FIT 100 Question

What are the five largest cities in the United States?

(Please jot down your answer on a piece of scratch paper.)

FIT 100 A BIG DIFFERENCE Between People and Computers

- People are very good at:
  - Resolving ambiguity
  - Taking context into account
- Computers are <u>not</u> very good at:
  - Resolving ambiguity
  - Inferring “right” interpretations based on context

Thus if we want to tell a computer what to do, we must do so *precisely* and *unambiguously*.

Programming Basics

FIT 100

To specify algorithms, we must be precise. To be precise, we need a language that is more exact than English. A programming language offers this advantage. All programming languages have a basic set of features.

FIT 100 What's Special About Programming Languages?

- Though the Alphabetize CDs algorithm was precise enough for a person to execute successfully, computers demand greater precision
- Programming languages are formal notations specifically designed for specifying algorithms – that means *each "word" or "sentence" in a programming language has one and only one interpretation*
- The programming language we will study this quarter is Visual Basic 6.0 (VB6)

FIT 100 Our Approach To Programming

- The plan …
  - Introduce you to general concepts of programming languages in lecture – these ideas apply to virtually all programming languages
  - Use VB6 in lecture and lab to illustrate these ideas
    - In your next lab you'll write your first VB6 program
  - Practice the ideas by writing programs
  - Add a few more language features…
  - Practice with a few more programs…
  - Some issues we'll explore:
    - How does a computer search for information?
    - How does a computer make a decision?
    - Can a computer create something aesthetic?

FIT 100 Order Matters

- CONCEPT: Programming languages execute ("do") instructions in order (unless told to do otherwise… more about this later)
- A program looks a bit like a long "to do" list; the first things on the list get done first
- Each instruction is executed one at a time – then the computer goes on to execute the next instruction
- Example: When you wrote your Web pages, the computer executed the HTML code in the order you of the statements you wrote.

FIT 100 Storing, "Remembering", Changing, and Accessing Data

- Imagine a programming language that was just made up entirely of "sentences" like "Print this" or "Display that"
- It would be very limited
- The program would have to do exactly the same thing every time

- CONCEPT: Being able to store, "remember", change, and access data allows us to write programs that do the same thing but with different data each time.

FIT 100 Variables

- CONCEPT: *Variable* is the term for a place in memory where the program can store, access, and restore information.

All variables have the following 3 properties:

1. A *name* so that the program can refer to the variable (a location in memory)
2. A means to store a (new) value in the variable
3. A means to "see" (or make a copy of) the value stored in the variable

FIT 100 Variables in Exchange Sort with CDs

When we acted out Exchange Sort, we used variables as follows:

1. Each student holding a CD represented a variable (a location in memory)
2. Each variable (student) stored a value (a CD)
3. Each time we looked at the name of a CD we were "seeing" the value stored in that variable
4. When two students exchanged CDs, a new value (CD) was stored in each of the variables (students)

FIT 100 On Variable Names

- The term "variable" reminds us the value can change
- The names used for variables are arbitrary provided:
  - Variable names must begin with a letter
  - Variable names can contain any letter, numeral or _
  - Most languages are *case sensitive*: a ≠ A
- Good variable names are meaningful and accurate
  - total, averageOverClass, etc, but not x, o0OO0o, etc

VB6: In all programming for FIT100, variable names should start with lowercase letters so as to avoid confusion with other names in VB6 … *ignore this convention at your peril!*

FIT 100 On Variable Values

- Values refer to the information stored in the variable (location in memory)
- Variables can take on different *types* of values
  - Whole numbers or *integers*: `2, -9, 1048576`
  - Character sequences or *strings*: `"2", "&^%$#@", " "`
  - Decimal numbers or *doubles*: `2.0, 3.14159, -999.99`
- In most programming languages, each variable can only hold one type of value
  - So the computer will know how much memory will be needed to store the value (e.g., integer vs. double)
  - To allow the computer to help detect errors in the code (e.g., when the program tries to put the wrong sort of value in a variable the programmer receives an error message)

FIT 100 Declaring Variables

- Declaring variables is a way of telling the computer:
  - That you want a location in memory (*the variable*)
  - What you will call (how you will refer to) that location in memory throughout your program (*the variable name*)
  - What type of information you will store in that location in memory, so the computer will know how much space to set aside (*the variable type*)
- VB6: Some examples of declaring variables:
  - `Dim num1 As Integer`
  - `Dim letter1 As String`
  - `Dim averageOverClass As Double`

FIT 100 Assigning Values to Variables

- CONCEPT: Computers must be told what value to assign to variables
- CONCEPT: The general form of an assignment statement is
  <*variable name*> <*assignment symbol*> <*expression*>
  - Languages use different assignment symbols: `=` `:=` ←
  - Read assignment as “is assigned”, or “becomes” or “gets”
  - All three components must always be present
- CONCEPT: Fundamental property of assignment
  The “flow” of information is always right-to-left
- VB6: Some examples of variable assignment
  - `destination = 12`
  - `changedVariable = value`

Meta-brackets < > enclose language defining terms

FIT 100 Let’s Try This Out…

A VB6 Example….

```
Dim snap As Integer
Dim crackle As Integer
Dim pop As Integer
snap = 6
pop = 3
snap = 5
snap = pop
pop = 10
crackle = pop
```

Question: What’s in snap? What’s in crackle?