

Python’da Kilit Oyunu Simülasyonu

Python ile kilit oyununu oynayabileceğiniz bir program



Bager Kurbanoğlu

İÇİNDEKİLER

Önsöz 2

Programcı Kataloğu 3

**Nasıl Bakılmalı** 3

Fonksiyonlar: 4

main(): {1} 4

input\_alici(): {2} 5

alan\_yapici(): {3} 6

alan\_yazdirici(): {4} 7

etraftaki\_sembol\_bulucu(): {5} 8

tas\_alma\_kontrolu(): {6} 9

oyun\_update(): {7} 11

İntegerlar: 12

Listeler: 14

Stringler: 17

Test Kataloğu 18

# Önsöz

Bu program ile amaçlanan kilit adlı oyunu python ile simule etmektir. Bu raporda programın genel olarak nasıl çalıştığı ile ilgili bilgiler verilecek, değişkenler ve fonksiyonlar **Programcı Kataloğu** adı altındaki bölümde ele alınacaktır. Deneme inputları sorun çıkarabilecek girdiler ve bu girdilerin çıktıları ise **Test Kataloğu** bölümü altında incelenecektir.

# Programcı Kataloğu

## **Nasıl Bakılmalı**

## Bu programcı kataloğunda öncelikle programda kullanılan fonksiyonlar açıklanacak.

Bu açıklanan fonksiyonların adlarının yanında **{}** içinde bir sayı bulunuyor. Bu sayı hangi değişkenlerin hangi fonksiyona ait olduğunu açıklamak için konulmuştur. İlk sayı hangi fonksiyona ait olduğunu açıklarken sonraki sayılar ise bu değişken başka fonksyonlara argüman olarak verilmşse, hangi fonksiyonlara verildiğini açıklamak için verilmiştir.

Bazı değişkenler birden fazla kez tekrar edebilir. Buralarda olduğu fonksiyon içinde ne işe yaradığı yazılmıştır.

## Fonksiyonlar:

**main(): {1}**

**Argümanlar: ()**

**Fonksiyonun amacı:**

Genel olarak diğer fonksiyonları içine alan ve önemli listelerin ilk olarak yapıldığı fonksiyon.

**Fonksiyonun çalışma prensibi:**

devam değişkeni “E” ya da “e” olduğu sürece oyunun devam etmesini sağlar.

kar1, kar2 değişkenleri kullanıcıdan alınan değerlere eşitlenir.

sutun\_satir değişkeni input\_alici() fonksiyonu ile kullanıcının girdiği değere eşitlenir.

alan\_yapici() fonksiyonu ile kullanılacak oyun\_alani listesi hazırlanır.

Kullanıcı görsün diye alan\_yazdır() fonksiyonu ile oyun alanı yazdırılır.

Kazanan değeri oyun\_update() fonksiyonun returnlediği değere eşitleyerek hem kazanan kişi bulunur hem de oyun oynanmaya başlanır.

**input\_alici():{2}**

**Argümanlar:**

(min=-1000000, max=1000000, normal\_mesaj="", hata\_mesaji="")

**Fonksiyonun amacı:**

Genel olarak kullanıcıdan alınan inputların değer kontrollerini yapmak.

**Fonksiyonun çalışma prensibi:**

try ve except ile sayı olmayan değer kontrolü ve koşullarla min max a göre değer kontrolü yapılır.

Sonra input returnlenir.

**alan\_yapici(): {3}**

**Argümanlar:**

(oyun\_alani, sutun\_satir, kar1, kar2, suanki\_oyuncu\_tas\_kordinat, siradaki\_oyuncu\_tas\_kordinat)

**Fonksiyonun amacı:**

Alınan argümanlar ile oyun\_alani iki boyutlu listesini yaratan ve karakterleri olmaları gereken yerlere yerleştiren fonksiyon

**Fonksiyonun çalışma prensibi:**

append metodu ile oyun\_alani listesini iki boyutlu listeye çevirir.

