BYTES AND BEATS

An Introduction to Programming with MATLAB

Instructor Guide

Module 5: Shopping for our Musical Party

Prerequisite Domain Knowledge: Basic MATLAB® calculations, variables

Expected Completion Time: 50 minutes

Table of Contents

Shopping for our Musical Party

Expected Duration: 50 minutes

Learning Objectives

- Revisit variables.
- Learn about data classes – string, logical and double.
- Understand when and how to create vectors.

Motivation

Variables are useful to make our program remember values we want to use during its course. But sometimes, we might end up making too many variables and it can become difficult to manage them. Is there a way to group similar types of data?

Materials

- MATLAB
- Worksheet “Party Planning – Shopping List”

Solution

```
open ShoppingList_sample_solution.mlx
```

Steps

Part A – Task 1

- Tell the students that over the next few activities we will write a program to create a shopping list for our musical party.

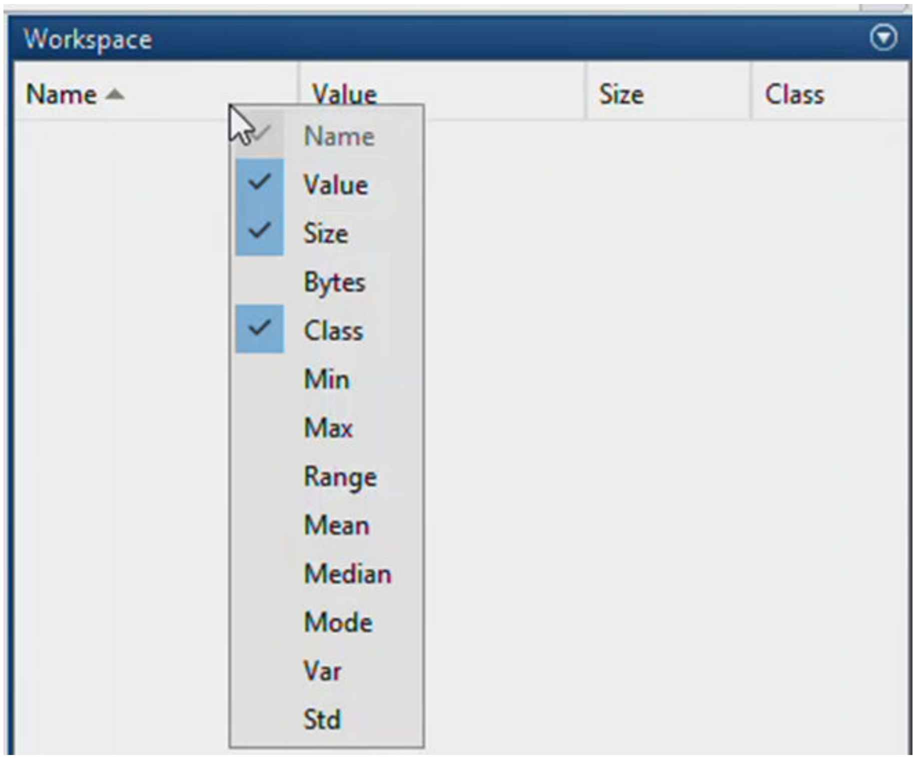
There are some party items to buy and we have a choice of two stores – ‘Utopia’ and ‘Paradise’. We want to keep within a budget and spend the least we can.

We will create a MATLAB program to create a list of items we should purchase from ‘Utopia’ by comparing prices of our shopping list for the two stores. We can then buy the remaining items from ‘Paradise’.

- Before moving ahead with the exercise, we will first set the **Workspace** to display Size and Class of variables along with the Value.

To display these, tell the students to go to the **Workspace** browser, right-click on an existing field like ‘Name’ or ‘Value’ and select the additional required fields in the context menu.

Every time the students create a variable or run some code, encourage them to observe the **Workspace** and check these columns.



- Give the students a copy of the Party Planning Worksheet which contains the list of items to choose from and their prices at both stores.

Tell the students to open the Live Script '*ShoppingList.mlx*' using the command below:

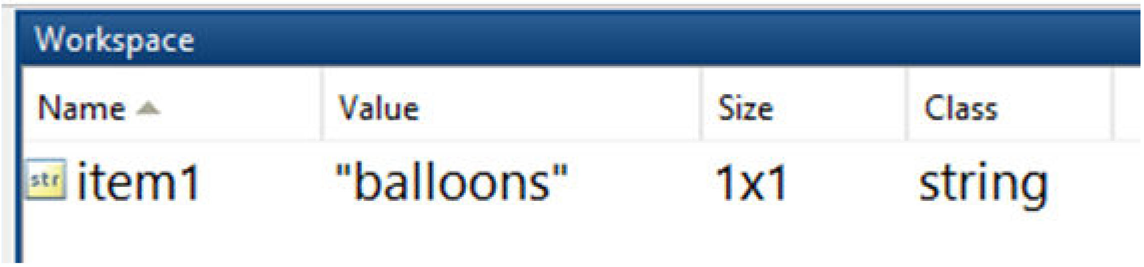
```
>> open ShoppingList.mlx
```

```
open ShoppingList.mlx
```

- At this point, the students should be familiar and comfortable with variables.
- Go to Section 1 in the code. Tell the students to observe the first line:

```
item1 = "balloons"
```

- Tell them to turn their attention to your screen while you execute this command in your **Command Window**
- Then, look at the **Workspace** and tell the students to observe the **Class** of this variable. It is a 'string'.

