Flowcharts and pseudocode are both ways to represent an algorithm. (An algorithm is a finite set of instructions that solves a problem).

Flowcharts are easy to understand, because they graphically represent the logical steps. The alternative, pseudocode, is popular with modern programmers. But everyone recognizes that both flowcharts and pseudocode are useful.

Part 1 - Your First algorithm

To introduce you to Flowgorithm, we'll start with a very simple algorithm, expressed in pseudocode. You don't have to understand what pseudocode is, or even what an algorithm is, to understand this example.

```
Display "Hello, World!"
Display "My name is Joe Gunchy!"
```

There. That's an algorithm. Not much of an algorithm, but an algorithm all the same. It is a complete, finite set of instructions to solve the problem of displaying a welcoming message.

A flowchart to represent that same algorithm would look like this:
NOTE: Textbook Flowcharts vs. Flowgorithm

The textbook uses the word “Display” to indicate that text will be shown to the user. Flowgorithm, by default, uses the word "Output".

You can tell Flowgorithm to use the book’s flowchart format by using clicking on Edit  Choose Chart Style  and then selecting "Gaddis" from the dropdown box.

Part 2 - A flowchart using Flowgorithm

Now we’ll create a flowchart using the Flowgorithm program.

1. Begin the Flowgorithm program. (Your instructor will tell you how to start the program.)

2. As you see, the “Main” and “End” ovals are already there for you. All you are going to have to do is put in the two “Output” parallelograms. For the first one, click the mouse somewhere along the arrow that runs from “Main” to “End”. (When using Flowgorithm for this operation, you can double-click or left-click.)

3. Look at the group of flowchart operation symbols presented. Find the one for “Output”, and select it with the mouse.

4. You should get a parallelogram with **Output**. Notice that is gray. This means that the shape is not yet complete. So, **double-click or left-click** anywhere inside the parallelogram.

5. Inside the window that pops up on your screen, enter the expression that we want to output. Specifically, enter "Hello, world!" You **must include the double-quotes**. Then, click OK.

6. Now repeat that same steps for the second output operation. Be sure you click on the arrow in the right place – we want to insert this second operation **after** the first one, so you have to click somewhere along the arrow between the first output operation and the “End” symbol. Create the output box and put “My name is your name!” as the output expression. Use **your** name!

The flowchart is now complete. In the old days, that would be the end of the flowcharting – you draw a flowchart (usually with pencil and paper), and that’s that. But this isn’t the old days! This is the digital twenty-first century!

The Flowgorithm application does much more than helping you to draw a flowchart. It can actually **execute** the logic of the flowchart. That’s why we’ll be using it all semester. We can learn all about creating algorithms during lecture and from the lecture textbook. In lecture, you’ll also learn to “desk-check” your algorithms – sort of “play execute” them.

But with Flowgorithm, you can actually get the computer to carry them out! And it is time to do just that.

7. On the Flowgorithm toolbar, click the green triangle (it looks like the play button on Youtube). You can also press F5 or “Run” from the Program Menu.

Voila! The first output shows up on the screen! (If you got an error message instead, you didn’t follow the instructions exactly. If you can’t figure out how to fix your flowchart, call for help.)

The flowchart has been executed. Your algorithm was designed to display (if your name was Joe Gunchy):

Hello, World!
My name is Joe Gunchy!
…on the screen, and it did exactly that. Cool!

NOTE: The Console Window

Notice that, with Flowgorithm, it looks like you received a text message from the computer.

Typically, text-based programs look pretty plain. They are, well, just text! There is an icon on the top of the window can toggle between the chat and classic appearance.

Part 3 - Syntax Errors

As the semester progresses, you’ll learn that there are three different kinds of errors which can occur when creating and executing programs (and “executable flowcharts”). The first kind, and the only kind we’ll talk about right now, is a “syntax error”. In a “syntax error”, you provide some sort of command to the computer (specifically, here, to the Flowgorithm program), but Flowgorithm does not understand what you are asking it to do, that is, your command does not conform to the grammar (syntax) rules of Flowgorithm.

Since you are using Flowgorithm, all of the operations are selected from a graphical list. You won’t be making any syntax errors by putting in the wrong command. (Alas, it is all too easy to make such an error when writing programs in various programming languages.) But it is still possible to make a syntax error when you provide the output expression. Let’s make one, just to see what happens.

1. Using the current flowchart, double-click inside the first Output parallelogram (the one with “Hello, world!”). This opens the Expression dialog box.

2. Change the expression. Remove the opening double-quote mark. Then click “OK”.

Uh-oh. Flowgorithm is not happy. It does not understand the expression you put in. Although it really doesn’t care what you want to display, it wants it enclosed in double-quotes.

3. Fix it and execute again. All should be well.

Part 4 – Output of Numeric Values

Here’s a new, very small, algorithm (in pseudocode).

```
Display 1947
```

Hmm. There are no quotes around 1947. (This is the year that Sac State was founded). Let’s see what this looks like as a flowchart.

