Chapter 2: Decision Making

CSI 112
George Mason University
What is a decision statement?

1. \texttt{age = get the age from the user}
2. \texttt{if the age > 20:}
   \hspace{1cm} \texttt{True:}
   \hspace{1cm} \texttt{result = "Congrats! You can now rent the Two Door Speck!"}
   \hspace{1cm} \texttt{False:}
   \hspace{1cm} \texttt{result = "Enjoy your bicycle, uphill both ways in the snow."}
3. \texttt{return result}

- Must always evaluate to true or false
- The decision must be composed of variables we have already given values to
Two types of decisions

- Nested decisions remember the results of decisions made before them (in the same nesting)
- Independent decisions do not
**Boolean values and calculations**

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>x and y</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>x or y</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

- A boolean value must evaluate to true or false
- Two boolean values can be compared with **and** or **or**
- Use parentheses if you want to combine **and** and **or**, i.e. 
  \[(x \text{ and } y) \text{ or } z \text{ or } x \text{ and } (y \text{ or } z)\], to disambiguate
Calculations that evaluate to boolean values

- $<$, $\le$, $>$, $\ge$ all evaluate to true or false
  - $3 < 2$ is false
- $==$, $!=$ also evaluate to true or false
  - $3 == 3$ is true
  - $3 == 4$ is false
  - "jello" != "blue" is true
if statements in Python

5. if operation is addition:
   True: result = number1 + number2
   False: do nothing

6. if operation is subtraction:
   True: result = number1 - number2
   False: do nothing

7. if operation is multiplication:
   True: result = number1 * number2
   False: do nothing

8. if operation is division:
   True: result = number1 / number2

if operation == "addition":
    result = number1 + number2
if operation == "subtraction":
    result = number1 - number2
if operation == "multiplication":
    result = number1 * number2
if operation == "division":
    result = number1 / number2

- **if** is a keyword; the if statement must end in a colon
- what belongs to a particular if statement is indented
elif statements in Python

- **elif** is a keyword; it stands for else if
- **elif** is attached to an `if` or `elif` before it, and indicates this `elif` is nested
- (you cannot have a standalone `elif`)

```python
5. if operation is addition:
    True: result = number1 + number2
    False: if operation is subtraction:
        True: result = number1 - number2
        False: if operation is multiplication:
            True: result = number1 * number2
            False: if operation is division:
                True: result = number1 / number2
```
if versus elif

```python
operation = "addition"

if operation == "addition":
    result = 6
elif operation == "addition":
    result = 5
elif operation == "addition":
    result = 4

return result
```

- The one on the left returns 6
- The one on the right returns 4
- Why? **if-elif** statements are nested, linked, and mutually exclusive. The plain **if** statements are not mutually exclusive, don’t know about each other, and all get executed
else statements in Python

1. number1 = get the first number from the user
2. number2 = get the second number from the user
3. if ((number1 - number2) is 1) or ((number1 - number2) is -1):
   True: result = "consecutive"
   False: result = "not consecutive"
4. return result

- **else** is a keyword; it is linked to an **if** or **elif**, and gets executed if the **if/elif** above it is false
- The **input** function is used to get user input (but we will never use this because it’s not practical for testing)
else statements in Python

```python
if operation == "addition":
    result = number1 + number2
elif operation == "subtraction":
    result = number1 - number2
elif operation == "multiplication":
    result = number1 * number2
elif operation == "division":
    result = number1 / number2
else:
    result = "operation undefined"
```

- **else** only gets executed if none of the **if** or **elif** before it are true
Indentation matters!

Link to example: http://pythontutor.com/visualize.html#code=def+template(num1,num2)%3A%0A%09result+%3D+%22%22%0A%09if+num1+%3D%3D+0%3A%0A%09%09result+%3D+%22num1+is+0%22%0A%09elif+num1+%3D%3D+1%3A%0A%09%09result+%3D+%22num1+is+1+%22%0A%09%09if+num2+%3E+3%3A%0A%09%09%09result+%3D+result+%2B+%22num2+%3E+3%22%0A%09%09elif+num2+%3E+4%3A%0A%09%09%09result+%3D+result+%2B+%22THIS+WILL+NEVER+RUN%22%0A%09%09else%3A%0A%09%09%09result+%3D+result+%2B+%22num2+%3C%3D+3%22%0A%09%09result+%3D+result+%2B+%22+finished+num1%22%0A%09else%3A%0A%09%09result+%3D+result+%2B+%22num1+is+not+0+or+1%22%0A%09%09return+result%0A%0Aprint+template(0,1)%0Aprint+template(1,3)%0Aprint+template(1,2)%0Aprint+template(2,1)&mode=edit&origin=opt-frontend.js&cumulative=false&heapPrimitives=false&textReferences=false&py=2&rawInputLstJSON=%5B%5D

This is another type of “nesting”, and what people usually refer to as nested if-else statements
Programming TRAP

Assignment Statement \( x = 5 \)

Boolean Expression \( x == 5 \)
boolean types in Python

```python
value1 = (1 == 1)
value2 = True
value3 = False
return (value1 and value2 and not value3)
```

- **True** and **False** are both keywords and types in Python (capitalization matters)
- **not** is a keyword that negates a boolean value
- the code above returns **True**