6) Python Exercise:
Using python lists and strings, demonstrate the difference between mutable and immutable data types

Answer:
The following is a python Idle session using list and string operations in order to demonstrate difference between mutable and immutable data objects.

The id() function returns a unique numeric identifier for the data object passed in as the input argument. The return value corresponds to a memory location. It is hardly ever used in Python applications but I use it here for nest ruction all purposes. Notice that the id() function returns the same number before and after changing a list. However the number is NOT the same when called before and after changing a string.

Also notice the differences between changing lists and strings using built-in class methods. A list.method() will change the list without reassignment whereas a string.method() requires reassignment. If the assignment is to original variable name, the id() changes which means that the variable name is referencing a different string data object.

What is often stated is that mutable objects can be changed “in place” whereas an immutable object cannot be changed “in place”.

---

```
Suzannes-MBP:~ suzanneberger$ python
Python 2.7.10 (default, Jul 14 2015, 19:46:27)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.39)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> this_list = [1, 2, 3]
>>> id(this_list)
4357789744
>>> this_list.append(17)
>>> this_list
[1, 2, 3, 17]
>>> id(this_list)
4357789744
>>> this_list[1] = "foobar"
>>> this_list
[1, 'foobar', 3, 17]
>>> id(this_list)
4357789744
>>>
```

```
>>>
>>> this_string = "Hello world"
>>> id(this_string)
4357937696
>>> this_string.upper()
'HELLO WORLD'
>>> this_string
'Hello world'
>>> this_string = this_string.upper()
>>> this_string
'HELLO WORLD'
>>> id(this_string)
4357937792
>>> this_string[0]
'H'
>>> this_string[0] = 'F'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>>
```