## İntroducing Lists

Lists sizə məlumat dəstlərini bir yerdə saxlamağa imkan verir — istər bir neçə element olsun, istər milyonlarla element. Lists Python-un yeni başlayanlar üçün asanlıqla istifadə edilə bilən ən güclü xüsusiyyətlərindən biridir və proqramlaşdırmanın bir çox vacib anlayışlarını birləşdirir.

**What is a list ?**

List müəyyən ardıcıllıqla düzülmüş elementlər toplusudur. Siz list yarada bilərsiniz ki, orada əlifbanın hərfləri olsun, 0–9 rəqəmləri olsun və ya ailənizdəki bütün insanların adları olsun. List-ə istədiyiniz hər şeyi əlavə edə bilərsiniz və list daxilindəki elementlərin bir-biri ilə əlaqəli olması tələb olunmur. Çünki list adətən birdən çox elementdən ibarət olur, buna görə də list-in adına çoxluq formasında ad vermək yaxşı fikirdir: məsələn letters, digits, names.

Python-da kvadrat mötərizələr ([]) list yaratmaq üçün istifadə olunur və list daxilindəki elementlər vergüllə ayrılır. Aşağıda müxtəlif velosiped növlərindən ibarət sadə bir list nümunəsi var:

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']

print(bicycles)
```

Əgər siz Python-dan list-i print etməyi istəsəniz, Python kvadrat mötərizələr daxil olmaqla list-in öz təmsilini qaytaracaq:

['trek', 'cannondale', 'redline', 'specialized']

**Accessing Elements in a List**

Lists ardıcıllıqla düzülmüş kolleksiyalardır, buna görə list-də istənilən elementə çıxmaq üçün Python-a həmin elementin mövqeyini — yəni index-i — demək lazımdır. List daxilində bir elementə çıxmaq üçün list-in adını yazırsınız, sonra isə kvadrat mötərizədə həmin elementin index-ni göstərirsiniz.

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']

print(bicycles[0])
```

List-dən tək bir element istədikdə, Python həmin elementi kvadrat mötərizə olmadan qaytarır:

trek

Siz həmçinin list-dəki istənilən elementə 2-ci fəsildə öyrəndiyiniz string metodlarını tətbiq edə bilərsiniz. Məsələn, 'trek' sözünü daha səliqəli göstərmək üçün **title()** metodundan istifadə etmək mümkündür:

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']

print(bicycles[0].title())
```

Bu nümunə əvvəlki nümunə ilə eyni nəticəni verir, sadəcə bu dəfə nəticə **'Trek'** olaraq baş hərfi böyük formada yazılır.

**Index Positions Start at 0, Not 1**

Python list-də ilk elementi 1-ci mövqedə yox, **0-cı mövqedə** hesablayır. Bu, əksər proqramlaşdırma dillərində belədir və səbəbi list əməliyyatlarının daha aşağı səviyyədə necə həyata keçirilməsi ilə bağlıdır. List-də ikinci elementin index-i **1** olur. Bu sayma sistemindən istifadə edərək, list-də istədiyiniz elementi sadəcə onun mövqeyindən **bir vahid çıxmaqla** əldə edə bilərsiniz. Məsələn, list-də dördüncü elementə çıxmaq üçün index **3** göstərilir.

Aşağıdakı nümunədə **index 1** və **index 3**-dəki bicycles elementləri çağırılır:

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']

print(bicycles[1])

print(bicycles[3])
```

Bu kod list-dəki ikinci və dördüncü bicycles elementlərini qaytarır:

```
cannondale

specialized
```

Python-da list-in son elementinə çıxış üçün xüsusi bir sintaksis var. Əgər siz **index -1** istəsəniz, Python həmişə list-dəki **son elementi** qaytarır:

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']

print(bicycles[-1])
```

Bu kod **'specialized'** dəyərini qaytarır. Bu sintaksis olduqca faydalıdır, çünki çox vaxt list-in tam uzunluğunu bilmədən son elementlərə çıxmaq istəyəcəksiniz. Bu qayda digər mənfi index-lərə də aiddir:

Index -2 (sondan ikinci element)

Index -3 (sondan 3-cu element) ve s.

**Using Individual Values from a List**

List-dən götürdüyünüz ayrı bir dəyəri digər variable-lar kimi istifadə edə bilərsiniz. Məsələn, list-dən alınmış bir dəyər əsasında mesaj yaratmaq üçün f-string istifadə edə bilərsiniz.

Gəlin list-dəki ilk bicycle elementini götürüb həmin dəyərdən istifadə edərək bir mesaj yaradaq:

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']

message = f"My first bicycle was a {bicycles[0].title()}."

print(message)
```

Burada **bicycles[0]** dəyərindən istifadə edərək bir cümlə qurulur və **message** adlı variable-a yazılır. Çıxış isə list-dəki ilk bicycle haqqında sadə bir cümlədir:

My first bicycle was a Trek.

