statement – if all conditions above end up failing then the statements inside the ELSE block will execute.

- However, using an ELSE statement at the end of your decision structure is optional.

- There is no logical need for an IF-ELIF-ELSE statement. You can always write a program without it by using a standard IF-ELSE block. The advantage of an IF-ELIF-ELSE statement is that your code may end up being more readable / understandable.
Logical Operators
Logical Operators

- All programming languages provide a set of “logical operators”
- These operators can be used to create complex Boolean expressions that evaluate more than one condition at the same time
Logical Operators

```python
x = 10
y = 5
a = 20
b = 25
if x > y and a < b:
    print ('yes!')
else:
    print ('no!')
```
Logical Operators

- Logical operators are used to combine Boolean expressions into a composite Boolean expression.

- There are three main logical operators that we use regularly in programming:
  - `and`
  - `or`
  - `not`
The “and” operator

- “and” can be used to combine two Boolean expressions

- The resulting Boolean expression will evaluate to be True if the two Boolean expressions it is connecting both evaluate to be True

<table>
<thead>
<tr>
<th>Boolean Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>True and True</td>
<td>True</td>
</tr>
<tr>
<td>True and False</td>
<td>False</td>
</tr>
<tr>
<td>False and True</td>
<td>False</td>
</tr>
<tr>
<td>False and False</td>
<td>False</td>
</tr>
</tbody>
</table>
Let’s evaluate!

a = 5

b = 10

print (a > b and a > 1)

print (a > 1 and b > a)

print (a == 5 and b < 100)

print (a > 1 and b < 1 and b > a)

print (a > 1 and b > 1 and b > a)
“and” Example

Loan Qualifier

```python
salary = float(input('How much do you make? '))
years = float(input('How long have you been at your job? '))

if salary >= 50000 and years >= 2:
    print ('You qualify for a loan!')
else:
    print ('You do not qualify for a loan')
```
The “or” operator

- “or” can also be used to combine two Boolean expressions
  - True or True => True
  - True or False => True
  - False or True => True
  - False or False => False

- The resulting Boolean expression will evaluate to be True if EITHER of Boolean expressions it is connecting evaluates to be True
Let’s evaluate!

\[ a = 5 \]
\[ b = 10 \]

print \((a > b \text{ or } a > 1)\)

print \((a > 1 \text{ or } b > a)\)

print \((a == 5 \text{ or } b < 100)\)

print \((a > 1 \text{ or } b < 1 \text{ or } b > a)\)

print \((a > 1 \text{ or } b > 1 \text{ or } b > a)\)
“or” Example

Guppy Temperature

temp = float(input('What is the temperature of your fish tank? '))

if temp < 72 or temp > 86:
    print ("The temperature is too extreme!")
The “not” operator

- The “not” operator is a unary operator that reverses the logical value of its argument.

- This means that it will “flip” a True value into a False value, and vice versa.
“not” example

username = input('username? ')

if not (username == 'Harry'):
    print("invalid input!")
else:
    print("Welcome, Harry!")
Programming Challenge: Username and Password

- Write a program that asks a user for a username and a password
- Check to see if BOTH the username and password are correct
- If so, provide a Welcome message to the user
- If not, provide a Login Failure message to the user