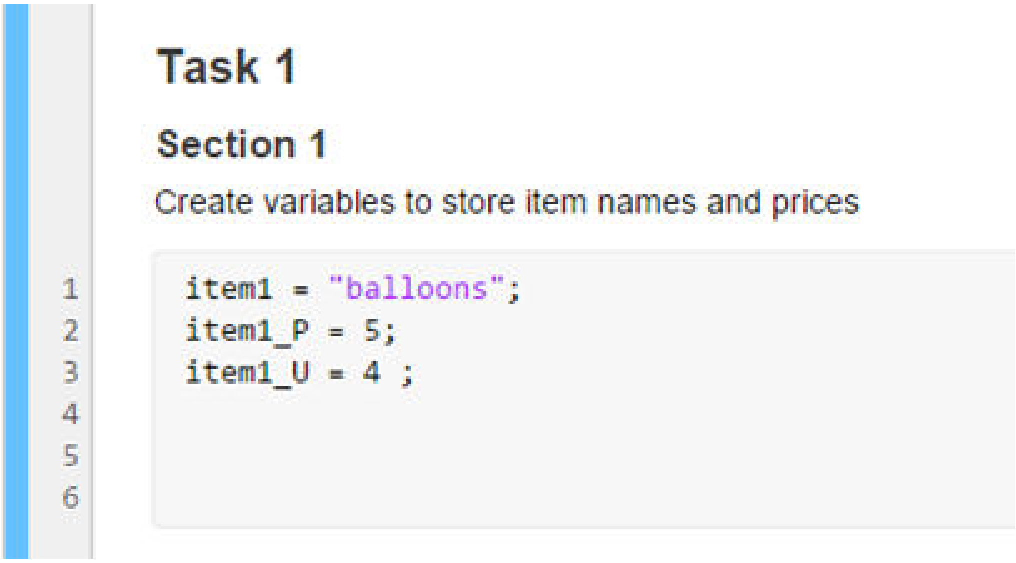


Name	Value	Size	Class
item1	"balloons"	1x1	string

- Tell them that until now, we have created numeric variables or variables that store numbers as their value. In MATLAB, you can also store names or text or words or sentences in a variable. This is called a 'string'. To make MATLAB store a value as a string, you must enclose the value in double quotes.
- Tell the student to try it out in their **Command Window** by creating a variable that stores their name as a string. For e.g.

```
my_name = "Hulk"
```

- Ask the students what they think the three lines of code in Section 1 are doing.



Task 1

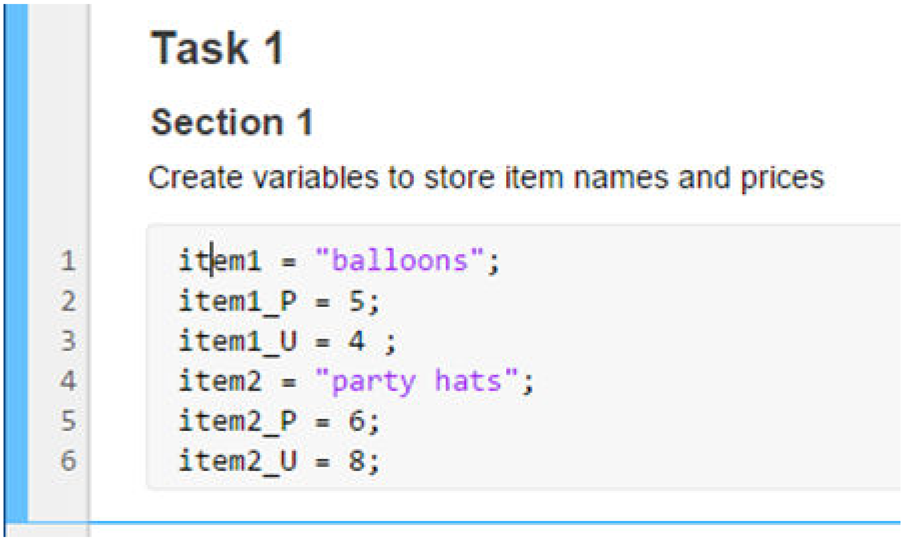
Section 1

Create variables to store item names and prices

```
item1 = "balloons";
item1_P = 5;
item1_U = 4 ;
```

- Ans: **`item1`** is saving the name of the first item we want to buy (balloons). **`item1_P`** is saving its price from the ‘Paradise’ store. **`item1_U`** is saving its price from ‘Utopia’.
- Tell them to run this section and observe the **Workspace**.
- Ask them what the class of numeric variables is. It is ‘double’.

- Below the given code, tell the students to use the space provided in Section 1 to create three more variables to save name and prices for party hats.
- They can refer to the ‘Party Planning - Shopping List’ worksheet to look up prices for hats.



