Unit 4 Arrays

# Learning goals

1. Perform basic operations with arrays, such as initializing arrays, accessing elements by index, determining the size of an array, updating elements, adding or deleting elements, and reverse look-up of indices.
2. Develop basic algorithms for processing arrays, such as summing the elements, finding the maximum value, searching for a value, and systematically changing each value.
3. Write programs that use arrays of strings, numbers, ordered pairs, colour values, and graphical objects.
4. Write programs that use modular arithmetic on the index of an array to cycle through the array repeatedly.
5. Use arrays to animate a cluster of objects of the same type, e.g. falling snow flakes.
6. Solve applied problems using arrays. (Example: Given an array of player names and a phrase, program the game eenie meenie miney mo.)

|  |  |
| --- | --- |
| **Lesson** | **Content** |
| **Lesson 1** Array operations | - Creating an array at compilation time by listing its elements - Accessing individual elements  - The range of the indices 0..n-1 - Printing the elements of an array  - Changing the elements of an array - Practice: Array Challenges.doc |
| **Lesson 2** Array algorithms | - Investigation: Finding the maximum value in an array - An algorithm for search and replace - Splitting a sentence into an array of words - How to initialize and fill an array at run-time (using append) - Practice: Array Challenges Part 2.doc |

|  |  |
| --- | --- |
| **Lesson 3**  Cycling through an array | - Take up Problem 2 (quiz generator) in Array Challenges Part 2 - Cycling through an array using modular arithmetic on the indices  \* Plain example of printing the elements repeatedly  \* Cycling through the hours of the day: *Hours of the day.py*  \* Cycling through a color array: *Simple color cycling.py*  \* Students practice: Array Challenges Part 3 |
| **Lesson 4**  Application: Timing animations with modular arithmetic | Take up Array Challenges Part 3 using *Blinking light.py* and *Blinking traffic light 1.py* - Adding randomized graphics: *Blinking traffic light 2.py* - Combining independently-timed animations: *Blinking traffic light 3.py* - Students experiment with these programs. - Introduce the Eenie Meenie Miney Moe game |
| **Lesson 5**  Application: Eenie Meenie Miney Moe | - Playing a sample EMMM game using students - Analysis of the sample game using a table of values - The nested-loop structure of the game - Students work in teams to derive the details of the EMMM algorithm - Students begin programming the algorithm |
| **Lesson 6** Application: Eenie Meenie Miney Moe, day 2 | - Quiz on Arrays - Deriving the EMMM algorithm - Implement EMMM as a Python program - Finding and fixing bugs in the EMMM program |
| **Lesson 7** Application: Circular string art | - Drawing the pattern by hand - Identifying the input and output of the algorithm - Deriving the formulas for the nails’ (x, y) coordinates using trig - Challenge 1: Write the for-loop for drawing the nails - Take up solution to Challenge 1 in Python broadcast - Storing the (x, y) values in an array for the loop that draws the lines - Challenge 2: Write the for-loop for drawing the lines |
| **Lesson 8** Animating a cluster of objects | - Finish and take up Challenge 2 - Review of how to create a simple, one-object animation - Generalizing to multiple objects using arrays of parameters & shapes - Begin Assignment 4 |
| **Lesson 9** Generating random arrays | - Replacing constant arrays in the cluster animation with random ones - Work on Assignment 4 |
| **Lesson 10** Documenting programs | - Making programs easier to read & debug w/ comments & white space - Work on Assignment 4 |