Burada ayrıca satir\_sutun sayısı kadar eleman yaratılmak yerine fazladan iki eleman da yaratılır ve bu sütün ve satırlar “/” karakterine eşitlenir. Bunun nedeni hem bir karakterin köşede olduğunu daha kolay bulmak, hem de çıkacak index errorların önüne geçmek içindir.

Bu konu hakkında daha detaylı bilgi için oyun\_alani değişkeninin açıklamasına bakınız

**alan\_yazdirici(): {4}**

**Argümanlar:**

(oyun\_alani, sutun\_satir, harfler)

**Fonksiyonun amacı:**

Bu fonksiyonun tek görevi verilen oyun\_alani listesini göze uygun bir şekilde yazdırmaktır.

**Fonksiyonun çalışma prensibi:**

Fonksiyon genel olarak karmaşık gözükse de kodların hepsi sadece spesifik bir görünüşü yakalamak için birçok kez tekrar eden kodlardır. Arada eklenmesi gereken detaylar için birçok sayıda print kullanılmıştır.

**etraftaki\_sembol\_bulucu(): {5}**

**Argümanlar:**

(oyun\_alani,istenilen\_kordinat)

**Fonksiyonun amacı:**

Verilen kordinat etrafındaki sembolleri bulmak

**Fonksiyonun çalışma prensibi:**

Bu fonksiyon verilen kordinat etrafındaki(komşuları) oyun\_alani listesine erişerek bulur ve bu değerleri yukarıdaki eleman 0. eleman olmak üzere (saat yönünde dönerek) 4 elemanlı bir listeye atar ve bu listeyi returnler. Bu listede 0.eleman üst, 1.eleman sağ, 2.eleman alt, 3.eleman soldaki karakteri saklar.

**tas\_alma\_kontrolu(): {6}**

**Argümanlar:**

(oyun\_alani,oynatilan\_tas\_kordinat, suanki\_oyuncu, siradaki\_oyuncu)

**Fonksiyonun amacı:**

etraftaki\_sembol\_bulucu fonksiyonunu da kullanarak karşı oyuncunun taşının alınıp alınmadığına bakmak ve alınmışsa taşın kordinatlarını döndürmek.

**Fonksiyonun çalışma prensibi:**

Koşullar ve etraftaki\_sembol\_bulucu() yardımı ile taşın gittiği yerin etrafını tarar ve bunları bir listeye atar. Sonra for döngüsü ile bu sembolleri gezer ve zıt oyuncunun sembolü ise alınma şartlarına bakmaya başlar. Burada zıt taşın konumuna göre taş arasında kalıyor mu ona bakar ve kalıyor ise kordinatlarını geri döndürür.

Burada köşelerin alınması diğer durumlara benzemediğinden mantığı biraz farklıdır:

Öncelikle oynatılan taşın etrafındaki her zıt karakterin de etrafına bakılır. Bu köşede olup olmadığını anlamak içindir.

Eğer ki oyun alanının oynanan kısmının köşelerine bakacak olursak görürüz ki mutlaka etrafında iki tane ”/” karakteri olmalıdır. Bir taşı almak için zaten iki zıt taş gerekeceği için şartlara, eğer etrafında iki zıt taş ve iki tane “/” karakteri varsa yazılabilir. Bu durumda taş kesin alınmalıdır.

**oyun\_update(): {7}**

**Argümanlar:**

(oyun\_alani, oyuncu1, oyuncu2, sutun\_satir, harfler, suanki\_oyuncu\_tas\_kordinat, siradaki\_oyuncu\_tas\_kordinat)

**Fonksiyonun amacı:**

Genel olarak oyunda hamlelerin yapılması, taşların hareket ettirilmesi, oyunun çizimi ve oyunun maç içindeki döngüsünden sorumludur.

**Fonksiyonun çalışma prensibi:**

kazanan değişkeni ile basit bir while döngüsü kurulur. Oyuncuların taş sayıları değişkene atanır. Sonra hareket ettirme girdisi ve bu girdinin kontrolleri yapılır. Bu girdi replace ile anlaşılabilecek kordinatlara dönüştürülür ve map() fonksiyonu ile bu değerler stringden her biri bir eleman olan bir hareket kordinat listesine dönüştürülür. Sonra taş hareket ettirilir ve taş alma kontrolü yaptırılır. Taş alınmışsa sayı düşürülür. Kazanan kontrolü yapılır ve en sonda ise şuanki oyuncunun değişkenleri ile sıradaki oyuncunun değişkenleri değiştirilir. En sonda kazanan döndürülür.

