LABORATUVAR ÇALIŞMASI 6 – Yineleme

Bu Çalışmanın Amacı

Bu çalışmadaki amacımız; yineleme ve bazı karakter işlemleri konularında öğrendiklerimizi pekiştirmek, ayrıca "Python' da Görüntü İşleme" konusuna giriş yapmaktır.

While Döngüsü

Programlama yaparken "Şu şart sağlandığı sürece şu işlemleri yap, şart sağlanmadığı andan itibarense yapmayı durdur ve bir daha yapma." şeklinde bir kod tasarımına gereksinim duyuyorsak "**while**" yapısını kullanabiliriz. **while** komutu da tıpkı **if** komutu gibi kendi sağında yer alan kriterin geçerliliğini kontrol eder ve kriter geçerli ise kendi kapsamındaki kodları (altında ve girinti olarak kendisinden daha sağda yer alan kod parçalarını) 1 kez çalıştırır. Bu çalıştırma esnasında, kriter içerisinde kıyaslama yaptığı değişkenlerin değerinde değişiklik olabilir. Çalıştırma sona erdikten sonra döngü tekrar başa döner ve kriterin geçerliliği tekrar kontrol edilir ve geçerli ise aynı kod parçaları tekrardan çalıştırılır. Bu işlemler, kriter geçersiz kalana kadar tekrarlanır, kriter geçersiz kaldıktan sonra bir daha tekrarlanmaz. Anlatımı somutlaştırmak için aşağıdaki geri sayım örneğini inceleyebiliriz:

```
def geri_say(n):
    while n > 0:
        print n
        n = n-1
    print "Son!"
```

Yukarıdaki fonksiyon, argüman olarak bir pozitif **n** tamsayısı almaktadır. **while** komutu, her adımda bu **n** tamsayısının 0' dan büyük olup – olmadığını kontrol etmekte, eğer büyük ise kendi kapsamına giren kod parçalarının (*'print n'* ve *'n = n-1'*) çalıştırılmasına müsaade etmektedir. Kendi kapsamındaki kod parçaları her çalıştırıldığında program akışı **while** komutunun bulunduğu satıra gelmekte, **n** sayısının 0' dan büyük olup-olmadığı tekrardan kontrol edilmekte ve büyük ise aynı işlemler tekrar yapılmaktadır. **n** sayısının, **while** döngüsü içerisinde, her adımda 1 azaltıldığına dikkat etmemiz gerekir. **n** sayısı 0' a eşit olduğunda ise **while** komutunun işlevi sona erecek, program akışı ise **while** komutunun kapsamının sona erdiği noktaya (*'print "Son!"'* ifadesinin bulunduğu satıra) gidecektir. **n=5** için bu fonksiyonun çalıştırılmasına ait Python Shell görüntüsünü inceleyelim:

```
>>> geri_say(5)
5
4
3
2
1
Son!
```

Eğer fonksiyona **n=-2** değeri girilirse, **while** komutunun kapsamına giren kod parçaları hiç çalışmayacaktır:

```
>>> geri_say(-2)
Son!
```

Eğer fonksiyonumuzdan **n** sayısının değerini 1 azaltan kod parçasını kaldırırsak,

```
def geri_say(n):
    while n > 0:
        print n
##          n = n-1
    print "Son!"
```

n=5 için bu fonksiyonun sonsuza kadar çalıştığını göreceğiz:

```
>>> geri_say(5)
5
5
5
5
5
5
5
...
```

Bunun (**sonsuz döngü**nün) nedeni, **n** sayısının azalmamasının sonucu olarak daima 0' dan büyük kalması (daima 5 olması) ve dolayısı ile **while** komutunda yer alan şartın daima sağlanıyor olmasıdır.

Aşağıdaki dikdörtgen çizdirme fonksiyonu örneğini incelememiz de faydalı olacaktır:

```
def DikdortgenCiz(en, boy):
    isaret = ''
    while en > 0:
        isaret = isaret + '*'
        en = en - 1
    while boy > 0:
        print isaret
        boy = boy - 1
```

Betik dosyası kullanılarak çalıştırılmak üzere hazırlanmış ve dikdörtgen çizmeye yarayan bu fonksiyonda kullanıcıdan dikdörtgenin boyu ve eni alınmakta ve bu değerlere uygun bir dikdörtgen çizdirilmektedir.

