BYTES AND BEATS

An Introduction to Programming with MATLAB

Instructor Guide

Module 4: Using Variables

Prerequisite Domain Knowledge: Basic MATLAB® calculations

Expected Completion Time: 50 minutes

Table of Contents

Creating and Using Variables

Expected Duration: 35 minutes

Learning Objectives

- Use Live Scripts to easily reuse and modify a sequence of commands.
- Create and use numeric variables.
- Understand the rules for creating, combining and renaming variables.

Motivation

We know how to perform calculations in MATLAB using numbers. What if we want to change some values? Do we have to write all the commands again?

Is there a way for MATLAB to remember numbers or calculations that we will use repeatedly in our program?

Materials

- MATLAB
- Worksheet “Calculations in MATLAB”

Solution

```
open DinerShares_sample_solution.mlx
```

Steps

PART A – Worksheet

- Tell the students that they will work in teams for this activity. Pair up the students and give one worksheet ("Calculations in MATLAB") to each pair.
- Suppose that each pair went out to eat at a diner with their friend, Hulk and wanted to calculate the price that everyone must pay.
- Have the students decide their order from the menu in the worksheet and then calculate the cost (by hand) in the table provided.
- Now, tell them to use MATLAB’s **Command Window** to make the same calculations and fill the last column of the table.
- Now, have them change the price of soda to $1 in their calculations in the last two columns of the “Calculations in MATLAB” worksheet (by hand and then in MATLAB).
- The aim is to get the students to realize that it is tedious to have to find who ordered soda and change the calculations everywhere.
- For the MATLAB calculations, have them realize that it would be nice if they did not have to type their commands again when we change the price of soda.

Wouldn’t it be great if they could just change one value instead of typing each command again?

The **Command Window** does not let you go back and change only one value in your commands. You must type the whole command again. In the next part, we will learn how to make MATLAB reuse our sequence of commands and remember our values/numbers.

PART B – Live Scripts and Variables

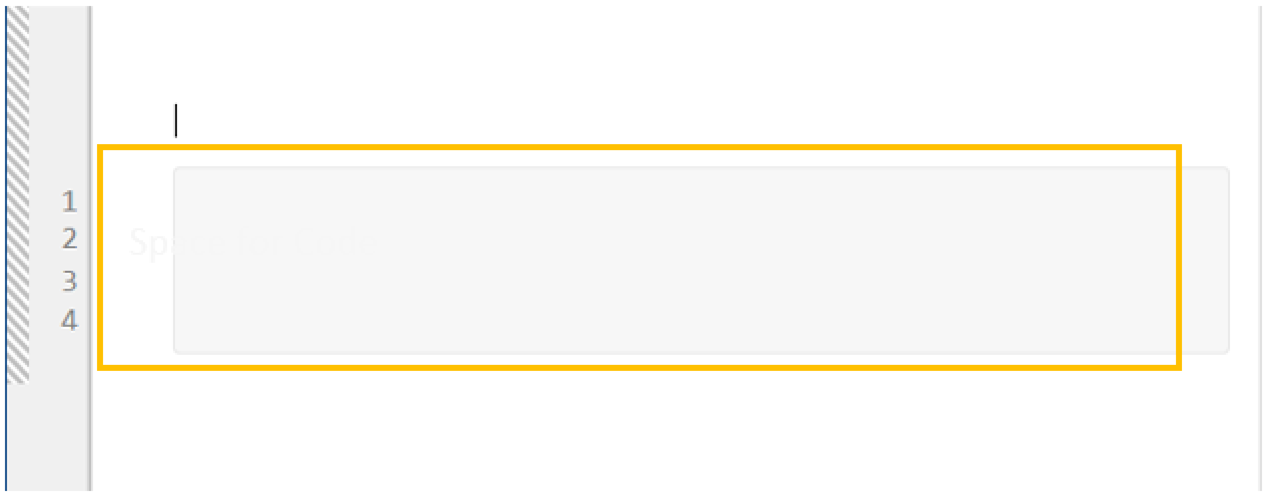
- In the MATLAB Command Window, have the students type the following command:

```
>> open DinerShares.mlx
```