## İntegerlar:

**MIN\_SUTUN: {1,2}**

Oyun alanının alabileceği minimum değer

**MAX\_SUTUN: {1,2}**

Oyun alanının alabileceği maximum değer

**KARAKTER\_UZUNLUGU: {1}**

Kullanıcıdan alınacak karakterin uzunluğu

**sutun\_satir: {1,3,4,7}**

Kullanıcıdan oyun alanının kaç satır ve sütün olacağının değerinin saklandığı yer.

**deger: {2}**

returnlenen kontrol edilmiş input değeri.

**sembol\_no: {6}**

etraftaki\_semboller değişkeninde index değeri olarak kullanılacak değer.

**suanki\_oyuncu\_tas\_sayi: {7}**

Şuanki oyuncuda olan taş sayısı.

**siradaki\_oyuncu\_tas\_sayi: {7}**

Sıradaki oyuncuda olan taş sayısı.

**KORDINAT\_GIRDI\_MAX\_UZUNLUK: {7}**

Hareket girdisinin alabileceği max uzunluk.(boşluk dahil değil)

## Listeler:

**oyun\_alani: {1,3,4,5,6,7}**

Oyundaki sembolleri içinde bulunduran iki boyutlu liste.

Bu liste sembol kontrolü yaparken hata kodu vermesin ve köşe ve kenarlar daha kolay anlaşılsın diye satir\_sütün sayısından 2 fazla satır ve sütün tutar. Bu fazladan satır ve sütünlara ise “/” karakteri atanır Aslında liste kordinat sistemi olarak düşünülürse şu şekilde görülür.

0 1 2 3 4 5

0 / / / / / /

1 / y y y y /

2 / /

3 / /

4 / x x x x /

5 / / / / / /

Bu şekilde komşuları iki “/” olan yerlerin köşeler olduğu anlaşılır ve şartlar daha kolay yapılır. Ayrıca sembollerin kordinatları tahtadakiler ile birebir aynı sayılarda olur.

**harfler: {1,3,4,7}**

Verilen sınırlara göre kullanılacak harflerin saklandığı liste

**suanki\_oyuncu\_tas\_kordinat: {1,3,7}**

Şuanki sırada olan oyuncunun taşlarının konumlarını saklayan iki boyutlu liste

**siradaki\_oyuncu\_tas\_kordinat: {1,3,7}**

Sıradaki oyuncunun taşlarının konumlarını saklayan iki boyutulu liste

**temp\_liste: {3}**

Başka bir listeye appendlemek için kullanılan liste. Sabit bir değeri yok geçici bir değişken.

**etraftaki\_semboller : {6}**

Oynatılan taş etrafındaki sembolleri saklayan liste.

**kordinat : {6}**

Alınan taş varsa o taşın konumunu saklayan, yoksa [-1, -1] değerini saklayan liste.

**hareket\_konumları : {7}**

Kullanıcıdan alınan hareket girdilerinin saklandığı yer.

**alinan\_tas\_kordinat\_gecici : {7}**

Alınan taş değerinin şart kontrolünde kullanılmak için geçici olarak atandığı değişken.

**alinan\_tas\_kordinat\_sakla: {7}**

Kullanıcıya alınan taşı yazdırırken kullanılacak değeri saklar.

## Stringler:

**devam: {1}**

Ana oyun loopunun devamını sağlayan değer

**kar1: {1,3,6,7}**

Oyuncudan alınan ilk oyuncunun karakterinin saklandığı string

**kar2: {1,3,6,7}**

Oyuncudan alınan ilk oyuncunun karakterinin saklandığı string

**kazanan: {1,7}**

Verilen sınırlara göre kullanılacak harflerin saklandığı liste

**suanki\_oyuncu: {7}**

Şuanki oyuncunun karakteri.

