Tuple Basics

This set of notebooks covers the third iterable data structure we will discuss in this course - `Tuples`.

Tuples are very similar to lists except for one important difference: Tuples are *immutable*. That means that the contents of a tuple cannot be altered. In this respect they are similar to strings. Because of this property, tuples are more efficient in terms of memory use and performance than lists.

Tuples are ordered and hence we can use indexes to access individual elements. The values in a tuple are enclosed inside parenthese and separated by commas.

Tuples are written as a collection of values separated by commas and enclosed in parentheses.
Tuples can also be created using the `tuple()` function. The `tuple()` function accepts an (optional) iterable parameter.

In [7]:

```
t1 = tuple()
t2 = tuple([8, 9, 6, 12])
t3 = tuple(range(5, -26, -4))
print(type(t1))
print(t1)
print(t2)
print(t3)
```

```
<class 'tuple'>
()
(8, 9, 6, 12)
(5, 1, -3, -7, -11, -15, -19, -23)
```

The values in a tuple are enclosed inside parenthese and separated by commas. This cells shows a few examples of tuples

In [8]:

```
from random import randrange
print((5,4,3,2,1))
print((randrange(8),randrange(8),randrange(8),randrange(8)))
print(('a', 'e', 'i', 'o', 'u'))
```

```
(5, 4, 3, 2, 1)
(6, 2, 3, 5)
('a', 'e', 'i', 'o', 'u')
```

Tuple variables can be created by assigning a tuple constant to a variable name.
A tuple can contain elements of different types.

In [9]:

```
test_tuple = ('5', 4, 3, 2, '1', 5.8, [7,8], (5, 9))
print(test_tuple)
```

```
('5', 4, 3, 2, '1', 5.8, [7, 8], (5, 9))
```

In [10]:

```
'''
Note that if you want to create a tuple with a single element then there must be a comma after the
element.
'''
test_tuple = ('5')
print(type(test_tuple))
test_tuple = ('5',)
print(type(test_tuple))
```

```
<class 'str'>
```

```
<class 'str'>
<class 'tuple'>
```