Python 5 Collection Types

3 Kasım 2022 Perşembe 19:05

AWS = Doküman okumadır.

Collection Types hold multiple values. In our programs, we usually need to group several items to render as a single object. We use collection types of data to do this job.

input :

```
country = ['USA', 'Brasil', 'UK', 'Germany', 'Turkey', 'New
    Zealand']

print(country)

```

output :

```
['USA', 'Brasil', 'UK', 'Germany', 'Turkey', 'New Zealand']

```

You do this when you want to create a `list` from an iterable object: that is, type of object whose elements you can import individually. The lists are iterable like other collections and string types. Let's create another `list` using `list()` function and compare with 👉'`[]`'.

List ler iterable dır.

input :

```
string_1 = 'I quit smoking'

new_list_1 = list(string_1)  # we created multi element list
print(new_list_1)

new_list_2 = [string_1]  # this is a single element list
print(new_list_2)

```

output :

```
['I', ' ', 'q', 'u', 'i', 't', ' ', 's', 'm', 'o', 'k', 'i', 'n',
    'g']
['I quit smoking']

```

Yukarıda iki farklı liste oluşturulmuş. Bunlardan ilki list() komutuyla. Böyle oluşturulduğunda string_1 e atanan değer olan "I quit smokinq" boşlukları da dahil olmak üzere her bir karakteri listenin elemenaları olmuş. Yani listeler aynı değeri birdan fazla sayıda barındırabiliyor.

Diper yöndemde ise [] komutu kullanılmış. Bunun çıktısında ise 'I quit smoking' tek bir liste elemanı oluyor.

The components of a `list` are not limited to a single data type, given that Python is a dynamic language: e.g.

```
mixed_list = [11, 'Joseph', False, 3.14, None, [1, 2, 3]]
```

Tips:

- As you see above, one or more of the **list elements** can even be a `list`.

List in içindeki elementler in hepsi aynı datatype olmak zorunda değildir. Bir list bile başka bir listin elemanı olabilir.

```
warning = 'You must quit smoking!'

print(len(list(warning)))

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
                                 Focus folder in explore
a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/CLARUSWAY_
$ cd c:/Users/a/Desktop/CLARUSWAY_Python_Dersl

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/CLARUSWAY_
$ C:/Users/a/AppData/Local/Programs/Python/Pyt
22
```

Yukarıda warning i list e çevirip uzunluğunu belirtmesini istediğimizde her elemanını saydığını görüyoruz.

In most cases, we'll have to make an empty `list` to fill it later with

the data you want.

```
empty_list_1= []

empty_list_2 = list()

```

We can add an element into a `list` using `.append()` or `.insert()` methods.

Pythonda çoğu zaman boş liste oluşturup ona sonradan eleman ekleyeceğiz. Bunun için iki komut kullanacağız. Birisi .append() komutu diğeri ise .insert()

list.append(element) syntax ı ile listenin en sonun eleman ekleriz. Ancak bunun sonucunu görmemiz için ekledikten sonra print komutuyla çıktı almamız gerekir.

```
empty_list_1 = []
empty_list_1.append('114')
empty_list_1.append('plastic-free sea')

print(empty_list_1)


PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/CLARUSWAY_Python_
$ cd c:/Users/a/Desktop/CLARUSWAY_Python_Dersleri

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/CLARUSWAY_Python_
$ C:/Users/a/AppData/Local/Programs/Python/Python311/
['114', 'plastic-free sea']
```

```
city = ['New York', 'London', 'Istanbul', 'Seoul', 'Sydney']
city.append('Adis Ababa')
print(city)

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/CLARUSWAY_Python_Dersleri
$ cd c:/Users/a/Desktop/CLARUSWAY_Python_Dersleri

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/CLARUSWAY_Python_Dersleri
$ C:/Users/a/AppData/Local/Programs/Python/Python311/python.exe c:/Users/a/D
['New York', 'London', 'Istanbul', 'Seoul', 'Sydney', 'Adis Ababa']
```

