16 Arrays

Algorithms & Programming

Python

A data structure is a format for organising and storing data. You will probably have come across **files** plenty of times before. We've used **variables** and **lists** in previous tasks. Other data structures include **arrays**, **records, hash tables, queues** and **trees**.

Aim: *To learn about data structures such as arrays.*

Task 1 – Data Structure Match

Use your experience, common sense or information on the internet to match the data types below to their common use.

	Data Type				Use
1	**Variable**	•	•	**a**	Data organised into nodes; a root and then branching structures.
2	**List**	•	•	**b**	A structure with keys and values to look up, a bit like a dictionary.
3	**Record**	•	•	**c**	A number of items which can easily be changed in length or value.
4	**Hash Table**	•	•	**d**	A collection of fields of different type e.g. a 'row' in a database.
5	**Tree**	•	•	**e**	An item holding a single value.
6	**Queue**	•	•	**f**	A structure often containing a large number of lines.
7	**Array**	•	•	**g**	Data kept in order, inserted at one end and removed at the other.
8	**File**	•	•	**h**	A fixed number of values in one or more dimensions.

Task 2 – Two-Dimensional Arrays

An array is a simple data structure. Whereas a variable is a single box containing an item of data, an array is like a series of boxes all tied together. A one-dimensional array is similar to the lists we used previously. There are a few differences:

- Arrays must contain only a single data type (strings, integers etc.). Lists can contain a mixture.
- Arrays tend to be more static. You can't insert data or sort an array easily.

An array called *three_letter_words_1D* might be assigned the following values:

```
three_letter_words_1D(0) = "And" #As with a list, the 1st index is zero.
three_letter_words_1D(3) = "Dab" #The 4th word is ‘Dab’
```

A two-dimensional array can be visualised as a grid. Two indexes are used; the first can be thought of as horizontal and the second, vertical. Use the array on the right to answer the questions.

a. What value is stored at the location *three_letter_words_2D(0 , 0)*? ___________

b. What value is stored at the location *three_letter_words_2D(3 , 1)*? ___________

c. What is the location of the value "Fib"? ______________________________________

d. What is the location of the value "Car"? ______________________________________

Array named *three_letter_words_2D*

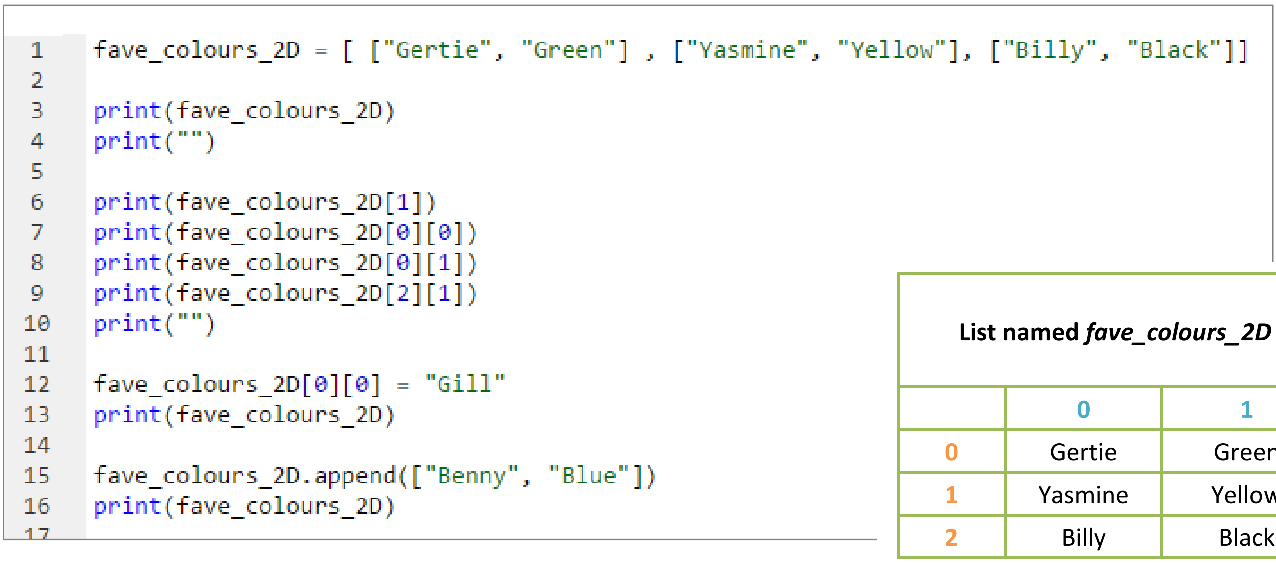
	0	1
0	And	Ear
1	Bat	Fib
2	Car	Gas
3	Dab	Hen

Arrays (page 2)

Task 3 – Working with 2D Arrays (or Lists)

It is great to think about arrays; they are common to all general programming languages. Having said this, they are not actually standard in Python. You can use arrays if you import the *array* module, but for the programs we will create it's easier to use lists.

a. Type the program below in *repl.it* and name it '**16.3 2D Lists**'. It creates the 2D array shown in the table. Add some comments to explain what is happening on each line.

