Programlama Dilleri Laboratuvar Föyü
Python Programlama Dili 2
String, Liste, Çokuzlu ve Sözlükler

Şevket Umut ÇAKIR

1 Giriş

Bu deneyde Python programlama dilinin çok kullanılan ve güçlü veri türleri olan string, liste, çokuzlular(tuple) ve sözlük incelenecektir. Başka programlama dillerinde de bulunan bu yapıları Python programlama dilinde kullanmak gayet kolaydır. Bu föyün hazırlanmasında emeği geçen Selahattin AKKAŞ'a teşekkür ederim.

2 Konu Anlatımı ve Deney Hazırlığı

Bu bölümde string, liste, çokuzlular(tuple) ve sözlük veri türlerinin yapısı ve kullanımı anlatılmaktadır.

2.1 String Veri Türü

String veri türü metinleri temsil etmek için kullanılmaktadır. Metinler tek veya çift tırnak ile tanımlanabilir. Tek tırnak içinde değiştirme damgasına(escape character) gerek kalmadan çift tırnak sembolü yada tersi kullanılabilir. Değiştirme damgasını geçersiz kılmak için metnin önüne *r* sembolü kullanılabilir. String türünün farklı kullanımları aşağıda verilmiştir.

```
print('merhaba') # merhaba
print("dünya") # dünya
print('Bana "merhaba" dedi!') # Bana "merhaba" dedi!
print("Bana \"merhaba\" dedi!") # Bana "merhaba" dedi!
print(r"c:\users\pau") # c:\users\pau
print('Çok\nsatırlı\nmetin') # 3 satır içerir
print("""Çok
satırlı
metin""")
```

Aritmetik operatörler string veri tipinde farklı işlevlere sahiptir. Bir metnin herhangi bir karakterine ya da bir kısmına köşeli parantezler ile erişilebilir. Kö-

şeli parantez içerisine tek bir indis değeri veya **:** sembolü ile ayrılmış bir aralık verilebilir[1].

```
print(3*'ab'+'cd') # abababcd yazdırır. * tekrarlar, +
↪  birleştirir
print('merhaba ' 'dünya') # yan yana stringler birleştirilir
s='Merhaba dünya!' # s metnini tanımlar
print(s[2]) # r yazar
print(s[3:7]) # haba yazar
print(s[-1]) # ! yazar
print(s[5:]) # ba dünya! yazar
print(s[:]) # Merhaba dünya! yazar
m='Python' # indislere göre konumlar aşağıdadır.
# +---+---+---+---+---+---+
# | P | y | t | h | o | n |
# +---+---+---+---+---+---+
#   0   1   2   3   4   5
#  -6  -5  -4  -3  -2  -1
```

String veri türünün farklı işlevler için kullanılan metotları bulunmaktadır. Bunlardan bazıları aşağıdaki gibidir:

- **lower(), upper():** Metnin sırasıyla küçük ve büyük harfli hallerini verir.
- **strip():** Metni başındaki ve sonundaki boşluklar olmadan verir.
- **startswith(str), endswith(str):** Metnin başka bir metinle başladığını ve bittiğini True ve False olarak verir.
- **find(str):** Metnin içinde başka bir metni arar; bulursa konumunu, bulamazsa -1 döndürür.
- **count(str):** Metnin içinde başka bir metnin kaç defa geçtiğini verir.
- **replace(eski, yeni):** Metnin içinde eşleşen değerleri başka bir metinle değiştirir.
- **split(ayraç):** Metni ayraca göre parçalara ayırır ve sonucu liste olarak döndürür.
- **join(liste):** Bir listenin elemanlarını aralarına metni koyarak birleştirir.

