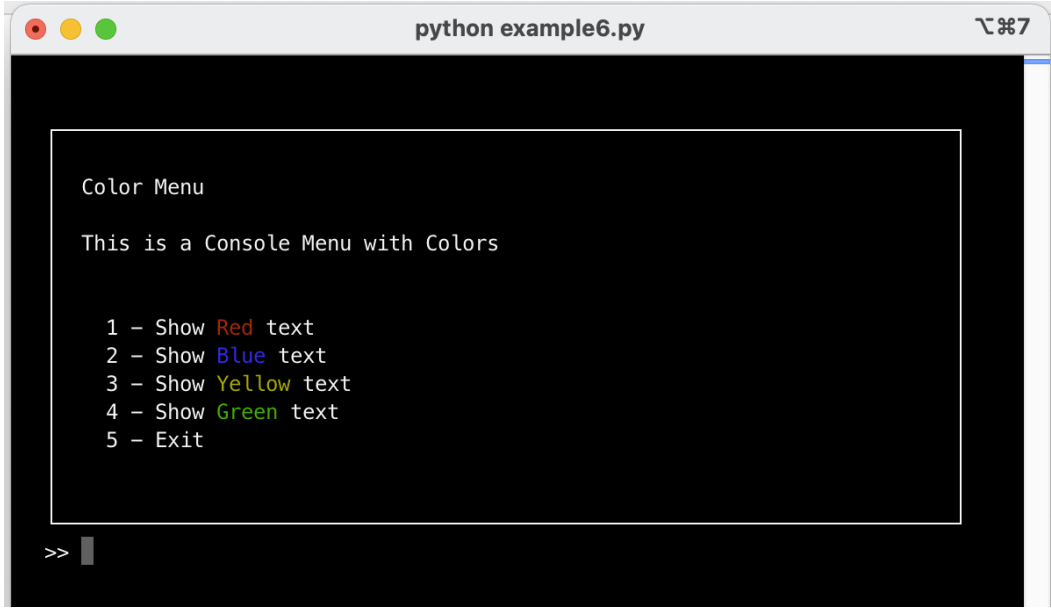


KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ
2023-2024 GÜZ DÖNEMİ
YZM217/BLM217 PROGRAMLAMA LAB 1 DERSİ
ÖDEV 1

Bilgilendirme:

- Ödev teslimi Google Classroom üzerinden yapılacaktır.
- Classroom'a bütün projenizi değil sadece py/ipynb dosyalarınızı yükleyiniz.
- Ödevin son yükleme tarihi **25.10.2023 23:55'tir.**
- Başarılar.

CONSOLE MENU



```
python example6.py

Color Menu

This is a Console Menu with Colors

1 - Show Red text
2 - Show Blue text
3 - Show Yellow text
4 - Show Green text
5 - Exit

>>
```

Bu ödevinizde sizden Python kullanılarak bir “console menu” oluşturmanız istenmektedir. (Yukarıda örnek bir console menu görebilirsiniz.) Menüde birtakım seçenekler olması ve herhangi bir seçenek seçildiğinde ilgili fonksiyonu çalıştırması beklenmektedir. Seçenek seçildikten sonra sırasıyla; fonksiyon giriş verilerini istemelidir, fonksiyon ara işlemlerini tamamlamalıdır ve fonksiyon yaptığı işlemlerin sonuçlarını menüde geri döndürmelidir. Seçeneklerde çalıştırılması istenen fonksiyonlar aşağıda detaylı bir şekilde anlatılmıştır. Menüde olması beklenen seçenek sayısı aşağıda yer alan fonksiyon sayısının bir fazlası kadardır. Fazla olan seçenek çıkış seçeneği olmalıdır ve çıkış seçeneği seçilene kadar menünün çalışması beklenmektedir.

1. k’nıncı En Küçük Elemanı Bulma

Bir liste içindeki k’nıncı en küçük elemanı bulan ve sonuç olarak menüde geri döndüren fonksiyonu yazınız. Liste boyutu dinamiktir, bu nedenle fonksiyonun her liste için çalışabilmesi gerekmektedir. Fonksiyonun istenen prototipi aşağıda gösterilmiştir. Fonksiyonun birinci girişi bir tam sayı olan k değeridir, ikinci girişi ise listedir.

Örnek: k_kucuk(3, [7, 10, 4, 3, 20, 15])

```
def k_kucuk(int, list):  
    ...  
    ...  
    ...  
    return k_kucuk_sonuc
```

2. En Yakın Çifti Bulma

Liste içindeki herhangi iki sayının toplamı, belirtilen sayıya en yakın olan sayı çiftini sonuç olarak menüde geri döndüren bir fonksiyon yazılması istenmektedir. Fonksiyonun istenen prototipi aşağıda gösterilmiştir. Fonksiyonun birinci girişi bir tam sayı (sorunun birinci cümlesinde bahsedilen sayı), ikinci girişi ise listedir. (Listeyi fonksiyon içinde sıraladıktan sonra sayı çiftini bulunuz.)