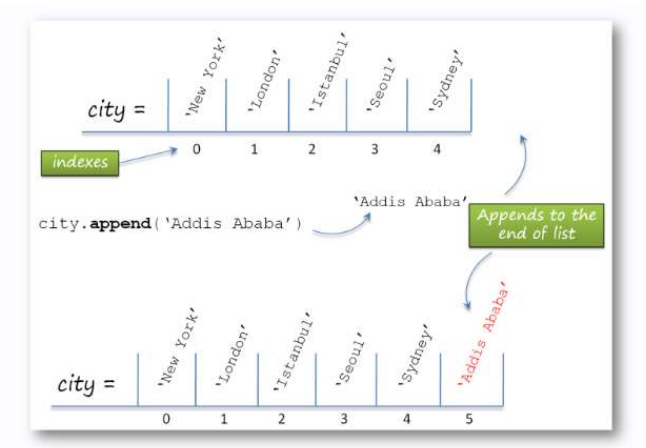


Diagram of '.append( )' Method

list.insert(index, object) ise bir list e index ile eleman ekler. Syntax ı yazarken önce index i sonra ekleyeceğimiz object i ekleriz.

```
city = ['New York', 'London', 'Istanbul', 'Seoul', 'Sydney']
city.append('Adis Ababa')
city.insert(2, 'Stockholm')
print(city)


PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/CLARUSWAY_Python_Dersleri
$ cd c:/Users/a/Desktop/CLARUSWAY_Python_Dersleri

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/CLARUSWAY_Python_Dersleri
$ C:/Users/a/AppData/Local/Programs/Python/Python311/python.exe c:/Users/a/Desktop/
['New York', 'London', 'Stockholm', 'Istanbul', 'Seoul', 'Sydney', 'Adis Ababa']
```

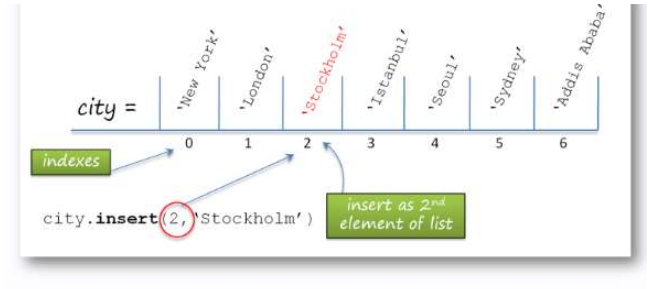


Diagram of 'insert()' Method

We can remove the elements in lists using `list.remove()` method or sort the elements using `list.sort()` method. Examine the example :

list.remove() komutuyla bir elemanı listeden kaldırırız.

List.sort() ile de liste elemanlarını sıralayabiliriz:

```
67  city = ['New York', 'London', 'Istanbul', 'Seoul', 'Sydney']
68  city.append('Adis Ababa')
69  city.insert(2, 'Stockholm')
70
71  city.remove('London')
72  print(city)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/CLARUSWAY_Python_Dersleri
$ cd c:/Users/a/Desktop/CLARUSWAY_Python_Dersleri

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/CLARUSWAY_Python_Dersleri
$ C:/Users/a/AppData/Local/Programs/Python/Python311/python.exe c:/Users/a/D
['New York', 'Stockholm', 'Istanbul', 'Seoul', 'Sydney', 'Adis Ababa']
```

```
67  city = ['New York', 'London', 'Istanbul', 'Seoul', 'Sydney']
68  city.append('Adis Ababa')
69  city.insert(2, 'Stockholm')
70
71  city.remove('London')
72  city.sort()
73  print(city)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/CLARUSWAY_Python_Dersleri
$ cd c:/Users/a/Desktop/CLARUSWAY_Python_Dersleri

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/CLARUSWAY_Python_Dersleri
$ C:/Users/a/AppData/Local/Programs/Python/Python311/python.exe c:/Users/a/Desk
['Adis Ababa', 'Istanbul', 'New York', 'Seoul', 'Stockholm', 'Sydney']
```

Tips:

- Remember! Elements of a list are counted from left to right and start with zero as in string types.