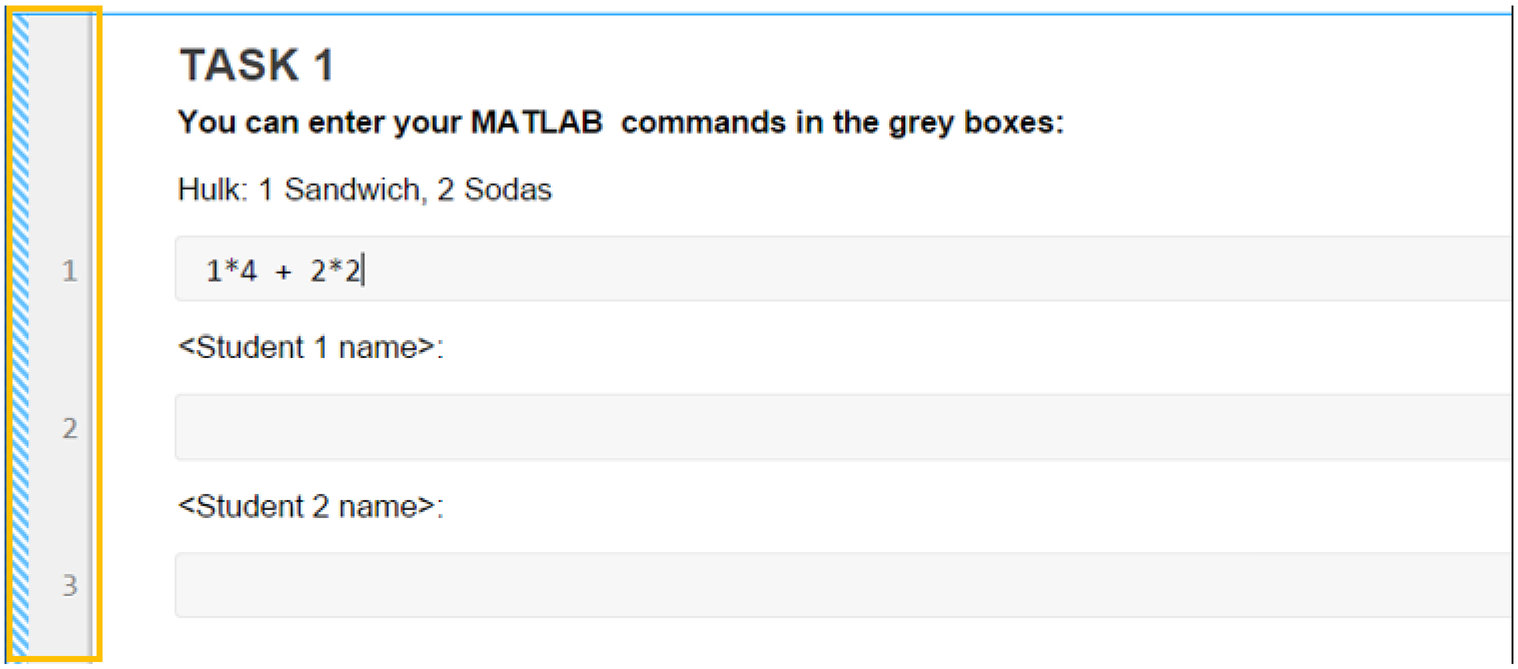
```
open DinerShares.mlx
```

- This will open a **Live Script** named ‘DinerShares’ in MATLAB’s **Live Editor**. The extension ‘mlx’ indicates that this is a Live Script.

- Notice that it contains a picture, some text and instructions. We will learn more about how to interact with a Live Script over this activity.
- Explain that a script is a sequence of commands that can be easily reused and changed. MATLAB has '**Live Scripts'** which let you add pictures and other text along with all your commands.
- You can enter code/commands in the grey space and any other text in the white space. MATLAB will only run commands that are typed in the grey boxes. It ignores everything in the white area.



- Tell the students to enter the MATLAB Commands for calculating costs in the space provided in the **Live Script** (these commands are same as the ones in the last column of the Worksheet).
- Use the lines numbered 2 and 3 to enter these. The first one is already filled for you.



TASK 1

You can enter your MATLAB commands in the grey boxes:

Hulk: 1 Sandwich, 2 Sodas

```
1   1*4 + 2*2
```

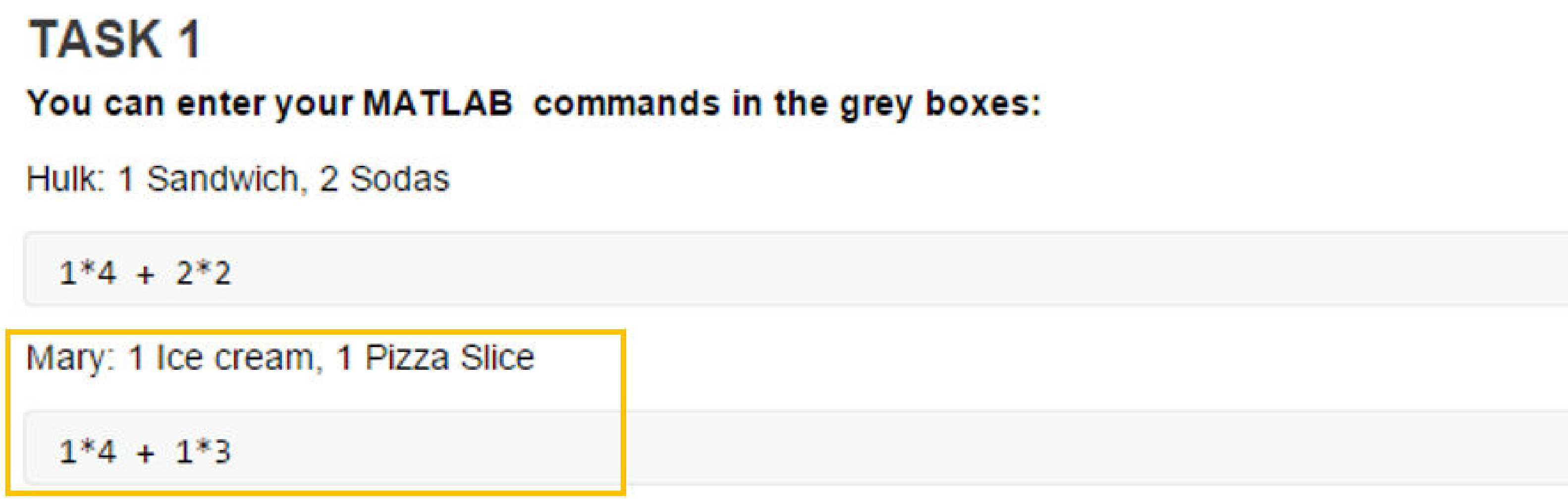
<Student 1 name>:

```
2
```

<Student 2 name>:

```
3
```

- Shown below is an example of what a student named Mary might enter in line 2:



TASK 1

You can enter your MATLAB commands in the grey boxes:

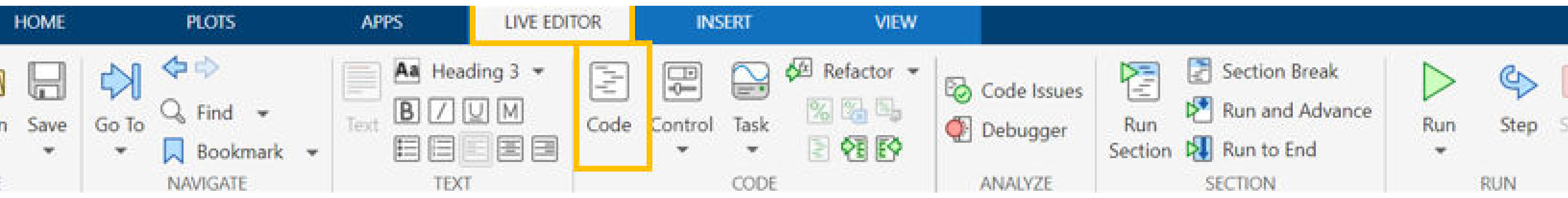
Hulk: 1 Sandwich, 2 Sodas

```
1*4 + 2*2
```

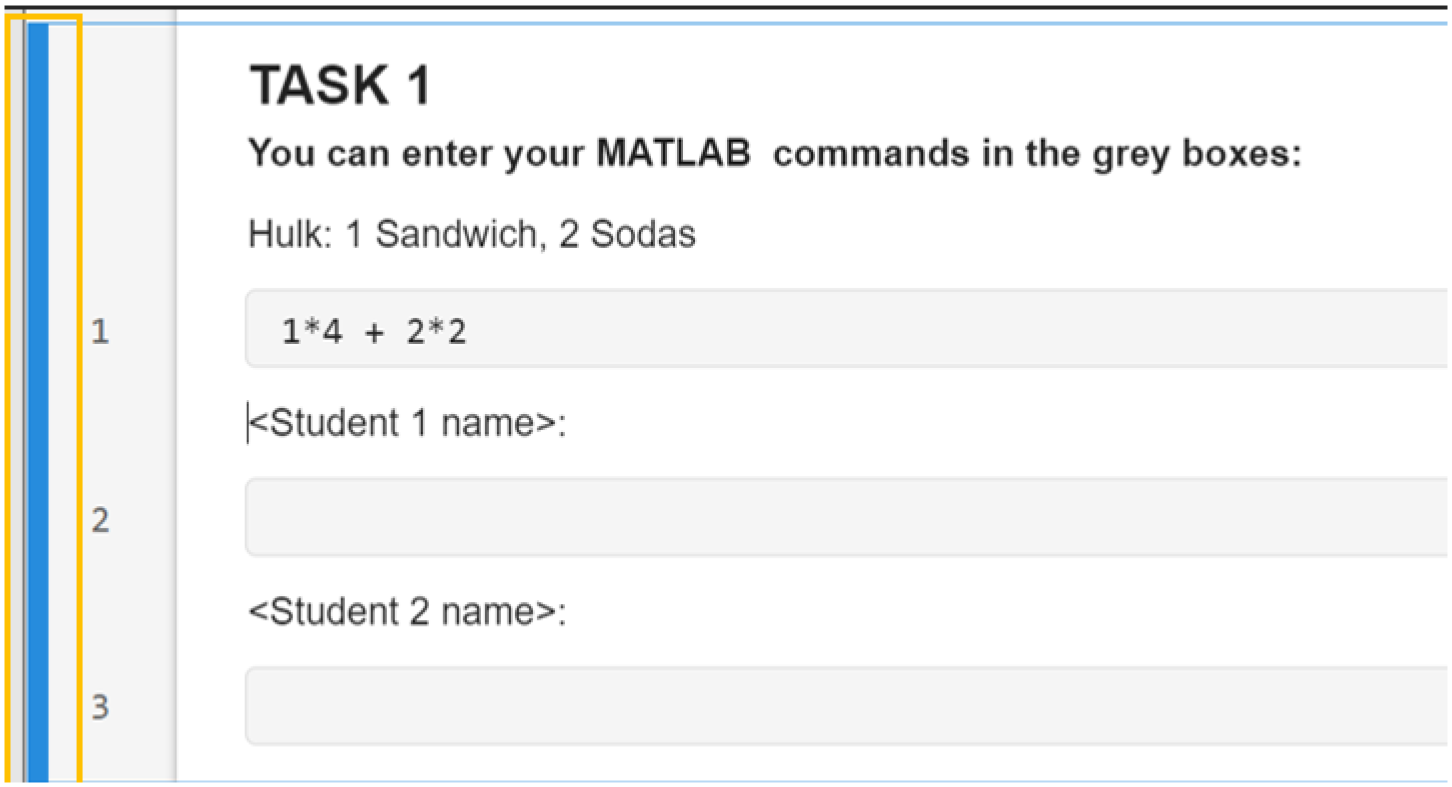
Mary: 1 Ice cream, 1 Pizza Slice

```
1*4 + 1*3
```

