

#2 Madlibs

We are going to work on a new project, a game called Madlibs. It will give us some more practice dealing with variables.

We are going to follow a defined set of steps in programming this. They are:

1. Decide on the goal.
2. “Sandbox” to acquire necessary knowledge
3. Write pseudocode, which describes in English what we want to do
4. Code
5. Test
6. Do it all again, that is “enhance.”

Probably you have played Madlibs before, not using a computer. You interview someone, writing down a noun, the name of a vegetable, or whatever, using the person you are talking to as a random generator. Then you put the words into a story that you had already written, that has blanks for the noun, the name of a vegetable, and so on. The result is pleasantly nonsensical, and you can both enjoy it.

To do the project, we need to follow our programming steps!

1. Decide on the goal

In this step you need to think about exactly what you want the procedure to accomplish. There are always a few little decisions to be made, and now is the time to make them.

We will capture words from the user, save them, and then plug them into a story that prints out on Page1. We will use our knowledge of variables. We want to create a variable that will capture a word typed in by the user. Then we will keep the words stored until time to use them. So we will need as many variables as there are blanks in the story. Here is our assignment:

- Make a madlib using at least five variables.
- Ask the user to give us words to put in the variables.
- Print out a story using the five variables into a text box.
- Make a procedure called **madlibs** that works with a **Main** and a **Start** procedure.

2. Sandbox to acquire necessary knowledge

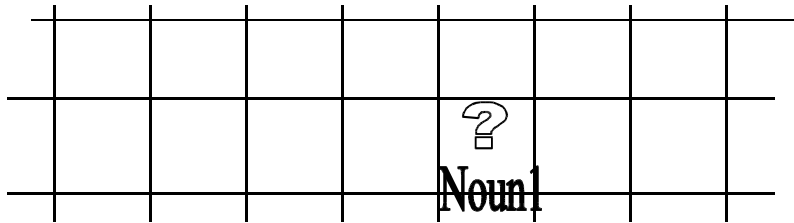
This step is a *learning* step. Here you look at your goal and see if there are some aspects that you need to learn more about before you can write the code. If so, then you write some “sandbox” or “throw-away” code to teach yourself what you need to know.

We need to know how to accept words typed in by the user. Here is an example. Suppose

we create and name a variable called **noun1**. On your Procedures Page, start a new procedure called **Madlibstest**. Then add a line to create a variable, **noun1**:

```
to madlibstest
  local "noun1
```

Here's what we have created:



Mailbox with Unknown Contents Called Noun1

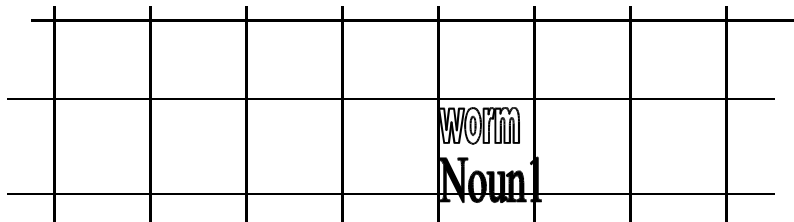
Now we type in this code:

```
question [What is your favorite animal?]
```

This procedure, **question**, asks the user a question, and gives space for an answer.

```
make "noun1 answer
```

The procedure **answer** outputs whatever the user just typed. The procedure **make** gives the variable **noun1** the value of answer. Let's say the user typed "worm." At this point, the variable **noun1** would contain "worm."



Mailbox With New Contents

Let's see what's going on. Your preliminary procedure looks like this:

```
to madlibstest
  local "noun1
  question [Please type in an animal name.]
  make "noun1 answer
  show :noun1 ;this means show what's in noun1
end
```

Go to the Command Center and type **madlibstest**. Type in an animal name as requested. Does the computer show the animal name at the bottom of the screen? It is supposed to.

