Lecture 4 Object: Mutable/Immutable, Attributes/Methods

In Python:

- An object's **identity** never changes once it has been created;
- Whether its **value** can change or not really depends:
  - If the value can be changed, the object is called *mutable* -- it is more flexible!
  - If the value cannot be changed, the object is called *immutable* -- it is safer!

For beginners, mutable/immutable objects can easily lead to errors that are very difficult to debug. (https://florimond.dev/blog/articles/2018/08/python-mutable-defaults-are-the-source-of-all-evil/)

- Whether the object is mutable or not? It depends on its **type**:
  - (for built-in types) **List**, Dictionary and Set are mutable;
  - Int, Float, String, Bool, Tuple ... are immutable.
  - Numpy array is also mutable (will talk about it later)

In [4]:
```
a = [1,2,3]
print(id(a))
a[0]=0
print(a)
print(id(a))
```

```
140717235953008
[0, 2, 3]
140717235953008
```

Compare it with the following two examples:

In [3]:
```
a = 1
print(id(a))
a = 0
print(id(a))
```

```
4516390016
4516389984
```

In [5]:
```
a = [1,2,3]
print(id(a))
a = [0,2,3]
print(a)
print(id(a))
```

```
140717235967472
[0, 2, 3]
140717235953008
```

Now it's time to test your understandings. Recall our examples in Lecture 2 and solve it yourself!

In [6]:
```
a = 1000
b = a
b = 1
print(a)
```

```
1000
```

In [7]:
```
a = [1000,1]
b = a
b = [1,1]
print(a)
```

```
[1000, 1]
```

In [8]:
```
a = [1000,1]
b = a
b[0] = 1
print(a)
```

```
[1, 1]
```

Indeed, what is the solution if we really want to "copy" a list?

There are multiple solutions to this (https://www.geeksforgeeks.org/python-cloning-copying-list/), and we will mention one here using the copy *method*.

In [9]:
```
a = [1,2,3]
b = a.copy()
a[0] = 0
print(a)
print(b)
```

```
[0, 2, 3]
[1, 2, 3]
```

Attributes and Methods of Python Object

Roughly speaking,

- attributes are the variables stored within object;
- methods are the functions stored within object.

String attributes/methods

In [29]:
```
text = "Data Science"
text.__doc__
```

Out[29]:
```
"str(object='') -> str\nstr(bytes_or_buffer[, encoding[, errors]]) -> str\n\nCreate a n
ew string object from the given object. If encoding or\nerrors is specified, then the o
bject must expose a data buffer\nthat will be decoded using the given encoding and erro
r handler.\nOtherwise, returns the result of object.__str__() (if defined)\nor repr(obj
ect).\nencoding defaults to sys.getdefaultencoding().\nerrors defaults to 'strict'."
```

In [46]:
```
text.upper() # return a new string object with upper case
```

Out[46]:
```
'DATA SCIENCE'
```

In [44]:
```
text # See? the original text is not affected
```

Out[44]:
```
'Data Science'
```

In [35]:
```
text.lower() # return a new string object
```

Out[35]:
```
'data science'
```

In [34]:
```
text.capitalize() # return a new string object
```

Out[34]:
```
'Data science'
```

Lists attributes/methods

In [36]:
```
numbers = [1, 4, 0, 2, 9, 9, 10]
numbers.__class__
```

Out[36]:
```
list
```

In [37]:
```
print(numbers)
print(id(numbers))
numbers.reverse() # does NOT return a new LIST object! just modify the original list --
 remember that list is mutable object
print(numbers) # [10, 9, 9, 2, 0, 4, 1]
print(id(numbers))
```

```
[1, 4, 0, 2, 9, 9, 10]
140717238799328
[10, 9, 9, 2, 0, 4, 1]
140717238799328
```

It is INCORRECT to write in this way:

In [43]:
```
numbers_reverse = numbers.reverse() # it is the INCORRECT way to reverse a list!!!
print(numbers_reverse)
```

```
None
```

In [38]:
```
numbers.sort()
print(numbers)
```

```
[0, 1, 2, 4, 9, 9, 10]
```

In [11]:
```
dir(text)
```

Out[11]:
```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mod__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmod__',
 '__rmul__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'capitalize',
 'casefold',
 'center',
 'count',
 'encode',
 'endswith',
 'expandtabs',
 'find',
 'format',
 'format_map',
 'index',
 'isalnum',
 'isalpha',
 'isascii',
 'isdecimal',
 'isdigit',
 'isidentifier',
 'islower',
 'isnumeric',
 'isprintable',
 'isspace',
 'istitle',
 'isupper',
 'join',
 'ljust',
 'lower',
 'lstrip',
 'maketrans',
 'partition',
 'replace',
 'rfind',
 'rindex',
 'rjust',
```

```
 'rpartition',
 'rsplit',
 'rstrip',
 'split',
 'splitlines',
 'startswith',
 'strip',
 'swapcase',
 'title',
 'translate',
 'upper',
 'zfill']
```

In [17]:
```
dir(numbers)
```

Out[17]:
```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__imul__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__rmul__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'append',
 'clear',
 'copy',
 'count',
 'extend',
 'index',
 'insert',
 'pop',
 'remove',
 'reverse',
 'sort']
```