2 Dimensional Collections in Python

A 2 Dimensional **list** is effectively a **list** made up of other **lists**.

```
2dlist = [list1, list2, list3]
```

These are useful if you need a grid or matrix of data, similar to an Excel spreadsheet.

```
fruits =     ['apple', 'banana', 'orange', 'coconut']
vegetables = ['celery', 'carrots', 'potatoes', 'beans']
meats =      ['chicken', 'turkey', 'fish']

groceries = [fruits, vegetables, meats]

print(groceries[2][1])
```

`print(groceries[0])` ← will print the list of 'fruits' because it is the first lit item

`print(groceries[0][2])` ← will print the 'orange' – it is in row 0, col 2

`print(groceries[2][2])` ← will print the 'fish' – it is in row 2, col 2

Exercise: what references are needed to print potatoes

`print(groceries[  ][  ])` ← will print the 'orange' – it is in row , col

To access an element from a 2D list you need 2 indices. Using a single index returns the entire row (or list).

`print(groceries[2])` ← will print the entire 'meats' list

Alternatively, we could create the same 2D list without using list names:

```
groceries = [['apple', 'banana', 'orange', 'coconut'],
             ['celery', 'carrots', 'potatoes', 'beans'],
             ['chicken', 'turkey', 'fish']]

print(groceries[2][1])
```

To iterate over the elements in a 2D list, we need to use "**nested 'for' loops**". Using a single **for** loop, we'll only iterate over the rows.

```
groceries = [['apple', 'banana', 'orange', 'coconut'],
             ['celery', 'carrots', 'potatoes', 'beans'],
             ['chicken', 'turkey', 'fish']]

for food_groups in groceries:
    print(food_groups)
```

To iterate through the elements found in each row as well, we use a nested loop:

```
groceries = [['apple', 'banana', 'orange', 'coconut'],
             ['celery', 'carrots', 'potatoes', 'beans'],
             ['chicken', 'turkey', 'fish']]

for food_groups in groceries:
    for food in food_groups:
        print(food)
```

To make this look more organised:

```
groceries = [['apple', 'banana', 'orange', 'coconut'],
             ['celery', 'carrots', 'potatoes', 'beans'],
             ['chicken', 'turkey', 'fish']]

for food_groups in groceries:
    for food in food_groups:
        print(food, end=" ")
    print()
```

Exercise:

Use a 2D tuple to create a phone number-pad that looks like the output below.

```
1 2 3
4 5 6
7 8 9
* 0 #
```

Solution: (using 2D **list** of **lists**):

```
num_pad = [[1, 2, 3],
           [4, 5, 6],
           [7, 8, 9],
           ["*", 0, "#"]]

for row in num_pad:
    for num in row:
        print(num, end=" ")
    print()
```

Solution: (using 2D **list** of **tuples**):

```
num_pad = [(1, 2, 3),
           (4, 5, 6),
           (7, 8, 9),
           ("*", 0, "#")]

for row in num_pad:
    for num in row:
        print(num, end=" ")
    print()
```

Solution: (using 2D **list** of **sets**):

```
num_pad = [{1, 2, 3},
           {4, 5, 6},
           {7, 8, 9},
           {"*", 0, "#"}]

for row in num_pad:
    for num in row:
        print(num, end=" ")
    print()
```

and so on…

```
# 2D tuple of lists
num_pad = ([1, 2, 3],
           [4, 5, 6],
           [7, 8, 9],
           ["*", 0, "#"])

# 2D tuple of tuples
num_pad = ((1, 2, 3),
           (4, 5, 6),
           (7, 8, 9),
           ("*", 0, "#"))

# 2D tuple of sets
num_pad = ({1, 2, 3},
           {4, 5, 6},
           {7, 8, 9},
           {"*", 0, "#"})

# 2D set of lists (NOT VALID)
num_pad = {[1, 2, 3],
           [4, 5, 6],
           [7, 8, 9],
           ["*", 0, "#"]}

# 2D set of tuples
num_pad = {(1, 2, 3),
           (4, 5, 6),
           (7, 8, 9),
           ("*", 0, "#")}

# 2D set of sets (NOT VALID)
num_pad = {{1, 2, 3},
           {4, 5, 6},
           {7, 8, 9},
           {"*", 0, "#"}}

for row in num_pad:
    for num in row:
        print(num, end=" ")
    print()
```