Tuples

In Python tuples are very similar to lists, however, unlike lists they are *immutable* meaning they can not be changed. You would use tuples to present things that shouldn't be changed, such as days of the week, or dates on a calendar.

In this section, we will get a brief overview of the following:

```
1.) Constructing Tuples
2.) Basic Tuple Methods
3.) Repetition in Tuple
4.) Slicing in Tuples
5.) Immutability
6.) When to Use Tuples
```

You'll have an intuition of how to use tuples based on what you've learned about lists. We can treat them very similarly with the major distinction being that tuples are immutable.

1. Constructing Tuples

The construction of a tuples use () with elements separated by commas. For example:

In [1]:

```
# Create a tuple
t = (1,2,3)
```

In [2]:

```
# Check len just like a list
len(t)
```

Out[2]:

```
3
```

In [3]:

```
# Can also mix object types
t = ('one',2)

# Show
t
```

Out[3]:

```
('one', 2)
```

In [4]:

```python
# Use indexing just like we did in lists
t[0]
```

Out[4]:

```
'one'
```

In [5]:

```python
# Slicing just like a list
t[-1]
```

Out[5]:

```
2
```

2. Basic Tuple Methods

Tuples have built-in methods, but not as many as lists do. Let's look at two of them:

In [6]:

```python
# Use .index to enter a value and return the index
t.index('one')
```

Out[6]:

```
0
```

In [7]:

```python
# Use .count to count the number of times a value appears
t.count('one')
```

Out[7]:

```
1
```

3. Repetition in Tuples

In [2]:

```python
# Code to create a tuple with repetition

tuple3 = ('python',)*3
tuple3
```

Out[2]:

```
('python', 'python', 'python')
```

4. Slicing in Tuples

In [3]:

```python
# code to test slicing

tuple1 = (0 ,1, 2, 3)
print(tuple1[1:])
print(tuple1[::-1])
print(tuple1[2:4])
```

```
(1, 2, 3)
(3, 2, 1, 0)
(2, 3)
```

5. Immutability

It can't be stressed enough that tuples are immutable. To drive that point home:

In [8]:

```python
t[0]= 'change'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-8-1257c0aa9edd> in <module>()
----> 1 t[0]= 'change'

TypeError: 'tuple' object does not support item assignment
```

Because of this immutability, tuples can't grow. Once a tuple is made we can not add to it.

In [9]:

```python
t.append('nope')
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-9-b75f5b09ac19> in <module>()
----> 1 t.append('nope')

AttributeError: 'tuple' object has no attribute 'append'
```

6. When to use Tuples ¶

You may be wondering, "Why bother using tuples when they have fewer available methods?" To be honest, tuples are not used as often as lists in programming, but are used when immutability is necessary. If in your program you are passing around an object and need to make sure it does not get changed, then a tuple becomes your solution. It provides a convenient source of data integrity.

You should now be able to create and use tuples in your programming as well as have an understanding of their immutability.

Since tuples are immutable, iterating through tuple is faster than with list. So there is a slight performance boost.

Up next Sets and Booleans!!