## Changing, Adding, and Removing Elements

Yaratdığınız Lists-in çoxu dinamik olacaq — yəni proqram çalışdığı müddətdə list qurulacaq və lazım olduqca ora elementlər əlavə olunacaq və silinəcək.

**Modifying Elements in a List**

Bir elementi dəyişdirmək üçün istifadə olunan sintaksis list-də elementə çıxış sintaksisinə çox bənzəyir. Hər hansı elementi dəyişmək üçün list-in adını yazırsınız, ardından dəyişdirmək istədiyiniz elementin index-ni göstərirsiniz və daha sonra həmin elementə vermək istədiyiniz yeni value-ni yazırsınız.

Məsələn, gəlin belə qəbul edək ki, bizdə motorcycles adlı list var və list-də ilk element `'honda'`-dır. Bəs bu ilk elementi necə dəyişərdik?

```
motorcycles = ['honda', 'yamaha', 'suzuki']

print(motorcycles)
```

```
motorcycles[0] = 'ducati'

print(motorcycles)
```

**Adding Elements to a List**

**List-in sonuna element əlavə etmək (append())**

Bir list-ə yeni element əlavə etməyin ən sadə yolu append() metodundan istifadə etməkdir. Append etdikdə yeni element list-in sonuna əlavə olunur.

Eyni əvvəlki list-dən istifadə edərək, yeni `'ducati'` elementini list-in sonuna əlavə edək:

```
motorcycles = ['honda', 'yamaha', 'suzuki']

print(motorcycles)


motorcycles.append('ducati')

print(motorcycles)
```

append() nəticəsində list-dəki digər elementlər dəyişmədən qalır:

['honda', 'yamaha', 'suzuki']

['honda', 'yamaha', 'suzuki', 'ducati']

**Boş list-ə append() ilə elementlər əlavə etmək**

Append() metodu dinamik lists yaratmağı çox asanlaşdırır.

Məsələn, ilk öncə boş list yarada bilərsiniz, sonra append() çağırışları ilə elementləri ardıcıl olaraq əlavə edə bilərsiniz:

```
motorcycles = []


motorcycles.append('honda')

motorcycles.append('yamaha')

motorcycles.append('suzuki')
```

```
print(motorcycles)
```

Bu üsul çox geniş istifadə olunur, çünki istifadəçi hansı datanı daxil edəcək — adətən proqram işləyərkən məlum olur. Bu zaman əvvəlcə boş list yaradılır, sonra istifadəçinin daxil etdiyi hər bir yeni value həmin list-ə append edilir.

**Inserting Elements into a List - List-in istənilən mövqeyinə element daxil etmək (insert())**

Siz insert() metodundan istifadə edərək list-in istənilən mövqeyinə yeni element daxil edə bilərsiniz. Bunun üçün insert() iki parametr alır:

1. elementin yerləşdiriləcəyi index,
2. həmin elementin value-u.

```
motorcycles = ['honda', 'yamaha', 'suzuki']

motorcycles.insert(0, 'ducati')

print(motorcycles)
```

Burada insert(0, 'ducati') ilk mövqeyə `'ducati'` əlavə edir.

Insert() həmin mövqedə yer açır və digər bütün elementləri sağa sürüşdürür

**Removing Elements from a List**

Çox zaman list-dən bir elementi və ya bir neçə elementi silmək lazım olur. Siz bir elementi:

- ya onun list-dəki mövqeyinə görə,
- ya da value-suna görə silə bilərsiniz.

