

Python for Data Science

Çalışma Dökümanı

Recep Aydoğdu

İçindekiler

[Data Science Kullanılan Alanlar 6](#_Toc37122515)

[Data Science Proje Döngüsü 6](#_Toc37122516)

[Veri Bilimine Giriş Alıştırmalar – 1 7](#_Toc37122517)

[Veri Bilimine Giriş Alıştırmalar – 2 10](#_Toc37122518)

[Python Programlama 14](#_Toc37122519)

[Temel Hareketler 14](#_Toc37122520)

[Integer, Float ve String 14](#_Toc37122521)

[Integer 14](#_Toc37122522)

[Float 14](#_Toc37122523)

[String 14](#_Toc37122524)

[Type 14](#_Toc37122525)

[String Metodları 15](#_Toc37122526)

[len() 15](#_Toc37122527)

[upper() & lower() 15](#_Toc37122528)

[isupper() & islower() 15](#_Toc37122529)

[replace() 15](#_Toc37122530)

[strip() 15](#_Toc37122531)

[dir() 16](#_Toc37122532)

[capitalize() 16](#_Toc37122533)

[title() 16](#_Toc37122534)

[Substring 16](#_Toc37122535)

[Type Dönüşümleri 16](#_Toc37122536)

[print() fonksiyonu 16](#_Toc37122537)

[Python Programlama Alıştırmalar – 1 17](#_Toc37122538)

[Python Programlama Alıştırmalar – 2 20](#_Toc37122539)

[Python Programlama Alıştırmalar – 3 25](#_Toc37122540)

[Veri Yapıları (Data Types) 29](#_Toc37122541)

[Listeler 29](#_Toc37122542)

[Liste Elemanlarına Ulaşma 29](#_Toc37122543)

[Liste İçi Type Sorgulama 29](#_Toc37122544)

[Liste elemanlarını değiştirme 30](#_Toc37122545)

[Listeye eleman ekleme 30](#_Toc37122546)

[Listeden eleman silme 30](#_Toc37122547)

[append ve remove metodlari 30](#_Toc37122548)

[insert metodu 30](#_Toc37122549)

[pop metodu 30](#_Toc37122550)

[count metodu 31](#_Toc37122551)

[copy metodu 31](#_Toc37122552)

[extend metodu 31](#_Toc37122553)

[index metodu 31](#_Toc37122554)

[reverse metodu 31](#_Toc37122555)

[sort metodu 31](#_Toc37122556)

[clear metodu 32](#_Toc37122557)

[Tuple (Demet) 32](#_Toc37122558)

[Tuple Oluşturma 32](#_Toc37122559)

[Eleman İşlemleri 32](#_Toc37122560)

[Dictionary (Sözlük) 32](#_Toc37122561)

[Dictionary Nedir? 32](#_Toc37122562)

[Dictionary Oluşturma 32](#_Toc37122563)

[Eleman Seçme İşlemleri 33](#_Toc37122564)

[Eleman Ekleme & Değiştirme 33](#_Toc37122565)

[Sets (Kümeler) 34](#_Toc37122566)

[Set Oluşturma 34](#_Toc37122567)

[Set’lere eleman ekleme ve çıkarma işlemleri 35](#_Toc37122568)

[Set’lerde Fark İşlemleri 37](#_Toc37122569)

[Set’lerde Kesişim ve Birleşim İşlemleri 37](#_Toc37122570)

[Set’lerde Sorgu İşlemleri 38](#_Toc37122571)

[Veri Yapıları Özet 38](#_Toc37122572)

[Python Programlama Alıştırmalar – 4 39](#_Toc37122573)

[Python Programlama Alıştırmalar – 5 42](#_Toc37122574)

[Python Programlama Alıştırmalar – 6 47](#_Toc37122575)

[Fonksiyonlar 52](#_Toc37122576)

[Fonksiyon Nedir? 52](#_Toc37122577)

[Matematiksel İşlemler 52](#_Toc37122578)

[Üs Alma 52](#_Toc37122579)

[Fonksiyon Nasıl Yazılır ? 52](#_Toc37122580)

[Bilgi Notuyla Çıktı Üretmek 53](#_Toc37122581)

[İki Argümanlı Fonksiyon Tanımlamak 53](#_Toc37122582)

[Ön Tanımlı Argümanlar 54](#_Toc37122583)

[Argümanların Sıralaması 54](#_Toc37122584)

[Ne Zaman Fonksiyon Yazılır? 54](#_Toc37122585)

[Fonksiyon Çıktılarını Girdi Olarak Kullanmak 54](#_Toc37122586)

[Local ve Global Değişkenler 55](#_Toc37122587)

[Local Etki Alanından Global Etki Alanını Değiştirme 56](#_Toc37122588)

[Karar-Kontrol Yapıları (Koşullar) 57](#_Toc37122589)

[Koşul Nedir? 57](#_Toc37122590)

[True – False Sorgulamaları (Boolean) 57](#_Toc37122591)

[if – else – elif 57](#_Toc37122592)

[**Uygulama:** if ve input ile kullanıcı etkileşimli program 59](#_Toc37122593)

[Döngüler 59](#_Toc37122594)

[For Döngüsü 59](#_Toc37122595)

[Döngü ve Fonksiyonların Birlikte Kullanımı 60](#_Toc37122596)

[**Uygulama:** if, for ve fonksiyonların birlikte kullanımı 60](#_Toc37122597)

[break & continue 60](#_Toc37122598)

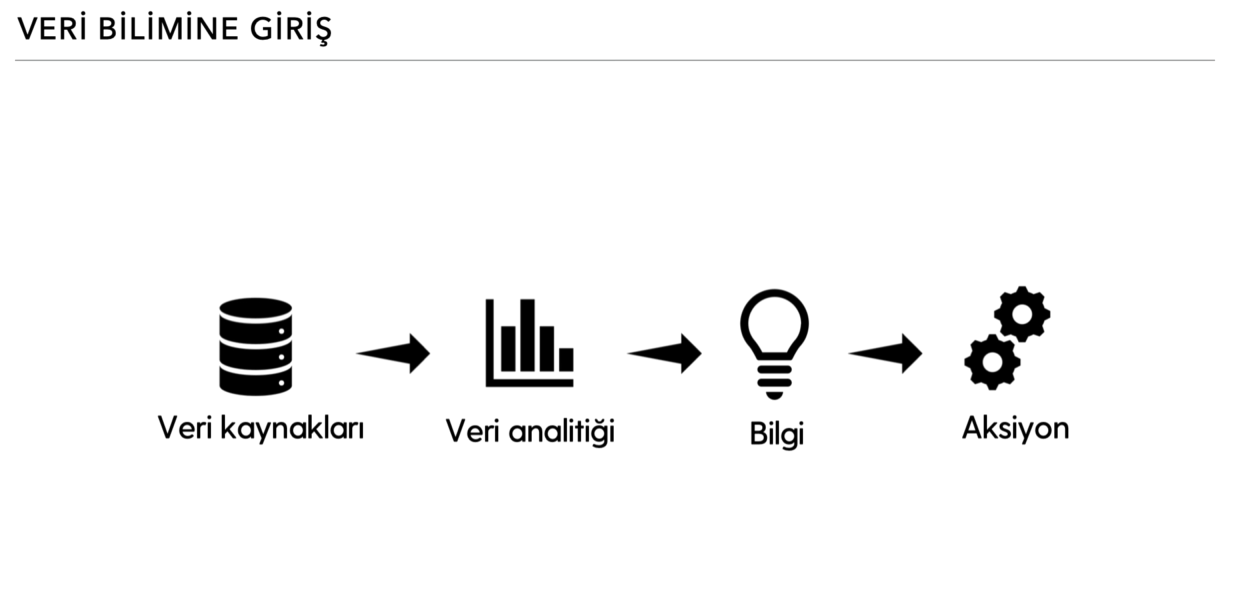
[while 61](#_Toc37122599)

[Python Programlama Alıştırmalar - 7 62](#_Toc37122600)

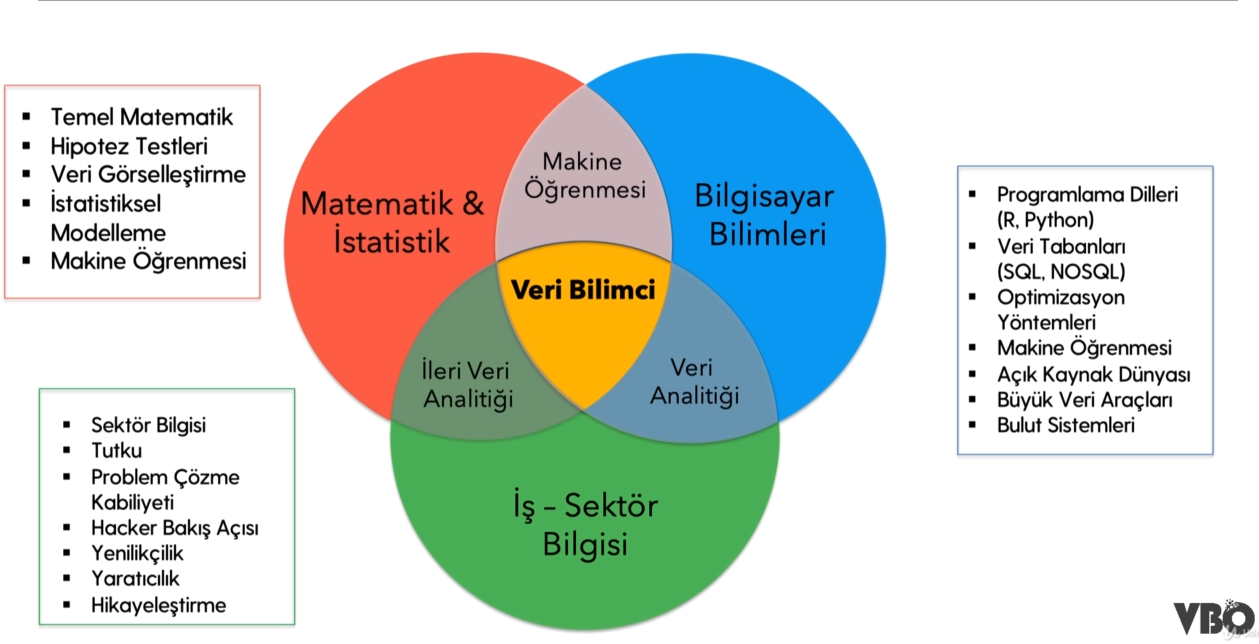
[Python Programlama Alıştırmalar - 8 66](#_Toc37122601)