Task 1

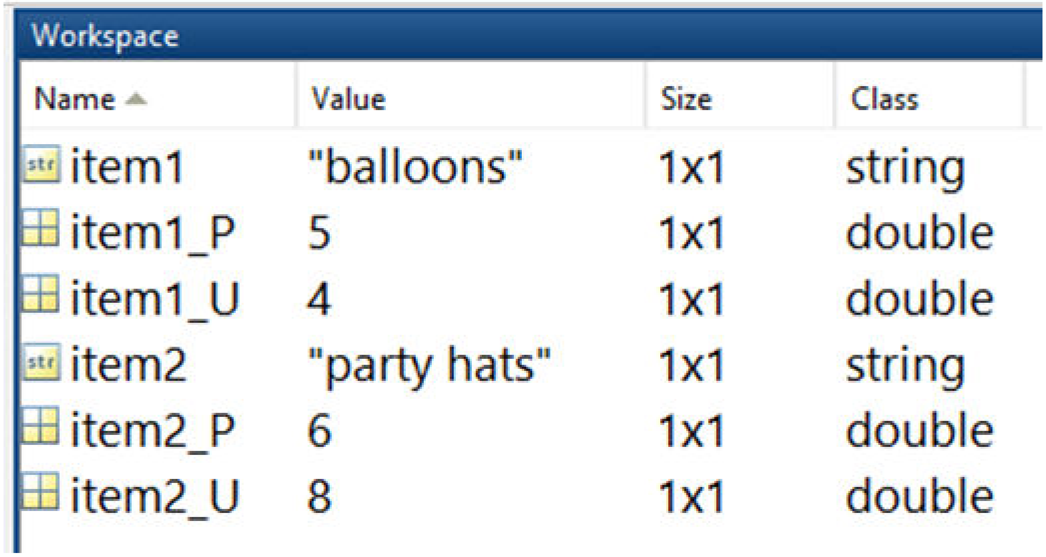
Section 1

Create variables to store item names and prices

```
item1 = "balloons";
item1_P = 5;
item1_U = 4 ;
item2 = "party hats";
item2_P = 6;
item2_U = 8;
```

```
item1 = "balloons";
item1_P = 5;
item1_U = 4 ;
item2 = "party hats";
item2_P = 6;
item2_U = 8;
```

- Now run the completed section and observe the **Workspace**.



Workspace

Name	Value	Size	Class
item1	"balloons"	1x1	string
item1_P	5	1x1	double
item1_U	4	1x1	double
item2	"party hats"	1x1	string
item2_P	6	1x1	double
item2_U	8	1x1	double

- Tell the students to turn their attention to your screen after completing Section 1.
- We have two items and respective prices in our shopping list. The students will now perform an operation in MATLAB to find out which item is cheaper in Utopia.
- This is what the code given in Section 2 is performing for item1.

On your screen, type the following line in the **Command Window** while discussing with the students what it does. This is a new operation which introduces a new class of variables. While typing the next line, explain that this operation is for comparing prices. The ‘<’ operation will display ‘1’ if the value on the left is less than the value on the right and ‘0’ otherwise.

```
buy1_U = item1_U < item1_P
buy2_U = item2_U < item2_P
```

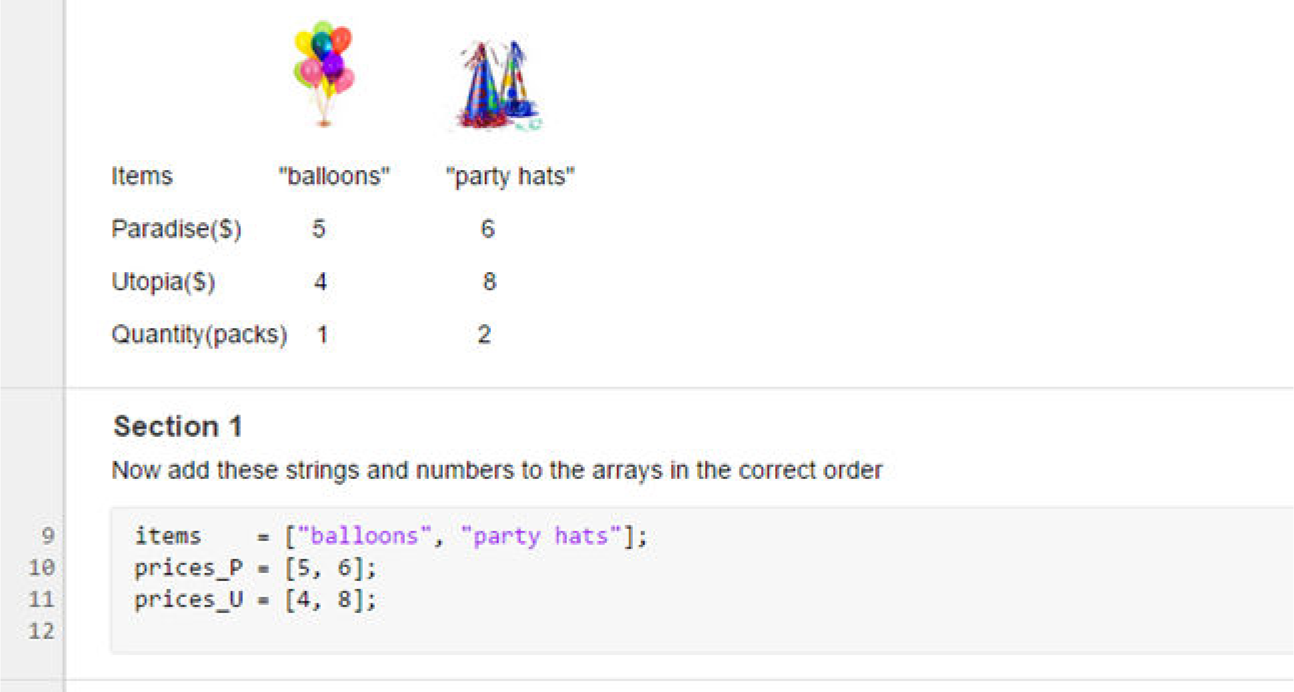
- Execute this line in your **Command Window** and observe that the value returned is 1 and the word ‘logical’ appears above it. Look at the **Workspace** and observe the class of the variable. It is ‘logical’.
- Ask the students about the other classes we have seen so far. They should remember ‘double’ and ‘string’. Tell them that ‘logical’ is another type which means that the value can be either ‘true’ or ‘false’.
- MATLAB returns logical 1 when the expression is **true** and logical 0 when the expression is **false**. Spend some time observing the values of ‘**item1_U**’ and ‘**item1_P’**to understand what the value in ‘**buy1_U**’ indicates. It is logical 1 i.e. true and indicates that we should buy this item from Utopia.
- Tell the students to go back to their Live Script and perform the similar calculations for item 2 (party hats) in the space provided in Section2.
- Now tell them to run the section and then ask the class whether they should buy item 2 in Utopia. The answer is **no** since the comparison returns logical 0.
- Once everyone has completed this, bring their attention to you and start this discussion.
- Ask the students:
- How many variables did we create so far to compare prices for only two items? Answer is 8.

