MISSION 16: Breakout

Time: 60-120 minutes

Overview:

Now that you've conquered *Handball* you are all set to code one of the all-time arcade classics! Imagine that you've been tasked by Atari's CEO to create the next hit game for the company. *Ready to break some bricks* on? This is the second of a two-part mission sequence. It starts where Mission 15 finished.

Cross Curricular:

- **MATH:** The mission introduces and uses the concept of a matrix. If your students are able, have a lesson or practice creating and using a matrix.
- **SCIENCE:** The mission goes into more detail about collisions, but the result of the collision is simple. Discuss what happens in a real collision, like with cars or atoms, etc.
- Supports **language arts** through reflection writing.

Materials Included in the learning portal Teacher Resources:

Mission 16 Assignment

The assignment is the worksheet for students to complete as they work through the mission. It should be given digitally to each student before the mission starts. They answer questions in the document during the mission and turn it in at the completion of the mission.

Mission 16 Lesson Plan

The lesson plan comes from the original CodeX Teacher Manual and is included here for easy reference.

Mission 16 Solutions **(Breakout)**

Formative Assessment Ideas:

- Exit ticket(s)
- Quizzes in CodeSpace
- Assignment document completion
- Completed **Breakout** program
- Daily student reflections
- No specific reviews are prepared for this mission. New concepts are not introduced but are applied. You can use the previous review Kahoots as needed.

Vocabulary:

- **Prototype:** A model of something, or an early sample created to test a concept
- **Matrix**: a structure with rows and columns – a 2D array

Preparing for the lesson:

This mission will create a one-player “breakout” game, building on the code from Mission 15. The code will use a matrix, for the bricks and their collision status. If the math is too advanced for your students, they can still complete the mission using the code that is given and reading through the hints. They are not required to understand all the math equations and concepts. However, a lot of detail is given to help understand how a matrix works in the code.

Also, students will create a lot of functions and include many global variables and constants. It is a lot to keep track of, so students will need to pay attention to details and type carefully. They should scroll through the code in CodeTrek to make sure they are placing new code in the correct functions and with the correct indenting. Make sure each objective is correct and code works properly before going to the next objective. Otherwise it will be very difficult to debug.

Students will use the Codex throughout the lesson. Decide if they will work in pairs or individually.

- There is no slide deck or workbook for this mission. All instructions are included in CodeSpace and CodeTrek.
- Be familiar with the Assignment document and the questions they will answer.
- Prepare the Assignment document for each student digitally using your LMS.
- The mission program does not need to be portable. If you want students to use the CodeX without a cable, then have batteries available.

Lesson Tips and Tricks:

🧑‍🤝‍🧑 Pre-Mission Discussion:

A pre-mission prompt is not given for this lesson. Students will continue their code from Mission 15. After reviewing the lesson, you may come up with your own pre-mission discussion prompts, or you may want to just jump in and start the mission. You can also use the quizzes from Mission 15 as a review before starting this Mission.

💻 Mission Activities:

Most of this lesson is on the computer, writing code to create a single player game.

- Each student will complete an Assignment Document.
- Students could work in pairs through the lesson, or can work individually.
- Students will need the CodeX and USB cable.

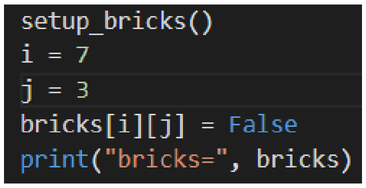
💡 **Teaching tip: Mission Introduction**
This objective introduces the mission by giving the background on the game "Breakout." The completed program will be a one-player version of "Breakout". It starts with the program from Mission 15 and adds code to it.

💡 **Teaching tip: Objective #1**
Students start with a "File-Save As" so they have their original code from Mission 15 and then also start with it for Mission 16. Students add two functions to create the rows of bricks across the top.

Students will answer a question on the assignment document.

🗝️ **NOTE:** This objective uses a for loop with a step, and also variables in drawing the rectangles. Review as needed.

💡 **Teaching tip: Objective #2**
Students will add a function to create a list of lists for the Boolean values of hit or not hit. In the goals, they need to fill in the value for i and j shown →



```
setup_bricks()
i = 7
j = 3
bricks[i][j] = False
print("bricks=", bricks)
```

Students will answer a question on the assignment document.

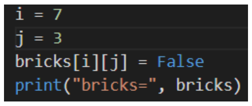
🗝️ **NOTE:** This objective introduces a matrix. This can be a really new concept for students. You may want to do a lot of practice with it before moving on.

💡 **Teaching tip: Quiz**

Students take a ❓ short quiz. The 4 Quiz questions are below. Remember: rows and columns start at 0, not 1.