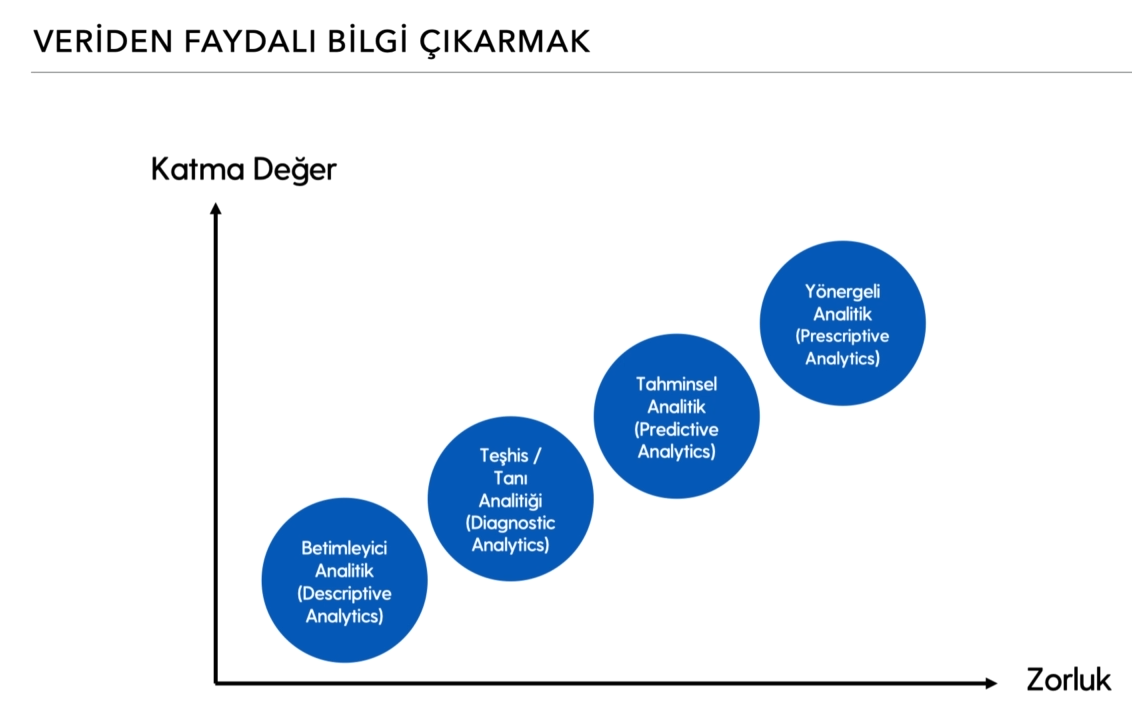
[Python Programlama Alıştırmalar – 9 71](#_Toc37122602)

**Data Science**

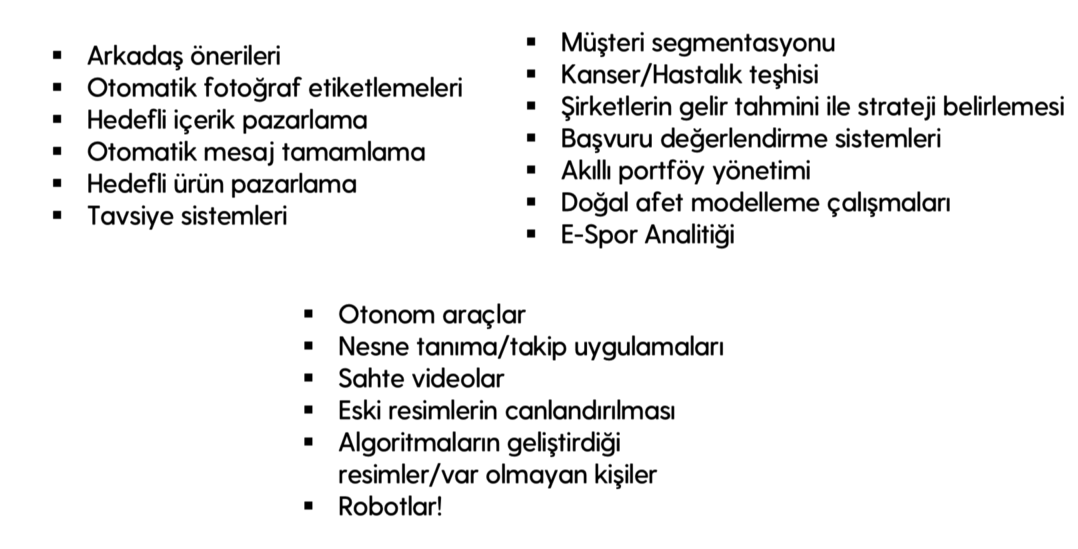


Veri Bilimci, veriden faydalı bilgi çıkarma sürecini yöneten kişidir.

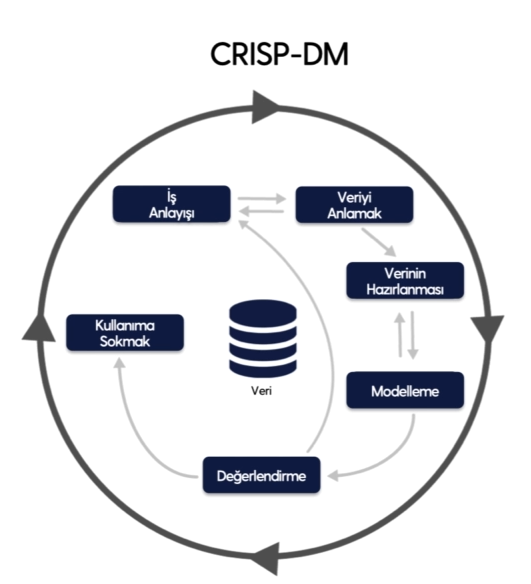




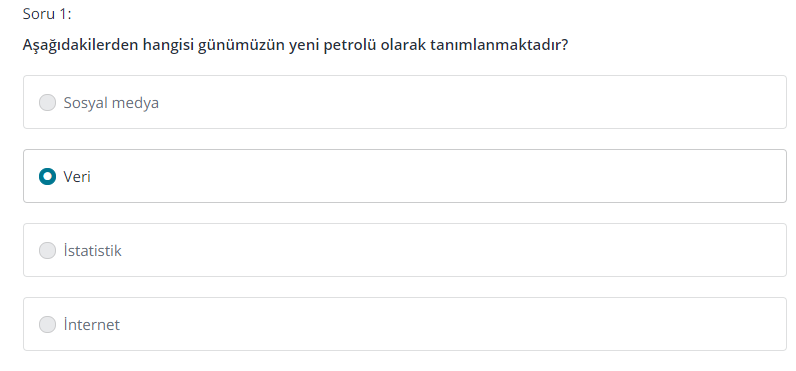
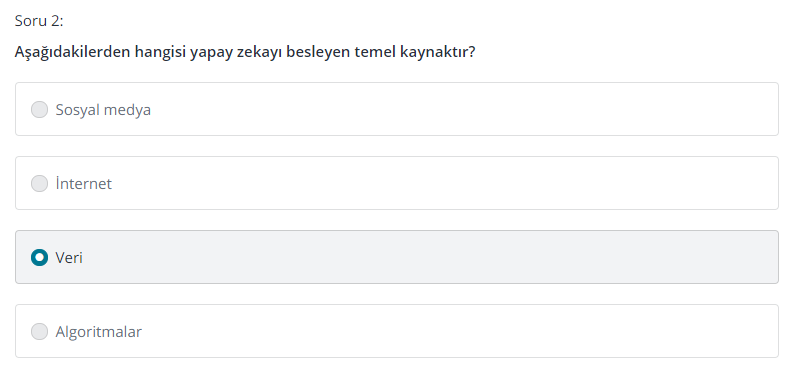
# Data Science Kullanılan Alanlar

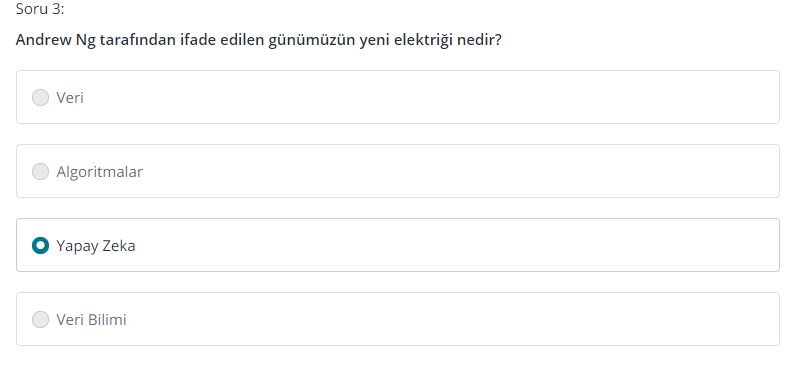


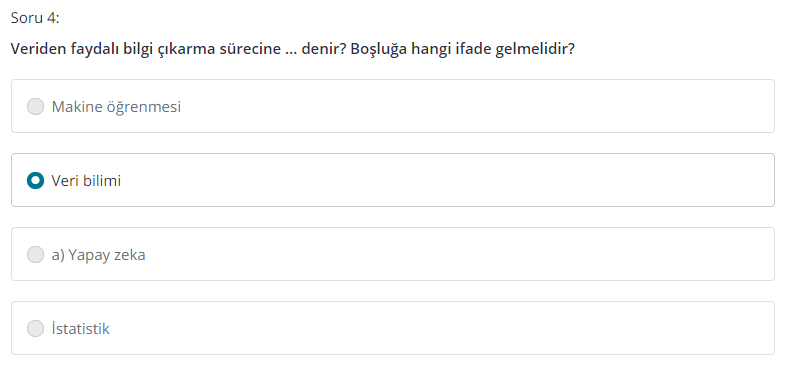
# Data Science Proje Döngüsü

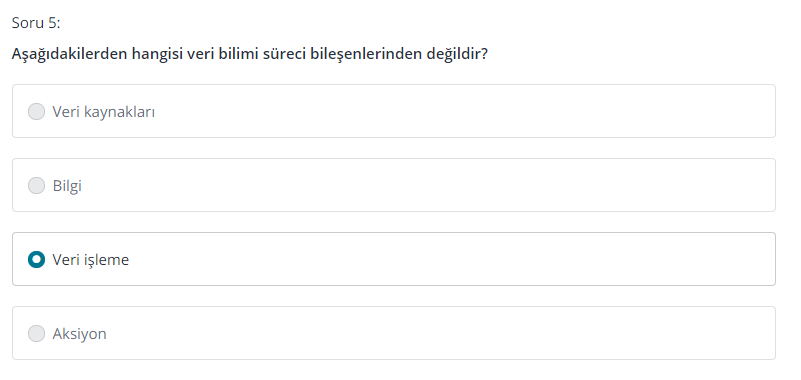


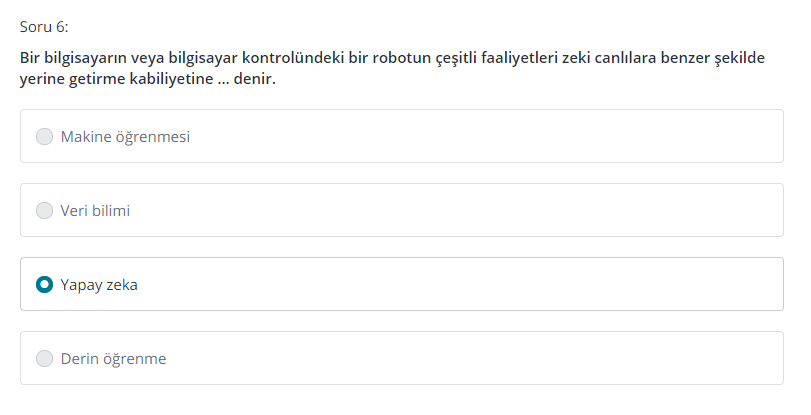
### Veri Bilimine Giriş Alıştırmalar – 1