- How many more will we need if we add 3 more items to our shopping list? Answer is 12. So, the total is 20!
- There is a repetition in the computations we are performing for each item. Would it be better if we could do this for all items in one or two lines instead of repeating the same process?

They should realize that this is tedious. Tell them that there is a way in MATLAB to use very few variables and perform the calculations all at once. If we rearrange the items and prices, we can group similar values and put them in one variable. This is called a vector or an array. This is what we will do in the next Task.

Part B – Task 2

Go to Section 1 in Task 2. Ask the students to describe the code in this section.



Items	"balloons"	"party hats"
Paradise($)	5	6
Utopia($)	4	8
Quantity(packs)	1	2

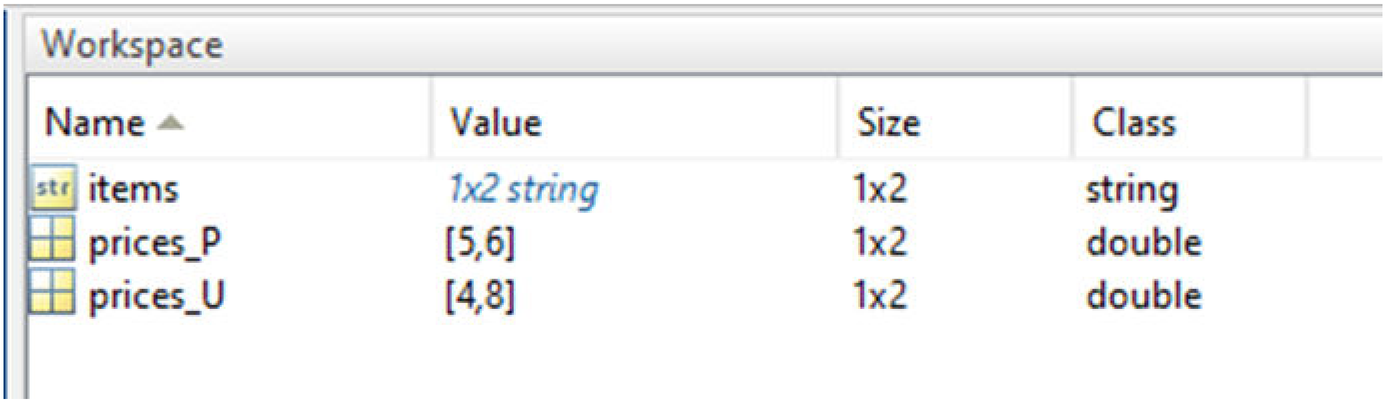
Section 1

Now add these strings and numbers to the arrays in the correct order

```
items    = ["balloons", "party hats"];
prices_P = [5, 6];
prices_U = [4, 8];
```

- How many variables are created?
- 3 variables: *items, prices_P, prices_U*
- What is stored in them?
- Look at the table above this section to draw parallels- `items` stores the names of 2 items, `prices_P`stores their prices in Paradise and `prices_U`stores their prices in Utopia. Note that the order in which the prices are stored is the same as the order of item names in *`items`*. This is important for our computations later.
- How are the values stored?
- Each value is separated by a comma, and everything is enclosed in square brackets - `[]`.
- Tell the students that these are called vectors. Vectors are used to group and store values of the same Class.

- Execute this section and observe the Value, Size and Class of the new variables created in the **Workspace**. The size of each of them is `1x2`. This indicates that each of these variables has 1 row with 2 values.



Workspace

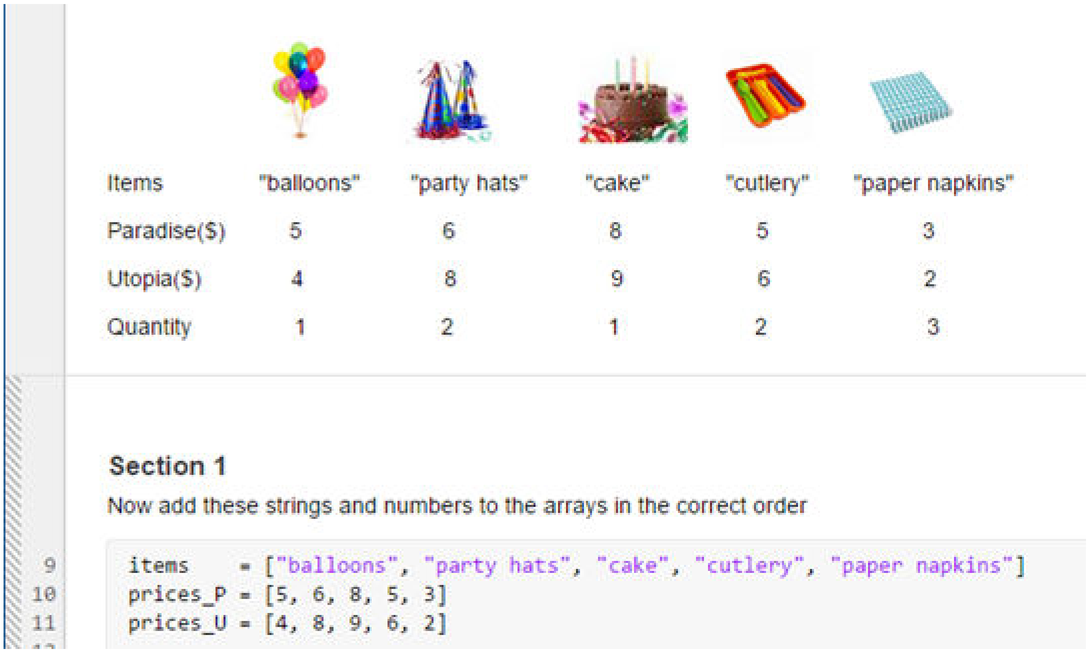
Name	Value	Size	Class
items	*1x2 string*	1x2	string
prices_P	[5,6]	1x2	double
prices_U	[4,8]	1x2	double