Bu metotlardan bazılarının kullanımı aşağıda verilmiştir:

```
s='Türkçe karakter uyuşmazlığı'
print(s.upper()) # TÜRKÇE KARAKTER UYUŞMAZLIĞI
print(s.upper().lower()) # türkçe karakter uyuşmazliği
print(s.startswith('Türkçe')) # True
print(s.replace('karakter', 'harf')) # Türkçe harf uyuşmazlığı
l=s.split(' ') # ['Türkçe', 'karakter', 'uyuşmazlığı']
```

```
print('- . -'.join(l)) # Türkçe- . -karakter- . -uyuşmazlığı
print(s.count('a')) # 3
print(len(s)) # 27 (uzunluğu verir)
```

2.2 Listeler

Listeler içinde birden fazla aynı veya farklı türden elemanı barındırabilen bir veri türüdür. Dizilerden farklı olarak listeler değişebilir bir yapıya sahiptir. Örneğin bir listeye oluşturulduktan sonra eleman eklenebilir ayda listeden elemanlar silinebilir. Listeleri tanımlarken köşeli parantezler ya da `list()` fonksiyonu kullanılır. Listelerde indis değeri sıfırdan başlar. Aşağıda örnek liste kullanımları verilmiştir.

```
l = ['kırmızı', 'yeşil', 'mavi', 'sarı']
print(l) # ['kırmızı', 'yeşil', 'mavi', 'sarı']
print(l[0]) # kırmızı
print(l[1:3]) # ['yeşil', 'mavi']
print(len(l)) # 4
l2 = l # yeni liste oluşturulmaz, her ikisi de aynı listeyi
↪  gösterir
bos_liste = [] # boş liste oluşturur
bos_liste = list() # boş liste oluşturur
l3 = ['siyah', 'mor']
print(l[2:] + l3) # ['mavi', 'sarı', 'siyah', 'mor']
```

Liste elemanları `for` döngüsü ile dolaşılabilir. Bir elemanın listede olup olmadığı da `in` işleci ile sorgulanabilir. Aşağıda `for` döngüsü ve `in` işlecinin listelerde kullanımı verilmiştir.

```
renkler = ['kırmızı', 'yeşil', 'mavi']
for renk in renkler:
    print(renk)
print('yeşil' in renkler) # True
print('sarı' in renkler) # False
```

Listelerin kullanışlı bir çok metodu bulunmaktadır. Bunlardan bazıları aşağıdaki gibidir:

- **append(eleman):** Listenin sonuna eleman ekler.
- **insert(indis, eleman):** Listenin belirtilen indisine araya eleman ekler.
- **extend(list):** Listenin sonuna başka bir listeyi ekler. Orjinal liste değişir.
- **index(eleman):** Elemanı listede arar, varsa konumunu döndürür. Yoksa ValueError hatası verir.
- **remove(eleman):** Verilen elemanı listeden siler. Döngü içinde yapılmamalıdır.

- **sort():** Listeyi sıralar. Orjinal liste değişir. Liste değiştirilmeden sıralanmak isteniyorsa `sorted(liste)` kullanılmalıdır.
- **reverse():** Elemanların sırasını tersine çevirir.
- **pop(indis):** Verilen indisteki elemanı siler ve geri döndürür. İndis verilmemişse son eleman için bu işlemi yapar.

Aşağıda liste metodlarının bazılarının kullanımı gösterilmiştir:

```
liste = ['Ali', 'Ahmet', 'Mehmet']
liste.append('Burak') # sona Burak eklenir
liste.insert(0, 'Hasan') # başa Hasan eklenir
liste.extend(['Ayse', 'Fatma']) # listenin sonuna Ayse ve Fatma
↪  eklenir.
print(liste) # ['Hasan', 'Ali', 'Ahmet', 'Mehmet', 'Burak',
↪  'Ayse', 'Fatma']
print(liste.index('Ahmet')) # 2
liste.remove('Burak')
liste.pop(1) # Ali silinir
print(sorted(liste)) # ['Ahmet', 'Ayse', 'Fatma', 'Hasan',
↪  'Mehmet']
print(liste) # ['Hasan', 'Ahmet', 'Mehmet', 'Ayse', 'Fatma']
liste.sort() # Liste sıralanmış hali ile değişir
```

2.3 Çokuzlular(Tuple)