Üstteki **while** döngüsünde, dikdörtgenin eni kadar uzunlukta ve "*" işaretlerinden oluşan yatay bir çizgi oluşturulmaktadır. İlk başta, "işaret" adlı karakter dizisinin içi boştur. Örneğin "en" değeri 6 ise, **while**' ın olduğu satırda 6' nın 0' dan büyük olduğu ifadesi doğrulanacaktır. 6 sayısı 0' dan büyük olduğu için **while**' ın kapsamına giren satırlardaki (hemen alttaki 2 satır) kodlar çalıştırılacak, "isaret" değişkeninin içeriği "*" olacak, "en" değeri ise 5 olacaktır. Sonra programın akışı tekrar **while** kısmına gelecek, "en" değerinin 0' dan büyük olduğu ifadesi doğrulanmaya çalışılacaktır. 5 sayısı da 0' dan büyük olduğu için "isaret" değişkeninin değeri "**" olacak, "en" değeri ise 4 olacaktır. Bu işlem, "en" değeri 0' a inene kadar devam edecek, daha sonra **0 > 0** şartı sağlanamayacağı için sona erecektir. Sonuçta "isaret" değişkeninin içeriği "******" olacaktır.

Daha sonra program akışı alttaki **while** döngüsüne geçecektir. Örneğin "boy" değeri 4 ise, üsttekine benzer biçimde, bu döngü de 4 kere çalışacak, ekrana 4 kere "******" yazdırılacaktır ("isaret" değişkeninin içeriği "******" olduğu için.). **print** komutu her kullanılışında yeni bir satıra geçtiği için bu işaretler alt alta yazdırılacak, sonuç olaraksa 6x4 boyutlarında bir dikdörtgen elde edilecektir.

Örnek kullanımı inceleyelim:

```
>>> DikdortgenCiz(6, 4)
******
******
******
******
>>> DikdortgenCiz(3, 5)
***
***
***
***
***
```

Python' da Görüntü İşlemeye Giriş

Python' da görüntü işleme (resimler üzerinde, çeşitli hesaplama teknikleri kullanarak birtakım işlemler yapma, resimlere yeni özellikler kazandırma gibi) işlemleri yapmak da mümkündür. Bunun için, "Python Image Library (Python Görüntü Kütüphansei)" ismi verilen bir kütüphane kullanılmaktadır. Öncelikle aşağıdaki adresten, kullanmakta olduğumuz Python sürümüne ve işletim sistemine uygun olan 'PIL' sürümünü indirerek kurulumunu yapmalıyız:

http://www.pythonware.com/products/pil/

Kurulum tamamlandıktan sonra, artık bu kütüphane içerisinde yer alan fonksiyonlardan faydalanmamız mümkün olacaktır (Bu kütüphanenin kullanımı hakkında pek çok yararlı bilginin yer aldığı İngilizce dokümanı www.pythonware.com/media/data/pil-handbook.pdf adresinde bulabilirsiniz.).

Bir betik dosyasına yazılmış olan aşağıdaki örneği inceleyelim:

```
import Image

#Resmi okuyup "res" degiskeni icerisine atma
res = Image.open("OrnekResim.jpg")

#Resme ait bilgileri yazdirma
print "Tur   :", res.format, "\nBoyut :", res.size, "\nRenk  :", res.mode

#Resmin aynadaki goruntusu
ters = res.transpose(Image.FLIP_LEFT_RIGHT)

#Resmin aynadaki goruntusunu goster
ters.show()
```

Yukarıdaki örnekte, PIL kurulumu ile birlikte gelen "Image" modülü çağrılmıştır. Daha sonra, **open** fonksiyonu kullanılarak betik dosyası ile aynı dizinde bulunan "*OrnekResim.jpg*" dosyası okunmuş, bu resim dosyasının bilgileri, **res** değişkeni içerisine atılmıştır. Bu **res** değişkeni aslında "format", "size", "mode" gibi farklı alt alanlara sahip olan bir nesnedir. **print** komutu yardımı ile de, bu alt alanlarda tutulan tür, boyut ve renk bilgileri ekrana yazdırılmaktadır. Ardından, **transpose** fonksiyonuna uygun değerler girilerek bir bakıma aynadaki görüntüsü elde edilmiş ve elde edilen bu yeni görüntünün (resmin) bilgileri de **ters** isimli değişkene atılmış, son olarak da bu değişkende yer alan yeni resim görüntülenmiştir.