Örnek: en_yakin_cift(54, [10, 22, 28, 29, 30, 40])
22 ve 30

```
def en_yakin_cift(int, list):  
    ...  
    ...  
    ...  
    return en_yakin_cift_sonuc
```

3. Bir Listenin Tekrar Eden Elemanlarını Bulma

Kendisine giriş olarak verilen bir listenin tekrar eden elemanlarını “list comprehension” yöntemi kullanarak bulan ve bu değerleri sonuç olarak menüde geri döndüren bir fonksiyon yazmanız istenmektedir. Fonksiyonun istenen prototipi aşağıda gösterilmiştir.

Örnek: tekrar_eden_elemanlar([1, 2, 3, 2, 1, 5, 6, 5, 5, 5])
[1, 2, 5]

```
def tekrar_eden_elemanlar(list):  
    ...  
    ...  
    ...  
    return tekrar_eden_elemanlar_sonuc
```

4. Matris Çarpımı

Kendisine giriş olarak verilen “list of list” şeklindeki iki matrisin çarpımını, “list comprehension” yöntemiyle hesaplayan ve sonucu menüde geri döndüren bir fonksiyon yazmanız istenmektedir. (zip() metodundan yararlanınız ve zip() metodunun örneği aşağıda verilmiştir.)

Örnek (zip()):

```
no = [23,56,87,47,12,36,45,47]
```

```
isim=["Ahmet","Mehmet","Ayşe","Zeynep","Elif","Kemal","Fatma","Can"]
```

```
zip1 =zip(no,isim)
```

```
print(list(zip1))
```

```
[(23, 'Ahmet'), (56, 'Mehmet'), (87, 'Ayşe'), (47, 'Zeynep'), (12, 'Elif'), (36, 'Kemal'), (45, 'Fatma'), (47, 'Can')]
```

Örnek (matris çarpımı):

```
A = [[1, 2, 3], [4, 5, 6]]
```

```
B = [[7, 8], [9, 10], [11, 12]]
```

```
sonuc= [[58, 64], [139, 154]]
```

```
def matris_carpimi(list1, list2):  
    ...  
    ...  
    ...  
    return matris_carpimi_sonuc
```

5. Bir Text Dosyasındaki Kelimelerin Frekansını Bulma

Bir text dosyası içindeki her bir kelimenin text içinde kaç adet geçtiğini map(), reduce() gibi metotları kullanarak bulan bir fonksiyon oluşturmanız istenmektedir. Fonksiyona giriş olarak string veri tipinde ilgili text dosyasının dosya konumu verilecektir ve fonksiyonun her bir kelimenin frekansını menüde sonuç olarak geri döndürmesi beklenmektedir.

Örnek:

-----giris_metni.txt-----

elma armut elma kiraz elma armut kiraz armut kiraz armut kiraz armut kiraz armut kiraz armut kiraz kiraz

elma=3
armut=7
kiraz=8

```
def kelime_frekans(str):  
    ...  
    ...  
    ...  
    return kelime_frekans_sonuc
```

NOT: Bu aşamadan sonraki soruların fonksiyonları özyinelemeli fonksiyon olacak şekilde oluşturulmalıdır.

6. Liste İçinde En Küçük Değeri Bulma

Kendisine giriş olarak verilen bir liste içindeki en küçük değeri bulan ve sonucu menüde geri döndüren bir fonksiyon yazınız.

Örnek: en_kucuk_deger([1,4,6,91,2,5])

1

en_kucuk_deger([1,2,3,-1,4,-2,5])

-2

```
def en_kucuk_deger(list):  
    ...  
    ...  
    ...  
    return en_kucuk_deger(list)
```

7. Karekök Fonksiyonu

Verilen bir N sayısının karekökünü bulmak için Babil döneminden beri kullanılan tekrarlamalı (iteratif) bir yöntem vardır: Önce, karekök için bir tahminde bulununuz ve buna x_0 adını veriniz. Bir sonraki tahminimiz

$$x_1 = \frac{1}{2} (x_0 + N/x_0)$$

olacaktır. Genel olarak,

$$x_{n+1} = \frac{1}{2} (x_n + N/x_n)$$

kuralıyla ardışık iterasyonlar yapılırsa, x_n değerleri hızlıca N 'nin kareköküne yakınsayacaktır. Bu yöntem, fonksiyon köklerini bulmak için kullanılan en iyi algoritmalarından biri olan Newton yönteminin özel bir durumudur.