**siradaki\_oyuncu: {7}**

Sıradaki oyuncunun karakteri.

# Test Kataloğu

**Taş Hareket Ettirme:**

**Girdiler:**

X

Y

6

1c 3c

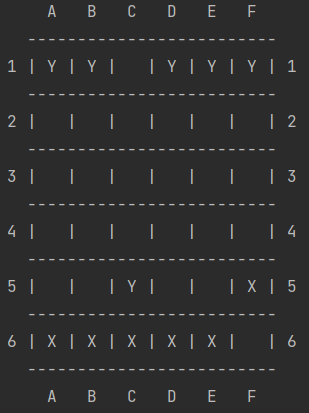
**Beklenen çıktı:**

Seçilen yerde taşınız yok

**Üretilen çıktı:**

Seçilen yerde taşınız yok

**Olası durum:**



**Girdiler:**

5f 5a

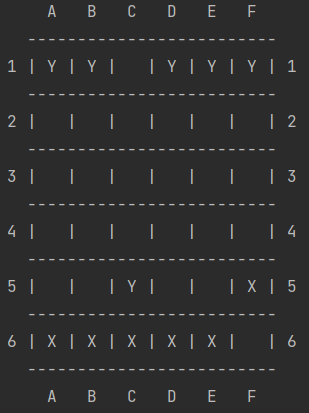
**Beklenen çıktı:**

Gidilecek yer ile taş arasında başka bir taş var.

**Üretilen çıktı:**

Gidilecek yer ile taş arasında başka bir taş var.

**Olası durum:**



**Girdiler:**

5f 2a

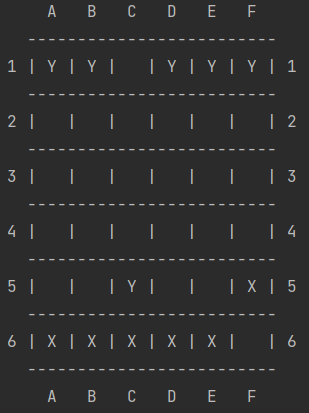
**Beklenen çıktı:**

Taşlar sadece çizgisel hareket edebilir.

**Üretilen çıktı:**

Taşlar sadece çizgisel hareket edebilir.