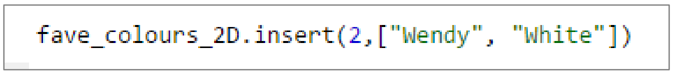


```
fave_colours_2D = [ ["Gertie", "Green"] , ["Yasmine", "Yellow"], ["Billy", "Black"]]

print(fave_colours_2D)
print("")

print(fave_colours_2D[1])
print(fave_colours_2D[0][0])
print(fave_colours_2D[0][1])
print(fave_colours_2D[2][1])
print("")

fave_colours_2D[0][0] = "Gill"
print(fave_colours_2D)

fave_colours_2D.append(["Benny", "Blue"])
print(fave_colours_2D)
```

List named *fave_colours_2D*

	0	1
0	Gertie	Green
1	Yasmine	Yellow
2	Billy	Black

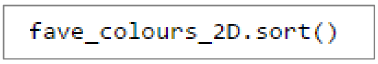
b. Try inserting an extra row of data into your list using the code below.



```
fave_colours_2D.insert(2,["Wendy", "White"])
```

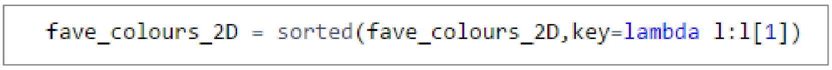
Note: *Inserting data into an array can often be difficult in programming. Python's lists are easy to use!*

c. Test the code below and find out how the list is sorted.



```
fave_colours_2D.sort()
```

d. Investigate the effect of the code below on your list. Look carefully at the way the list is now sorted.

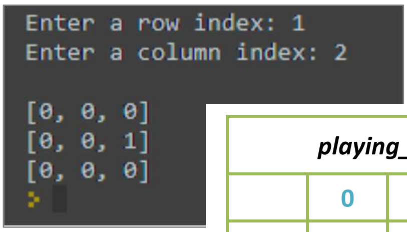


```
fave_colours_2D = sorted(fave_colours_2D,key=lambda l:l[1])
```

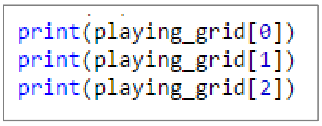
Task 4 – The Playing Grid

Create a 2D list to act as a playing grid like the one on the right. Name the program '**16.4 The Playing Grid**'. Remember that numbers don't need to be placed in quotes.

Add a mechanism so that the user can enter a pair of coordinates into the console and that location is changed to a 1. So, for example, if the user enters a 1 then a 2, the location (1,2) is changed to a 1. Print the array one row at a time after the change has been made (use code such as that shown).



```
Enter a row index: 1
Enter a column index: 2

[0, 0, 0]
[0, 0, 1]
[0, 0, 0]
```



```
print(playing_grid[0])
print(playing_grid[1])
print(playing_grid[2])
```

playing_grid

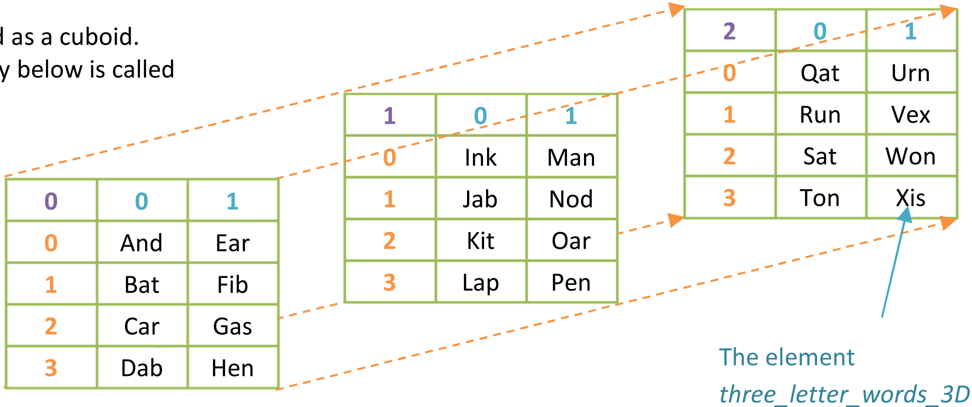
	0	1	2
0	0	0	0
1	0	0	0
2	0	0	0

Add any other functionality that you want. We will pick up on this idea in a later task.

