Yönergeler

```
iki tane zarımız (1-6) var
bu zarlar atıldıktan sonra toplamı;
```

- eğer 7,11 ise kazandınız
- eğer 2,3,12 ise kaybettiniz olacak.
- bunların dışındaysa toplam sizin puanınız olacak ve zar atmaya devam edeceksiniz yeni atılan zarların toplamı puanınıza eşit olursa kazanıyorsunuz ancak toplamı 7 değerini alırsa kaybediyorsunuz. Döngü kazanıncaya yada kaybedinceye kadar devam edecek. fonksiyon kullanmadan tasarlayınız.

  2+3=5 6+6=12 4+3=7 kaybettiniz

  4+3=7 kazandınız

  6+5=11 kazandınız

  6+6=12 kaybettiniz

birinci çözüm

```
1  import random
2
3  # iki zarın atılması 1
4
5  zar1 = random.randint(1,6)# [1-6] arası tam sayı dödürür
6
7  zar2 = random.randint(1,6) # [1-6] arası tam sayı dödürür
8  #zarlar toplandı 1
9  toplam = zar1 + zar2
10
11 print("Zarlar atıldı! Toplam: ", toplam)
12
13 # ilk tur için kazanma veya kaybetme durumlarının kontrol edilmesi
14
15 if toplam == 7 or toplam == 11:
16     #kazanma mesajı 1
17     print("Kazandın!")
18
19 elif toplam == 2 or toplam == 3 or toplam == 12:
20     #kayetme mesajı 1
21     print("Kaybettin!")
22
23 else:
24     puan = toplam
25     print("Puanın: ", puan)
26
27 # puanın yeniden elde edilmesi
```

```
28
29  while True:
30      # iki zarın atılması 2
31      zar1 = random.randint(1,6)# [1-6] arası tam sayı döndürür
32      zar2 = random.randint(1,6)# [1-6] arası tam sayı döndürür
33
34      #zarlar toplandı 2
35      toplam = zar1 + zar2
36      print("Zarlar atıldı! Toplam: ", toplam)
37
38      if toplam == 7:
39      #kaybetme mesajı 2
40          print("Kaybettin!")
41          break
42
43      elif toplam == puan:
44      #kazanma mesajı 2
45          print("Kazandın!")
46          break
```

```
Zarlar atıldı! Toplam:  4
Puanın:  4
Zarlar atıldı! Toplam:  6
Zarlar atıldı! Toplam:  11
Zarlar atıldı! Toplam:  10
Zarlar atıldı! Toplam:  5
Zarlar atıldı! Toplam:  7
Kaybettin!
```

- kazanma, kaybetme , (zar atma) , (toplam gösterilmesi) tekrarlandığı için fonksiyon tanımlanması uygun görüldü

```
1 def kazandinMesaji():
2     print("Kazandın!")
3
```

```
1 def kaybettinMesaji():
2     #print("Kaybettin!")
3     #print("tebrikler kumarhane kazandı")
4     print("geçmiş olsun")
```

```
1 def zarAt():
2     zar1 = random.randint(1,6)# [1-6] arası tam sayı döndürür
3     zar2 = random.randint(1,6) # [1-6] arası tam sayı döndürür
4     return zar1,zar2
```

```
1 def toplamGoster(z1,z2):
2     toplam = z1 + z2
3     #buradan istediğiniz mesajı yazdırabilirsiniz (tek biryerden değiştirerek işinizi kolayla
4     #print("Zarlar atıldı! Toplam: ", toplam)
5     print(f"{z1}+{z2} = {z1+z2}")
6     return toplam
```

kodunuza dokunmadan "sadece" fonksiyonlarda (tek bir yerden) oynama yaparak istediğiniz sonuca ulaşabilirsiniz

yukarıdaki fonksiyonları yerine yazıldıktan sonra çözüm bu şekilde olur

```
import random

# iki zarın atılması 1
z1,z2 = zarAt()
#zarlar toplandı 1
toplam = toplamGoster(z1,z2)

# ilk tur için kazanma veya kaybetme durumlarının kontrol edilmesi

if toplam == 7 or toplam == 11:
    #kazanma mesajı 1
    kazandinMesaji()

elif toplam == 2 or toplam == 3 or toplam == 12:
    #kayetme mesajı 1
    kaybettinMesaji()

else:
    puan = toplam
    print("Puanın: ", puan)

# puanın yeniden elde edilmesi

while True:
    # iki zarın atılması 2
    z1,z2 = zarAt()

    #zarlar toplandı 2
    toplam = toplamGoster(z1,z2)

    if toplam == 7:
        #kaybetme mesajı 2
        kaybettinMesaji()
        break

    elif toplam == puan:
        #kazanma mesajı 2
        kazandinMesaji()
        break
```

```
3+3 = 6
Puanın:  6
```

```
1+5 = 6
Kazandın!
```

