TCSS 142 – Introduction to Programming

Autumn 2014
Day 03

Day 3 Overview

- Connecting to H: drive from home
- Number and field formatting
- Interactive input
- Boolean expressions
- Simple if
- Sequential ifs

Working From “Home”

- You can connect to your H: drive from home, but you need to do that through the Institute's server
  - Every student in our class gets an account with the institute
  - Username is your netid
  - Password is …
  - And change it today to something you can remember

- Directions are located on Institute's Lab Web pages
  - Home page in Firefox
  - http://css.insttech.washington.edu/~lab/

Working From “Home”

- Once you have your account set up, you will need a software that will connect you to the Institute's server and your H: drive
  - PuTTY and a window-based secure transfer program, such as WinSCP
  - SSH

Number and field formatting

- Using built-in `format` function
  - Two arguments:
    - Numeric value to be formatted
    - Format specifier
  - Returns string containing formatted number
  - Format specifier typically includes precision and data type

- As a lab exercise

- Formatting does NOT change variable value – formats for display purposes only!!!

Input

- Most programs need to read input from the user

- Built-in `input` function reads input from keyboard
  - Returns the data as a string
  - Format: *`variable`* `= input(`*`prompt`*`)`

- In order to interpret the value as an int or as a float, the string value has to be converted
  - `int(`*`item`*`)` converts *`item`* to an `int`
  - `float(`*`item`*`)` converts *`item`* to a `float`

Example

- Open `myFirstProg.py` and save as `areaIO.py`

- Let's change the program so that we prompt for input, read it, and then calculate the area of a circle

- Let's extend the program by prompting for a height as well and calculating the volume of a cylinder

- What happens when a wrong value is entered?

Relational Expressions

- `if` statements and `while` loops both use logical tests that are based on Boolean logic (true or false).

- Tests use *relational operators*:

Operator	Meaning	Example	Value
`==`	equals	`1 + 1 == 2`	`true`
`!=`	does not equal	`3.2 != 2.5`	`true`
`<`	less than	`10 < 5`	`false`
`>`	greater than	`10 > 5`	`true`
`<=`	less than or equal to	`126 <= 100`	`false`
`>=`	greater than or equal to	`5.0 >= 5.0`	`true`

Logical operators

- Logical operators are used to create more complex Boolean expressions: `and`, `or`, `not`

- "Truth tables" for each, used with logical values *p* and *q*:

p	**q**	**p and q**	**p or q**
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

p	**not p**
true	false
false	true

Logical operators

- Tests can be combined using *logical operators*:

Operator	Example	Result
and	`(2 == 3) and (-1 < 5)`	`??`
or	`(2 == 3) or (-1 < 5)`	`??`
not	`not(2 != 3)`	`??`

Operator	**Meaning**	**Associativity**
( )	Parenthesis	Left
-	Negation	Right
**	Exponentiation	Right
*, /, // , %	Multiplication, Division, Modulus	Left
+ , -	Addition, Subtraction	Left
<, <=, >, >=, ==, !=	Relational Operators	Left
not		
and, or	Logical Operators	Left
=	Assignment	Right

Expressions

- Assuming `x = 4, y = 6, z = 0`
  evaluate the following expression as either T or F on paper and then run in Python

```
x + 2 < y
x + 3 >= y
y == x
not x < 3
z == not 6
x < 6 < 10
not y
not z
not "hello"
not ""
```

Write an Expression

- taxRate is over 25% and income is less than $20000

- temperature is less than or equal to 75 or humidity is less than 70%

- 21 < age < 60

- age is 21 or 22

- age is either < 10 or > 20

Truth Tables

- Verify whether

  not (A and B) *and* (not A) or (not B) are equivalent

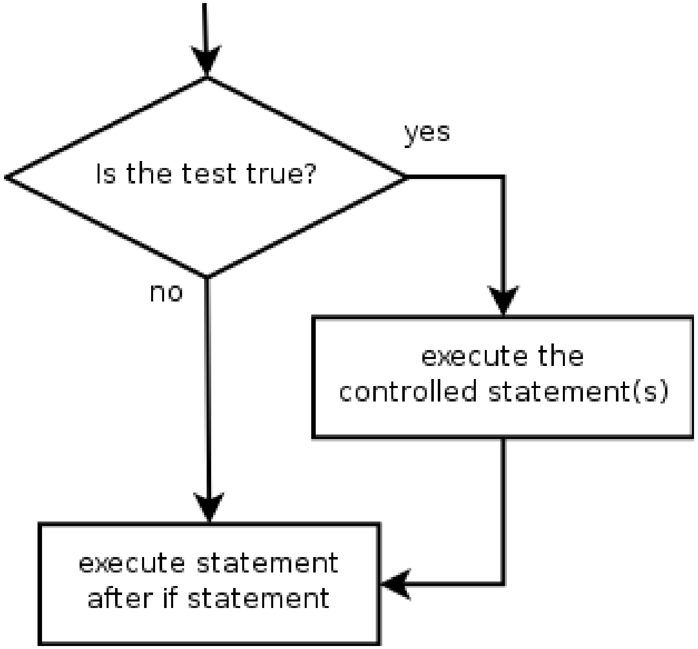
  not A && not B *and* not (A or B) are equivalent