Now, tell the students to add three more items to the shopping list table. They can either

- Fill the table provided in the worksheet.

OR

- Click and Drag the pictures in the Live Script and add them to the table. They can look up the prices from the worksheet and insert them in the table.
- Now, they can modify the code in Section 1 of Task 2 by adding the three new names to *`items`* and their corresponding prices in *`prices_U`* and *`prices_P`.* The table they filled should help with this task.



Items	"balloons"	"party hats"	"cake"	"cutlery"	"paper napkins"
Paradise($)	5	6	8	5	3
Utopia($)	4	8	9	6	2
Quantity	1	2	1	2	3

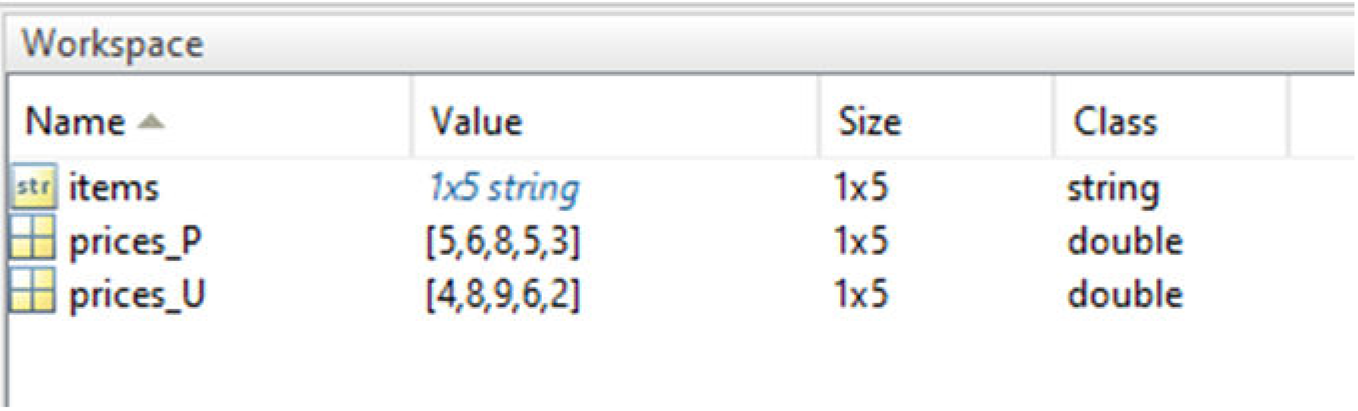
Section 1

Now add these strings and numbers to the arrays in the correct order

```
items    = ["balloons", "party hats", "cake", "cutlery", "paper napkins"]
prices_P = [5, 6, 8, 5, 3]
prices_U = [4, 8, 9, 6, 2]
```

- Execute this section and observe the change in Size of the variables in the **Workspace**.

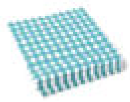
```
items    = ["balloons", "party hats", "cake", "cutlery", "paper napkins"]
prices_P = [5 6 8 5 3]
prices_U = [4 8 9 6 2]
buy_U = prices_U < prices_P
```



Workspace

Name ▲	Value	Size	Class
items	*1x5 string*	1x5	string
prices_P	[5,6,8,5,3]	1x5	double
prices_U	[4,8,9,6,2]	1x5	double

- Discuss with the students that using vectors allowed us to simply add items to the list without having to create more variables. In fact, we can keep adding more items and prices without making new variables.
- Now the question is, how can we compute with vectors and how can we get the list of items that are cheaper in Utopia?
- Tell them to check if the '<' operation works with the `prices_U` and `prices_P` vectors. They will type the following line to the section 1 to do this:
- buy_U = prices_U < prices_P
- Execute this section and observe the Value, Size and Class of the new variable created in the **Workspace**.
- `buy_U`is a vector of logical Class. We have now seen vectors of string, double and logical classes. We know they are vectors by looking at the Size which is 1 x 5. This means that there are 5 values stored in a single row.
- You can now discuss what the students interpret the logical values in `buy_U` to mean. We saw before when comparing a single value, that logical 1 means that the item is cheaper in Utopia. The five logical values in `buy_U` correspond to the 5 items in `items` variable. 1 indicates that the corresponding item is cheaper in Utopia.



Items	"balloons"	"party hats"	"cake"	"cutlery"	"paper napkins"
Paradise($)	5	6	8	5	3
Utopia($)	4 ✓	8 ✗	9 ✗	6 ✗	2 ✓
buy_U =	**[1**	**0**	**0**	**0**	**1]**

The next two steps are to be performed on the instructor's computer in the **Command Window.**

Now that we know which items to buy, we will learn to extract their names in a separate vector. The position of each value in a vector is called an 'index'.

