ITU ACM Student Chapter Course Program

Introduction to Python

Week 5

Instructor
Serra Bozkurt

Assistants
Zafer Yıldız
Hüseyin Averbek

Week 5

FONKSİYONLAR

Fonksiyon Nedir?

Fonksiyonlar kısaca belirli bir işlevi yapmak için oluşturulan komutlar kümesidir. Örneğin print() fonksiyonu ekrana çıktı vermemizi sağlayan işlevi yapmak için belirli komutlar bulundurur.

Neden Fonksiyon?

Karmaşık programlar yazılmaya başlandığında ve kod uzunluğu arttığında aynı işlemleri tekrar tekrar yapmak gereken yerler olabilir ve bunları her seferinde tek tek yazmak kodun okunabilirliğini azaltır.

Bu gibi tekrar tekrar yapılması istenen işlemler için her seferinde kod yazmak yerine bir defa fonksiyon oluşturulur ve aynı işlem her yapılmak istendiğinde çağırıp kullanılır. Böylece kod karmaşasından kurtulunur.

Fonksiyonlar sayesinde:

- Aynı kodu defalarca yazmak gerekmez. Dolayısıyla bellek (RAM) gereksiz yere dolmaz.
- Programcıların aynı proje üzerinde beraber çalışmasını kolaylaştırır.
- İşleri küçük birimlere bölmek, programlama hatalarını bulmayı (debugging) kolaylaştırır.
- Programlama dilinin çekirdek tanımında bulunmayan üst seviye işlemleri tek komutla yapmayı sağlar.

Fonksiyonlara Giriş

Fonksiyonlar, matematikteki f(x) = y ifadelerindeki mantıkla düşünülebilir. Tek farkı girdi veya çıktı sayısının tek olmak zorunda olmayışıdır. Yani f(x, y, z, t) = a, b şeklinde fonksiyonlar da Python'da yazılabilir.

Python'da bir fonksiyon tanımlanırken def ifadesi kullanılır. İngilizcedeki "define" kelimesinden gelmektedir. Bir fonksiyonun tanım satırı syntax'i şu şekildedir:

```
def fonksiyonun_ismi(fonksiyonun girdi parametreleri virgülle ayrılmış)):

...
...    fonksiyonun işlemleri...
...

return çıktı_parametre
```

Örneğin;

```
def toplam(sayi_bir, sayi_iki):
                                                # fonksiyonun içinde bir değişken␣
 ↪oluşturduk
    toplamlari = sayi_bir + sayi_iki            # aldığımız girdileri kullandık

    return toplamlari                           # çıktıyı döndürdük
```

Yukarıdaki kod satırı tek başına hiçbir iş yapmamaktadır. Çünkü bir fonksiyonun çalışması için program içinde çağırılması gerekir.

```
def toplam(sayi_bir, sayi_iki):                 # bu
                                                # kısım
    toplamlari = sayi_bir + sayi_iki            # fonksiyon
                                                # bloğudur
    return toplamlari                           #

a, b = 5, 3         # burada iki integer tanımladık
c = toplam(a, b)    # burada fonksiyonu çağırıp girdi olarak a ve b yi kullandık,
                    # fonksiyon çıktı olarak tek sayı return'leyeceğinden
                    # tek bir sayıya fonksiyonun sonucunu atadık

print(c)
```

```
8
```

Fonksiyonun çıktısını hiçbir değişkende tutmadan doğrudan da bastırabilirdik:

```
def toplam(sayi_bir, sayi_iki):
    toplamlari = sayi_bir + sayi_iki
    return toplamlari

a, b = 5, 3
print(toplam(a, b))
```

```
8
```

Çoklu çıktılarda çıktı sayısıyla çıktıyı atama yaptığımız değişken sayısı eşit olmazsa fonksiyon tuple döndürür.

```
def toplamfark(sayi_bir, sayi_iki):
    toplamlari = sayi_bir + sayi_iki
    farklari = sayi_bir - sayi_iki
    return toplamlari, farklari

a, b = 5, 7

c = toplamfark(a, b)
```

```
print(c)
```

```
(12, -2)
```

Ancak çıktı sayısı artınca atanan değer sayısının birden fazla olması hataya yol açar.

```
def toplamfarkcarpim(sayi_bir, sayi_iki):

    toplamlari = sayi_bir + sayi_iki
    farklari = sayi_bir - sayi_iki
    carpimlari = sayi_bir * sayi_iki

    return toplamlari, farklari, carpimlari

a, b = 5, 7

c = toplamfarkcarpim(a, b)      # bu satır hata vermez
                                # çünkü tek tuple değeri tamamen c ye atanır
print(c)

c, d = toplamfarkcarpim(a, b)  # bu satır hata verir
```

```
(12, -2, 35)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-17-64fcd7942964> in <module>
     13 print(c)
     14
---> 15 c, d = toplamfarkcarpim(a, b)  # bu satır hata verir

ValueError: too many values to unpack (expected 2)
```

Hiçbir değer döndürmeyen fonksiyonlar da olabilir:

```
def listeyi_arttirma(liste):
    for i in range(len(liste)): # listenin her elemanını indexing methoduyla␣
 ↪dolaşır
        liste[i] += 1           # ve bir arttırır

liste = [1, 2, 3]

listeyi_arttirma(liste)     # fonksiyon hiçbir çıktı vermeyeceğinden atama␣
 ↪yapılmaz

print(liste)
```

```
[2, 3, 4]
```