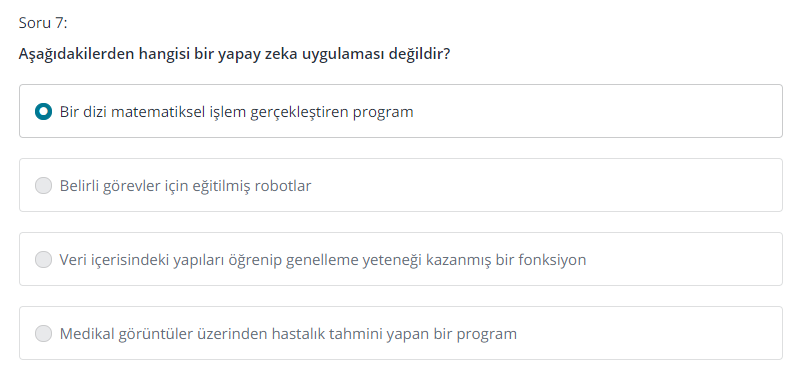
 

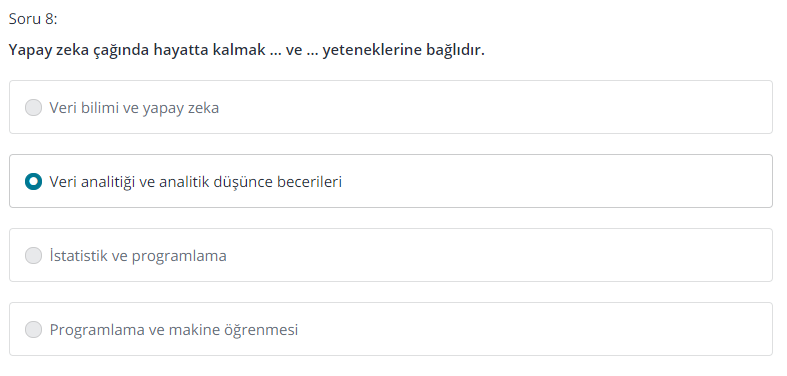


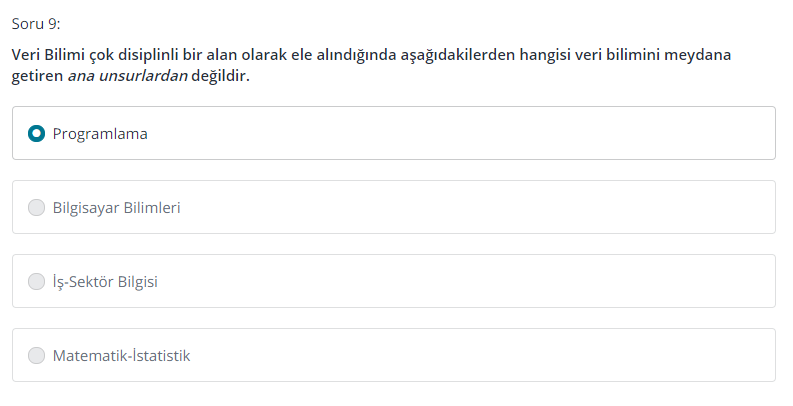
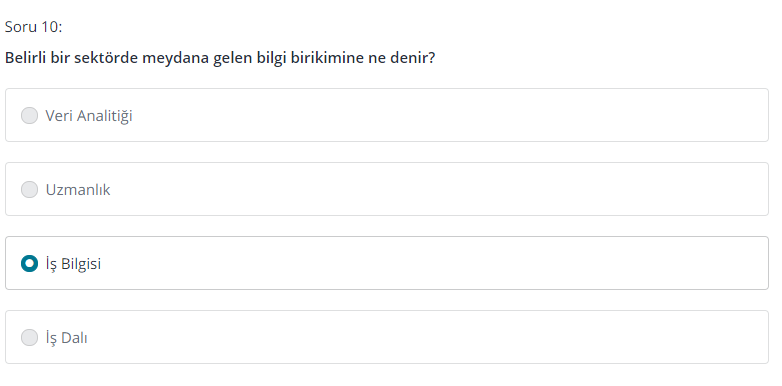




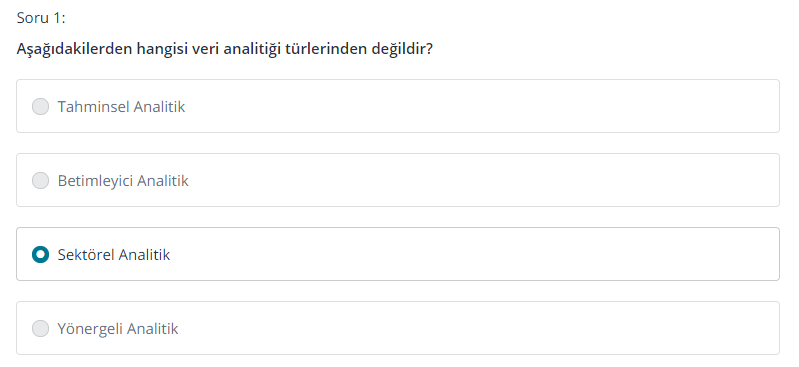


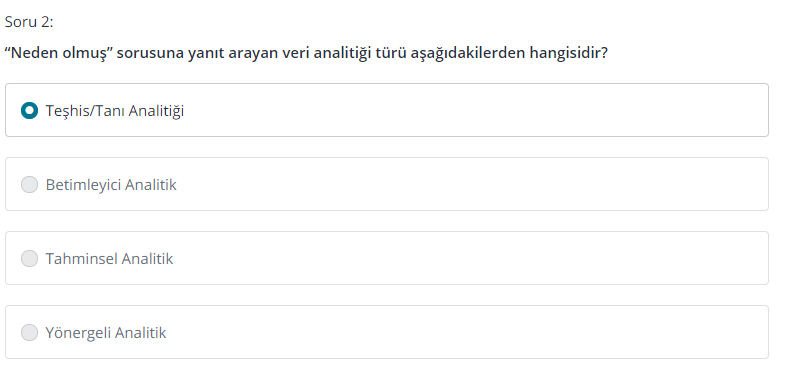


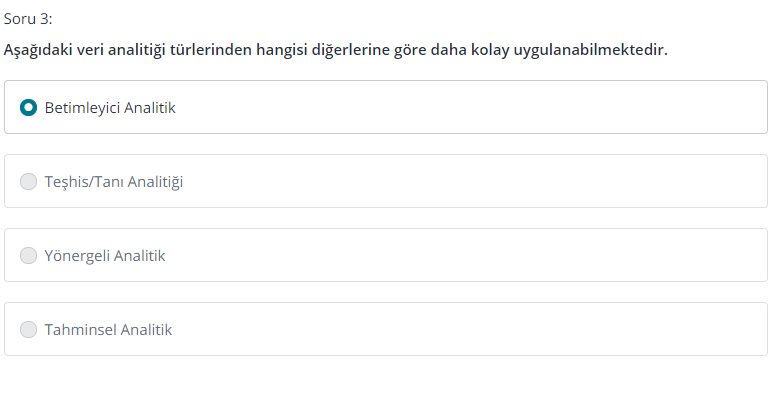


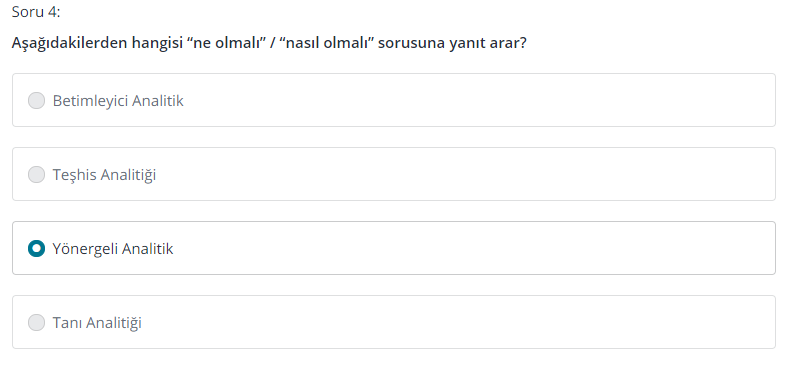
 

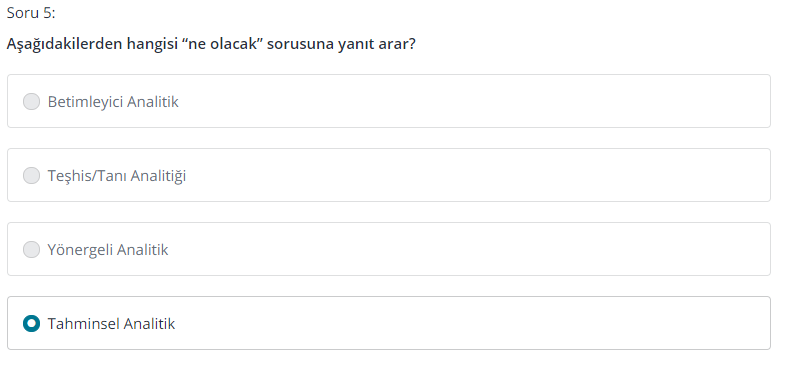
### Veri Bilimine Giriş Alıştırmalar – 2