But, let's make a text box and print our output in that. That way, we can use our game without using the Command Center. We would need that if we posted it on the Web. To make a text box, go to Page1. Click the icon box with ABC in it. Then click on the screen, and size your text box to make it at least half the size of the screen. You can right-click on the text box and tell it to hide the box name.

To print into the text box, we use the command **print**, instead of **show** for the Command Center.

The MicroWorlds help button index tells us this about **print**:

print

print (pr) word-or-list

Prints a word or list in the current text box. The text is followed by a carriage return and line feed sequence. See insert.

Example:

repeat 5 [print "hello]

The suggested format is **print word-or-list**. A *word* has a " mark right before it. A *list* is enclosed in [brackets]. So our input needs to have one or the other, " mark or brackets. Try these in the Command Center:

```
print "hello
```

```
print [hello]
```

```
print [[hello] [Mom]]
```

Can you get it to print "Hello out there!" ? Try it before you turn the page.

Our “Hello out there!” would look like this:

```
print [Hello out there!]
```

How do we get it to print out *what's in* a variable? On the Procedures Page, let's try this:

```
to test  
local "noun1  
make "noun1 "me  
print [Hello :noun1]  
end
```

Type **test** in the Command Center, and push <enter>. Oops!! It printed “Hello :noun1” in the text box! That's not what we want! We want *what's in noun1!*

Let's look at a procedure called **sentence**. Here's what Help has to say about **sentence**:

sentence (se) word-or-list1 word-or-list2

(sentence word-or-list1 word-or-list2 word-or-list3...)

Reports a list which is made up of its inputs (words or lists). Sentence can take more than 2 inputs when sentence and the inputs are enclosed in parentheses. See list.

Examples:

show sentence "a "b

a b

show (sentence "hi "there [Bill])

hi there Bill

This is telling us to use this format:

```
print (sentence input1 input2 input3)
```

Notice the parentheses that include the word **sentence** and all the inputs. Each of these inputs needs to be one of these:

1. a word, with a " mark
2. a list, using [brackets]

3. “what’s in” a variable, for example **:noun1**.

I want you to look at these commands, and guess which ones should work. Then test them in a test procedure. To make them work, first type this into the procedure:

```
to test  
  local [animal age]  
  make "animal "cow  
  make "age 13  
  {put test sentence here}  
end
```

To test each one, make sure you have a text box. Then type *test* in the Command Center to run the **test** procedure. You don’t have to keep typing *test*; just put the cursor right after the word and press enter. That will run it too.

1. **print (sentence "the :animal "ran)**
2. **print (sentence [The very large] :animal [ran through the woods])**
3. **print [Hello there] [How are you?]**
4. **print sentence [Hello there][How are you?]**
5. **print (sentence [Hello there] [How are you] [today?])**
6. **print (sentence [My age is] :age [years.])**

For grins, change the animal and the age in the variables and run them again. You could make a very large mouse run through the woods!

Now that we know more on this subject, let’s get interactive, and collect those variable contents from the user. In our **madlibstest** procedure on the Procedures Page, add these underlined lines:

```
To madlibstest  
  local "noun1  
  question [Please type in an animal name.]  
  make "noun1 answer  
  show :noun1  
  print (sentence "hello "there :noun1)  
  print (sentence [What are we talking about?] :noun1)  
end
```

Now go to the Command Center and type **madlibstest**. Push <enter>.

What do you get? Do some more experimenting.

We're well on our way to making a madlib!

Now, what if we want to create several variables at once? We'll need at least five variables. Let's guess that **local** might be able to help us. Look up **local** under the Help index. It says this:

local

local word-or-list

So we can use either a single word (with quote mark) or a list (with brackets) as an input for local. For instance, we can say:

local "fred

for one variable, or for several,

local [fred sam bob]

We need a list because we want to create five variables. Let's try this at the top of the **madlibstest** procedure:

local [noun1 noun2 noun3 noun4 adj1 adv1]

In fact, it is standard procedure in programming to create your variables at the top of the procedure, and then do the rest of the coding. So let's plan to do that.

