Objects of built-in type that are **mutable** are:

* Lists
* Sets
* Dictionaries
* User-Defined Classes (It purely depends upon the user to define the characteristics)

—>>

Objects of built-in type that are **immutable** are:

* Numbers (Integer, Rational, Float, Decimal, Complex & Booleans)
* Strings
* Tuples
* Frozen Sets
* User-Defined Classes (It purely depends upon the user to define the characteristics)

**Mutable Objects in Python**

#Creating a list which contains name of Indian cities  
(*Hint şehirlerinin adını içeren bir liste oluşturma*)

cities = [‘Delhi’, ‘Mumbai’, ‘Kolkata’]

# Printing the elements from the list cities, separated by a comma & space  
(*Bir virgül ve boşlukla ayrılmış, liste şehirlerinden öğeleri yazdırmak*)

for city in cities:

print(city, end=’, ’)

Output [1]: Delhi, Mumbai, Kolkata

#Printing the location of the object created in the memory address in hexadecimal format  
(*Bellek adresinde oluşturulan nesnenin konumunu onaltılık formatta izleme*)

print(hex(id(cities)))

Output [2]: 0x1691d7de8c8

#Adding a new city to the list cities  
(*Liste şehirlerine yeni bir şehir eklemek*)

cities.append(‘Chennai’)

#Printing the elements from the list cities, separated by a comma & space

(*Bir virgül ve boşluk ile ayrılmış, liste şehirlerinden öğeleri yazma*)

for city in cities:

print(city, end=’, ’)

Output [3]: Delhi, Mumbai, Kolkata, Chennai

#Printing the location of the object created in the memory address in hexadecimal format

*(Bellek adresinde oluşturulan nesnenin konumunu onaltılık formatta izleme)*

print(hex(id(cities)))

Output [4]: 0x1691d7de8c8

The above example shows us that we were able to change the internal state of the object ‘cities’ by adding one more city ‘Chennai’ to it, yet, the memory address of the object did not change. This confirms that we did not create a new object, rather, the same object was changed or mutated. Hence, we can say that the object which is a type of list with reference variable name ‘cities’ is a MUTABLE OBJECT.

*(Yukarıdaki örnek bize bir şehir daha 'Chennai' ekleyerek 'şehirler' nesnesinin iç durumunu değiştirebildiğimizi gösteriyor, ancak nesnenin bellek adresi değişmedi. Bu, yeni bir nesne oluşturmadığımızı doğrular, bunun yerine aynı nesne değiştirildi veya mutasyona uğradı. Dolayısıyla, referans değişken adına sahip bir liste olan nesnenin, 'şehirler' değişebilir bir nesne (MUTABLE OBJECT) olduğunu söyleyebiliriz.)*

## ****Immutable Objects in Python****

#Creating a Tuple which contains English name of weekdays

*(Hafta içi İngilizce adı içeren bir tuple oluşturma)*

weekdays = ‘Sunday’, ‘Monday’, ‘Tuesday’, ‘Wednesday’, ‘Thursday’, ‘Friday’, ‘Saturday’

# Printing the elements of tuple weekdays

print(weekdays)

Output [1]: (‘Sunday’, ‘Monday’, ‘Tuesday’, ‘Wednesday’, ‘Thursday’, ‘Friday’, ‘Saturday’)

#Printing the location of the object created in the memory address in hexadecimal format

print(hex(id(weekdays)))

Output [2]: 0x1691cc35090

#tuples are immutable, so you cannot add new elements, hence, using merge of tuples with the # + operator to add a new imaginary day in the tuple ‘weekdays’

*(Tuples değişmezdir, bu nedenle yeni bir unsur ekleyemezsiniz, bu nedenle # + operatör ile tuples birleştirme kullanarak ‘Hafta içi’ Tuple’e yeni bir hayali gün eklemek için)*

weekdays += ‘Pythonday’,

#Printing the elements of tuple weekdays

print(weekdays)

Output [3]: (‘Sunday’, ‘Monday’, ‘Tuesday’, ‘Wednesday’, ‘Thursday’, ‘Friday’, ‘Saturday’, ‘Pythonday’)

#Printing the location of the object created in the memory address in hexadecimal format

print(hex(id(weekdays)))

Output [4]: 0x1691cc8ad68

This above example shows that we were able to use the same variable name that is referencing an object which is a type of tuple with seven elements in it. However, the ID or the memory location of the old & new tuple is not the same. We were not able to change the internal state of the object ‘weekdays’. The Python program manager created a new object in the memory address and the variable name ‘weekdays’ started referencing the new object with eight elements in it.  Hence, we can say that the object which is a type of tuple with reference variable name ‘weekdays’ is an IMMUTABLE OBJECT.

*(Yukarıdaki bu örnek, içinde yedi element bulunan bir tuple türü olan bir nesneye atıfta bulunan aynı değişken adını kullanabildiğimizi göstermektedir. Ancak, eski ve yeni tuple'nin kimliği veya bellek konumu aynı değildir. "Hafta içi" nesnesinin iç durumunu değiştiremedik. Python program yöneticisi bellek adresinde yeni bir nesne oluşturdu ve "Hafta içi" değişken adı yeni nesneyi sekiz öğe ile atıfta bulunmaya başladı. Bu nedenle, referans değişken adına sahip bir tuple türü olan nesnenin, 'hafta içi' değişmez bir nesne (IMMUTABLE OBJECT) olduğunu söyleyebiliriz.)*