Note: If a student accidentally deletes a code line, you can insert code lines back by clicking on the 'Code' button in the 'Live Editor' tab. Similarly, you can insert text lines by using the 'Text' button.



- The code in this script is divided into sections. To run the section of commands for Task 1, students can do one of these two things:
- Click on the vertical blue bar to the left of the code.



TASK 1

You can enter your MATLAB commands in the grey boxes:

Hulk: 1 Sandwich, 2 Sodas

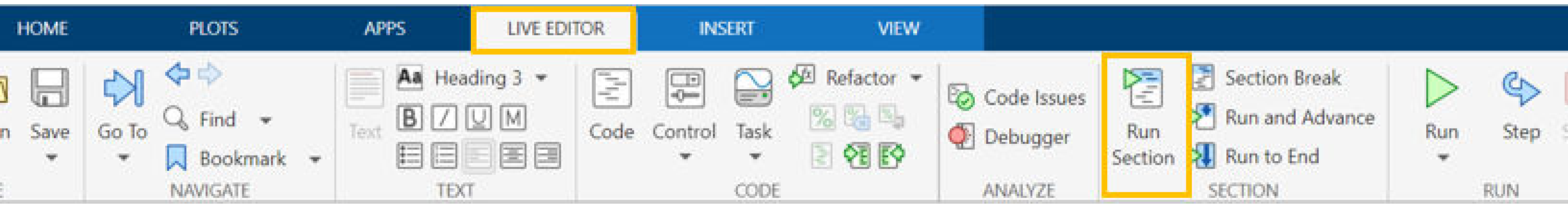
```
1*4 + 2*2
```

<Student 1 name>:

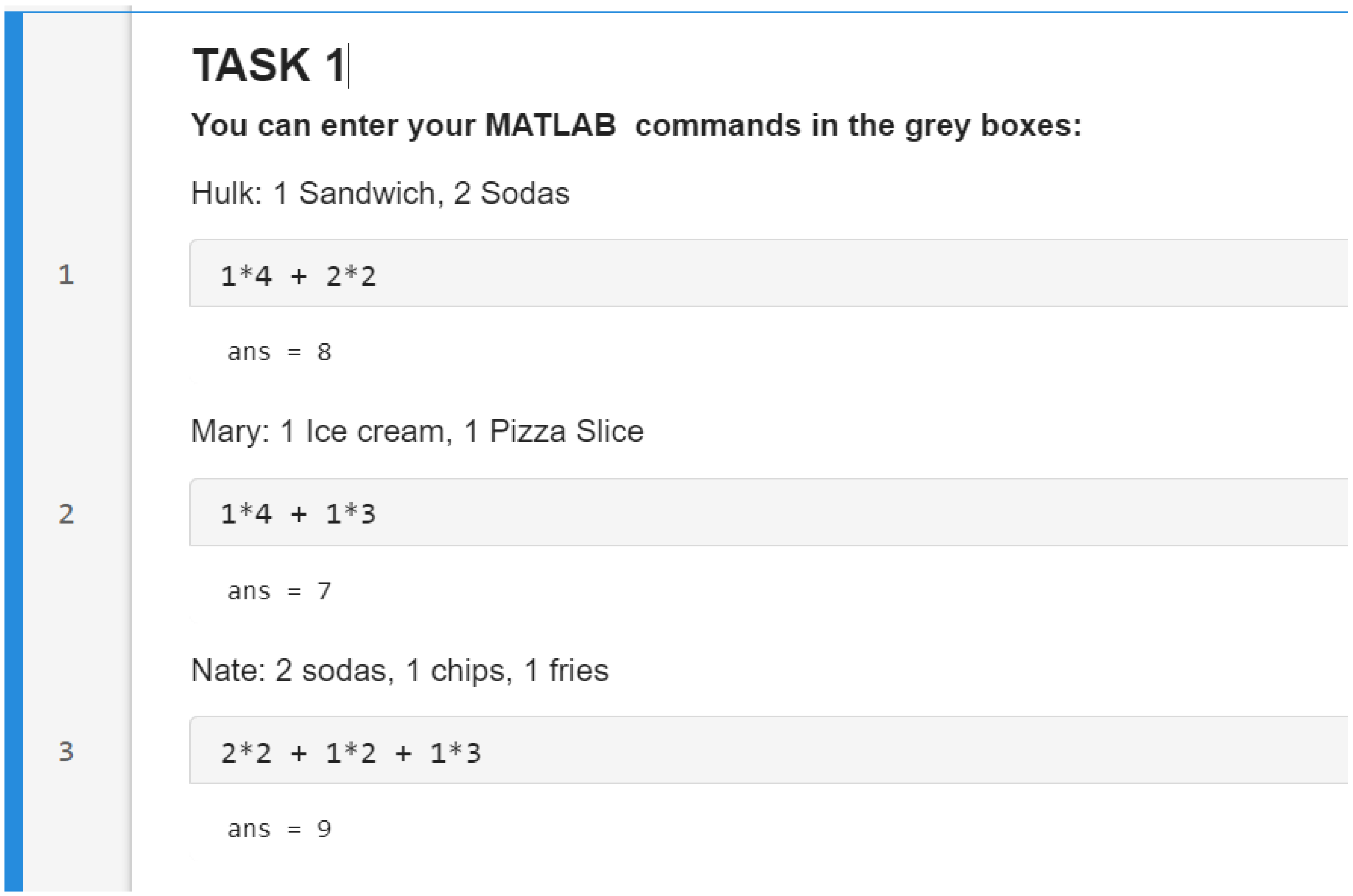
<Student 2 name>:

OR

- Click anywhere in the Task 1 section (the section should be enclosed by two blue lines as above), then click 'Run Section' button in the '**Live Editor'** tab's 'Section' section.



- Here is a sample output after a student enters all their commands and runs the completed section:


TASK 1

You can enter your MATLAB commands in the grey boxes:

Hulk: 1 Sandwich, 2 Sodas

```
1*4 + 2*2
```

```
ans = 8
```

Mary: 1 Ice cream, 1 Pizza Slice

```
1*4 + 1*3
```

```
ans = 7
```

Nate: 2 sodas, 1 chips, 1 fries

```
2*2 + 1*2 + 1*3
```

```
ans = 9
```

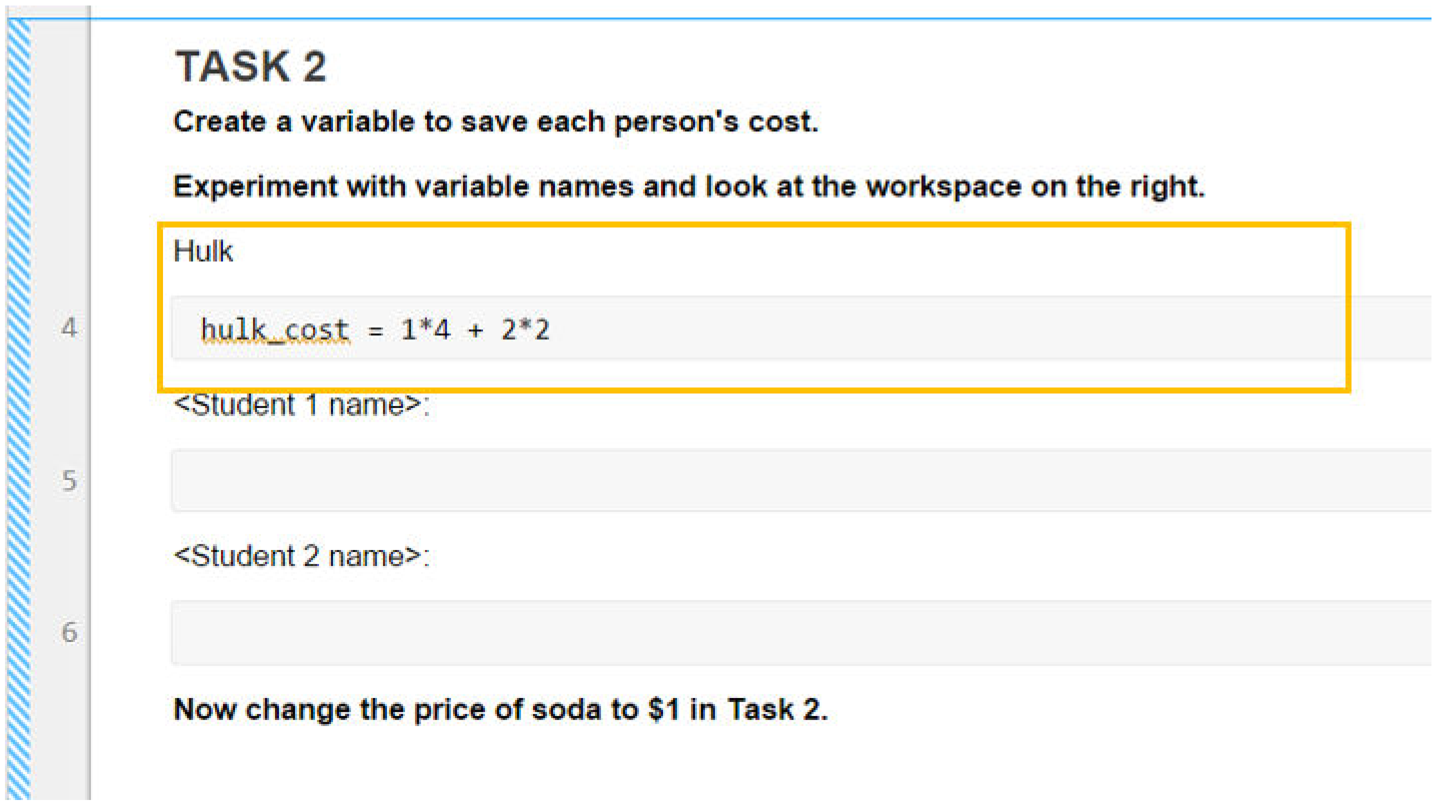
- Notice that the output/answer is displayed in the white space after each command.