Bu örneği gerçekleştirmek için, betik dosyanızın olduğu dizine “OrnekResim.jpg” adında bir resim koyabilir, ya da farklı isimde bir resim dosyası koyarak koddaki “OrnekResim.jpg” kısmını değiştirebilirsiniz. Aşağıdaki örnek resim için,

OrnekResim.jpg

anlatılan program şu resmi görüntüleyecektir:

Python Shell ekranında da aşağıdaki çıktı görülecektir:

```
>>>
Tur   : JPEG
Boyut : (600, 450)
Renk  : RGB
```

PIL kütüphanesinden yararlanarak siz de resimler üzerinde bu tür işlemler yapabilirsiniz.

Ek Bilgi : En Uzun Kelimeyi Bulma

Aşağıdaki kutucukta, bir cümle içerisindeki en uzun kelimeyi (en uzun kelime ile aynı uzunlukta birden fazla kelime varsa bunlardan en sonuncusunu) ve bu kelimenin uzunluğunu ekrana yazan, "**en_uzun_keLime**" isminde bir fonksiyon yer almaktadır:

```
def en_uzun_keLime(cumle):
    depo = dict()
    kelime = str.split(cumle)
    for i in range(0, len(kelime)):
        depo[kelime[i]] = len(kelime[i])
    en_uzun = depo[kelime[0]]

    j = 0
    for i in range(0, len(kelime)):
        if depo[kelime[i]] >= en_uzun:
            j = i
            en_uzun = depo[kelime[i]]
    print "En uzun : ", kelime[j], "\nUzunluk : ", depo[kelime[j]]
```

Bu fonksiyon, argüman olarak boşluklarla ayrılmış kelimelerden oluşan bir karakter dizisi almaktadır. **split** fonksiyonu ise bir **liste** oluşturup, birbirinden boşluklarla izole edilmiş olan her bir kelimeyi, cümledeki sırasına göre bu dizinin içerisine koymaktadır. Yukarıdaki **kelime** değişkeni, aslında bir dizidir. Ayrıca, **depo** adında, **sözlük** veri türünde bir değişken tanımlanmıştır. Sözlükler de aslında birer dizidir, ancak dizilerde indisler tamsayılardan oluşurken sözlüklerde indisler tamsayılardan, ondalıklı sayılardan ya da karakter dizilerinden oluşabilir. **len** fonksiyonu ise bir listede/sözlükte kaç eleman bulunduğunu ya da bir karakter dizisinin kaç karakterden oluştuğunu döndürür. **range** fonksiyonu ise iki argüman (sayı değeri olarak) alır ve bu iki argümanın değerinin arasında yer alan **tamsayıları** sıralı olarak, bir liste içerisinde döndürür. **for** döngüsü ile kullanımında da her bir adımda, **range** fonksiyonunun döndürmüş olduğu dizinin her bir elemanı, "for" ifadesinin sağında yer alan değişkenin içerisine atılarak işlem yapılacaktır. **Bu bölümde anlatılan konular ileride anlatılacağı için, sadece fonksiyonu çalıştırarak aşağıdaki işlemleri yapmanız yeterli olacaktır:**

```
>>> en_uzun_keLime('bir makinenin calismadigini ispat etmen
gerektiginde kesin calisir')
En uzun :  calismadigini
Uzunluk :  13
>>> en_uzun_keLime('yanlis numara cevirdiginde cevrilen numara
kesinlikle mesgul degildir')
En uzun :  cevirdiginde
Uzunluk :  12
>>> en_uzun_keLime('yere dusen her sey ulasilmasi en zor koseye
yuvarlanir')
En uzun :  yuvarlanir
Uzunluk :  10
```

