Part 2.2 – How a high-level (human-like/assumption-based) language is translated to low-level (computer/math-based)

Explanation of print vs raw\_input – why is syntax different?

 Because the people who made the language chose it to be.

 A language usually goes through more than one step

 In order for it to be run on a computer.

 Computer only knows binary. Which is basically math.

 It’s a system of logic. 1 is yes and 0 is no. Everything gets converted to that.

Computer hardware uses transistors to do logic. A transistor has 3 connections. If the switching input is true (has a charge) then the charge on the other input is passed to the output. There are also analog computers, but so far we have the most success and control using digital circuits.

The first step of translating to machine language (binary) is called Compiling. This takes the file (or files) and creates something like assembly (a list of memory copying and math operations). The next step is linking: this creates a single binary file (for a single-file program, or does the same for each component of a program with multiple components—such as dll files [shared library of data and functions]).

What happens when you have a weird English-like phrase, such as:

 Print ‘your name is’,myname

 It is converted to a function, probably something like

 Print(“your name is roger”)

 [roger was was found in memory that ‘myname’ pointed to]

 During compiling, it is converted to something like assembly, depending on the language:

 db ah, 0, 18, ‘$’

 (copies from memory location ah, starting at 0, for 18 characters, as a string)

 Then converted to binary, something like:

 00000001 00001111 00010010 00000010

 (if opcode 1 were a string copy command, and 2 [00000010] specified a string)

 If this were real binary for some computer that could read it, it would be a program 4 bytes long that displays a string of text