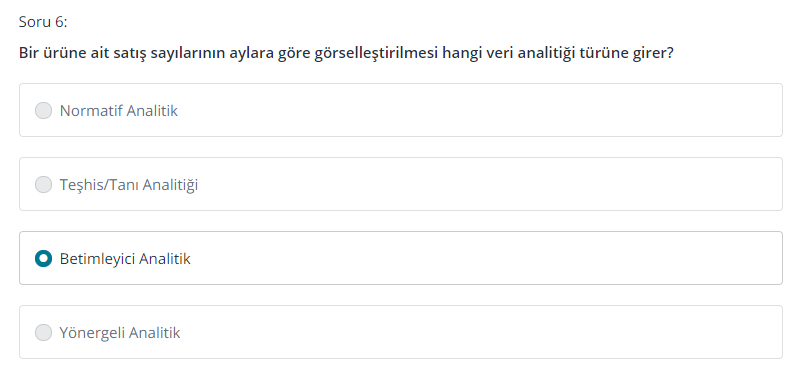


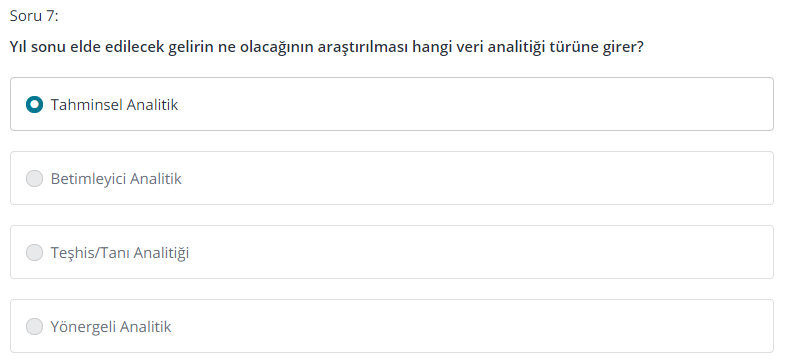


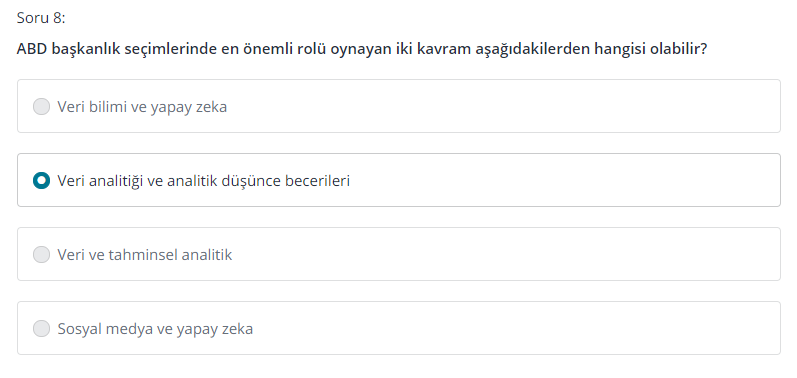


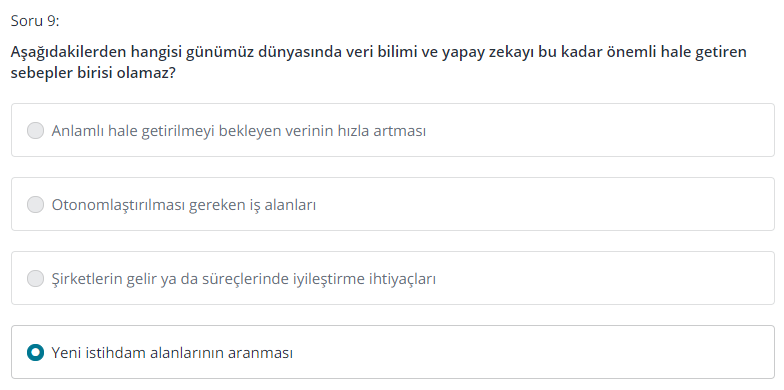


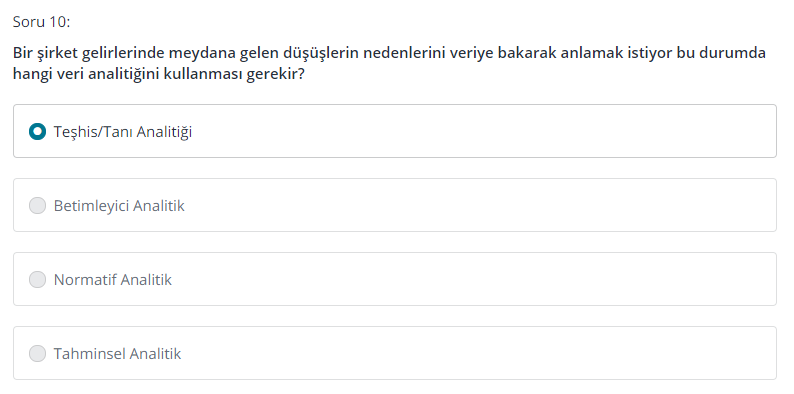












# Python Programlama

* Python, Google tarafından destekleniyor.
* Python’ın yorumlayıcı özelliği vardır. Etkileşim özelliğine sahiptir. (Soru-cevap mantığıyla çalışır.)
* High Level bir programlama dili.
* OPP (nesneye dayalı) ve FP(Fonksiyonel programlama).

## Temel Hareketler

* Seçili alanı F9 tuşu ile çalıştırabiliriz.
* Python programlama dilinde oluşturulan her şey bir nesnedir.
* Yorum satırı oluşturmak için satır başına # koyarız.

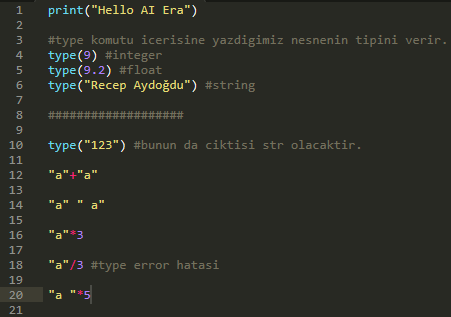
### Integer, Float ve String

Integer **=** 9 gibi ondalıksız sayılar.

Float **=** 9.2 gibi ondalıklı sayılar.

String **=** Karakter dizileri. “Çift tırnak” veya ‘Tek tırnak’ içinde yazılır.

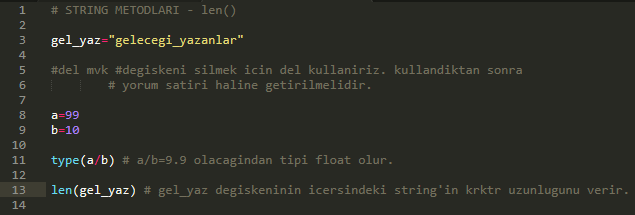
Type **=** type() içersine yazılan nesnenin tipini verir.



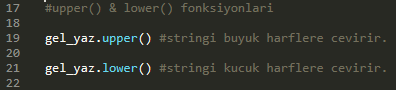
* “a”+”a” 🡪 aa
* “a””a” 🡪 aa
* “a”\*3 🡪 aaa
* “a”-“b” 🡪 TypeError alırız. Bu operatör sadece numeric ifadelerde kullanılır.
* “a”/3 🡪 TypeError

### String Metodları

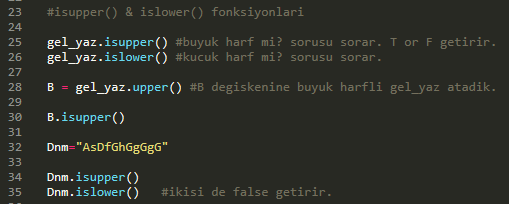
len()**=** içerisine yazılan değişkenin uzunluğunu verir.



upper() & lower() **=**



isupper() & islower() **=**



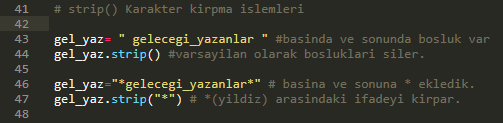
replace() **=**



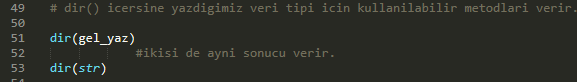
replace(“eski\_karakter”,”yeni\_karakter”)

gelecegi\_yazanlar 🡪gelecegi\_yızınlır

strip() **=** Karakter kırpma işlemleri



dir() **=**



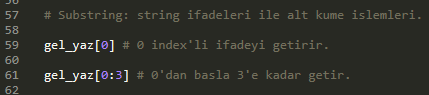
capitalize() **=** İlk harfi büyütür.

gel\_yaz.capitalized()

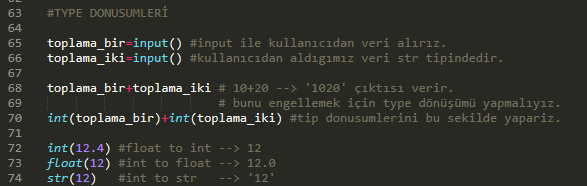
title() **=** Her kelimenin ilk harfini büyütür.

gel\_yaz.title()