💡 **Teaching tip: Objective #3**
Students will delete the "draw_bricks()" and "brick_row()"functions and replace ithem with the "brick_place()" function to use the i and j. Then modify the "setup_bricks()" function to also create a brick at the same time as creating the list of Bools.

🗝️ **NOTE:** Also remember to delete the test code →



```
i = 7
j = 3
bricks[i][j] = False
print("bricks=", bricks)
```

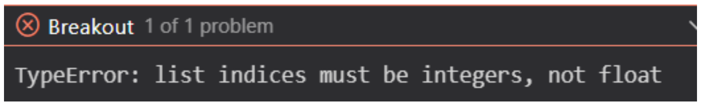
🗝️ **NOTE:** Two hints in this objective, and they are very helpful. The first hint is a strategy for working on a model, and also for debugging. Good to review!

🗝️ **NOTE:** The second hint helps clear up the difference between x and y, and i and j. It gives a really good example. You may want to make a visual of this to put up in the room.

💡 **Teaching tip: Objective #4**
Students create a function to check if a brick has been collided into. In CodeTrek, the last bubble is in the main program, not a function.

Students will answer a question on the assignment document.

🗝️ **NOTE:** Without the int() conversion, this error will occur. You have seen it before! The position x, y must be integers! The auto-grader will accept the code without int(), but the code will through a runtime error.



```
Breakout 1 of 1 problem
TypeError: list indices must be integers, not float
```

Adding the int() conversion is a little tricky because you have to add another set of () , not just int

💡 **Teaching tip: Quiz**

Students take a ❓ short quiz. The 2 Quiz questions are below.

💡 **Teaching tip: Objective #5**
Students will add a way to check the previous ball position with the current ball position to know where the ball is hitting the brick. Code will be added to two different functions – be careful!

Students will answer a question on the assignment document.

🗝️ **NOTE:** Be careful with the indenting of the last bubble – the if statements. They are indented inside the if statement!

💡 **Teaching tip: Objective #6**
The objectives look like a lot of work, but this is a fairly straightforward objective. The last thing to do is add comments to the code. One comment must include the secret word, which is embedded in a comment you see in CodeTrek. This is also the answer to the question on the answer document.

Students will answer a question on the assignment document.

💡 **Teaching tip: Objective #7**
At this point, the game is complete! The next two objectives give a walk-through of adding a feature to a game (iterative design process).

Students will answer a question on the assignment document.

🗝️ **NOTE:** Remember to make “mute” global in the “check_buttons()” function.

💡 **Teaching tip: Objective #8**
Another function is added to “level-up” once all bricks are cleared, to make it an authentic arcade game.

💻 Mission Complete:

This mission ends with a completed, working program that will play an authentic game of “Breakout”. You need to decide how you will use the program for assessment. You could:

- Go to each student and check-off their code
- Have the students download their code to a text file and turn it in using your LMS or as an attachment in email
- Have students print their code (either download and then print the text file, or print a screenshot)
- Have students switch computers and run each other’s code. Fill out a simple rubric and turn in to teacher
- Any other way that works for you

🧑‍🤝‍🧑 Post-Mission Reflection:

The assignment document does not include a post-mission reflection. You can add one if you want to. End by collecting the Assignment document and any formative assessment you want to include.

💻 IMPORTANT Clearing the CodeX:

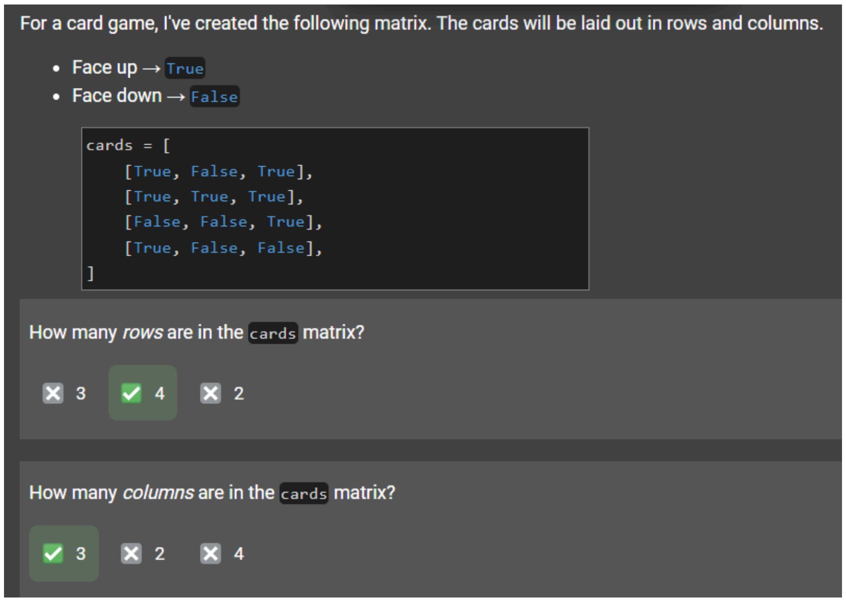
The students have already created a “Clear” program. Students should open and run “Clear” at the end of each class period.