Tell the students to turn their attention to your computer screen. Execute Task 2 -Section 1 of the sample solution and show the students your `items`, `prices_P` and `prices_U` variables.

```
items
prices_U
prices_P
```

Execute the following commands in your Command Window one by one while discussing what each does:

```
items(1)
```

This gets the first item name from the vector items

```
prices_U(1)
```

This gets the first value from vector prices_U which is the price in Utopia for first item

Ask the students how they would get the first items price in Paradise. The answer is:

```
prices_P(1)
```

Now, execute the following command and discuss

```
prices_U(2:4)
```

The ':' is used to indicate all numbers from 2 to 4. This will get values at indices 2, 3 and 4 from prices_U in a vector.

Next, tell them that we can also use another vector to index. For example, if we want to get the 1st, 3rdand 5th items and their prices in Utopia, we can use these commands:

```
ind = [1 3  5];
```

Create a vector with indices

```
items(ind)

prices_U(ind)
```

Spend some time here to ensure understanding of indexing.

NOTE - The above outputs are based on the sample solution.

Finally, test their understanding by typing this code on your screen and ask them how they would index to reorder the words in the following vector of strings to form a meaningful sentence:

```
words = ["We" "party" "musical"  "are" "5:00 PM"  "having"  "at" "a"];
```

The answer is

```
ind = [1 4 6 8 3 2 7 5];
words(ind)
```

1. **Optional**: You can make a compound statement instead of creating *ind,* like so*:*

```
words([1 4 6 8 3 2 7 5])
```

You can use the following command to join the words into a single string to make a sentence:

```
join(words)
```

NOTE – `join` is a *function* provided in MATLAB. We cover *functions* in detail in a later module.

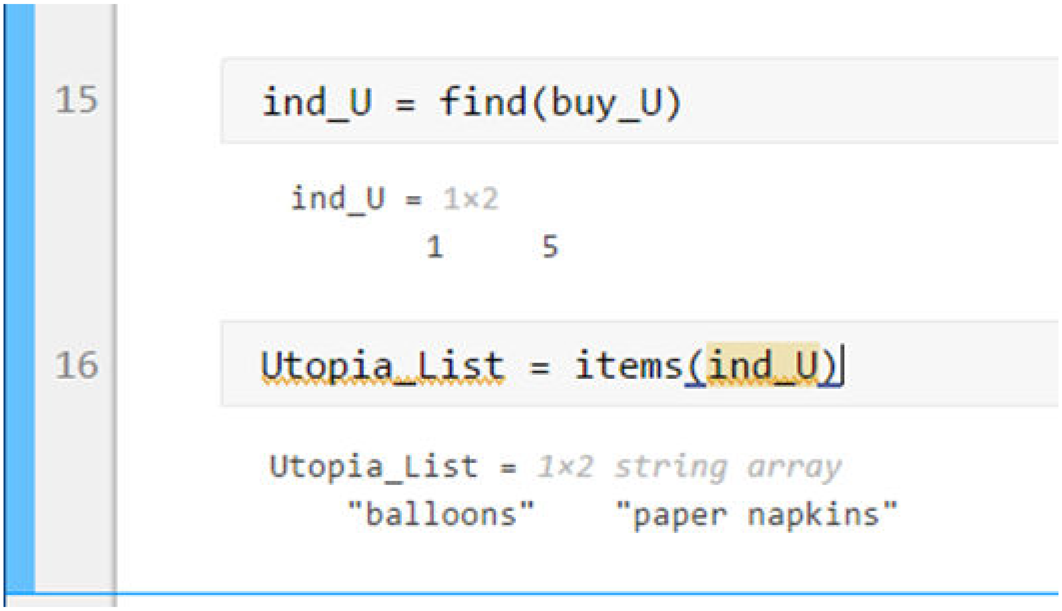
The students can now go back to the Live Script to pick up where we left off.

We created the variable buy_U in section 1 of this task. We have also learnt about indexing. If we can get the position numbers (indices) in buy_U that have the value 1, then we can get the names of items we want to buy from Utopia.

- The students can manually get the indices by observing buy_U and then perform indexing. See if they can do that.
- However, there is a command in MATLAB that will find the indices or positions of 1's in `buy_U`. Tell the students to add the following line of code in Section 2:

```
ind_U = find(buy_U)
```

- **NOTE –** `find` is a function provided in MATLAB. We cover functions in detail in a later module.
- Now execute the section again and look at the values in `ind_U`*.*
- Finally, ask the students to write a line of code to extract the names of items from `items` that are cheaper in Utopia by using `ind_U.` They would be able to do this based in their understanding of indexing.



```
ind_U = find(buy_U)

 ind_U = 1×2
         1     5

Utopia_List = items(ind_U)

Utopia_List = 1×2 string array
    "balloons"    "paper napkins"
```

We have finally written a program to get a list of items from our shopping list that are cheaper in Utopia!

Part C – Task 3

We have learned how to write code to compare prices and get a list of items that are cheaper in Utopia. The next two sections will introduce a little more complexity in the vector computations by using discounts and quantities.