1. Don’t start a new flowchart. You have this ability by selecting “New” from the File Menu. But let’s not do that. We are going to modify your existing chart.

2. Add an output shape to the flowchart, just like you did last time. Put it at the end, below your name.

3. For the expression, type: 1947 (No quotes). (What is the significance of 1947, you may ask? It is the year Sac State was founded.)

4. Execute the flowchart. Hmm. That worked. Maybe we really don’t need those double-quotes after all.

5. Re-open the expression dialog for the Output operation and replace 1947 with CSUS. No quotes.

6. Execute the flowchart.
Dang it! The program didn't cause a syntax error, but it failed when you executed it!

You're going to have to wait awhile before you get a full explanation. It has something to do with variables, and we'll be covering those next time. For now, you need to know the following:

- If you want to print words and stuff like that, put the expression in quotes.
- If you want to print numbers, you don't need the quotes.

Well, gee, what if the expression is some of both? Sacramento State's mailing address is: 6000 J Street. That's got numbers and words.

7. Change the output expression to **6000 J Street** (No quotes)
8. Nope, it doesn't like that.
9. Put quotes around the whole expression.
10. Execute. There, that worked. I wonder what would happen if we put numbers (without words) in quotes.
11. Change the expression to **"1947"**. Use the quotes.
12. Execute. It looks like we can have numbers as the output expression with or without quotes. And that's true, uh, sort of.
13. Let's make it look a tad more official. Change it to **"Sac State was founded in 1947"**. Execute it.

**Part 5 – What It All Means**

The real truth is that numbers are no different than any other characters (like the characters that make up "Hello, world!")), **unless we want to do some arithmetic on them**. If we want to do arithmetic, then numbers inside quotes aren't going to work. The computer treats anything inside quotes as just "characters", keystrokes on the keyboard. If you want them to actually be numeric values, don't put them in quotes.

**DEFINITION: String**

In computer programming, we refer to a group of characters ("Hello, world!" or "6000 J Street ") as a string.

If we put a number in quotes, like "8", that's a string, too. It treats it just like any other characters.

(On a slightly more subtle level, if we mean for them to be "numeric", don't put them in quotes. If we mean for them to just be characters, do put them in quotes. The "6" in "6000" is just a character.

Next time, when we start working with variables, you'll find that there's a little more to this "when to use double quotes" stuff. Can't wait, can you?

**Part 6 – Let's try some arithmetic.**

1. Change the output expression to: **12 + 7** (No quotes)
2. Execute the flowchart. Did it treat the twelve and the seven as "numeric"? Sure it did.
3. Put the expression in quotes: **"12 + 7"**
4. Execute.

That should make sense. They aren't numbers anymore. It's just a string with the keystrokes 1 and 2 and space and plus sign and space and 7.

5. Let's make it a tad more meaningful. Enter your current age and add 5 to it. Use the + 5 rather than just doing the math your head. *(That should be the year you probably will graduate from Sac State.)*

6. Execute it. Did the math work?

### Part 7 – Joining Strings Together

Some of you might wonder what would happen if you put each of the numbers in quotes, i.e., you had an expression that looked like:

```
"12" + "7"
```

Well, you are welcome to try it, but what it does isn't exactly what the textbook will teach you. The book combines the two strings into the single string “127”. Not entirely unreasonable, but there is another symbol recommended for joining together two strings rather than the plus sign. That symbol is an ampersand (&). Let's try it.

1. Change the first output expression to: "Hello" & "," & "world" & "!"

2. Execute the flowchart. There shouldn't be any surprises. If it doesn't print *Hello, world!*, you probably typed something wrong. Go look carefully at where the quotes and ampersands are.

<table>
<thead>
<tr>
<th>DEFINITION: Concatenation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joining two strings together to form a single string is called <em>concatenation</em>.</td>
</tr>
</tbody>
</table>

But why would you go to all of that trouble?

…Well, since you asked....

It turns out that you can concatenate numbers (actual numeric values) and strings to form a string.

3. Add a new output shape with the expression: "XYZ" & 123

4. Execute the flowchart.

Flowgorithm (and some programming languages) will allow you to concatenate that way. Flowgorithm converts the number to a string and concatenates. No big deal, of course, you could have made the expression "XYZ123" and gotten the same result. But....

5. Change that expression to: "Twice my age is " & `<put your age here>` * 2

(Put your own age in place of `<put your age here>`.) Note that in computer programming, we use an asterisk for multiplication. And be sure to leave a blank at the end of the string (right after is, inside the quotes).

6. Execute. There. Concatenation of a calculated number and a string! Whoopee!!

I know this doesn't seem like all that big a deal, but as soon as we start using variables, it will become a very big deal. That'll have to wait for next time.