List elemanları soldan sağa sayılır ve strginlerde olduğu gibi index 0 dan başlar.

List elemanları len() komutuyla sayılabilir.

```
1  my_list = [1, 3, 5, 7]
2  print(my_list * 3)
```

```
1  [1, 3, 5, 7, 1, 3, 5, 7, 1, 3, 5, 7]
```

List ler sıralı olduğu için yukarıdaki işlemin sonucu sırasıyla çıktı verir.
"index()" komutu bir elementin listenin ilk görünümündeki index ini verir:

Example

What is the position of the value 32:

```
fruits = [4, 55, 64, 32, 16, 32]
```

```
x = fruits.index(32)
```

Try it Yourself »

Note: The `index()` method only returns the *first* occurrence of the value.

"del" komutu pythonda nesneleri siler. List ler de object olduğu için onları da siler:

```
90  x = ["apple", "banana", "cherry"]
91
92  del x[0]
93
94  print(x)
95
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYT

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/CLARUSWAY_Pytho
$ cd c:/Users/a/Desktop/CLARUSWAY_Python_Dersleri

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/CLARUSWAY_Pytho
$ C:/Users/a/AppData/Local/Programs/Python/Python31
['banana', 'cherry']
```

List in içinden bir elemanı silmesini istediğimiz için, "del x[0]" ile köşeli parantez içinde index belirttik.

"pop()" fonksiyonu ile de list içindeki bir elemanı index iyle silebiliyoruz. (remove fonksiynunda nesneyi yazıyorduk):

```
96  fruits = ['apple', 'banana', 'cherry']
97
98  fruits.pop(1)
99  print(fruits)
100
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/CLARUSWAY_Python_Dersleri
$ cd c:/Users/a/Desktop/CLARUSWAY_Python_Dersleri

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/CLARUSWAY_Python_Dersleri
$ C:/Users/a/AppData/Local/Programs/Python/Python311/python.exe
['apple', 'cherry']
```

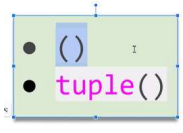
Each part of the slice has a default value, so they are **optional**. If we don't assign a value to the **start** index, it is considered to be **0**; if we don't assign a value to the **stop** index, it will be the **same as** the **length** of the sequence.

- `my_list[:]`: returns the full copy of the sequence
- `my_list[start:]` : returns elements from `start` to the end element
- `my_list[:stop]` : returns element from the 1st element to `stop-1`
- `my_list[::step]` : returns each element with a given `step`

Tuples List lerer göre daha hızlı çalışır.
Tuple ı nasıl oluştururuz:

Creating a `tuple`

- We have two basic ways to create a tuple.



Parantez içine almaya tuple deniyor. Değişmesini istemediğimiz verileri tuples içine koyuyoruz.

```
my_tuple = ("wakabayashi",)  # tuple olması için string yanına bir tane comma koymamız gerekiyor.
my_tuple2 = tuple("wakayabashi")
print(my_tuple, type(my_tuple))
print(my_tuple2, type(my_tuple2))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/CLARUSWAY_Python_Dersleri
$ cd c:/Users/a/Desktop/CLARUSWAY_Python_Dersleri

a@DESKTOP-TPH8ITT MINGW64 ~/Desktop/CLARUSWAY_Python_Dersleri
$ C:/Users/a/AppData/Local/Programs/Python/Python311/python.exe c:/Users/a/Desktop/CLARUSWAY_Python_Dersler
('wakabayashi',) <class 'tuple'>
('w', 'a', 'k', 'a', 'y', 'a', 'b', 'a', 's', 'h', 'i') <class 'tuple'>
```

Tuple ın indexini girebiliyoruz. Ancak ekleme veya çıkarma yapamıyoruz. Yani append() remove() gibi fonksiyonları tuple için kullanamıyoruz.

