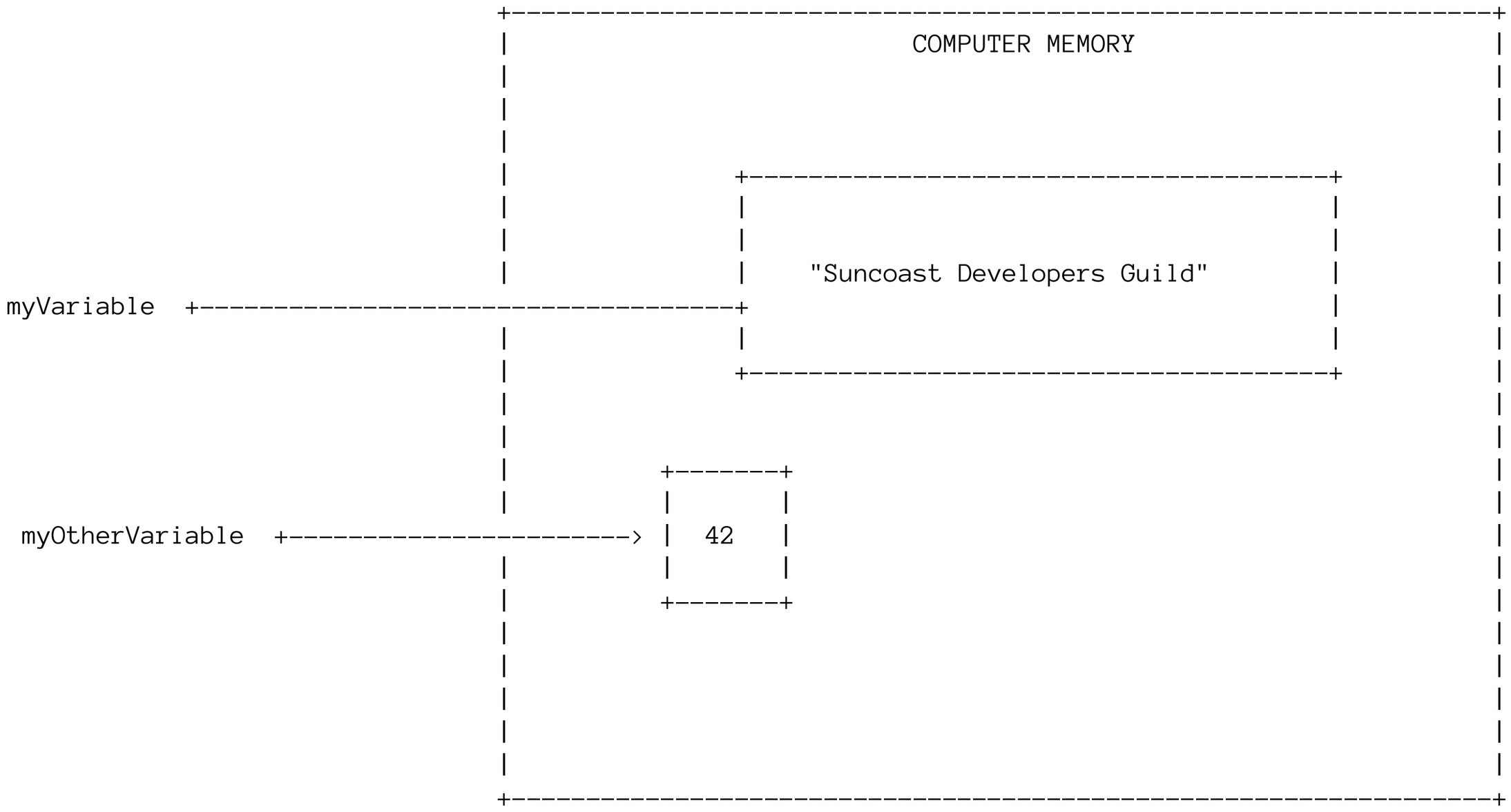
Computers

... are machines that accept input, manipulate data, and produce output.

Variable

... is an identifier for a value our program needs to keep track of


COMPUTER MEMORY
"Suncoast Developers Guild"
myVariable
myOtherVariable
42

Language Rules

- Naming variables
- The kinds (*types*) of information we can store in variables
- If the value of a variable, once assigned, can be changed
- The syntax for declaring a variable
- If the kind (*type*) of information stored in a variable can change once assigned

C# rules

- Must *declare* our variables before we can use them.
- Must tell `C#` what kind of information the variable will store.
- From our example `myVariable` stores a series of letters and spaces and `myOtherVariable` stores a number.
- These represent the variables `type`.