Clear this output by right-clicking anywhere in the script and selecting 'Clear All Output'

- Turn the students' attention to the fact that every answer below the code is `ans` and a value.
- Tell them that MATLAB is automatically saving their results with a name `ans`. This is called a "*Variable*", because the value in `ans` keeps varying after every command.
- Explain that a *variable* is like a box/container that we use to store our values or results. A *Variable* has a Name and a Value. We can just recall the Name when we want to use the Value stored in it and we can change its Value at any time in our program.
- So, given that `ans` is last equal to Nate's total, there is presently no way to recall Hulk or Mary's total using a name.

Have the students go back to the Live Script that is open. Tell them that we will create some variables in Task 2. Perform the same calculations as we did in Task 1, but this time save the answers to a different variable.

- The variable `hulk_cost` has been created for example in line 4.



TASK 2

Create a variable to save each person's cost.

Experiment with variable names and look at the workspace on the right.

Hulk

```
4    hulk_cost = 1*4 + 2*2
```

<Student 1 name>:

```
5
```

<Student 2 name>:

```
6
```

Now change the price of soda to $1 in Task 2.

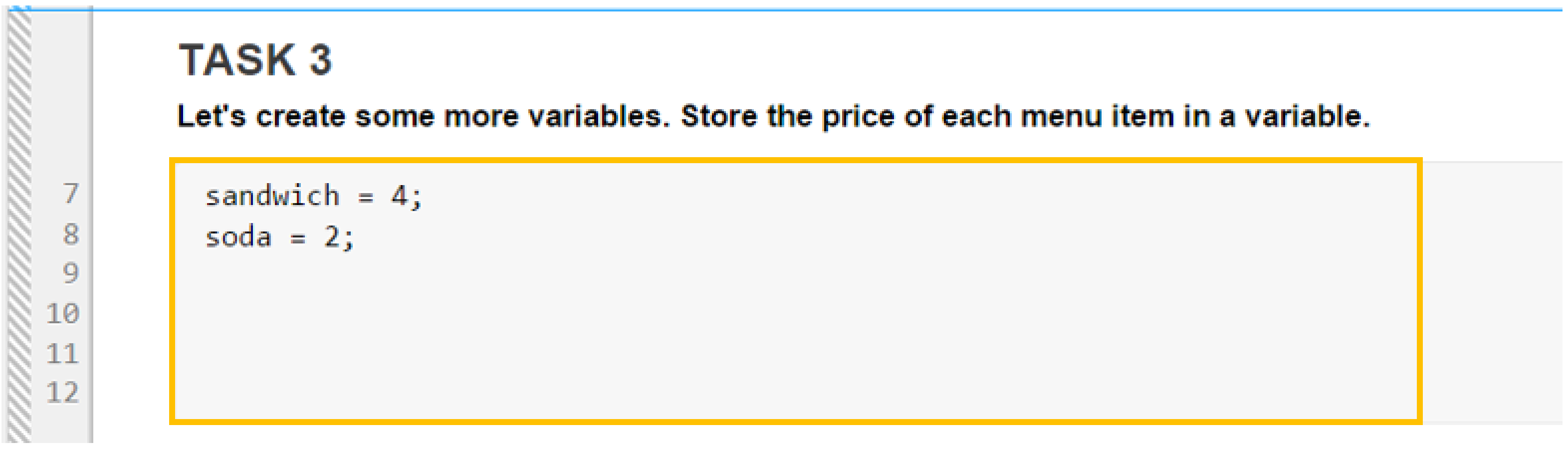
Similarly, tell the students to create two variables, one to store their cost at line 5 and another for their partner's cost at line 6.