- Verify whether A and B or C *and* not (A or B and C) are equivalent

The `if` statement

Executes a block of statements only if a test is true

```
if test:
    statement
    ...
    statement
```


Is the test true?
yes
no
execute the controlled statement(s)
execute statement after if statement

- Example:

```
gpa = float(input("Enter your gpa: "))
if gpa >= 2.0:
    print("Application accepted.")
```

The `if` statement

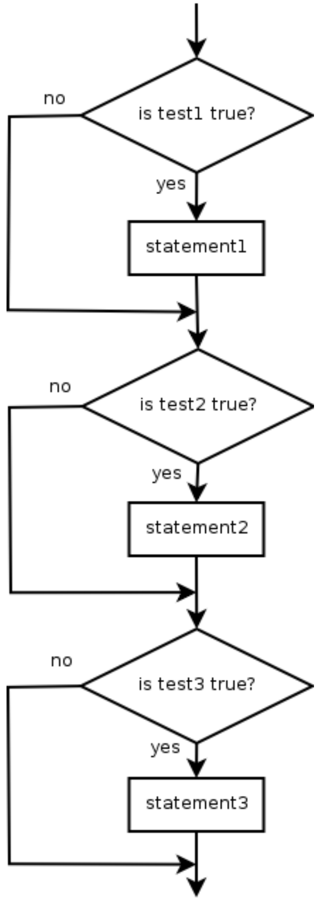
```
gpa = 1.0
if gpa >= 2.0:
    print("Application accepted.")
    print("Good job!")
```

```
gpa = 2.0
if gpa >= 2.0:
    print("Application accepted.")
    print("Good job!")
```

```
gpa = 1.0
if gpa >= 2.0:
    print("Application accepted.")
print("Good job!")
```

Sequential ifs

- On occasion, you will have a number of tests to perform
  - If these are independent tests - NOT mutually exclusive (one true condition does not preclude another to be true), use sequential ifs



```
if test:
    statement(s)
if test:
    statement(s)
if test:
    statement(s)
```

0, 1, or many paths may execute

(independent tests; not exclusive)

Example

- Taxes

if you have a child under age 17, deduct $1,000

if you have mortgage, deduct mortgage interest charged by the bank

if you own a car, deduct car registration fee

if you are over 70, deduct $1, 125

if you gave to tax-deductible charities, deduct the amount you gave

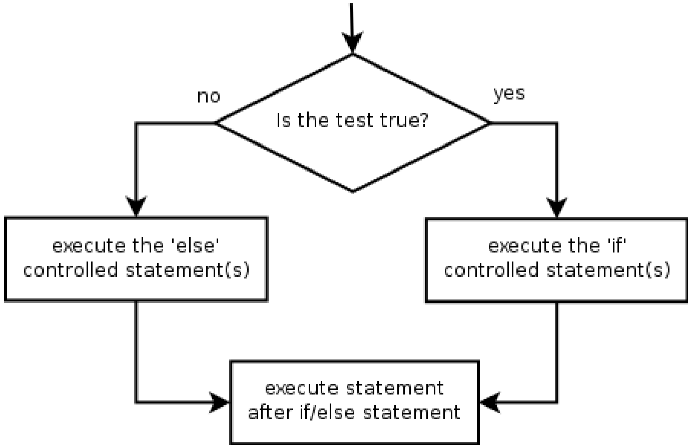
Exercise

- Write a program called `evens.py` that asks for 3 integers and prints how many of these values are even numbers:
  - How to determine if a number is even?
  - How to keep track of how many of these numbers are even?

The `if/else` statement

Executes one block if a test is true, another if false

```
if test:
    statement(s)
else:
    statement(s)
```


Is the test true?
no
yes
execute the 'else' controlled statement(s)
execute the 'if' controlled statement(s)
execute statement after if/else statement

- Example:

```
gpa = float(input("Enter your gpa: "))
if gpa >= 2.0:
    print("Welcome to Mars University!")
else:
    print("Application denied.")
```