Our first `C#` types

- A sequence of letters, numbers, digits, emoji 🚀, spaces, etc. is called a `string`
- A whole number (without any decimal places) is called an **`int`** (short for *integer*)

How do you make a variable?

```
string name = "Samantha";
int score = 95;
```

Break it down

```
string name = "Samantha";
int score = 95;
```

Breaking it down

```
// Type    Name    Value     Statement End
//  |        |       |          |
//  |        |       |          |
//  v        v       v          v
   string  name = "Samantha";
```

Break it down

```
string name = "Samantha";
int score = 95;
```

```
// Type    Name    Value     Statement End
//  |        |       |          |
//  |        |       |          |
//  v        v       v          v
    int     score = 95;
```

Let C# do the work

- Programmers like to be efficient Some might say lazy
- Let the computer do the work when we can

- *Type inference* Let C# figure out what to write where type goes.

Type Inference

```
var name = "Samantha";
var score = 95;
```

- var keyword allows C# to automatically automagically? determine the type of variable based on the value on the right hand side.

- This is how we will create variables most of the time.

So what else ya got?

- Numeric primitive types
- Non-numeric primitive types
- Reference types

Numeric types

- **int** : a whole number from –2,147,483,648 to 2,147,483,647
- **long** : a whole number from –9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
- **double** : a 64 bit floating point value that has an approximate precision of ~15-17 digits
- **float** : a 32 bit floating point value that has an approximate precision of 6-9 digits
- **decimal** : a more precise way to store decimal numbers, but has a smaller range

Non-numeric primitive types

- **char** : represents a unicode character, a single letter, or an emoji. This is what `strings` are made of.
- **bool** : `true` or `false`
- **byte** : represents a raw chunk of data (values from `0` to 255)

Examples

Integers

```
var score = 42;
```

Double

```
var total = 10.0;
var alsoTotal = 10d;
var pi = 3.14159265;
```

Floats

```
var total = 10f;
var price = 10.0f;
var pi = 3.14159265f;
```

Decimal

```
var total = 10m;
var price = 10.0m;
var pi = 3.14159265m;
```

Examples

Characters

```
var piratesFavoriteLetter = 'r';
var firstLetterOfTheEnglishAlphabet = 'a';
var lastLetterOfTheEnglishAlphabet = 'z';
var capitalLetter = 'Q';
var rocketShip = '🚀';
```

Boolean

```
var theCakeIsALie = true;
var worldIsFlat = false;
```

Strings

```
var name = "Zaphod Beeblebrox";
```

Math

Addition, subtraction, multiplication, division, remainder (modulus)

```
var firstNumber = 42;
var secondNumber = 12;

var sum = firstNumber + secondNumber;
var difference = secondNumber - firstNumber;
var product = firstNumber * secondNumber;
var division = firstNumber / secondNumber;
var remainder = firstNumber % secondNumber;
```

PEMDAS

PEMDAS

```
var answer = 2 + 3 * 4; // 14
var answer2 = (2 + 3) * 4; // 20
```

NOTE: There are other order of operations in code. We may see some of those later.

Strings have extra features!

More complex types, like `string` can be asked questions about itself.

How long is a string?

```
var sentence = "The quick brown fox jumped over the lazy dog";
var howLong = sentence.Length;
```

Break it down

```
//      variable  right-hand-side
//         |              |
//         v              v
    var howLong = sentence.Length;
```

```
// variable  property
//     |         |
//     v         v
    sentence.Length
```

Results

```
var sentence = "The quick brown fox jumped over the lazy dog";
var howLong = sentence.Length;
```

The howLong *variable will know it is an* `int`*, and its contents will be* `44`

DateTime

C# comes with a specialized class to deal with storing a date and time.

A `DateTime` is a `Year`, `Month`, `Day`, `Hour`, `Minute`, `Second`, and `Millisecond`.

To create a DateTime variable equal to the current time you can use this syntax:

```
var rightNow = DateTime.Now
```

If you then wanted to know the current month:

```
var rightNow = DateTime.Now;
var currentMonth = rightNow.Month;
```

About naming variables

- Use clear variable names.
- Longer variable names don't cost anything.
- Remember that code is **written once** but **read many** times.
- You may be the one reading it later.

Always code as if the developer who ends up maintaining your code knows where you live. Code for readability.
– A wise developer

Not good

```
var w = 52;
var n = "Arthur Dent";
var i = 1;
var resp = axios.get();
```

Better

```
var weeksInOneYear = 52;
var name = "Arthur Dent";
var currentDepartmentIndex = 1;
var response = axios.get();
```