Alıştırmalar

Alıştırma – 1

Görev

Matematikte 1' den büyük pozitif bir tamsayı ile **1** arasındaki (bahsedilen tamsayı da dahil) tüm tamsayıların çarpımına, ilgili tamsayının **faktöriyeli** 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**

"**faktoriyeL**" isimli bir faktöriyel alma fonksiyonu yazınız ve bunu "**Lab06_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 tamsayıların faktöriyel hesaplaması yapılamaz**." yazmalıdır. Sizden beklenen gerçekleştirime ait şablon aşağıdaki gibidir:

```
import doctest
def faktoriyeL(tamsayi):
    """
    >>> faktoriyeL(9)
    362880
    >>> faktoriyeL(4)
    24
    >>> faktoriyeL(1)
    1
    >>> faktoriyeL(0)
    1
    >>> faktoriyeL(-8)
    Negatif tamsayilarin faktoriyel hesaplamasi yapilamaz.
    """
    #Gerceklestiriminiz asagiya yazilacaktir.

doctest.testmod()
```

İpucu

"While Döngüsü" bölümünü inceleyiniz.

Sonuç

Gerçekleştiriminizi ve / veya karşılaştığınız problemleri raporunuza yazınız.

Alıştırma – 2

Görev

while döngüsü kullanarak; argüman olarak bir tamsayı alan ve değer olarak da girilen bu tamsayının basamak sayısını döndüren bir fonksiyon hazırlayınız. Fonksiyonunuza "**kac_basamak**" ismini veriniz ve "**Lab06_kac_basamak.py**" isimli betik dosyasına kaydediniz. Fonksiyonunuza ait şablon aşağıda verilmiştir:

```
import doctest

def kac_basamak(sayi):
    """
    >>> kac_basamak(67853)
    5
    >>> kac_basamak(600)
    3
    >>> kac_basamak(8)
    1
    >>> kac_basamak(0)
    1
    >>> kac_basamak(-8)
    1
    >>> kac_basamak(-9807)
    4
    """
    #Gerceklestiriminiz asagiya yazilacaktir.

doctest.testmod()
```

Sonuç

Gerçekleştiriminizi ve / veya karşılaştığınız problemleri raporunuza yazınız.

Alıştırma – 3

Görev

"Python' da Görüntü İşlemeye Giriş" bölümünde verilen betik kodunu modifiye ederek, **resmi başaşağı çevirmesini** sağlayınız. Yani, ilgili bölümde verilen örnek resim için elde edilecek sonuç görüntüsü şu şekilde olmalıdır:

İpucu

"Python' da Görüntü İşlemeye Giriş" bölümünde linki verilmiş olan dokümanın "Geometrical Transforms" bölümünü inceleyiniz.

Sonuç

Gerçekleştiriminizi ve / veya karşılaştığınız problemleri raporunuza yazınız.

Alıştırma – 4

Görev

"Ek Bilgi : En Uzun Kelimeyi Bulma" bölümünde verilen betik dosyasında yer alan fonksiyonu, **en kısa kelimeyi verecek şekilde** modifiye ediniz (En kısa kelime ile aynı uzunlukta birden çok kelime varsa bunlardan en sonuncusunu vermelidir.). Fonksiyonunuza "**en_kisa_keLime**" ismini vererek, "**Lab06_en_kisa_keLime.py**" isimli betik dosyasına kaydediniz. Fonksiyonun kullanımına dair örnek aşağıdadır:

```
>>> en_kisa_keLime('bir makinenin calismadigini ispat etmen
gerektiginde kesin calisir')
En kisa :  bir
Uzunluk :  3
>>> en_kisa_keLime('yanlis numara cevirdiginde cevrilen numara
kesinlikle mesgul degildir')
En kisa :  mesgul
Uzunluk :  6
>>> en_kisa_keLime('yere dusen her sey ulasilmasi en zor koseye
yuvarlanir')
En kisa :  en
Uzunluk :  2
```

Sonuç

Gerçekleştiriminizi ve / veya karşılaştığınız problemleri raporunuza yazınız.