hocanın çözümü

```
def zarAt():
    z1=random.randrange(1,7)
    z2=random.randrange(1,7)
    return z1,z2
def goster(a):
    z1,z2=a
    print(f"{z1}+{z2} = {z1+z2}")
```

```
x=zarAt()
goster(x)
toplam=sum(x)
if toplam == 7 or toplam == 11:
    print("kazandiniz")
elif toplam in (2,3,12):
    print("kaybettiniz")
else:
    toplam=sum(x)
while True:
    x=zarAt()
    goster(x)
    toplam2=sum(x)
    if toplam2 == toplam:
        print("kazandanız")
        break
    elif toplam2 == 7:
        print("kaybettiniz")
        break

```

```
3+1 = 4
3+4 = 7
kaybettiniz
```

```
import random # random.randrange()

```

```
from random import randrange # randrange() -> daha kolay kullanım

```

```
e=523

```

```
from math import * # math kutuphanesindeki herşeyi import eder
e

```

```
2.718281828459045
```

```
from math import sqrt

```

scope

```
numara = 20

```

```
print(f" ilk id =  { id(numara)}")
```

```
 ilk id =  2067903245200
```

```

def carpma(rakam):

    numara = 10
    print(f" numara id =  { id(numara)}")

    return numara * rakam
```

```
carpma(5)
```

```
 numara id =  2067903244880
50
```

```
print(numara) # numara değeri değişmedi
```

```
20
```

```
id(numara)
```

```
2067903245200
```

```
x = 20
x = 10
```

```
x
```

```
10
```

Local, Enclosing, Global, Built-In

```
benimAdim = "Cafer"
#Global
```

```
 3
 4 def benimFonksiyonum():
 5     benimAdim = "Sami"
 6     #Enclosing
 7     def icFonksiyon():
 8         benimAdim = "Damla"
 9         #Local
10         print(benimAdim)
11     icFonksiyon()
```

```
1 benimFonksiyonum()
```

```
    Damla
```

```
1 print(benimAdim)
```

```
    Cafer
```

```
1 y = 10
2
3 def ornekFonksiyon():
4     global y
5     y = 5
6     print(y)
```

```
1 ornekFonksiyon()
```

```
    5
```

```
1 import random  as r # random yazmak yerine r yazılır
2 r.randint(1,9)
```

```
    9
```

Listeler

```
1 l1= ["myString",20,[10,20,30,40],300,40,(1,2,3,4)]
2 type(l1)
```

```
    list
```

```
1 print(l1[0][2],l1[2][2],l1[5][2],l1[-1][2]) #son index 5 olduğu için -1 de yazılabilir
```

```
    S 30 3 3
```

```
1 l1[1+1]#=l1[2]
```

```
    [10, 20, 30, 40]
```

```
l1[1]=200
l1
```

```
['myString', 200, [10, 20, 30, 40], 300, 40, (1, 2, 3, 4)]
```

```
range_list=list(range(1, 10))
range_list
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
for i in range_list:
    print(i)

```

```
1
2
3
4
5
6
7
8
9
```

```
l2 = []
for i in range_list:
    l2 += [i]
l2
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
s="dfdd"
s[0]+=2

```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_1124\801500073.py in <module>
      1 s="dfdd"
----> 2 s[0]+=2

TypeError: can only concatenate str (not "int") to str
```

SEARCH STACK OVERFLOW

```
s[0]="a"
```

```
s+="d"
s
```

```
'dfddd'
```

```
l3 = []
for i in "range_list":
    l3 += [i]
l3
```

```
['r', 'a', 'n', 'g', 'e', '_', 'l', 'i', 's', 't']
```

```
l2+l3
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 'r', 'a', 'n', 'g', 'e', '_', 'l', 'i', 's', 't']
```

```

for i in range(len(range_list)):
    print(f"{i+1}. eleman = {range_list[i]} ")

```