We need one more procedure, one that allows us to talk to the user. Let's try **announce**. The Help menu tells us this:

announce

announce word-or-list

*Displays the message in an alert box. Clicking OK closes the box. See **question** and **answer**.*

So the input for **announce** can be a list of words, in brackets. Let's test this in the Command Center:

announce [Here is your madlib.]

Does it work? Let's add these ideas to our **madlibstest**. We can use these new tools: **question**, **print**, **sentence**, **local** using a list of variables, and **announce**.

To madlibstest

local [noun1 noun2 verb3 adj4 noun5]

question [Please type in an animal name.]

make "noun1 answer

announce [Here is your madlib.]

print (sentence [One day the] :noun1 [took a walk in the rain.]

end

Planning

The next thing you need to do for our **madlibs** procedure is write a simple story on a piece of paper, or in a word processor. Underline five words that you will be erasing, but don't choose verbs (action words). Verbs are too hard to get right when asking input from the user. Separately, write down a question for each word. (Remember, the question needs to ask the user for a word of the same type, not exactly the same word, or the madlib won't end up being funny.) Keep the questions in order. Next to each question, write down a good variable name for that underlined word, for example noun1, noun2, noun3, adj1, adv1, and so on.

3. Pseudocode

Pseudocode is just like program code except that it is written in English. It helps us make a quick and easy plan without worrying yet about getting details exactly right.

Go to the Procedures Page and type **to madlibs**, to begin your pseudocode. Put a semicolon at the beginning of the next line so we know it is pseudocode. First make pseudocode for creating and naming your variables. For example,

to madlibs

;create and name variables noun1, noun2, noun3, noun4, adj1, adv1

Your second line of pseudocode will ask the user a relevant question for the first missing word, such as, "Please describe a feeling," or "Give me an adverb, such as slowly, quickly, heavily, etc." Write that pseudocode, with semicolon. For example,

;ask the user to type an animal name

Your third line of pseudocode will take the word from the user and store it in the first

variable. For example,

;Store the animal name in a variable named noun1.

Do the same for the next variable, and the next until you have five variables. Write these lines of pseudocode.

Your next line of pseudocode will announce, “Here is your madlib.” Write this.

Finally we have the story-writing section. Write pseudocode lines for printing out your story into the text box, inserting the right variables in the right places. For example,

;print “There once was a noun1. One day The noun1 went to the park and saw a noun2.”

Now, show your teacher. Your teacher can compare it with the pseudocode in the answer key and give you pointers.

4. Code

Translate your pseudocode into code, that is, into Logo. In other words, after each line of pseudocode with a ; in front of it, put a real line of code that says the same thing in Logo. Your code should be based on our latest version of **madlibstest**. Look at **madlibstest** to figure out how to translate each line of pseudocode into code.

5. Test

Test your creation by typing **madlibs** in the Command Center. Then make a **Main** procedure that calls **madlibs** and a **Start** button that calls **Main**. Make a start button. Does it work? Now you have a complete program. Check with the answers in the back of the book if you are having trouble.

Exercise:

Make a second Madlib with a different story and different variables, totaling 7 variables instead of 5.



start

I wake up in my Delaware and my personal gravity is reversed. I am stuck to the ceiling. I have no way to get to the floor. Then a bell. Bump. Kids come splashing down the hall. So I guess it is 2 p.m. But I remembered that I can't go outside. Just then my personal gravity goes back to normal for just enough time for me to go outside. I grip the car for dear life. My hands slip and I begin to skating upwards. I can't stop! Just then I wake up. I was just dreaming I look up. Its true I am stuck to the roof! I try to find a way to get to the floor. I see a ladder. I climb down it. When I get to the bottom I try find something to grab onto. I see a basket. I grab it and accidentally drag it up to the roof with me. I get the phone and call the Alphas. They bring a ladder to get me down. As they get me down they tie an elephant to me. After several minutes my personal gravity goes back to normal.
THE END

Stephen's Madlib results