# Accessing Lists and Comparing Lists

**In this notebook we look at some more examples using lists:**

1. Accessing elements of a list
2. Comparing lists with strings. specifically, we demonstrate that lists are mutable but strings are not.

**Lists can be accessed using idexes, just like we did for strings. As in the case of strings, the index begins at 0 and ends at a value one less than the number of items in the list. Negative indexes can also be used to start the list from the end working backwards.**

In [1]:

```
list1 = ['physics', 'chemistry', 1997, 2000]
list2 = [1, 2, 3, 4, 5, 6, 7 ]
print(list1)
print(list2)
print(list1[3])
print(list2[-7])
```

```
['physics', 'chemistry', 1997, 2000]
[1, 2, 3, 4, 5, 6, 7]
2000
1
```

**Unlike strings, lists are mutable. This means that individual elements of a list can be changed or removed. However, since strings are immutable (or not mutable), the same cannot be done with strings.**

**The `list()` function can be used to convert a valid sequence to a list. Since strings are sequences, they can also be converted to a list.**

In [1]:

```
my_str = 'Hello, I am immutable'
my_lst = list('Hello, I am mutable')
print(my_str)
print(my_lst)
print('my_str',my_str[10]) #Extract the character at the 10th index from the string.
print('my_lst', my_lst[10]) #Extract the character at the 10th index from the list.

my_str[10] = 'n'  #Since strings are immutable, this statement will result in an error.
my_lst[10] = 'n'   #Since lists are mutable, this statement is valid.
print(my_lst)
```

```
Hello, I am immutable
['H', 'e', 'l', 'l', 'o', ',', ' ', 'I', ' ', 'a', 'm', ' ', 'm', 'u', 't', 'a', 'b', 'l', 'e']
my_str m
my_lst m


---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-1-79a02cf61b91> in <module>()
     15 print('my_lst', my_lst[10]) #Extract the character at the 10th index from the list.
     16
---> 17 my_str[10] = 'n'  #Since strings are immutable, this statement will result in an error.
     18 my_lst[10] = 'n'   #Since lists are mutable, this statement is valid.
     19 print(my_lst)

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