AND STARTING TODAY, ALL PASSWORDS MUST CONTAIN LETTERS, NUMBERS, DOODLES, SIGN LANGUAGE AND SQUIRREL NOISES.
Review Exercise

• Define:
  • Variable
  • Primitive Types
  • Conditional
  • Relational operator

• True/False
  • if(x = 34) tests to see if x is equal to 34
  • The number of bytes of memory used by a variable depends on its value.
  • A memory address is where a variable is stored.
Review Exercise

• If the user provides 1, what will print to the screen?

```cpp
#include <iostream>
using namespace std;

int main () {
    int num = 0;
    cout << "Give me a number: ";
    cin >> num;

    switch (num) {
    case 1:
        cout << "Go left" << endl;
        break;
    case 2:
        cout << "Go right" << endl;
        break;
    default:
        cout << "What ran?" << endl;
        break;
    }

    return 0;
}
```
Review Exercise

• What does this code output?

```cpp
#include <iostream>
using namespace std;

int main () {
    int x = 0;
    if (x == 2 || 1) {
        cout << "The number is 1 or 2" << endl;
    }
    else {
        cout << "The number is not 1 or 2" << endl;
    }
    return 0;
}
```
String

- C++ style strings are objects (revisit in 162)
- Come from `<string>`
- Allows us to take in more than numbers or single entities
- Examples:
  - “Hello world” -> 
  - “123 456 789”
  - “a b C”
Use getline

• There are two getline functions
  • `<string>` getline -> takes the istream, takes the string variable, extracts until delimiter or \n (newline)
  • `<istream>` getline -> c-string (week 7?)

• Use the one in the `<string>` library

• Example
  
  string my_str = "";
  cout << "Give me a string: ";
  getline(cin, my_str);
Why are strings cool?

• Most user interfaces don’t operate purely on numbers
• Can store more info (baby step into arrays -> week 6)
• Can do more interesting things such as error handle
• It’s an object so more functionality
Demo