Substring **=** Alt küme işlemleri



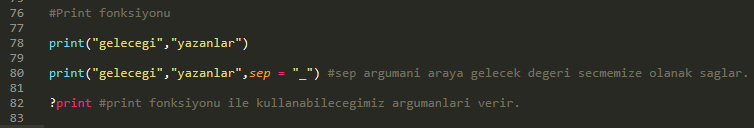
#### Type Dönüşümleri



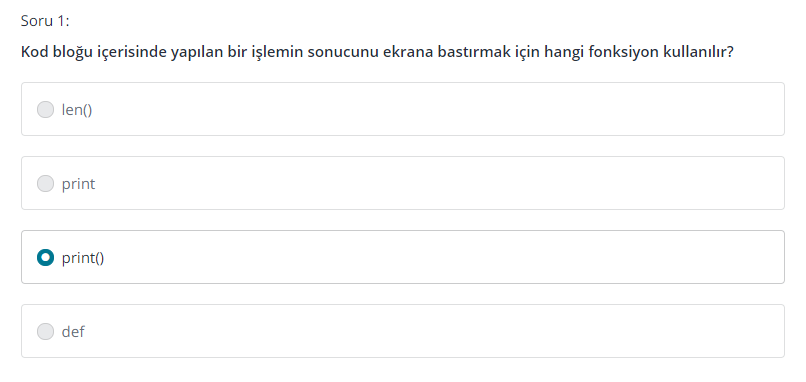
#### print() fonksiyonu

print(“gelecegi”,”yazanlar”) 🡪 gelecegi yazanlar

print(“gelecegi”,”yazanlar”,sep = (“\_”)) 🡪 gelecegi\_yazanlar

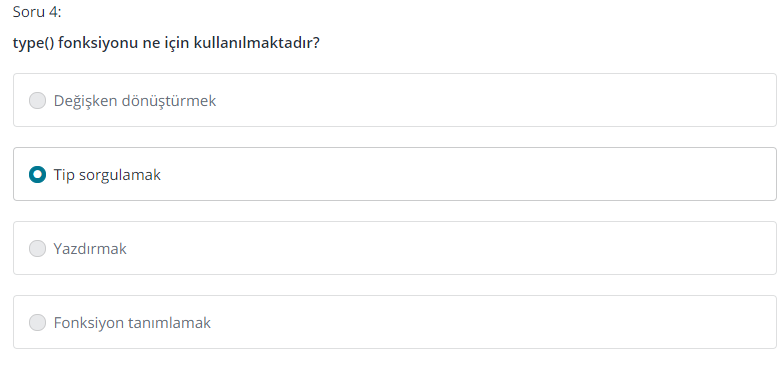


### Python Programlama Alıştırmalar – 1







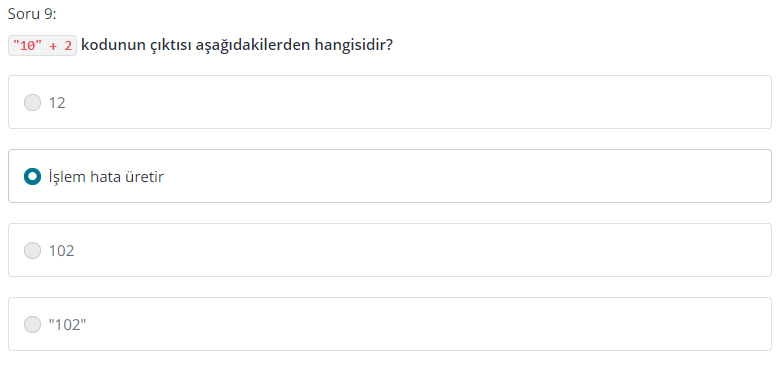








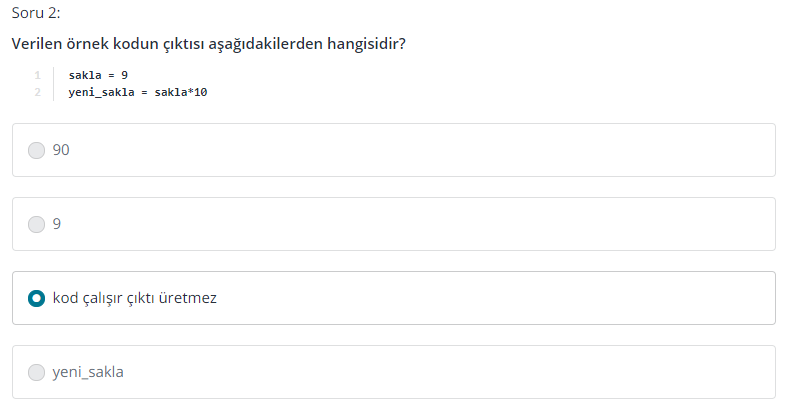


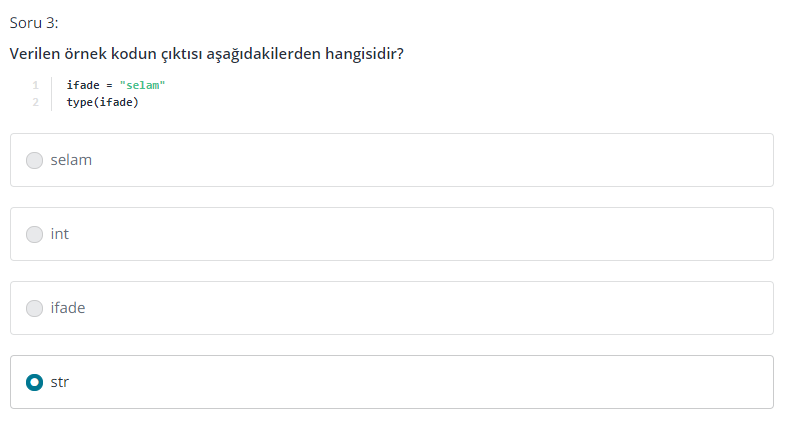




### Python Programlama Alıştırmalar – 2

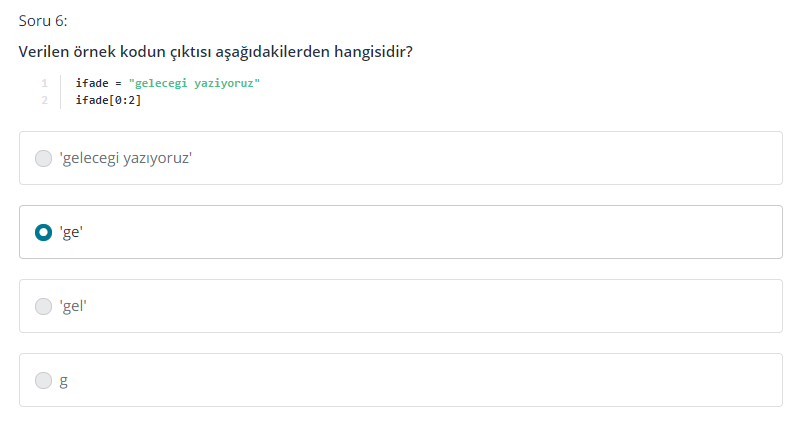


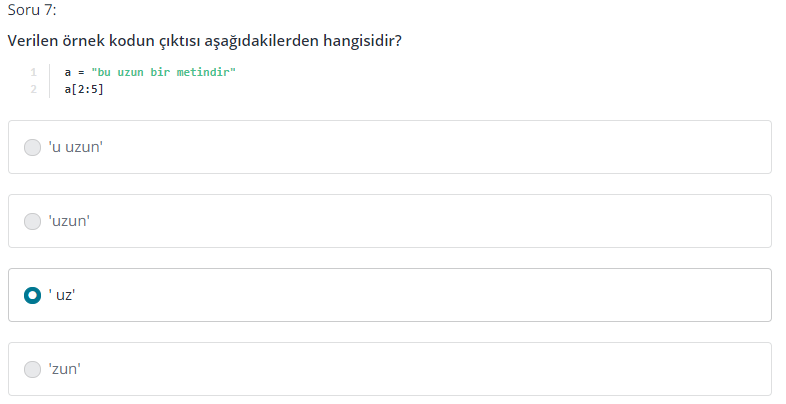




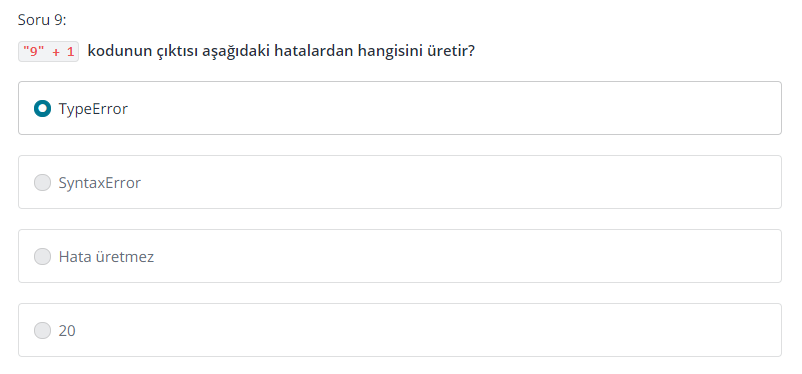


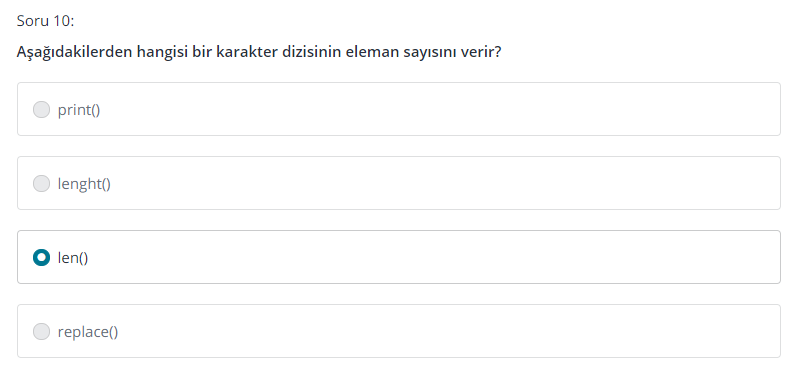




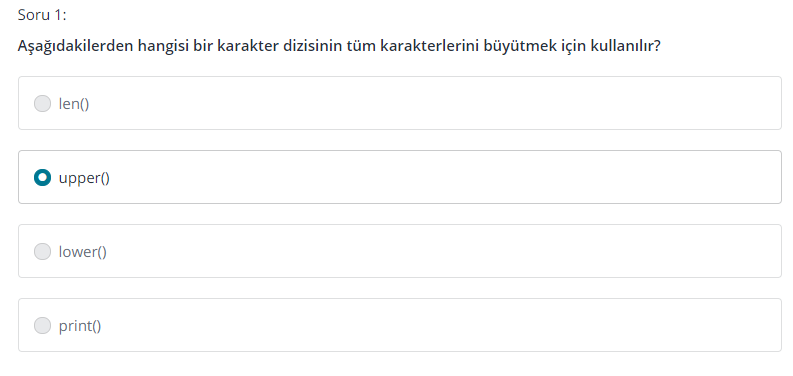


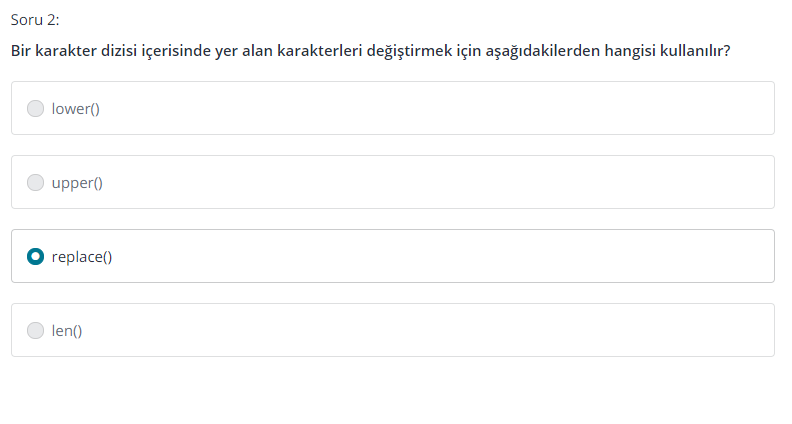


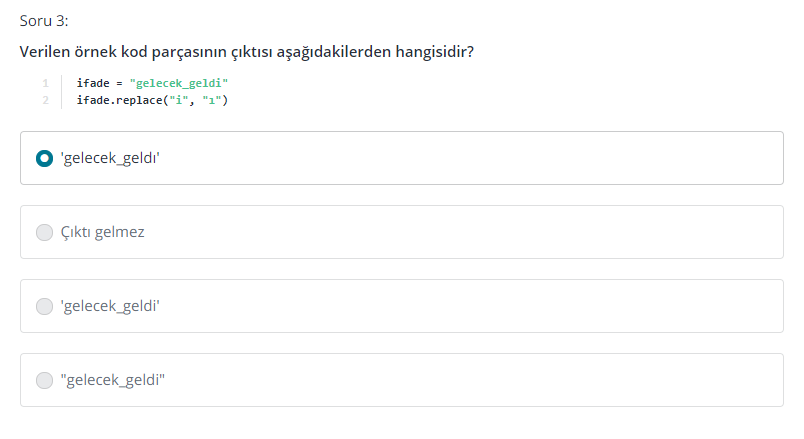


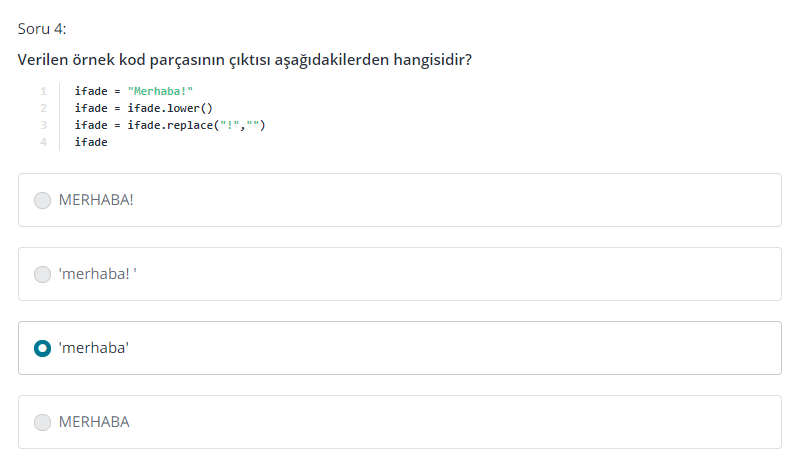


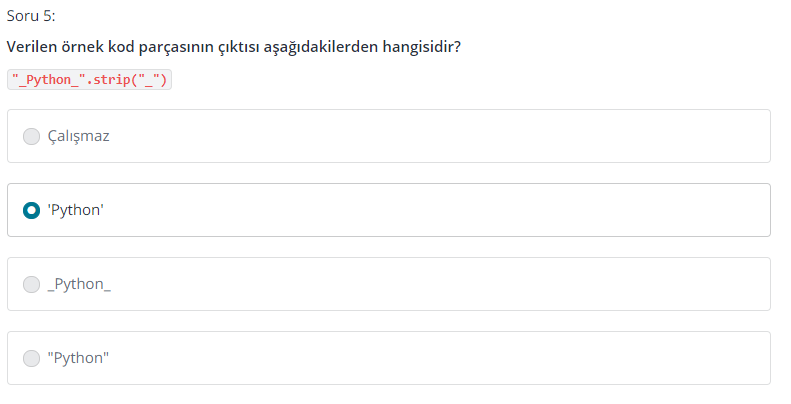
### Python Programlama Alıştırmalar – 3

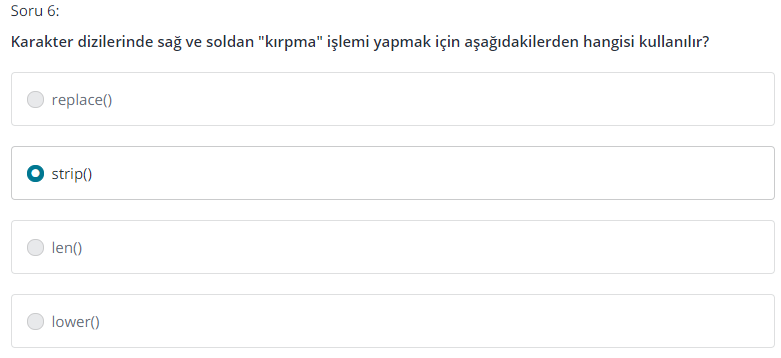


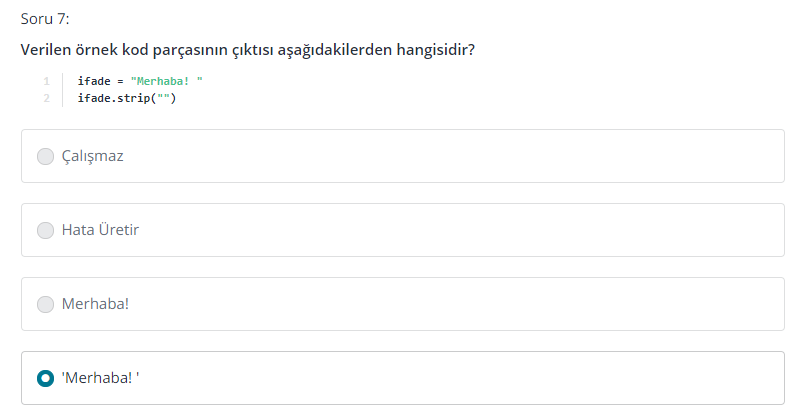


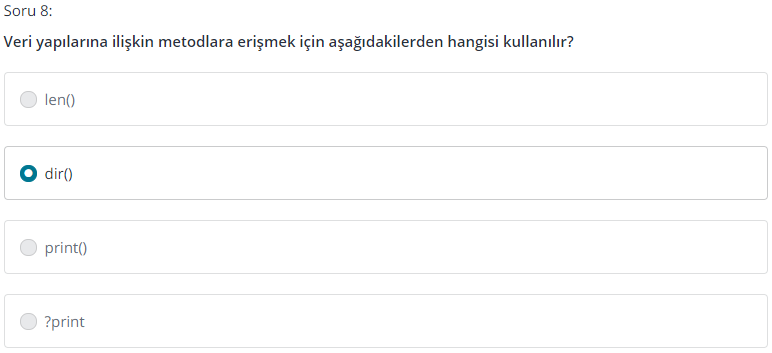