Bu algoritmayı kullanarak, bir sayının karekökünü veren ve sonucu menüde geri döndüren bir fonksiyon oluşturmanız istenmektedir.

Parametreler:

- * Karekökü alınacak sayı `N`.
- * İlk tahmin `x0`.
- * Tolerans `tol`. Varsayılan değer 10^{-10} .
- * Azami iterasyon sayısı `maxiter`. Varsayılan değer 10.

Sonuç:

- * `N`'nin karekökü için en iyi tahmin

Fonksiyon içinde, yukarıdaki formülü tekrar tekrar işleterek daha iyi tahminlere ulaşan bir döngü yazınız. Her adımda, hata değeri $|x_n^2 - N|$ ile tolerans `tol` karşılaştırılmalıdır. Hata, i toleransın altına düşünce döngü sonlandırılmalıdır. Ayrıca, döngü iterasyonları sayısı `maxiter`'den fazla olursa yine döngü sonlanmalıdır. Bu durumda bir uyarı mesajı ekrana yazılmalıdır ve fonksiyon döngüde elde edilen son sonucu vermelidir.

Örnek: `karekok(N=10, x0=1)`

3.162277660168379

`karekok(N=10000, x0=0.1)`

10 iterasyonda sonuca ulaşamadı. 'hata' veya 'maxiter' değerlerini değiştirin

103.38412392442035

`karekok(N=10000, x0=0.1, maxiter=15)`

100.0

Not: Tolerans değerinden ne kadar uzaklaştığınızla alakalı ayrı bir fonksiyon yazabilirsiniz.

```
def karekok(int, float):  
    ...  
    ...  
    ...  
    return karekok(int, float)
```

8. En Büyük Ortak Bölen

Kendisine giriş olarak verilen iki tam sayının en büyük ortak bölenini bulan ve sonucu menüde geri döndüren bir fonksiyon oluşturmanız istenmektedir.

Örnek: eb_ortak_bolen(18, 64)

2

eb_ortak_bolen(32, 64)

32

```
def eb_ortak_bolen(int1, int2):  
    ...  
    ...  
    ...  
    return eb_ortak_bolen(int1, int2)
```

9. Asallık Testi

Bir sayı 1'den büyük olmak koşulu ile kendisi hariç böleni yoksa asal sayıdır. Buradan hareketle, kendisine giriş olarak verilen bir tam sayının asal sayı olup olmadığını kontrol eden ve sonucu menüde geri döndüren bir fonksiyon oluşturmanız istenmektedir.

Örnek: asal_veya_degil(35)

False

asal_veya_degil(97)

True

```
def asal_veya_degil(int):  
    ...  
    ...  
    ...  
    return asal_veya_degil(int)
```

10. Daha Hızlı Fibonacci Hesabı

Özyineleme kullanırken birtakım olumsuzluklarla karşılaşmaktadır. Bunlardan bir tanesi de gereksiz hesaplama yapılmasıdır. Örneğin, fib(5)'e yapılan çağrıda fib(3)'ün çağırılması/hesaplaması 2 kere tekrarlanacaktır.

Daha az hızlı olan başka bir algoritmayı düşününüz. Dizi kağıt üzerinde hesapladığında ne yapılmaktadır?

0 1 1 2 3 5 8 13 21 34 ...

Önceki iki sayıya bakılmaktadır ve bir sonraki sayıya ulaşmak için toplamaları hesaplanmaktadır. Önceki ikisi yeniden hesaplanmamaktadır, sadece eklenmektedir (veya hatırlanmaktadır). Bu algoritmayı şöyle düşünmeniz önerilmektedir: Parametreleri,

- n (kaçıncı Fibonacci sayısı hesaplanacak olan)
- bir sayı k
- fib(k)
- fib(k-1) değerleri.

k'yi, diziyi yazarken bugüne kadar gelinen yer olarak düşününüz. k, n'ye ulaştığında işlemler bitmiş demektir. hizlandirici(n,k,fibk,fibk1). hizlandirici fonksiyonun özyinelemeli olduğu gözden kaçmamalıdır.

n,k,fibk,fibk1

8 1 1 0

8 2 1 1

8 3 2 1

8 4 3 2

8 5 5 3

8 6 8 5

8 7 13 8

8 8 21 13

```
def hizlandirici(int1,int2,int3,int4):  
    ...  
    ...  
    ...  
    return hizlandirici(int1,int2,int3,int4)
```

NOT: Herhangi bir kütüphane kullanarak arayüz ekleyenlere fazladan puan verilecektir.