

More symbol tables, conditionals, position and keyword arguments

Oliver W. Layton

CS151: Computational Thinking: Visual Media

Lecture 07, Spring 2021

Wednesday Feb 24

Admin

- Project extension grading: 29/30 is a badge of honor in CS courses!

Continued: Temperature symbol table

```
def convertFtoC(temp):  
    convertedTemp = (5/9)*(temp - 32)  
    print(temp, 'degrees in F is', convertedTemp, 'degrees in C.')
```

Main code

```
tempInF1 = 32  
convertFtoC(tempInF1)
```



```
tempInF2 = 212  
convertFtoC(tempInF2)  
tempInF1 = 100
```

- **Local scope:** variables only may be accessed inside their own symbol table. This is why we can't access convertedTemp from main code.

One more thing to add to symbol tables: modules

Let's draw symbol table for the following

```
import turtle
```

```
def right():  
    print('Oliver you are always right!')
```

```
dist = 10  
turtle.forward(dist)  
turtle.forward(20)
```

Conditional statements

New type of statement to allow us to prepare for the unexpected!

if statement

```
if [Boolean condition statement]:  
    [body]
```

If there's traffic, then we want to take a different route. If it's raining, go outside with an umbrella. If we're drawing a house, we'll want to draw a rectangle and triangle.

The condition **MUST** evaluate to True to enter the `if` statement body (indented).

Note the :

Example if statement

```
if i_am_happy:  
    print(':)')
```

```
print('hi 1')
```

```
if i_am_sad:  
    print(':(')
```

```
print('hi 2')
```

- Code executes line-by-line, top-to-bottom like usual. `i_am_happy` and `i_am_sad` variables MUST be either `True` or `False` (boolean type).
- `print(':)')` only run IF `i_am_happy` is assigned the value `True` (otherwise `print('hi 1')` is run next after Line 1).

`if else` statement

After the body of the `if` statement, you have the option of putting an `else` statement (not required), and Python will evaluate code in the body of the `else`, assuming the Boolean condition evaluates to `False`:

```
if [Boolean condition statement]:  
    [body]  
else:  
    [body]
```

- Python *skips over the body of the* `else`, if it enters the `if` clause, which makes sense.

Example if else

```
if i_am_happy:  
    print(':)')  
else:  
    print(':(')  
  
print('hi')
```

- if `i_am_happy` is `True`, we print :) then we print hi.
- if `i_am_happy` is `False`, we print :(then we print hi.

Adding more conditions to check with `elif` clauses

Short for `else if`.

```
if isSnowing:  
    getSkiJacket()  
elif isRaining:  
    getRainJacket()  
else:  
    getSunGlasses()
```

`isSnowing` and `isRaining` are booleans.

Relational operators

Assume `x = 1` and `y = 2`. All expressions **must** evaluate to `True` or `False` (Boolean value):

Operator	Definition	Example
<code><</code>	strictly less than	<code>x < y</code>
<code><=</code>	less than or equal to	<code>x <= 0</code>
<code>></code>	strictly greater than	<code>y > 1 + x</code>
<code>>=</code>	greater than or equal to	<code>y >= 2</code>
<code>==</code>	equal to	<code>x == y</code>
<code>!=</code>	not equal to	<code>x != 10</code>

Relational operators have **lower precedence** than math operators (*i.e. math is done before relational operators are applied*).