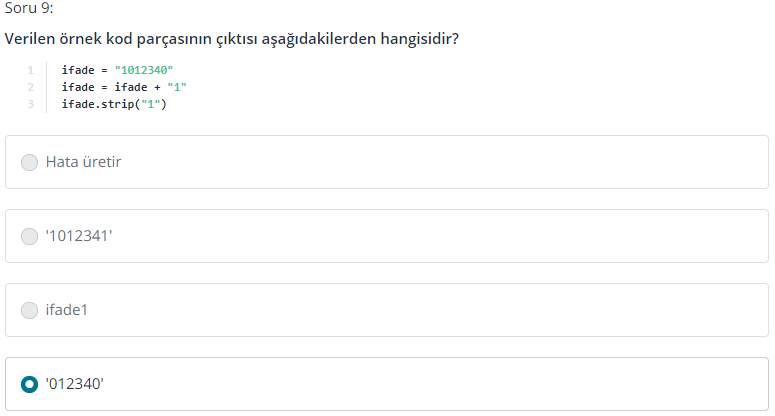
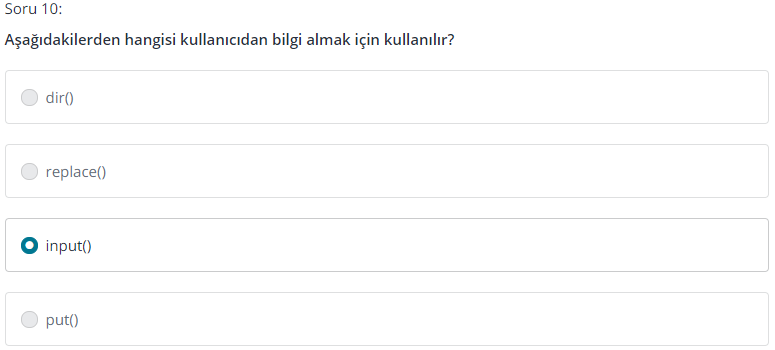










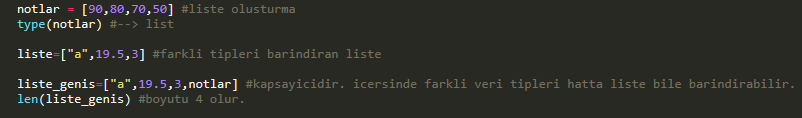
## Veri Yapıları (Data Types)

### Listeler

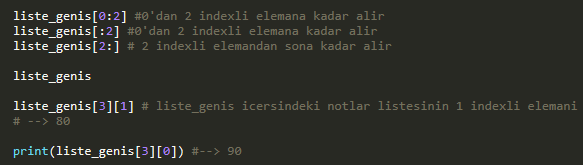
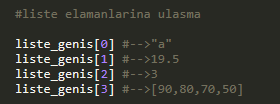
1. Değiştirilebilir
2. Kapsayıcıdır (Farklı tipte verileri tutabilir.)
3. Sıralıdır

Köşeli parantez [ ] ya da list() fonksiyonu ile liste oluşturabiliriz.

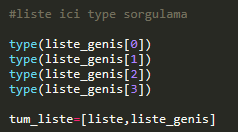
Liste bir üst type’dır içersinde farklı type’da veriler barındırabilir.



#### Liste Elemanlarına Ulaşma

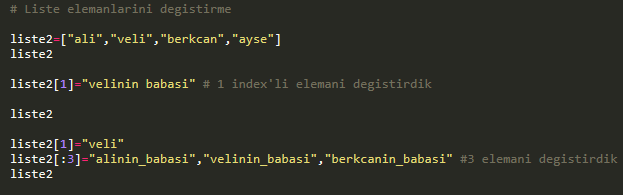


#### Liste İçi Type Sorgulama

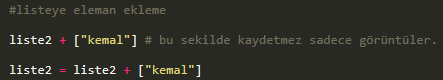


**del liste 🡪 liste’yi siler**

#### Liste elemanlarını değiştirme



#### Listeye eleman ekleme



#### Listeden eleman silme

del liste2[5] 🡪 5 index’li elamanı siler.

#### append ve remove metodlari

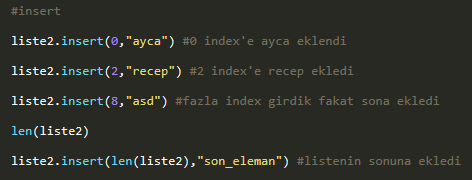
liste2.append("berkcan") 🡪sona ekleme yapar

liste2.remove("alinin\_babasi") 🡪silme yapar

liste2.remove("velinin\_babasi")

#### insert metodu

index’e göre ekleme yapar.