Arrays (page 3)

Extension – Three-Dimensional Arrays

A 3D array can be visualised as a cuboid.
The three-dimensional array below is called *three_letter_words_3D.*



0	0	1
0	And	Ear
1	Bat	Fib
2	Car	Gas
3	Dab	Hen

1	0	1
0	Ink	Man
1	Jab	Nod
2	Kit	Oar
3	Lap	Pen

2	0	1
0	Qat	Urn
1	Run	Vex
2	Sat	Won
3	Ton	Xis

The element *three_letter_words_3D(3,1,2)* holds the value 'Xis'.

a. What value is stored at the location *three_letter_words_3D(0 , 0 , 0)*? ____________

b. What value is stored at the location *three_letter_words_3D(3 , 1 , 2)*?____________

c. What value is stored at the location *three_letter_words_3D(1 , 0 , 1)*? ____________

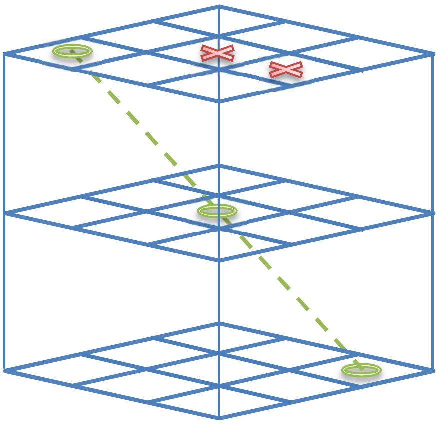
d. What is the location of the value "Nod"? ____________________________________

e. What is the location of the value "Run"? ____________________________________

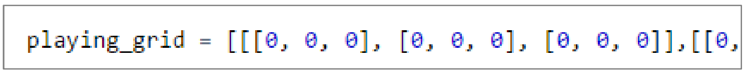
f. What is the location of the value "Fib"? ____________________________________

Task

Your task is to create a playing grid for a 3D Noughts and Crosses game. Name the program '**16E 3D Game**'. The game continues until all the spaces have been filled, then you count up the lines of three that each player has made.

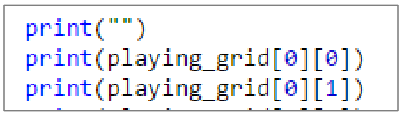


- Use a 3D list with dimensions 3 x 3 x 3. Each element should start off as a zero. (***Note:*** *You could use empty elements rather than zeros, but these are more difficult to visualise in the console.)*



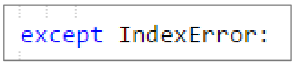
```
playing_grid = [[[0, 0, 0], [0, 0, 0], [0, 0, 0]],[[0,
```

- Find a way of displaying the lists in the console so that they look a little like the game.



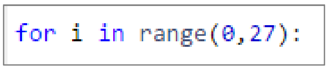
```
print("")
print(playing_grid[0][0])
print(playing_grid[0][1])
```

- Allow Player 1 to input 3 coordinates. These should change one location to a 1 (a naught). Test and adapt your inputs until they make sense.
- Allow this input to repeat, so that more locations are chosen.
- Every other turn should be Player 2. Their 3 coordinates should change one location to a 2 (a cross).
- Try and disallow a turn if the location has already been taken.
- Catch errors when the location isn't possible.

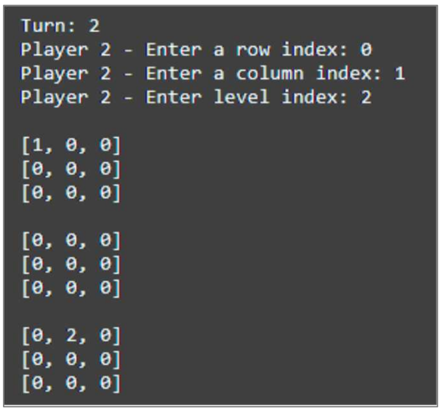


```
except IndexError:
```

- Only allow 27 turns, so that the game stops when all the locations have been used.



```
for i in range(0,27):
```



```
Turn: 2
Player 2 - Enter a row index: 0
Player 2 - Enter a column index: 1
Player 2 - Enter level index: 2

[1, 0, 0]
[0, 0, 0]
[0, 0, 0]

[0, 0, 0]
[0, 0, 0]
[0, 0, 0]

[0, 2, 0]
[0, 0, 0]
[0, 0, 0]
```