Fonksiyonlar girdi almayabilirler:

```
def talimatlar():                                   # talimatları bastıran fonksiyon
    print()
    print("Bakiyenizi öğrenmek için 1'i, yükleme yapmak için 2'yi, para çekmek␣
 ↪için 3'ü, ")
    print("programdan çıkmak için 4'ü tuşlayınız.")
    print()

def basarili():
    print("İşlem başarıyla gerçekleştirildi!")

print("Hoşgeldiniz!")
talimatlar()

istek = int(input())
bakiye = 500

while (istek != 4):

    if istek == 1:                      # talimatlardan 1 numara bakiyeyi␣
 ↪bastırmak içindi
        print("Bakiyeniz:", bakiye)

    elif istek == 2:                    # talimatlardan 2 numara bakiyeye ekleme␣
 ↪yapmak içindi

        ekleme = int(input("Lütfen yüklenecek miktarı giriniz: "))

        bakiye += ekleme

        basarili()

    elif istek == 3:                    # talimatlardan 3 numara bakiyeyi␣
 ↪eksiltmek içindi

        cekme = int(input("Lütfen çekmek istediğiniz miktarı giriniz: "))

        while cekme > bakiye:           # bu döngüyle bakiyeden yüksek miktar␣
 ↪girilmemesini sağlıyoruz

            print("Lütfen bakiyenizi aşmayınız!")

            cekme = int(input("Lütfen çekmek istediğiniz miktarı giriniz: "))

        # eğer yukarıdaki while döngüsünden çıkabilirse çekilmek istenen miktar␣
 ↪bakiyeyi aşmıyordur
```

```
        bakiye -= cekme

        basarili()

    else:
        print("Lütfen talimatlar doğrultusunda bir sayı giriniz!")

    talimatlar()
    istek = int(input())

print("Teşekkürler, iyi günler.")
```

```
Hoşgeldiniz!

Bakiyenizi öğrenmek için 1'i, yükleme yapmak için 2'yi, para çekmek için 3'ü,
programdan çıkmak için 4'ü tuşlayınız.

1
Bakiyeniz: 500

Bakiyenizi öğrenmek için 1'i, yükleme yapmak için 2'yi, para çekmek için 3'ü,
programdan çıkmak için 4'ü tuşlayınız.

2
Lütfen yüklenecek miktarı giriniz: 300
İşlem başarıyla gerçekleştirildi!

Bakiyenizi öğrenmek için 1'i, yükleme yapmak için 2'yi, para çekmek için 3'ü,
programdan çıkmak için 4'ü tuşlayınız.

1
Bakiyeniz: 800

Bakiyenizi öğrenmek için 1'i, yükleme yapmak için 2'yi, para çekmek için 3'ü,
programdan çıkmak için 4'ü tuşlayınız.

3
Lütfen çekmek istediğiniz miktarı giriniz: 900
Lütfen bakiyenizi aşmayınız!
Lütfen çekmek istediğiniz miktarı giriniz: 700
İşlem başarıyla gerçekleştirildi!

Bakiyenizi öğrenmek için 1'i, yükleme yapmak için 2'yi, para çekmek için 3'ü,
programdan çıkmak için 4'ü tuşlayınız.

1
Bakiyeniz: 100
```

```
Bakiyenizi öğrenmek için 1'i, yükleme yapmak için 2'yi, para çekmek için 3'ü,
programdan çıkmak için 4'ü tuşlayınız.

4

Teşekkürler, iyigünler.
```

Recursion Kavramı

Bir fonksiyon kendi kendini çağırıyorsa o fonksiyona **recursive** denir. Ancak fonksiyonun şartları ve bitiş sınırları iyi belirlenmelidir yoksa fonksiyon sonsuz döngüye girebilir.

[6]:
```
def fibonacci(a):

    if a == 1 or a == 0:       # fibonacci dizisinin 0 ıncı ve 1 inci indexinde 1␣
 ↪elemanı vardır
        return 1

    return fibonacci(a - 1) + fibonacci(a - 2)

print(fibonacci(5))            # 1 1 2 3 5 8
```

```
8
```

Kütüphane Fonksiyonları

Kütüphane, belli bir işlev için hazırlanan fonksiyonlar topluluğudur. Bir kütüphane matematik fonksiyonlarını toplarken başka bir kütüphane kelime işleme, bir başkası ağ iletişimi, bir başkası oyun modülleri barındırıyor olabilir. Kütüphaneler bir dilin resmi tanımına dahil olabilir (bu durumda onlara standart kütüphane denir) veya üçüncü kişiler tarafından hazırlanmış olabilir.

Örneğin: Python'da math kütüphanesi. Kütüphaneyi kullanmak için import math ifadesi kullanılır. Daha sonrasında math ile birlikte gelen fonksiyonlar math.fonksiyon_adi(girdiler) şeklinde kullanılabilir.

[1]:
```
import math
print(math.sqrt(3))

print(math.sin(math.pi / 2))
```

```
1.7320508075688772
1.0
```