İçerisinde birden fazla elemanı saklayabilen fakat oluşturulduktan sonra değiştirilemeyen veri tipleridir. Çokuzlu oluşturmak için parantez sembolleri kullanılır. Kullanımı aşağıdaki gibidir:

```
cu = (1, 3.14, 'merhaba')
for el in cu: # Tüm elemanları yazar
    print(el)
print(cu[1]) # 3.14
cu[2] = 0 # Hata: tuple değiştirilemez
```

2.4 Sözlükler(Dictionary)

Sözlükler anahtar-değer çiftlerinin tutulduğu bir veri türüdür. Köşeli parantezlerin içine anahtarlar yazılarak değerlere ulaşılabilir. Tanımlarken süslü parantezler ya da `dict()` fonksiyonu kullanılır. Aşağıda kullanımı verilmiştir:

```
sozluk ={'a': 'alpha', 'o':'omega', 'g':'gamma'}
print(sozluk) # {'a': 'alpha', 'o': 'omega', 'g': 'gamma'}
sozluk=dict()
sozluk['a']='alpha'
sozluk['o']='omega'
```

```
sozluk['g']='gamma'
print(sozluk) # {'a': 'alpha', 'o': 'omega', 'g': 'gamma'}
# for döngüsü varsayılan olarak anahtarlarda dolaşır
for anahtar in sozluk:
    print(anahtar + ':' + sozluk[anahtar])
# Aşağıdakiler listeye dönüştürülebilir.
print(sozluk.keys()) # dict_keys(['a', 'o', 'g'])
print(sozluk.values()) # dict_values(['alpha', 'omega', 'gamma'])
# anahtar değer çiftleri üzerinde dolaşma
for anahtar, deger in sozluk.items():
    print(anahtar,deger)

del sozluk['a'] # silmek için kullanılır
```

3 Deneyin Uygulanması

Bu kısımda deneyde gerçeklenmesi gereken kodlar anlatılmaktadır.

3.1 Sabit Başlangıç

Kullanıcıdan alınan metnin ilk harfi sabit kalmak koşulu ile içindeki ilk harfle aynı olan harfleri "*" sembolü ile değiştiren kodu yazınız. Örneğin *google* girdisi *goo*le* olarak çıktı verilmelidir.

3.2 Karıştır

Kullanıcıdan alınan iki adet metnin ilk iki harflerini yer değiştirerek birleştiren ve ekrana yazan kodu yazınız. Örneğin **mixer** ve **glider** girdileri için çıktı **glxer miider** olmalıdır.

3.3 Önce x Sırala

Kullanıcıdan alınan satırları(boş satır gelene kadar) önce x içerenleri sonra diğerlerini sıralayarak ekrana yazdıran programın yazılması istenmektedir. Girdi metninin bittiğini belirtmek için son satır boş metin olarak gönderilmektedir. Aşağıda örnek girdi ve çıktı bulunmaktadır.

Girdi:

```
mix
xyz
apple
xanadu
aardvark
```

Çıktı:

```
xanadu
xyz
aardvark
apple
mix
```

3.4 En Çok Tekrarlayan Kelime

Kullanıcıdan alınan metinde en çok tekrar eden 5 kelime ve tekrar etme sayısının ekrana yazılması istenmektedir. Girdi metni işlenmeden önce nokta(.), virgül(,), kesme işareti(') boş string ile; tire işareti(-) ise bir adet boşluk karakteri ile yer değiştirmelidir. Ayrıca metinler küçük harfe dönüştürülüp işlenmelidir. Bu problemin çözümü için sözlük yapısı kullanılabilir. Sözlüğün anahtar değeri sözcüğü, değeri ise tekrar sayısını saklayacaktır. Programa verilen tüm girdileri(dosya sonu karakterine kadar) okumak için `girdi = sys.stdin.read()` kullanılabilir. Sözlük yapısını değerlere göre tersten sıralayıp anahtar değerlerini almak için aşağıdaki döngü biçimi kullanılabilir.

```
for anahtar in sorted(sozluk, key=sozluk.get, reverse=True):
```

Kaynaklar

[1] *The Python Tutorial*. URL: `https://docs.python.org/3/tutorial/introduction.html` (son erişim: 26.2.2019).