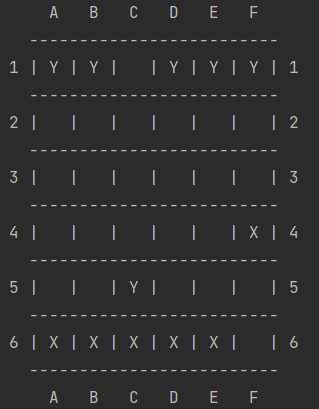
**Olası durum:**



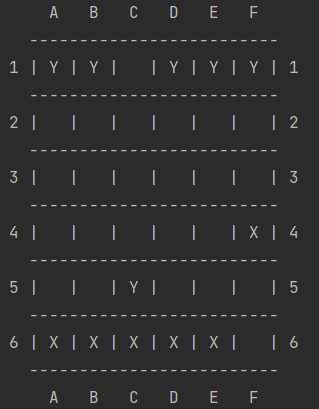
**Girdiler:**

5f 4f

**Beklenen çıktı:**

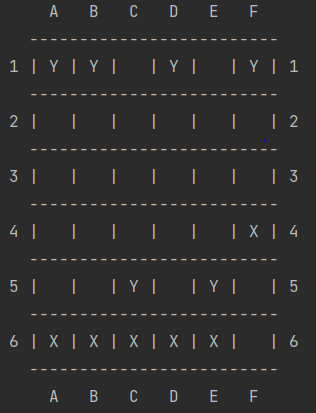


**Üretilen çıktı:**



,

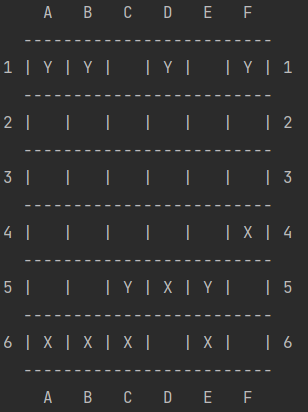
**Olası durum:**



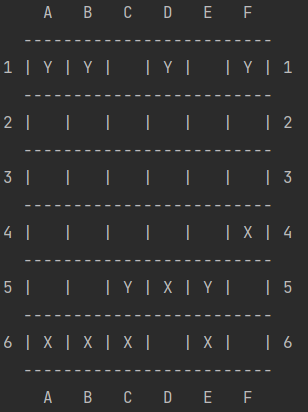
**Girdiler:**

6d 5d

**Beklenen çıktı:**

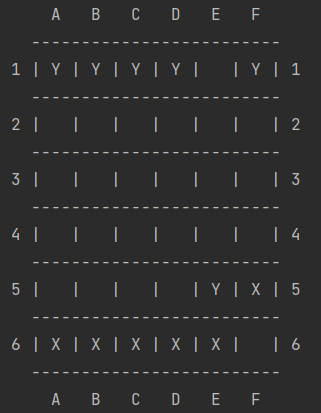


**Üretilen çıktı:**



**Taş Alma:**

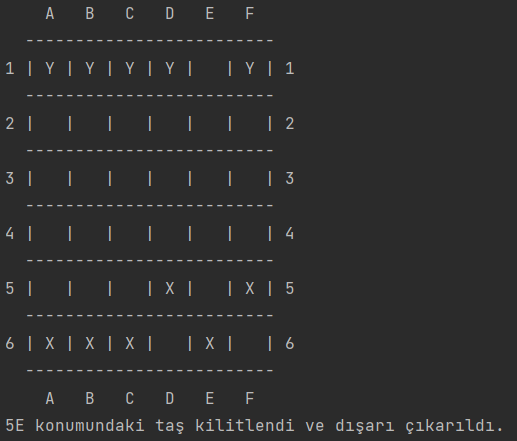
**Olası durum:**



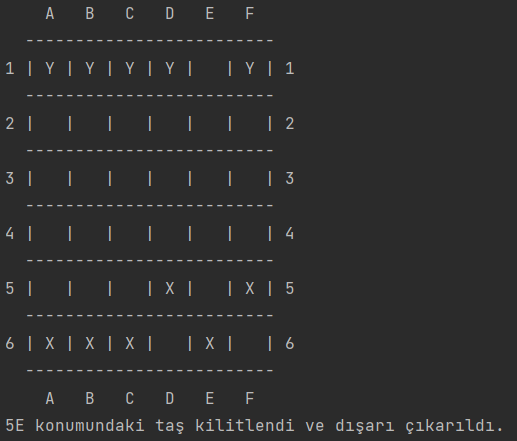
**Girdiler:**

6d 5d

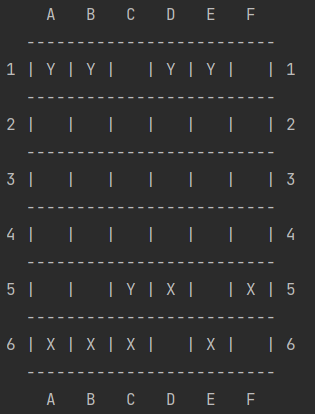
**Beklenen çıktı:**



**Üretilen çıktı:**



**Olası durum:**



**Girdiler:**

1e 5e

