**Lesson 10 Arrays**

**Getting Started**

We looked at arrays briefly in Lesson 8 when we reading in external text and putting it into a Python list (arrays are called “lists” in Python). Now it’s time to really understand them.

Before you can use an array, you have to *initialise* it. Initialising an array variable:

* tells Python that the variable is an array;
* sets the contents at the start.

Depending on whether you know how many elements will be in the array you can initialise the array in two different ways:

* If you don’t know how many elements will be used, make an empty array, like this:

namelist = []

This sets up an array called namelist which has nothing in it yet. You will need to append values later as you build it up.

* If you do know how many values will be stored you can initialise the array with values already in it like this:

namelist = [“Albert”,”Beatrice”,”Caroline”,”Dennis”]

This makes a 4 element array with names at each position. Alternatively, you might do this:

scores = [0,0,0,0,0,0,0,0,0,0]

In this case, every value is set to 0 in a 10 element array. The program will change the values as it progresses.

Notice that all the values in an array must be of the same data type so namelist is a string array while scores is an integer array.

**Reading Values**

We use index numbers to refer to the specific values in an array. These are put into square brackets and start from zero. So in the namelist array above, namelist[0] is “Albert” and namelist[3] is “Dennis”. If you try to refer to namelist[4] you will get a run-time error because the array only has 4 elements meaning the highest index number possible is 3.

You can assign values as long as there is an element there so this line would be OK:

 scores[8] = 27

But this one wouldn’t be OK as we only created 10 elements at the beginning and so indices can only go from 0 to 9:

 scores[10] = 53 #Run-time error!

Try this program to see the way this works. Notice the use of a for loop to show the values in an array, Because you can work out how many values there are in an array using the len() function a counter-controlled loop works really well.

names = [“Phil”,”Quentin”,”Rachel”,”Simone”]

scores = [45,27,83,63]

length = len(names) #How many names are there?

for count in range(0,length): #Loop that number of times

 line = names[count]+“ scored ”+str(scores[count])

 #Need to use str to change number into a string

 print(line)

This next program does much the same but asks you to type in the scores and uses append() to add them to the array.

names = []

scores = []

nextname = input(“Enter a name: ”)

while nextname != “”: #Loop until no more names

 nextscore = int(input(“What did ”+nextname+“ score? ”))

 names.append(nextname)

 scores.append(nextscore)

 nextname = input(“Enter a name: ”) #Next name then loop

#Now display them like before

length = len(names)

for count in range(0,length):

 line = names[count]+“ scored ”+str(scores[count])

 print(line)

** Over to You**

Make a new program that starts off an empty array. It should ask the user to enter an integer and then fill the array with the multiples of that integer from multiplied by 0 to multiplied by 12. At the end get the program to print the array.

The output on the screen will look something like this:

Please enter the factor: 4

The array is [0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48]

**Using Arrays with External Files**

If you want to store the results in an array into an external file you will need to use a for loop again. External files can only take string values so you may need to change the data type of the variables as you save them using the str() function.

Try this program:

scores = []

nextscore = input(“Enter a score (0 to finish): ”)

while nextscore != 0: #Loop until no more scores

 scores.append(nextscore)

 nextscore = input(“Enter a score (0 to finish): ”)

#Store in a text file

try:

 File = open(“data.csv”,”w”) #csv to open in Excel

 length = len(scores)

 for count in range(0,length):

 nextvalue = str(scores[count]) #convert to string

 File.write(nextvalue+”\n”) # \n moves to next line

 File.close()

except:

 print(“Problem saving file”)

**Two Dimensional Arrays**

So far we’ve only dealt with arrays that are one dimensional, a list of names or numbers but sometimes we want to store a table of values e.g. a league table. Imagine this situation where we want to store the details of a group table in a competition:

 **Pd W D L Pts**

Amersham 4 3 1 0 10

Bedminster 4 1 1 2 4

Charmouth 4 0 2 2 2

I can store the names in an array called names which will look like this [“Amersham”, “Bedminster”, “Charmouth”] and I could store the details in separate arrays: played, won, drawn, etc… but that would be rather clunky.

Much better to have the details in a table with 3 row and 5 columns. The way to do this is to have one array with 3 rows where each row is *another* array, like this:

 [4, 3, 1, 0, 10]

leaguetable [4, 1, 1, 2, 4]

 [4, 0, 2, 2, 2]

You obviously can’t write it like that in a program so you have to make an array called leaguetable and then append separate arrays to it.

leaguetable = [] #Initialise the array

leaguetable.append([4, 3, 1, 0, 10]) #Append the 1st row

leaguetable.append([4, 1, 1, 2, 4]) #And so on …

leaguetable.append([4, 0, 2, 2, 2])

Just like co-ordinates we refer to the items in a two dimensional array using two indices, the first one says which row in the main array you are in and the second one says which item in that row you are reading.

This is leaguetable[0][2]

 [4, 3, 1, 0, 10]

leaguetable [4, 1, 1, 2, 4]

And this is leaguetable[2][4]

 [4, 0, 2, 2, 2]

Type in and run this example to see how a two dimensional array can be built up.

#Times Tables

factor = int(input(“Which table would you like? ”))

timestable = [[0,0]] #Start with one row; it is always zero

for count in range(1,13): #Note last number must one higher

 timestable.append([0,0]) #Append a blank row

 result = factor \* count

 timestable[count][0] = count #store the number

 timestable[count][1] = result #store the result

#Now print out the table

for row in timestable: #A new way to use for – much easier!

 print(row)

Did you see how the second for loop was written to pick out each row in the timestable array? Python makes it very easy to work through arrays. Each time the loop repeats it assigns the next element in the array into the variable row. Because we are using a two dimensional array each row will be the next one dimensional array that is picked out. If we wanted to make the output slightly more helpful, we might write the last two lines like this:

for row in timestable: #A new way to use for – much easier!

 print(row[0],”times”,factor,”=”,row[1])

row[0] is the first element in the row and row[1] is the second element.

** Over to You**

Make a program that make an array to store all square and triangle numbers up to a given value. It should ask the user for that value. The formulas to calculate triangle numbers is:

nth triangle number = ½ × n × (n + 1)

Typical output would look like this:

Please enter limit: 3

0 0 0

1 1 1

2 4 3

3 9 6