Lecture 3.3 : Tuples

Introduction

- A *tuple* is essentially an *immutable* list. Because it is immutable it shares some characteristics with strings i.e. neither can be modified. However a tuple is different to a string in that while the latter consists solely of characters, a tuple can contain objects of any type (like a list).

Tuples

- To create a tuple we use the comma operator:

```
>>> t = 4, 5, 6
>>> t
(4, 5, 6)
```

- Parentheses can (and should) be used to make it obvious we are creating a tuple:

```
>>> t = (4, 5, 6)
>>> t
(4, 5, 6)
```

- Concatenation and repetition work as we would expect:

```
>>> t = (4, 5, 6)
>>> u = (7, 8, 9)
>>> t + u
(4, 5, 6, 7, 8, 9)
>>> u * 3
(7, 8, 9, 7, 8, 9, 7, 8, 9)
```

- Slicing also works as we would expect:

```
>>> u = (7, 8, 9)
>>> u[1]
8
>>> u[::-1]
(9, 8, 7)
```

- We can iterate over a tuple with `for` and check membership with `in`:

```
>>> t = ('cat', 'dog', 'mouse')
>>> for i in t:
...     print(i)
...
cat
dog
mouse
>>> print('dog' in t)
```

```
True
>>> print('hen' in t)
False
```

- Because a tuple is immutable we cannot alter its contents:

```
>>> t = ('cat', 'dog', 'mouse')
>>> t[0] = 'hen'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Named tuples

- Related data can be grouped together to form a single object called a *named tuple*. Suppose for example we wish to model a car. A car has numerous attributes including a make, model, age and number of miles. We can use a named tuple as follows to represent a single `Car` data type that has each of these attributes:

```
>>> from collections import namedtuple
>>> Car = namedtuple('Car', ['make', 'model', 'age', 'miles'])
>>> car1 = Car('Opel', 'Astra', 3, 30000)
>>> car2 = Car('VW', 'Golf', 5, 60000)
>>> car3 = Car(miles=12000, age=1, make='Mazda', model='MX5')
>>> print(car1.make)
Opel
>>> print(car1.model)
Astra
>>> print(car1.age)
3
>>> print(car1.miles)
30000
>>> car3.miles < car2.miles
True
>>> print(car3.miles)
12000
>>> print(car2.miles)
60000
```