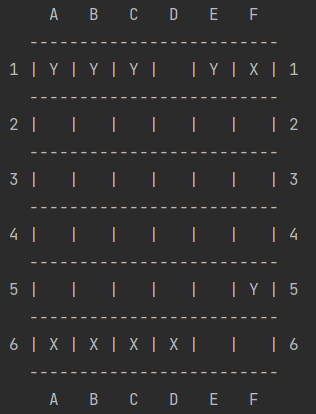
**Beklenen çıktı:**



**Üretilen çıktı:**



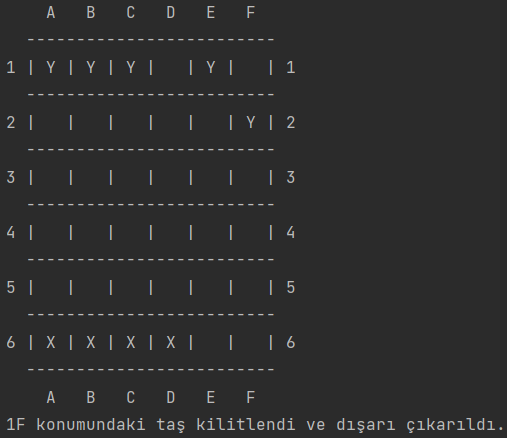
**Olası durum:**



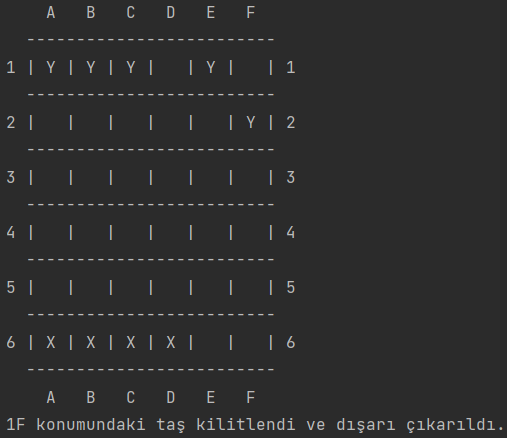
**Girdiler:**

5f 2f

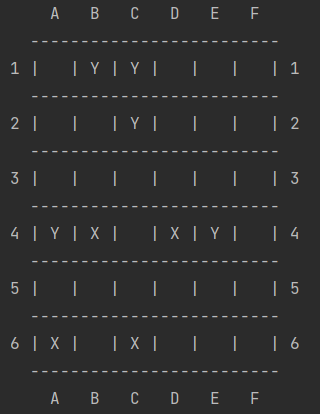
**Beklenen çıktı:**



**Üretilen çıktı:**



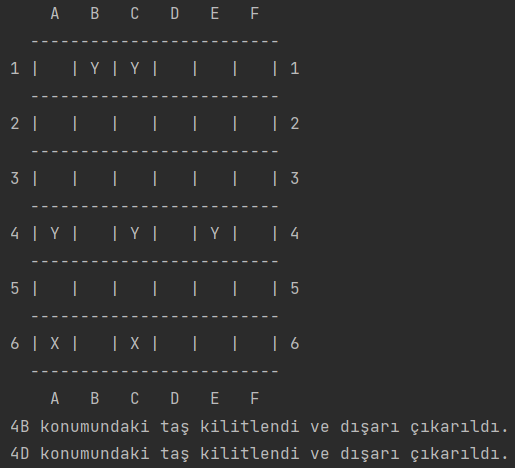
**Olası durum:**



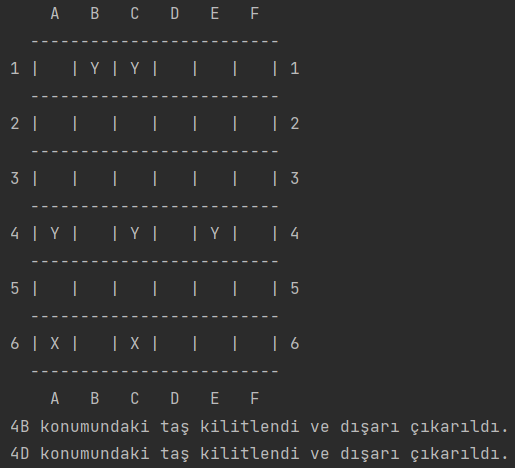
**Girdiler:**

2c 4c

**Beklenen çıktı:**



**Üretilen çıktı:**



**Tam Bir Oyun:**

**Girdiler:**

X

Y

4

4d 2d

1c 3c

4b 2b

3c 3d

2b 2d

1b 1c

4c 4d

1c 1b

4a 4c

1a 3a

4c 1c

1b 1a

2d 2a

1a 1b

4d 4a

h

**Çıktılar:**