**Bir elementi del operatoru ilə silmək**

Əgər list-dən silmək istədiyiniz elementin mövqeyini (index) bilirsinizsə, del operatorundan istifadə edə bilərsiniz.

```
motorcycles = ['honda', 'yamaha', 'suzuki']

print(motorcycles)

del motorcycles[0]

print(motorcycles)
```

Buradakı **del** list-dəki ilk elementi – `'honda'` – silir

Əgər index bilirsinizsə, **del** ilə list-in istənilən elementini silə bilərsiniz. Məsələn, ikinci elementi – `'yamaha'` – belə silmək olar:

```
motorcycles = ['honda', 'yamaha', 'suzuki']

print(motorcycles)

del motorcycles[1]

print(motorcycles)
```

Hər iki nümunədə, **del** istifadə edildikdən sonra silinmiş elementə artıq çıxış etmək olmur.

**Bir elementi pop() metodu ilə silmək**

Bəzən list-dən elementi sildikdən sonra həmin dəyərdən istifadə etmək istəyirsiniz. Məsələn:

- yeni vurulan bir obyektin koordinatlarına baxmaq,
- bir istifadəçini aktivlər siyahısından çıxarıb qeyri-aktivlər siyahısına əlavə etmək və s.

pop() metodu list-in sonuncu elementini silir, amma silinən elementi istifadə etməyə imkan verir. “pop” sözü list-i üst-üstə yığılmış elementlər toplusu (stack) kimi xatırlamaqdan gəlir — sonuncu elementi “çıxartmaq” kimi.

Gəlin, list-dən bir motosiklet pop edək:

```
motorcycles = ['honda', 'yamaha', 'suzuki']

print(motorcycles)

popped_motorcycle = motorcycles.pop()

print(motorcycles)

print(popped_motorcycle)
```

Burada:

- əvvəlcə list yaradılıb və çap olunub,

- daha sonra **pop()** ilə son element çıxarılıb və *popped_motorcycle* dəyişəninə yazılıb,
- sonra həm qalan list, həm də silinən element göstərilib.

Output:

['honda', 'yamaha', 'suzuki']

['honda', 'yamaha']

suzuki

Bu metodu necə istifadə etmək olar? Təsəvvür edin ki, list-də motosikletlər **alınma sırasına** görə düzülüb. Bu halda pop() ilə son aldığınız motosiklet haqqında cümlə qura bilərsiniz:

```
motorcycles = ['honda', 'yamaha', 'suzuki']

last_owned = motorcycles.pop()

print(f"The last motorcycle I owned was a {last_owned.title()}.")
```

Output:

The last motorcycle I owned was a Suzuki.

**pop() metodundan istifadə edərək istənilən mövqedən element silmək**

pop() metodunda index yazmaqla list-dən istənilən mövqedəki elementi silə bilərsiniz:

```
motorcycles = ['honda', 'yamaha', 'suzuki']

first_owned = motorcycles.pop(0)

print(f"The first motorcycle I owned was a {first_owned.title()}.")
```

Output:

The first motorcycle I owned was a Honda.

# del və pop() arasında seçim

- Silinən elementdən **heç bir şəkildə istifadə etməyəcəksinizsə → del**
- Silinən elementi **sonradan istifadə edəcəksinizsə → pop()**

# Bir elementi remove() metodu ilə dəyərə görə silmək

Bəzən silmək istədiyiniz elementin index-ını bilmirsiniz, amma **dəyərini** bilirsiniz. Bu halda **remove()** metodundan istifadə edə bilərsiniz.

Məsələn, list-dən `'ducati'` dəyərini silmək:

```
motorcycles = ['honda', 'yamaha', 'suzuki', 'ducati']

print(motorcycles)

motorcycles.remove('ducati')

print(motorcycles)
```

Output:

['honda', 'yamaha', 'suzuki', 'ducati']

['honda', 'yamaha', 'suzuki']

Remove() silmək üçün dəyərin list-də nə vaxt birinci dəfə göründüyünü **tapır** və həmin elementi silir.

**remove() metodundan istifadə edərək silinən dəyərdən istifadə etmək**

Dəyəri silib, sonradan həmin dəyər barəsində məlumat yazmaq istəyirsinizsə:

```
motorcycles = ['honda', 'yamaha', 'suzuki', 'ducati']

print(motorcycles)

too_expensive = 'ducati'

motorcycles.remove(too_expensive)

print(motorcycles)

print(f"\nA {too_expensive.title()} is too expensive for me.")
```

Output:

['honda', 'yamaha', 'suzuki', 'ducati']

['honda', 'yamaha', 'suzuki']

A Ducati is too expensive for me.

