Conditionals

Overview: Making Decisions

If “is it raining”:
  - take the umbrella
  - wear rainboots
  - wear raincoat
Else:
  - wear sandals
  - wear a summer dress

“Is it raining” is an expression that can return True or False. In a Python program we can use:
- True/False values
- Relational Expressions
- Logical Expressions
- Predicates (all evaluate to True/False) whenever the code needs to make a decision for what to do next.

Conditionals

Conditionals (if Statements)

Boolean expressions are used to choose between two courses of action in a conditional statement introduced by the keyword if.

```python
def abs(n):
    '''returns absolute value'''
    if n < 0:
        return -n
    else:
        return n
```

Above is the Python syntax for expressing conditional statements. To notice:
- Colons at the end of line for if and else
- Indentation for lines succeeding if and else

Flow Diagrams

The Road Not Taken

Two roads diverged in a yellow wood,
And sorry I could not travel both

IMPORTANT: Only one of the branches is ever executed when a conditional statement is encountered. That is what the Flow Diagram exemplifies.
Dropping `else`

When writing functions where both branches contain a `return` statement, we can drop the keyword `else`, as shown below. This is because `return` will exit the function when `if` is True. In the case it's False, the rest will be executed, which is exactly what `else` would do too.

```python
def abs(n):
    '''returns absolute value'''
    if n < 0:
        return -n
    return n
```

Notice the missing `else`.

Nested Conditionals

**Concepts in this slide:** Syntax for nested conditionals, example of nesting.

```python
def abs(n):
    '''returns absolute value'''
    if n < 0:
        return -n
    return n
```

Flow Diagram:

**Concepts in this slide:** Another example of the flow diagram model for branched execution.

**A Better Approach:**

Chained Conditionals

**Concepts in this slide:** New keyword: `elif`. Replace nesting with chaining of conditionals.

```python
def movieAge(age):
    if age < 8:
        return 'G'
    elif age < 13:
        return 'PG'
    elif age < 18:
        return 'PG-13'
    else:
        return 'R'
```

Compare this implementation of `movieAge` with that of the previous slide. For chained conditionals, we write less code, which is also easier to read because of fewer indentations.

**Flow Diagram:**

Chained Conditionals

**IMPORTANT:** In the moment one of the tests is True, the associated statements are executed and the chained conditional is exited. Only in the case when tests are False, we continue checking to find a True test.
Exercise 1: Define a function named `letterGrade` that takes one score (the average of all your individual scores in a class), and returns a letter grade.

Assume:
A >= 90, B >= 80, C >= 70, D >= 60, F < 60

Exercise 2: Define a function named `addArticle` that takes a string argument and returns a new string with the correct article (a or an) added to the front of the argument.

Exercise 3: Define a function named `daysInMonth` that takes a month (as an integer) as the parameter, and returns the number of days in it, assuming the year is not a leap year. If the month does not fall between 1 and 12, return an error message as a string. Make the function as concise as possible.

isVowel revisited

The following definition doesn't work. Why?

```python
def isVowel(s):
    l = s.lower()
    return l == ('a' or 'e' or 'i' or 'o' or 'u')
```

Because by Python's treatment of truthy/falsey values, it's equivalent to

```python
def isVowel(s):
    l = s.lower()
    return l == 'a'
```

Simplifying Boolean Expressions and Conditionals

There are several code patterns involving boolean expressions and conditionals that can be simplified. The unsimplified versions are considered to be bad style and will be flagged by our Codder tool. Below BE stands for any expression evaluating to a boolean, and STMS stands for any statements.

<table>
<thead>
<tr>
<th>Complex Expr/Stmt</th>
<th>Simpler Expr/Stmt</th>
<th>Complex Expr/Stmt</th>
<th>Simpler Expr/Stmt</th>
</tr>
</thead>
<tbody>
<tr>
<td>BE == True</td>
<td>BE</td>
<td>BE == False</td>
<td>not BE</td>
</tr>
<tr>
<td>if BE: return True else: return False</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>if BE: return False else: return True</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>if BE1: return BE2 else: return False</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>if BE: STMS return True else: STMS return False</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STMS return BE</td>
<td>result = BE</td>
<td>return result</td>
<td>return BE</td>
</tr>
</tbody>
</table>
Simplifying Boolean Expressions and Conditionals: Example

```python
def doesNotBeginWithVowel(s):
    if isVowel(s[0]) == False
        return True
    else:
        return False

def doesNotBeginWithVowel(s):
    if not isVowel(s[0])
        return True
    else:
        return False

def doesNotBeginWithVowel(s):
    return not isVowel(s[0])
```