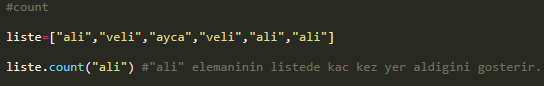
#### pop metodu

index’e göre silme yapar.

liste2.pop(0) #0 index degerli elemani siler

liste2.pop(1) #1 indexli elemani siler.

#### count metodu

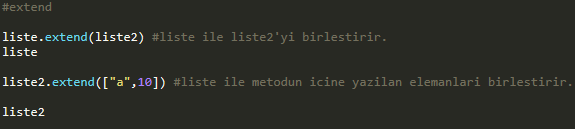


🡪 3

#### copy metodu

liste\_yedek=liste.copy() 🡪 liste'yi liste\_yedek'e kopyalar.

#### extend metodu

İki farklı listeyi birleştirir.

#### index metodu



#### reverse metodu

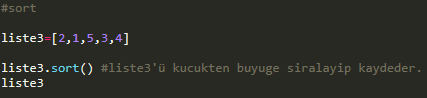
liste = [1,2,3]

liste.reverse() 🡪 liste elemanlarını ters sırayla kaydeder.

liste = [3,2,1]

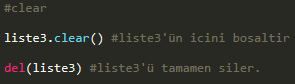
#### sort metodu

Elemanları küçükten büyüğe sıralar.



#### clear metodu

liste’nin içini boşaltır.



### Tuple (Demet)

1. Kapsayıcıdır
2. Sıralıdır
3. Değiştirilemez (Listeden farkı budur.)

#### Tuple Oluşturma



**NOT=** Tek elemanlı tuple oluştururken sonuna virgül koymalıyız. Aksi takdirde tuple oluşturmak istediğimiz anlaşılamaz.

Örneğin; t = (“eleman”,)

#### Eleman İşlemleri

Tuple’larda eleman işlemleri listeler ile birebir aynıdır. (index’e göre erişim vs.)

t=(1,2,3,4)

t[0] 🡪 1

t[-1] 🡪4 (sondan birinci eleman demektir.)

### Dictionary (Sözlük)

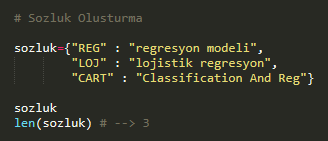
1. Kapsayıcıdır
2. Sırasızdır 🡪 Listelerden farkı budur.
3. Değiştirilebilirdir.

#### Dictionary Nedir?

Key’ler ve bu key’lerin karşılıklarının bir arada tutulduğu veri yapısıdır.

Listelerde olduğu gibi index’leme yapılmaz.

#### Dictionary Oluşturma



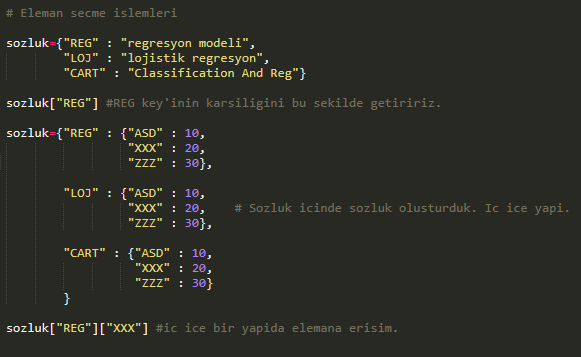
{“key” : “key’in karşılığı”}

**NOT=** Sözlüklerde key’ler sadece sabit veri yapılarından oluşabilir. list gibi yapılardan olamaz. String ve sayılar sabit ver yapılarıdır.

Sabit veri yapısı değiştirilemez demektir. Tuple’da buna dahildir.

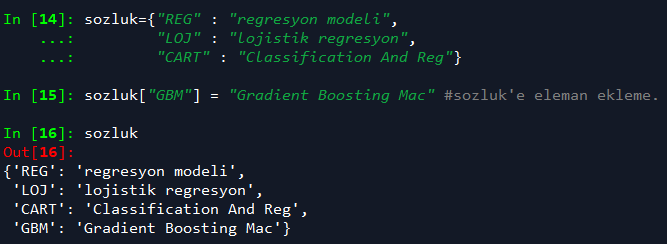
t = (“tuple”,) 🡪 sozluk = { t : “tuple’dan key olur” }

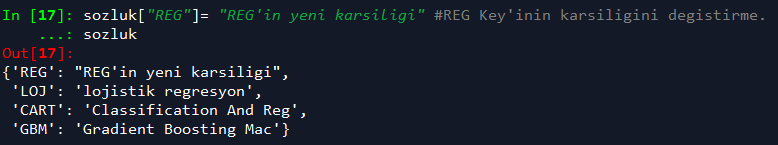
#### Eleman Seçme İşlemleri



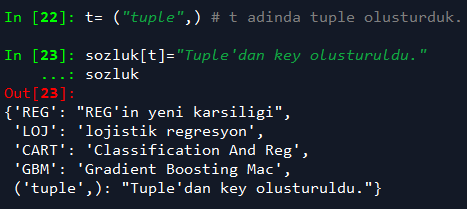


#### Eleman Ekleme & Değiştirme





REG key’i olmasaydı yeni key oluşturulacaktı.



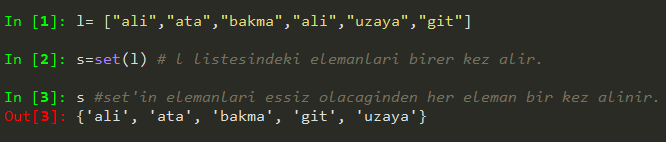
### Sets (Kümeler)

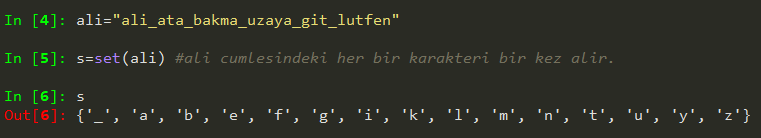
1. Sırasızdır (Index değerleri yok.)
2. Değerleri eşsizdir. (Tekrar eden değeri olmaz.)
3. Değiştirilebilir.
4. Kapsayıcıdır. Farklı türden veri yapıları barındırabilir.

Set’ler performans odaklı veri tipleridir. Programlama anlamında biraz daha hız istediğimizde kullanılır. Matematiksel anlamda bu veri yapıları kümelere benzer.

#### Set Oluşturma

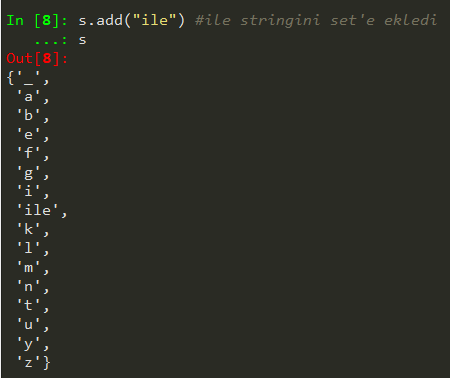
s = set() 🡪 s isminde bir set oluşturuldu.

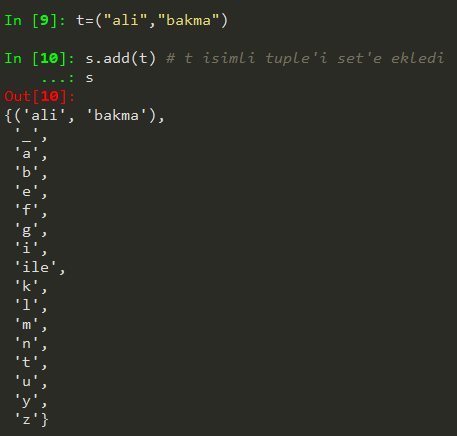


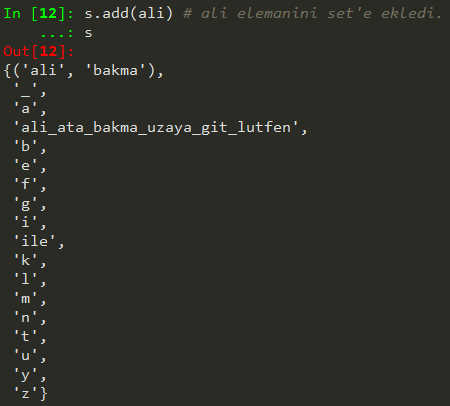


#### Set’lere eleman ekleme ve çıkarma işlemleri

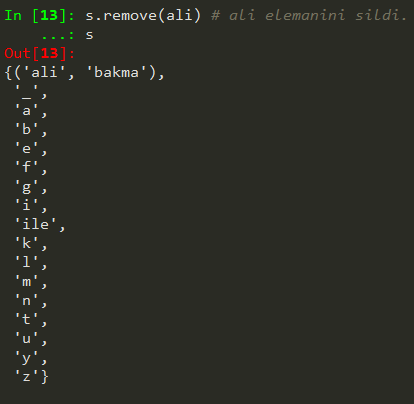
add() fonksiyonu ile ekleme yaparız.

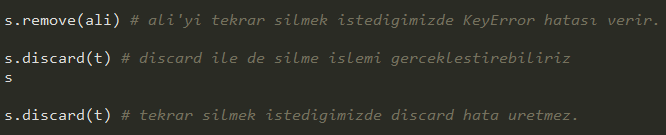






remove() fonksiyonu ile set’lerden eleman silebiliriz.

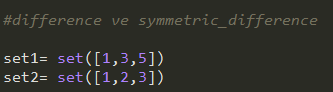




#### Set’lerde Fark İşlemleri

**difference & symmetric\_difference**

**difference =** kümelerin farkını verir.



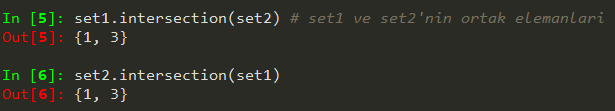
**symmetric\_difference =** ikisinde de ortak olmayan elemanları verir.



#### Set’lerde Kesişim ve Birleşim İşlemleri

**intersection & union & intersection\_update**

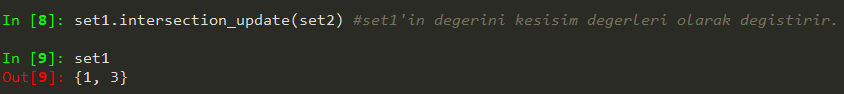
**intersection =** kesişim



**union =** birleşim



**intersection =** set1’in değerini kesişim değerleri olarak değiştirir.



#### Set’lerde Sorgu İşlemleri

**isdisjoint & issubset & issuperset**

**isdisjoint =** Ayrık küme mi?

İki kümenin kesişiminin boş olup olmadığını sorgular.

Boş ise True değil ise False döndürür.