SUCCESS CRITERIA:

- ❏ Add 8 rows of bricks to the handball program using a matrix.
- ❏ Detect if the ball collides with a brick.
- ❏ Delete a brick when the ball collides with it.
- ❏ Bounce the ball when it collides with a brick.
- ❏ Add a score and lives to the game.
- ❏ Add a mute button.
- ❏ Add a “level-up” feature
- ❏ Game works correctly and without errors

❓ Quiz Questions

Quiz 1



For a card game, I've created the following matrix. The cards will be laid out in rows and columns.

- Face up → `True`
- Face down → `False`

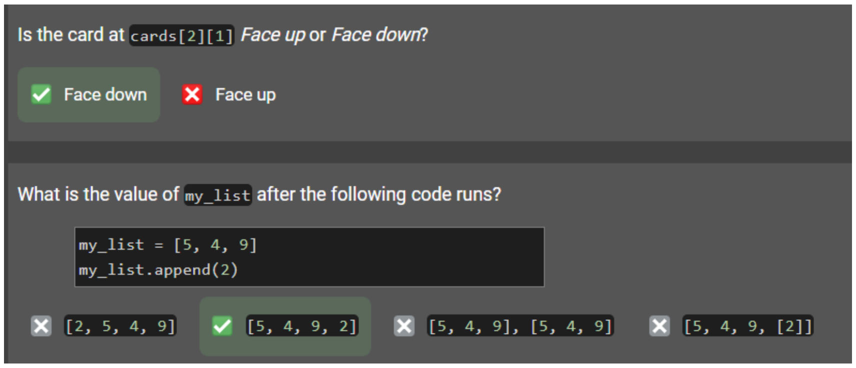
```
cards = [
    [True, False, True],
    [True, True, True],
    [False, False, True],
    [True, False, False],
]
```

How many *rows* are in the `cards` matrix?

✗ 3 ✓ 4 ✗ 2

How many *columns* are in the `cards` matrix?

✓ 3 ✗ 2 ✗ 4



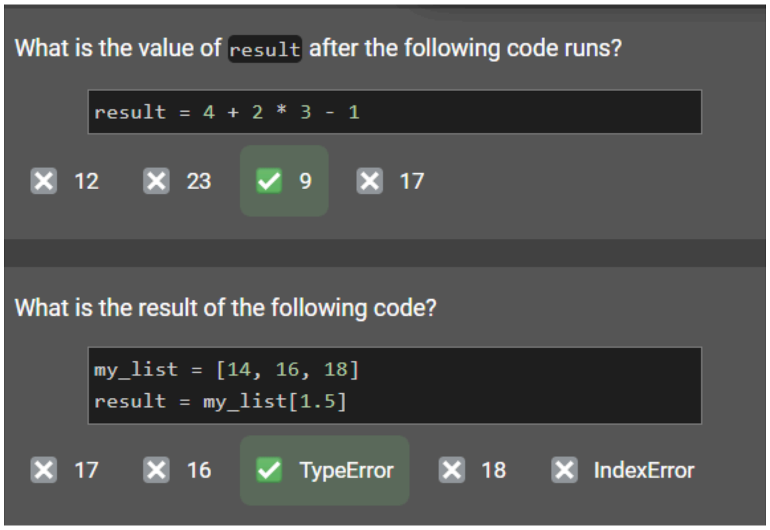
Is the card at `cards[2][1]` *Face up* or *Face down*?

✓ Face down ✗ Face up

What is the value of `my_list` after the following code runs?

```
my_list = [5, 4, 9]
my_list.append(2)
```

✗ `[2, 5, 4, 9]` ✓ `[5, 4, 9, 2]` ✗ `[5, 4, 9], [5, 4, 9]` ✗ `[5, 4, 9, [2]]` Quiz 2


What is the value of `result` after the following code runs?

```
result = 4 + 2 * 3 - 1
```

✗ 12 ✗ 23 [✓ 9] ✗ 17

What is the result of the following code?

```
my_list = [14, 16, 18]
result = my_list[1.5]
```

✗ 17 ✗ 16 [✓ TypeError] ✗ 18 ✗ IndexError