Tuples are commonly used for small collections of values that will not need to change, such as an IP address and port. If we have unchanged data, we should choose **tuples** because it is much **faster than lists**.

Tuples ları değişmeyecek küçük collectionlar için kullanıyoruz. (IP adresi port gibi) List ten daha hızlıdır.

The same indexing rules for lists also apply to tuples. Tuples can also be nested and the values can be any valid Python valid.

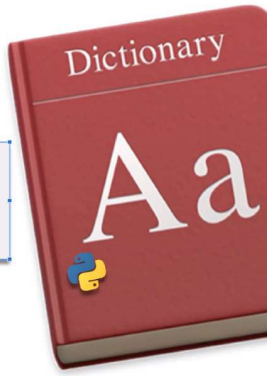
List lerde geçerli olan tüm indexleme kuralları tuple lar için de geçerlidir. İç içe kullanılabilir ve Python da geçerli herhangi bir değere sahip olabilirler

Tuple için parantez içine almak zorunda değiliz. String sonuna virgül koyarsak tuple olur.

Definitions

- Dictionaries

```
{key1 : value1,
 key2 : value2}
```



Dictionary bir key ve bir value çiftinden oluşur. Baktığınız zaman ilk çift birinci elemandır, ikincisi ikinci elemandır. Yani key ve value birlikte bir elemandır.

Key, bir dictionary içerisinde bir kere kullanılır. Her key unique olmak zorunda.

Creating a `dict` (review)

- A `dict` can be created by enclosing pairs, separated by commas, in curly-braces 👉 `{}`.
- Another way to create a `dict` is to call the `dict()` function.

- `{}`
- `dict()`

```
grocer1 = {'fruit':'apple', 'drink':'water'}
grocer2 = dict(fruit='apple', drink='water')
print(grocer1)
print(grocer2)
```

```
{'fruit': 'apple', 'drink': 'water'}
{'fruit': 'apple', 'drink': 'water'}
```

CLARUSWAY©

Manuel olarak yazarken key i string şeklinde yazıyoruz. Ancak dict fonksiyonu ile yazarken text= şeklinde yazıyoruz.

Creating a `dict`

Tips:

- Note that keys and values can be of different types.

```
mix_values = {'animal': ('dog', 'cat'),  # tuple type
              'planet': ['Neptun', 'Saturn', 'Jupiter'],  # list type
              'number': 40,  # int type
              'pi': 3.14,  # float type
              'is_good': True}  # bool type

mix_keys = {22 : "integer",
            1.2 : "float",
            True : "boolean",
            "key" : "string"}

```

CLARUSWAY©
WAY TO REINVENT YOURSELF

Key ve value farklı datatype ta olabilir. #unhashable hatası aldığımızda key olarak tuple, set ….

Creating a `dict`

- Now, it's time to create a `dict` using `dict()` function :

```
dict_by_dict = dict(animal='dog', planet='neptun', number=40, pi=3.14, is_good=True)

print(dict_by_dict)
```

```
{'animal': 'dog',
'planet': 'neptun',
'number': 40,
'pi': 3.14,
'is_good': True}
```

⚠ Avoid ! :

- Do not use quotes for `keys` when using the `dict()` function to create a dictionary.

Main Operations with `dicts` (review)

- You can access all;
  - items using the `.items()` method,
  - keys using the `.keys()` method,
  - values using the `.values()` method.

input :

```
country = ['USA', 'Brasil', 'UK', 'Germany', 'Turkey', 'New
    Zealand']

print(country)

```

output :

```
['USA', 'Brasil', 'UK', 'Germany', 'Turkey', 'New Zealand']

```

Scratch Time ! : Create this list with scratch.

Tips:

- All the country names are printed in the same order as they were stored in the list because lists are **ordered**.

The components of a `list` are not limited to a single data type, given that Python is a dynamic language: e.g.

```
mixed_list = [11, 'Joseph', False, 3.14, None, [1, 2, 3]]
```

Tips:

- As you see above, one or more of the **list elements** can even be a `list`.