Let them experiment with variable names. Make them meaningful and easy to recall based on the value they contain. \

Note: The rules for creating variable names can be found here: https://www.mathworks.com/help/matlab/matlab_prog/variable-names.html. They are also discussed below. Use these guidelines to help students create their variables.

Once complete, run the section of code in Task 2.

- Change the price of soda to $1 at the end of Task 2 and run the section again. This is much easier since you do not need to type the full set of commands again.
- In Task 3, students will create variables to store the prices listed on the menu.
- For example, the variables `sandwich` and `soda` have been created for them at lines 7 and 8. Their values are equal to the menu price of these foods.



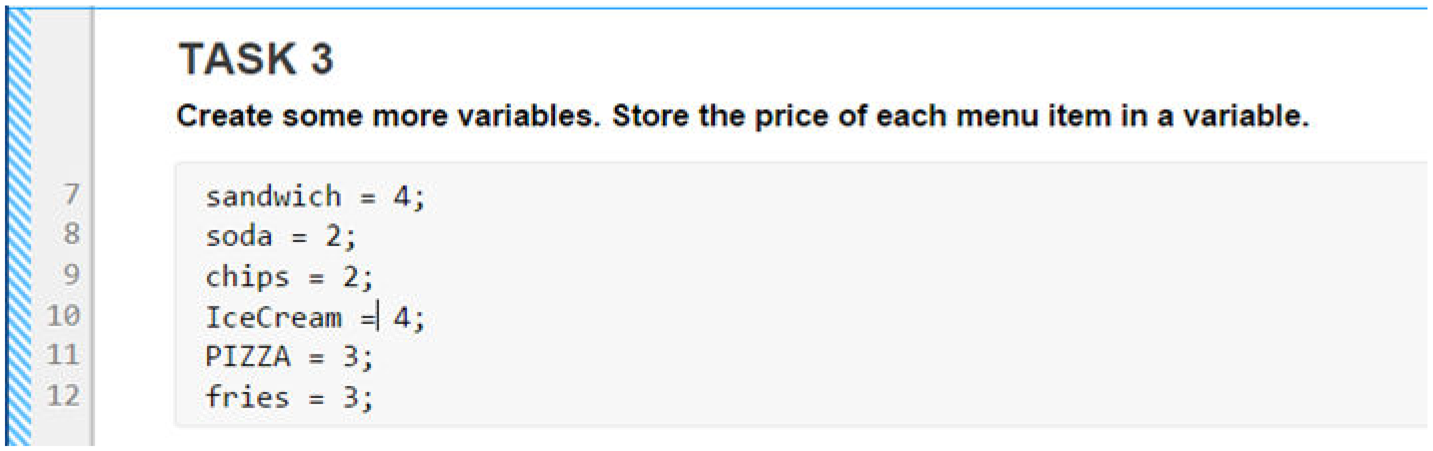
TASK 3

Let's create some more variables. Store the price of each menu item in a variable.

```
7   sandwich = 4;
8   soda = 2;
9
10
11
12
```

- ***Notice the semi-colon at the end of these lines. This will tell MATLAB that you do not want to display the output of these lines when you run them.***

- Tell the students to similarly create variables at lines 9 to 12 to store the prices of fries, pizza slice, ice cream and chips. They can use meaningful names for these variables and experiment with names. They may see an error when they do not follow certain rules for creating variable names. We will go over these rules later. Once completed, the script should look like this:

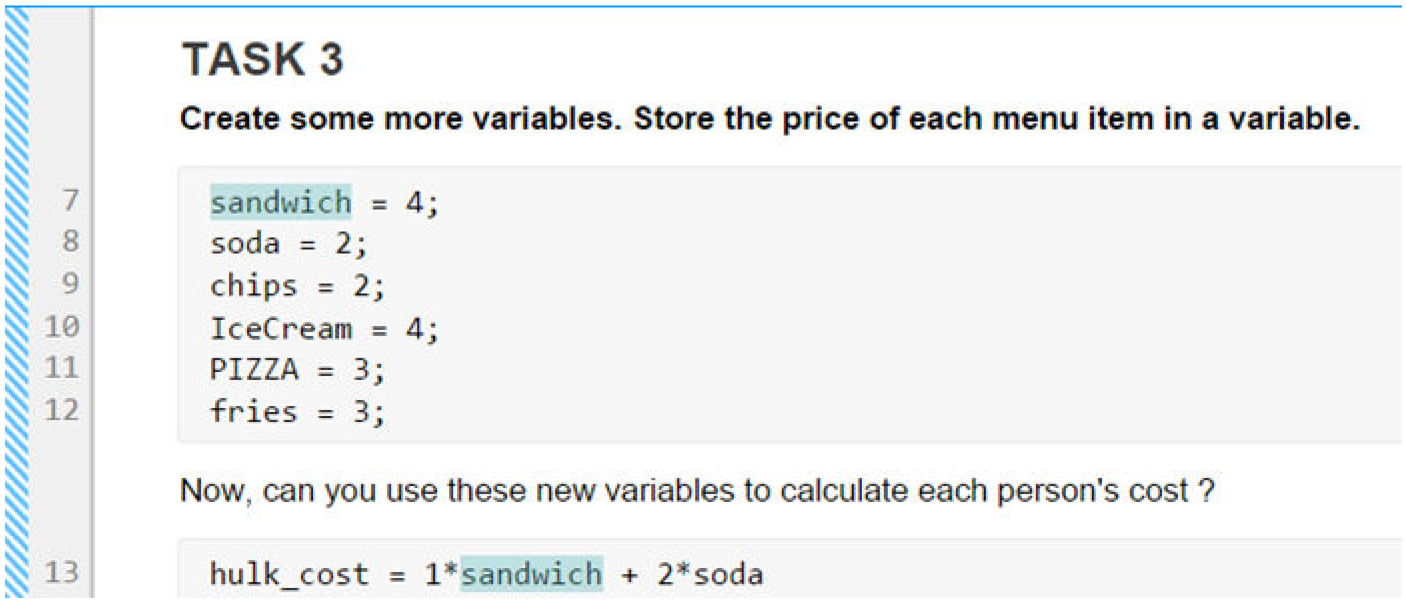


TASK 3

Create some more variables. Store the price of each menu item in a variable.

```
7   sandwich = 4;
8   soda = 2;
9   chips = 2;
10  IceCream = 4;
11  PIZZA = 3;
12  fries = 3;
```

- We can use the values we save in variables to make further calculations by just using the name of the variable instead of the number.
- Line 13 has already been entered for example. Discuss with them what they think line 13 is doing. They can get a hint by clicking on the name `sandwich` in this line. This will also highlight `sandwich` at line 7 where they saved a value in the variable.



TASK 3

Create some more variables. Store the price of each menu item in a variable.

```
7   sandwich = 4;
8   soda = 2;
9   chips = 2;
10  IceCream = 4;
11  PIZZA = 3;
12  fries = 3;
```

Now, can you use these new variables to calculate each person's cost ?

```
13  hulk_cost = 1*sandwich + 2*soda
```

- Tell them to similarly enter lines 14 and 15. This is the first time that the students will use existing variables on the right side of the '=' sign.
- **Note:** Line numbers may not match if students end up adding extra lines anywhere.
- Run the section of code in Task 3.
- Change the price of soda to $1 at the end of Task 3.
- Discuss this with the students.
- This time, changing the price of soda to $1 is a lot easier. You simply change the value assigned to the variable `soda`. You do not have to remember who ordered soda and change it in every calculation.
- The last line of the code is to use these variables to add all the costs and save them to a new variable.
- E.g. If their variables for Student 1 and Student 2 costs are called `my_cost` and `Other_cost`:

```
Total3 = hulk_cost + my_cost + Other_cost
```

The students can close this Live Script once completed.

Test Understanding of Variables

Expected Duration: 15 minutes

Materials

- MATLAB

Learning Objectives

- Understand reassignment of variables

Motivation

Students should now be comfortable with creating new variables and assigning them values. A further advancement of this concept is to reassign variables during course of a program.

Steps

- Tell the students to consider the following command and to think about what will happen for a moment, before you try it together:

```
>> soda = soda + 1
>> soda
```

- Some students might think that this will create an error because no number equals itself plus one. Others may see this as a valid command.
- After briefly discussing what might happen, execute the code and check the value of `soda` in the **Workspace**.

Note: Reminder that the Live Script should be closed and you should now be working within the **Command Window.**

- You <u>**can**</u> reassign variable names, but this concept is potentially a conceptual pitfall, so spend time to make sure the students understand what is happening.
- Variable names are just name tags attached to numbers. We can remove that name tag and put it on any new number we like, even on that number itself plus one.
- To put it another way, MATLAB computes the right side of the *assignment operation*. The result is just a number—with MATLAB it does not matter where that number came from. MATLAB then saves that number in the name on the left side. It also does not matter what was there before—it gets overwritten.
- Test for understanding by asking the students what the results of the following commands will be as you execute them one by one (the last line is the trickiest one):

```
>> soda

>> soda = soda + 1

>> soda = soda + 1

>> soda = soda - 2

>> sandwich = soda*2

>> soda = 1

>> sandwich
```

- Some students may think that `sandwich` should have a value of 2 at the end. However, tell them that no new calculation was performed to change `sandwich`*'s* value. On one of the lines above, a number was computed with `soda*2`, and it was assigned to `sandwich`. Even though the value of `soda` changes, the value of `sandwich` will not because there was no assignment operation made to change it.
- Demonstrate this in the **Command Window**.