**NOTE**

**remove()** metodu yalnız göstərilən dəyərin **birinci** dəfə göründüyü elementi silir. Əgər həmin dəyər list-də bir neçə dəfə varsa, hamısını silmək üçün **döngə (loop)** istifadə etmək lazımdır. Bu, 7-ci fəsildə öyrədilir.

### Sorting a List Permanently with the sort() Method

Python-un `sort()` metodu list-i çeşidləməyi nisbətən asan edir. Tutaq ki, bizdə avtomobillərin olduğu bir list var və bu list-in elementlərini əlifba qaydasına görə düzülmüş formada saxlamaq istəyirik. Sadəliyi qorumaq üçün fərz edək ki, list-dəki bütün sözlər kiçik hərflərlə yazılıb.

```
cars = ['bmw', 'audi', 'toyota', 'subaru']

cars.sort()

print(cars)
```

`sort()` metodu list-in ardıcıllığını **davamlı olaraq** dəyişir. Artıq avtomobillər əlifba sırası ilə düzülüb və orijinal ardıcıllığa geri qayıtmaq mümkün deyil:

['audi', 'bmw', 'subaru', 'toyota']

Bu list-i tərs əlifba sırasi ilə çeşidləmək üçün `sort(reverse=True)` istifadə edə bilərsiniz:

```
cars = ['bmw', 'audi', 'toyota', 'subaru']

cars.sort(reverse=True)

print(cars)
```

### Sorting a List Temporarily with the sorted() Function

List-in orijinal ardıcıllığını qoruyub saxlamaq, amma onu çeşidlənmiş formada göstərmək istədikdə `sorted()` funksiyasından istifadə edə bilərsiniz. `sorted()` yalnız görüntüləmə üçün list-i çeşidləyir, list-in özünü dəyişmir.

```
cars = ['bmw', 'audi', 'toyota', 'subaru']

print("Here is the original list:")

print(cars)
```

```
print("\nHere is the sorted list:")
print(sorted(cars))

print("\nHere is the original list again:")
print(cars)
```

Output:

```
Here is the original list:
['bmw', 'audi', 'toyota', 'subaru']

Here is the sorted list:
['audi', 'bmw', 'subaru', 'toyota']

Here is the original list again:
['bmw', 'audi', 'toyota', 'subaru']
```

`sorted()` həmçinin `reverse=True` arqumentini qəbul edir.

**Printing a List in Reverse Order**

List-in ardıcıllığını tərsinə çevirmək üçün `reverse()` metodundan istifadə edə bilərsiniz. Əgər avtomobillər list-i əvvəlcə xronoloji ardıcıllıqla saxlanılıbsa, `reverse()` istifadə edərək ardıcıllığı tərsinə çevirə bilərsiniz:

```
cars = ['bmw', 'audi', 'toyota', 'subaru']
print(cars)

cars.reverse()
print(cars)
```

Output:

```
['bmw', 'audi', 'toyota', 'subaru']
```

['subaru', 'toyota', 'audi', 'bmw']

**NOTE**

`reverse()` alfabetik tərs sıralama etmir; sadəcə list-in ardıcıllığını tam tərsinə çevirir.

`reverse()` metodu ardıcıllığı **davamlı olaraq** dəyişir, amma eyni metodu yenidən tətbiq etsəniz, list əvvəlki vəziyyətə qayıdar.

**Finding the Length of a List**

List-in uzunluğunu tez bir şəkildə tapmaq üçün `len()` funksiyasından istifadə olunur:

```
cars = ['bmw', 'audi', 'toyota', 'subaru']

len(cars)
```

Output:

4

## Index Error nədir?

List-də elementlər 0-dan başlayır:

motorcycles = ['honda', 'yamaha', 'suzuki']

Index-lər belədir:

- 'honda' → 0
- 'yamaha' → 1
- 'suzuki' → 2

Əgər sən `motorcycles[3]` yazsan — **4-cü elementi istəyirsən**, amma list-də 3 element var → **IndexError** çıxır.

**-1 index**

`list[-1]` — həmişə son elementi qaytarır.

Amma list boşdursa, yenə **IndexError** verəcək.