- Tell the students that we can get $2 off in Utopia once per item. We will perform the same task of comparing prices but this time, after subtracting the discount.
- Before starting the Task, get their attention to your computer screen. Tell them that we can subtract a number from a vector in MATLAB. In your Command Window, show them the following command:

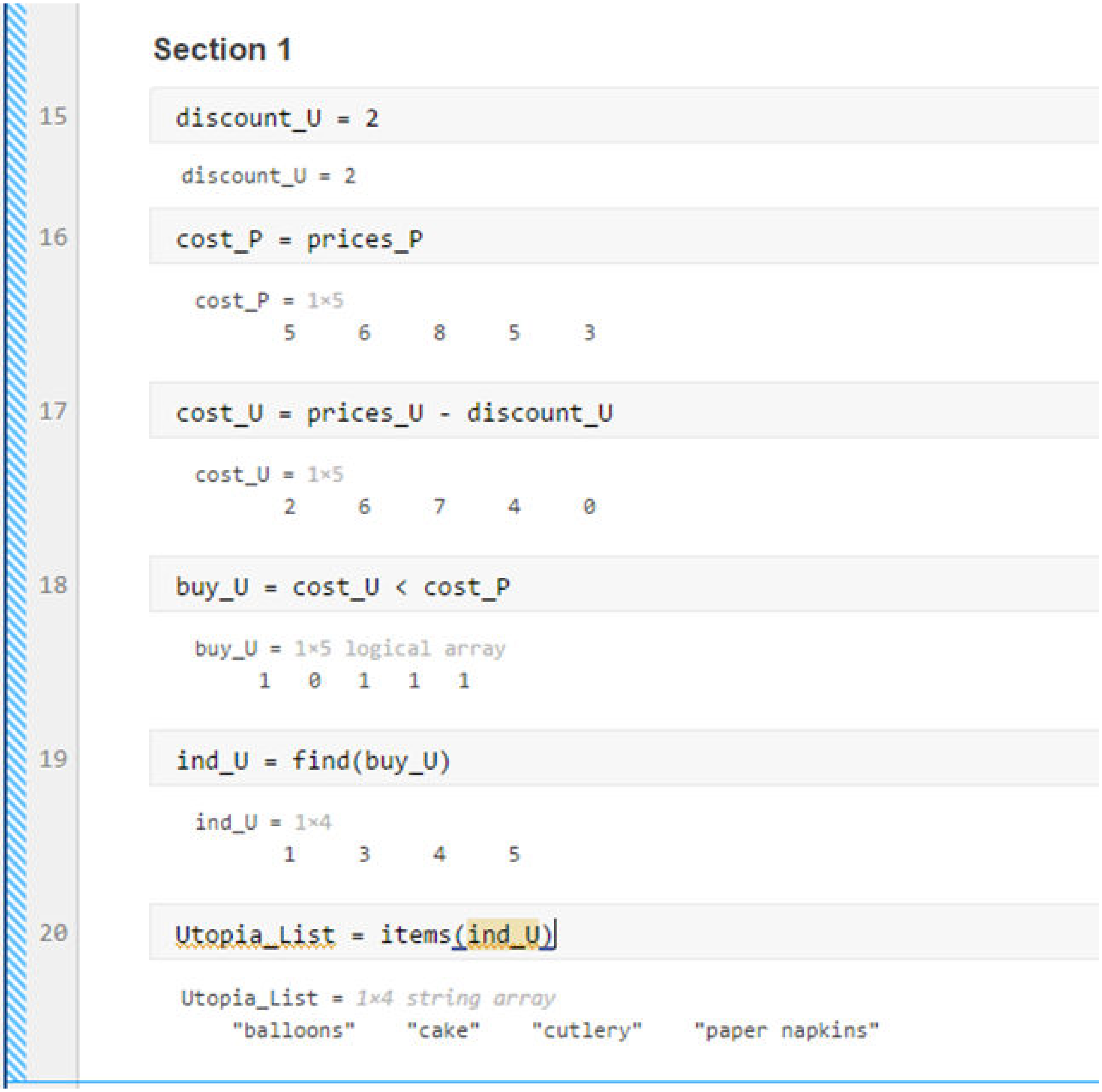
```
[2 3 4 5] - 2
```

- Look at the output and discuss. We are subtracting the number 2 from a vector and the output is another vector with 2 subtracted from each value.
- We also know that we can use a variable instead of a vector and we can save the output in a variable. Show them the following to reinforce this concept

```
my_vec = [2 3 4 5]
sub = my_vec-2
```

- Now, the students can complete Task 3- Section 1. We have created a variable to store discount to start. For this task, we will subtract the discount from prices in Utopia first to calculate `cost_U.`Since there is no discount in Paradise, the final `cost_P` is same as `prices_P.`
- Next, write a line of code to compare the final costs from the previous two statements and save result in `buy_U`.

- Finally, ask the students what the next two lines should be. They are same as the last lines from previous section.



Section 1

```
discount_U = 2

 discount_U = 2

cost_P = prices_P

  cost_P = 1×5
         5     6     8     5     3

cost_U = prices_U - discount_U

  cost_U = 1×5
         2     6     7     4     0

buy_U = cost_U < cost_P

  buy_U = 1×5 logical array
       1   0   1   1   1

ind_U = find(buy_U)

  ind_U = 1×4
         1     3     4     5

Utopia_List = items(ind_U)

 Utopia_List = 1×4 string array
     "balloons"    "cake"    "cutlery"    "paper napkins"
```

