LABORATUVAR ÇALIŞMASI 10 – Özyineleme

Bu Çalışmanın Amacı

Bu çalışmadaki amacımız, özyineleme konusunda öğrendiklerimizi pekiştirmektir.

Özyineleme

Özyineleme, "bir ifadenin kendi kendini tekrar etmesi", "bir fonksiyonun kendi içinde kendini çağırması" olarak tanımlanabilir. Örnek verecek olursak:

```
def ozyineleme():
    ozyineleme()
```

Yukarıdaki kodu bir betik dosyasına yazarak "ozyineLeme" fonksiyonunu çağırdığımızda bu fonksiyon defalarca kendini çağıracak ve **maksimum özyineleme sayısı** aşıldığında bu iç içe çağırma işlemi sona erecektir. Fonksiyon çalıştırıldığında görülecek sonuç, aşağıdakine benzer olacaktır:

```
  File "C:/Python25/ozyineLeme.py", line 2, in ozyineLeme
    ozyineLeme()
  File "C:/Python25/ozyineLeme.py", line 2, in ozyineLeme
    ozyineLeme()
  File "C:/Python25/ozyineLeme.py", line 2, in ozyineLeme
    ozyineLeme()
  ...
  ...
RuntimeError: maximum recursion depth exceeded
```

Başka bir örneği inceleyelim:

```
def geri_say(tavan):
    if tavan <= 0:
        print 'Basla!'
    else:
        print tavan
        geri_say(tavan-1)
```

Bir betik dosyasında yer alan bu fonksiyonu, "**tavan**" değerine **3** vererek çalıştıracak olursak şu sonucu elde ederiz:

```
>>> geri_say(3)
3
2
1
Basla!
```

Burada ilk önce "**geri_say**" fonksiyonuna **3** değeri verilmektedir. Fonksiyon, **tavan** değeri **3** olduğu için **if** bloğuna girmeyip, **else** içerisine girecektir, **tavan** değerini (3) ekrana yazdıktan sonra, "tavan – 1 (yani **2**)" değeri ile kendini çağıracaktır. Bu satırdaki işlemi "**geri_say(2)**" olarak düşünebiliriz. Daha sonra fonksiyon (aslında fonksiyonun başka bir kopyası), **2** değeri girilerek sil baştan çağrıldığı için aynı işlemler yeniden yapılacak, **if** bloğu atlanarak **else** içerisinde ekrana **tavan** değeri (2) yazılacaktır ve fonksiyon," tavan – 1 (yani **1**)" değeri ile çağrılacaktır. Bu işlemin sonunda da "**geri_say(1)**" çağrısı yapılacak, ekrana tavan değeri olan **1** yazılacak ve en sonda "**geri_say(0)**" çağrısı yapılacaktır. Ancak, **0** değeri verilerek fonksiyon tekrar çağrıldığında "**0 <= 0**" şartı sağlanacağı için bu sefer **if** bloğuna girilirken **else** bloğuna girilmeyecektir. **if** bloğuna girildiğinde ekrana "**Basla!**" yazısı yazılarak program sonlanacaktır. Programın sonlanmasının nedeni, en son olarak girmiş olduğu **if** bloğunun içerisinde kendi kendini yeniden çağırmasını söyleyen herhangi bir komut bulunmamasıdır.

Alıştırmalar

Alıştırma – 1

Görev

Matematikte 1’ den büyük pozitif bir tamsayı ile **1** arasındaki (bahsedilen tamsayı da dahil) tüm tamsayıların çarpımına, ilgili tamsayının **faktöriyel**i denir. 0 ve 1 sayılarının faktöriyeli **1**’ e eşittir. Negatif tamsayıların ise faktöriyel hesaplaması yapılamaz. Örneğin:

3’ ün faktöriyeli = **3!** = 3 * 2 * 1 = **6**

7’ nin faktöriyeli = **7!** = 7 * 6 * 5 * 4 * 3 * 2 * 1 = **5040**

Özyineleme kullanarak “**faktoriyel**” isimli bir faktöriyel alma fonksiyonu yazınız ve bunu “**Lab10_faktoriyel.py**” isimli bir betik dosyasına kaydediniz. Fonksiyonunuz argüman olarak bir tamsayı alıp, değer olarak ise bu tamsayının faktöriyelini döndürmelidir. Negatif bir tamsayı verildiğinde ise ekrana uyarı olarak “**Negatif tamsayilarin faktoriyel hesaplamasi yapilamaz**.” yazmalıdır. Sizden beklenen gerçekleştirimin çalıştırılmasına ait bir örnek gösterim aşağıdaki gibidir:

```
>>> faktoriyel(6)
720
>>> faktoriyel(-4)
Negatif tamsayilarin faktoriyel hesaplamasi yapilamaz.
>>> faktoriyel(0)
1
>>> faktoriyel(1)
1
```

İpucu

“Özyineleme” bölümünü inceleyiniz. Değer döndüren bir fonksiyon içerisindeki bir **if** bloğunda değer döndürmeden fonksiyonu sonlandırmak gerekiyorsa, tek başına “**return**” komutu kullanılmalıdır (‘return’ komutunun sağ tarafına herhangi bir şey yazılmamalıdır.). Örneğin, kendisine verilen iki sayıdan birinciyi ikinciye bölerek sonucu döndüren, ikinci sayı 0 ise sonuç döndürmeden uyarı veren bir fonksiyon aşağıdaki gibidir:

```
def tamsayi_bolme(bolunen, bolen):
    sonuc = 0
    if bolen != 0:
        sonuc = bolunen / bolen
    else:
        print 'Bolen sayi sifir olamaz.'
        return
    return sonuc
```

Çalıştırıldığında görülecek sonuç:

```
>>> tamsayi_bolme(7,3)
2
>>> tamsayi_bolme(8,0)
Bolen sayi sifir olamaz.
```

Sonuç

Gerçekleştiriminizi ve / veya karşılaştığınız problemleri raporunuza yazınız.