```
1. eleman = 1
2. eleman = 2
3. eleman = 3
4. eleman = 4
5. eleman = 5
6. eleman = 6
7. eleman = 7
8. eleman = 8
9. eleman = 9
```

```
for eleman in enumerate(l1):
    print(eleman)
```

```
(0, 'myString')
(1, 200)
(2, [10, 20, 30, 40])
(3, 300)
(4, 40)
(5, (1, 2, 3, 4))
```

```
(l3 > l1) and (l3 >l1) == (l3[0] > l1[0]) # ilk elemana göre karşılaştırır
```

```
True
```

```
l2
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
liste=[2,5,6,7,8,9]
#yukarıdaki listeyi normalleştiren fonksiyonu yazınız. yeni=l[i]/sum(l)
```

liste=[2,5,6,7,8,9]

yukarıdaki listeyi normalleştiren fonksiyonu yazınız. yeni=l[i]/sum(l)

```
def normalize_list(lst):
    total = sum(lst)
    normalized_lst = [val/total for val in lst]
    return normalized_lst
liste = [2, 5, 6, 7, 8, 9]
normalized_list = normalize_list(liste)
print(normalized_list)
```

```
[0.05405405405405406, 0.13513513513513514, 0.16216216216216217, 0.1891891891891892, 0.21621
```

farklı bir yöntem daha

```
def topla(x):
    toplam=0
    for i in x:
        toplam+=i
    return toplam
```

```

def normallestir(x):
    toplam=topla(x)
    l=[]
    for i in range(len(x)): # x listesinin uzunluğu kadar çalışır
        l.append(x[i]/toplam) # toplama bölünen yeni eleman listeye eklenir
    return l
normallestir(liste)
```

```
[0.05405405405405406,
 0.13513513513513514,
 0.16216216216216217,
 0.1891891891891892,
 0.21621621621621623,
 0.24324324324324326]
```

listeyi normalleştir

- normalleştirilen listenin özllikleri
    - elemanlarının toplamı 1 olmalı
    - tüm elemanları 0 ile 1 rasındadır

```
normalList=[eleman/sum(l2) for eleman in l2]
normalList
```

```
[0.022222222222222223,
 0.044444444444444446,
 0.06666666666666667,
 0.08888888888888889,
 0.1111111111111111,
 0.13333333333333333,
 0.15555555555555556,
 0.17777777777777778,
 0.2]
```

```
sum(normalList) # 1 ise doğrudur
```

```
1.0
```

tuple veri yapısı

```
* listeden farklı olarak boyutu sabit
* verileri silinemez yada değiştirilemez
```

```
t=()
type(t)
```

```
tuple
```

```
dir(t) #
```

```
['__add__',
 '__class__',
 '__class_getitem__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmul__',
```

```
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'count',
 'index']
```

```
1 dir(l2)
```

```
['__add__',
 '__class__',
 '__class_getitem__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__imul__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__rmul__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'append',
 'clear',
 'copy',
 'count',
 'extend',
 'index',
 'insert',
 'pop',
 'remove',
 'reverse',
 'sort']
```

```
1 len(t)
```

```
0
```

```
1 sum(t)
```

```
0
```

```
1 t="deneme",
```

```
1 type(t)
```

```
tuple
```

```
1 t=("deneme")
2 type(t) # tuple olması için üstteki gibi , bırakmamız gerek
```

```
str
```

```
1 t=(1,2,3,5)
2 print(t[0])
3 print(t[0]*45)
```

```
1
45
```

```
1 t[0]=100 # tuple değiştirilemez olduğundan hata verecektir
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_1124\1437634803.py in <module>
----> 1 t[0]=100 # tuple değiştirilemez olduğundan hata verecektir

TypeError: 'tuple' object does not support item assignment
```

SEARCH STACK OVERFLOW

```
1 t = (1,2,3,4)
2 t2= (10,20,30,40)
```

```
1 t2 = t # bunu yaptığımız anda t2'nin elemanlarına erişemiyeceğiz
```

```
1 t2
```

```
(1, 2, 3, 4)
```

```
1 t = (1,2,3,4)
2 t2= (10,20,30,40)
```

```
1 t2 + t
```

```
(10, 20, 30, 40, 1, 2, 3, 4)
```

```
1 t2 += t # aslında yeni bir tuple oluşşturup atandı
```

```
1 t2
```

```
(10, 20, 30, 40, 1, 2, 3, 4)
```

```
1 l2+t2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_1124\1209953008.py in <module>
----> 1 l2+t2