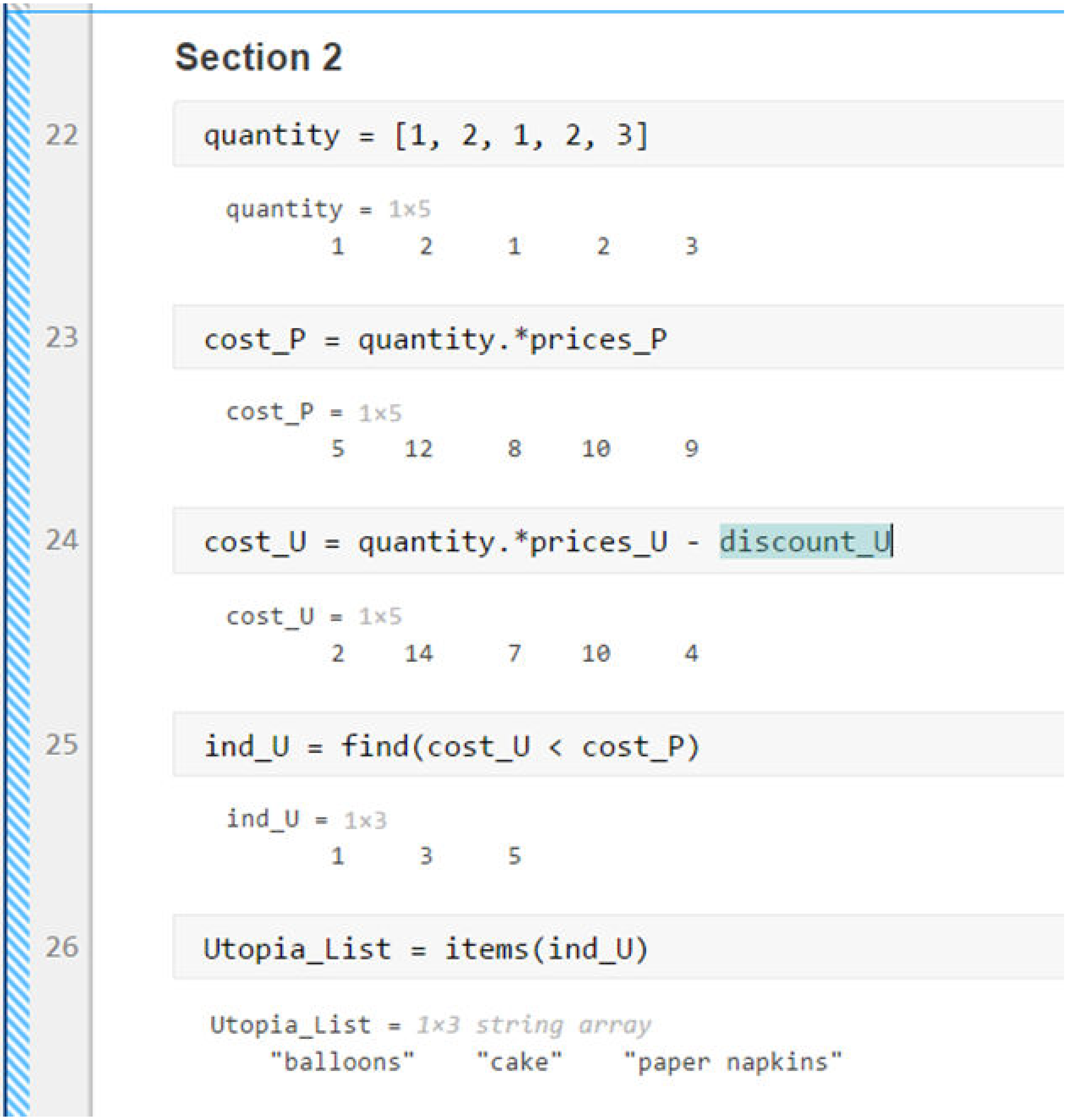
- **Optional**: What if Utopia had $2 discount only for the first 3 items and $1 for all the remaining items?
- We can make the discount variable a vector instead of a single number (scalar).
- Change the line `discount_U = 2` to discount_U = [2 2 2 1 1]

Execute the section. Notice that each number in `discount_U` is subtracted from the corresponding price without changing anything else! We just learnt how to subtract a vector from a vector. Keep in mind that to do so, the sizes of the vectors must be same. Ask the students why. (MATLAB will not know what to do with extra numbers. It will give an error)

- **Optional**: You may choose to just demonstrate the next section and allow the students to follow. Here, we will consider buying different quantities of each item instead of just 1 pack. Ask the students for ideas on how they might do that. The hint is in the provided code. (There are empty square brackets next to quantity).
- Create a vector by filling the empty square brackets with quantities for each item. They can first fill them in the table provided in the worksheet or at start of Task 2.
- The prices in both stores should now be multiplied by the corresponding quantities before discount is subtracted. Tell them about the '`.*`' operator. The dot before the multiplication operation tells MATLAB that we want to multiply two vectors element by element. You can add this line in the section to demonstrate:

```
cost_P = quantity.*prices_P
```

- From here on, the students should complete the section by adding a line to compute `cost_U`. Provide hints as necessary. Here is the solution with sample output:



Section 2

```
22  quantity = [1, 2, 1, 2, 3]

    quantity = 1×5
         1     2     1     2     3

23  cost_P = quantity.*prices_P

    cost_P = 1×5
         5    12     8    10     9

24  cost_U = quantity.*prices_U - discount_U

    cost_U = 1×5
         2    14     7    10     4

25  ind_U = find(cost_U < cost_P)

    ind_U = 1×3
         1     3     5

26  Utopia_List = items(ind_U)

    Utopia_List = 1×3 string array
        "balloons"    "cake"    "paper napkins"
```

- To calculate `cost_U`, `discount_U` must be subtracted after `quantity` is multiplied with `prices_U`
- Notice that we have used a compound statement at line 25. Instead of assigning the output of comparison to `buy_U,` we have directly used the comparison expression in the parenthesis. Compound statements can be tricky, and students may learn to use them only if they truly understand variables and assignment.
- Students may now clear the **Workspace** and **Command Window**.