TypeError: can only concatenate list (not "tuple") to list
```

SEARCH STACK OVERFLOW

```
1 l2 + list(t2)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 1, 2, 3, 4]
```

Referans tip

- listeler bellekt bir yer tutar ve atama yapıldığında adresini verirler
- adresini tutan diğer değişkenin üzerinde değişiklik yapılırsa orjinal liste değişir
    - orjinal liste değişmemesi için koyası alınması gerekmektedir (copy komutu ile)

Not 1

```
1 sublist = [1,2,3]
2 list2=[10,20,sublist]
3 list2
```

```
[10, 20, [1, 2, 3]]
```

```
1 sublist.clear()
2 list2
```

```
[10, 20, []]
```

Not 2 silme& ekleme işlemi

```
1 sublist = [1,2,3]
2 list2=[10,20,sublist]
3 list2
```

```
[10, 20, [1, 2, 3]]
```

```
sublist2 = sublist
sublist2.clear()
list2
```

```
[10, 20, []]
```

```
sublist2 = sublist
sublist2.append(2)
list2
```

```
[10, 20, [2]]
```

Not 3 kopy alırsak değerler değişmez

```
sublist = [1,2,3]
list2=[10,20,sublist]
list2
```

```
[10, 20, [1, 2, 3]]
```

```
sublist2 = sublist.copy()
sublist2.append(2)
list2
```

```
[10, 20, [1, 2, 3]]
```

tuple içindeki verileri silmemize veya değiştirmemize izin vermez demiştik !!

```
t=(1,2,3,sublist)
```

```
t
```

```
(1, 2, 3, [1, 2, 3])
```

```
sublist.clear()#append remove gibi metodlar kullanılabilir
t
```

```
(1, 2, 3, [])
```

```
a,b,c="ABC"
```

```
```

```
1 a
```

```
'A'
```

```
1 b
```

```
'B'
```

```
1 list1=[1,3,5]
2 a,b,c = list1
```

```
1 a
```

```
1
```

```
1 c
```

```
5
```

```
1 a,b,c=range(5,8)
2 a,b,c
```

```
(5, 6, 7)
```

```
1 a=45
2 b=62
3 a,b =b,a #swap
```

```
1 a,b
```

```
(62, 45)
```

slice

```
1 l=list(range(10))
2 l[0:5]# son index (5.)  dahil değil
```

```
[0, 1, 2, 3, 4]
```

```
1 l[5:]
```

```
[5, 6, 7, 8, 9]
```

```
1 l[:]# hepsini yazar
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
1 l[1:8:2] #başlangıç,bitiş,adım
```

```
[1, 3, 5, 7]
```

```
1 print(l)
2 print(l[::-1],"yöntem 1")#tüm elemanları tersten yazar
3 l.reverse()
4 l,"reverse metodu ile ters çevrilmiş" #kalıcı değişiklik
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0] yöntem 1
([9, 8, 7, 6, 5, 4, 3, 2, 1, 0], 'reverse metodu ile ters çevrilmiş')
```

```
1 l[:3]=['x','b','c','d','e']
2 l
```

```
['x', 'b', 'c', 'd', 'e', 6, 5, 4, 3, 2, 1, 0]
```

```
1 l[:3]
```

```
['x', 'b', 'c']
```

```
1 l[::2]=['d', 'e', 'd', 'e', 'd', 4]
2 l
```

```
['d', 'b', 'e', 'd', 'd', 6, 'e', 4, 'd', 2, 4, 0]
```

```
1 l=['x', 'b', 'c', 'd', 'e', 'd', 'e', 'd', 4, 5, 6, 7, 8, 9]
2 del l[:5]
3 l
```

```
['d', 'e', 'd', 4, 5, 6, 7, 8, 9]
```

```
1 del l[:] # del l ile aynı
2 l
```

```
[]
```

```
1 def a(x):
2     for i in range(len(x)):
3         x[i]*=2
4
```

- listenin kopyası değil referansı gittiği için değişiklikler kalıcı olur

  dolaylısıyla return kullanmaya gerek yoktur

```
1
2 l=[1,2,3,4]
```

```
3 a(l)
4 l
```

```
[2, 4, 6, 8]
```

```
1 t=(1,2,3,4)
2 t[::2]
```

```
(1, 3)
```

```
1 a(t)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_1124\3511233446.py in <module>
----> 1 a(t)

~\AppData\Local\Temp\ipykernel_1124\3497654507.py in a(x)
      1 def a(x):
      2     for i in range(len(x)):
----> 3         x[i]*=2

TypeError: 'tuple' object does not support item assignment
```

SEARCH STACK OVERFLOW

faydalı liste Metodları

```
1 str_List=["Murtaza","cafer","Damla","Sami"]
```

```
1 str_List.reverse()#listeyi ters çevirir
2 str_List
```

```
['Sami', 'Damla', 'cafer', 'Murtaza']
```

```
1 str_List.sort()#ascii değerine göre sıralar
2 str_List
```

```
['Damla', 'Murtaza', 'Sami', 'cafer']
```

```
1 benimListem = [10,20,30,40]
2 type(benimListem)
```

```
list
```

```
1 benimListem.append(50)#son index'e ekler
2 benimListem
```

```
[10, 20, 30, 40, 50]
```

```
1 benimListem.extend("denememem")
2 benimListem
```

```
[10, 20, 30, 40, 50, 'd', 'e', 'n', 'e', 'm', 'e', 'm']
```

```
1 benimListem.extend((1,2,3))
2 benimListem
```

```
[10, 20, 30, 40, 50, 'd', 'e', 'n', 'e', 'm', 'e', 'm', 1, 2, 3]
```

```
1 benimListem.pop()#son elemanı çıkarır & dödürür
```

```
3
```

```
1 benimListem
```

```
[10, 20, 30, 40, 50, 'd', 'e', 'n', 'e', 'm', 'e', 'm', 1, 2]
```

```
1 benimListem.remove(40)# 40 sayı sil
```

```
1 benimListem.count(20)# kaç tane 20 var
```

```
1
```

```
1 benimListem.append(20) # 20 ekle
```

```
1 benimListem.count(20) # kaç tane 20 var
```

```
2
```

```
1 benimListem
```

```
[10, 20, 30, 50, 'd', 'e', 'n', 'e', 'm', 'e', 'm', 1, 2, 20]
```

```
1 benimListem *2
```

```
[10,
 20,
 30,
 50,
 'd',
 'e',
 'n',
 'e',
 'm',
 'e',
 'm',
 1,
 2,
 20,
 10,
 20,
 30,
```

```
 50,
 'd',
 'e',
 'n',
 'e',
 'm',
 'e',
 'm',
 1,
 2,
 20]
```

```
karisikListe = [1,2,3.5,"cafer",9]
type(karisikListe)

```

```
list
```

```
sonucum = karisikListe[3]
karisikListe[3]
```

```
'cafer'
```

```
l2 = [1,5,"Asi",4,[6,"z"]]
```

```
zDegiskenimiz = l2[4][1]
zDegiskenimiz
```

```
'z'
```

```
karmasikListe = [[1,2,3,["a","b"],50],40,20,["z",5.5],[3,["a"]]]
```

```
bDegiskenimiz = karmasikListe[0][3][1]
bDegiskenimiz
```

```
'b'
```

```
benimAdim = "Murtaza"
```

```
benimAdimBuyukHarfli = benimAdim.upper()
benimAdimBuyukHarfli
```

```
'MURTAZA'
```

```
help(benimAdim.upper)
```

```
Help on built-in function upper:

upper() method of builtins.str instance
    Return a copy of the string converted to uppercase.
```

```
def yeniToplama(*args):
    #print(type(args))#tuple
    return sum(args)
```

```
yeniToplama(10,20,30,40,50,60)
```

```
210
```

```
def ornekFonksiyon(**kwargs):
    return(kwargs) # kwargs döndürülmeli !!!

```

```
sonucum=ornekFonksiyon(Sami = 90, Murtaza = 100,Cafer = 101)
type(sonucum)

```

```
dict
```

```
def keyWordKontrolu(kwargs):
    # girdi olarak dict olarak verilmeseydi ustteki fonksiyon gibi kullanıcıdan **kwargs alın
    if "Damla" in kwargs:
        print("Damla var")
    else:
        print("Damla yok")

```

```
keyWordKontrolu(sonucum)
```

```
Damla yok
```

```
def benimFonksiyonum(*args):
    return args
type(benimFonksiyonum(20,30,40))
```

```
tuple
```

```
1
```

```
1
```

```
1
```