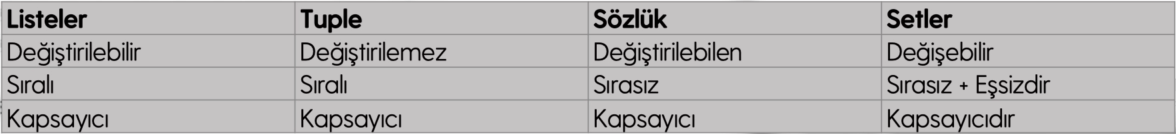
**issubset =** subset’i mi? Alt kümesi mi? sorgusunu yapar.



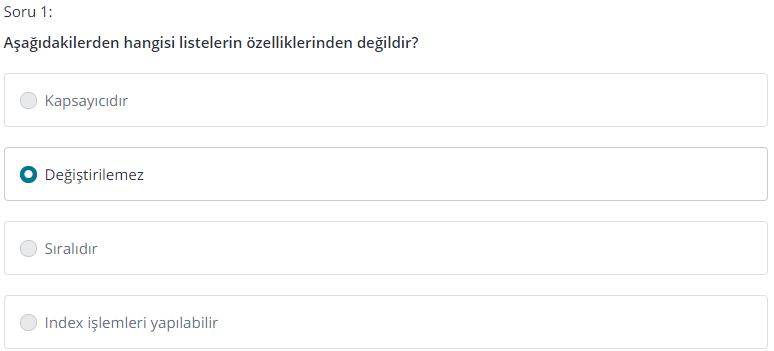
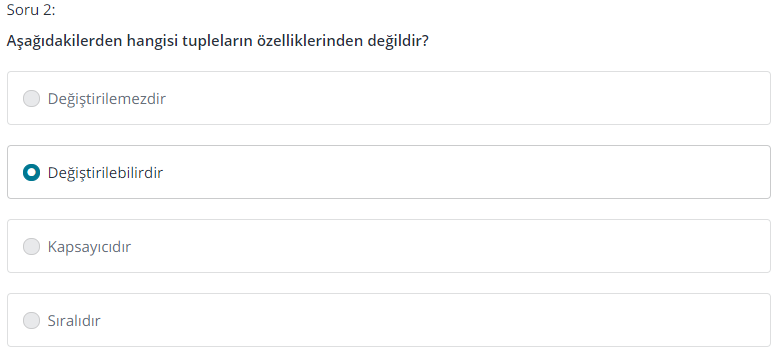
**issuperset =** Kapsar mı?

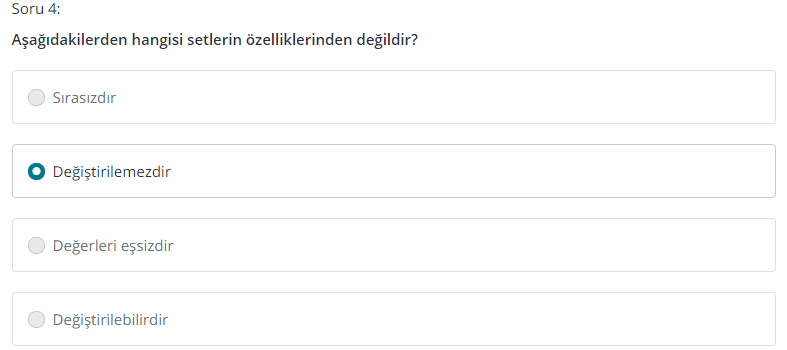
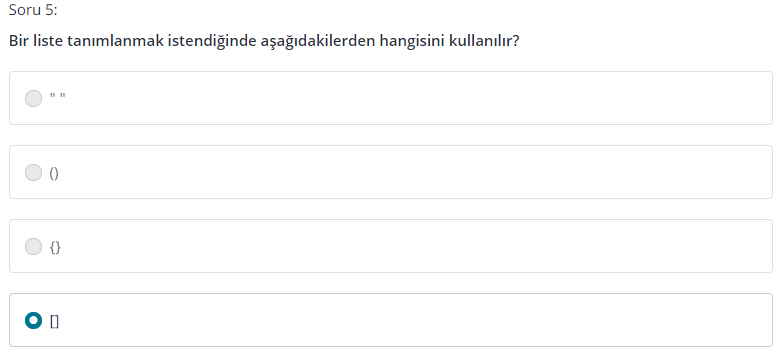
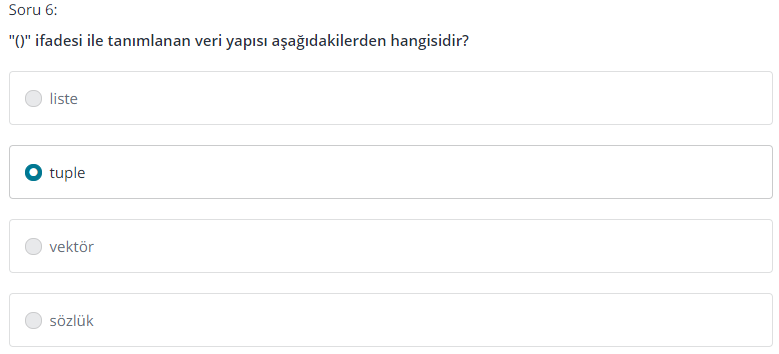


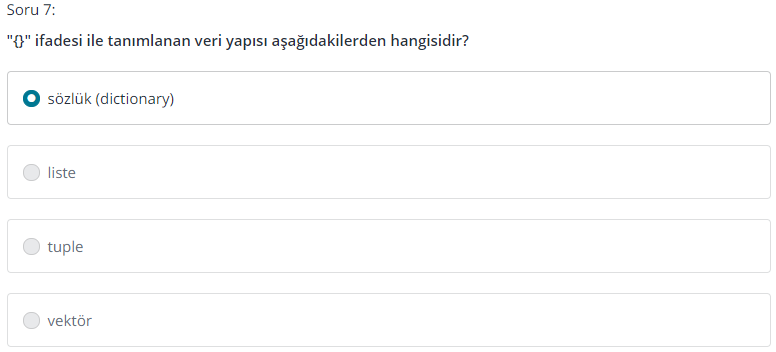
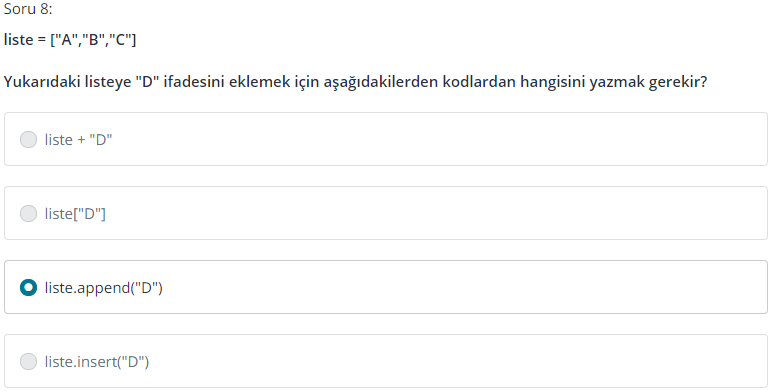
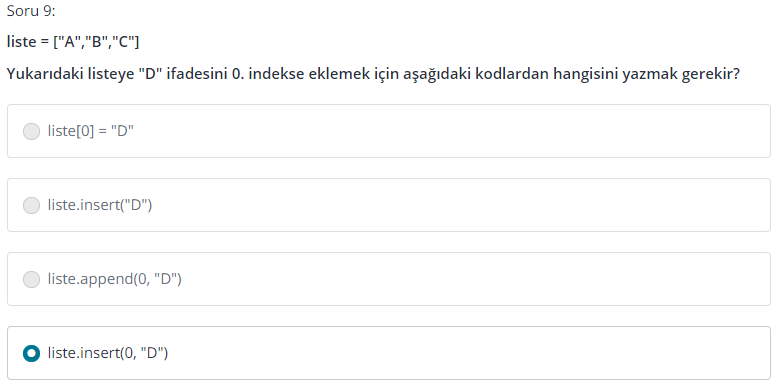
### Veri Yapıları Özet

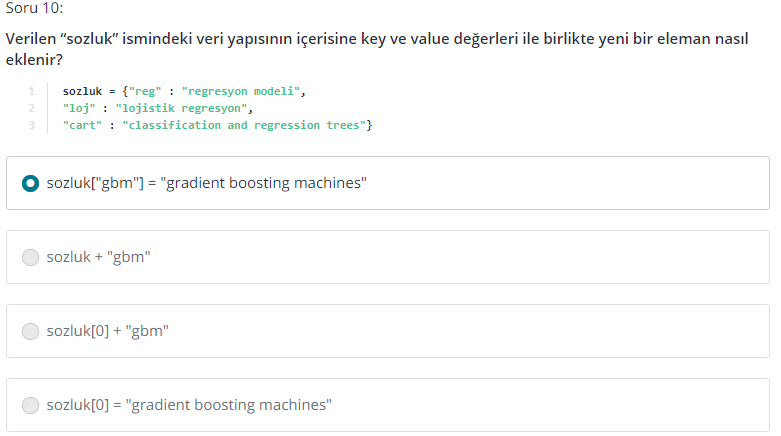


### Python Programlama Alıştırmalar – 4

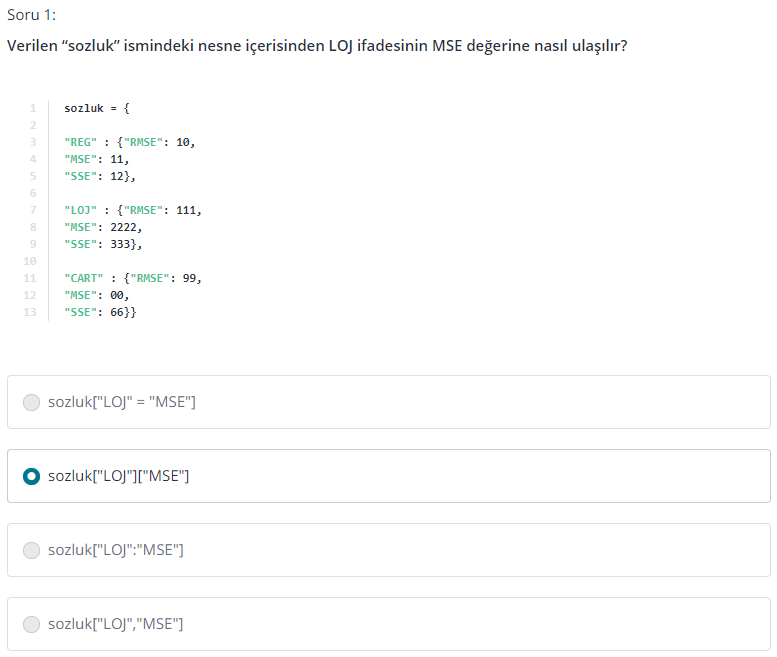
  

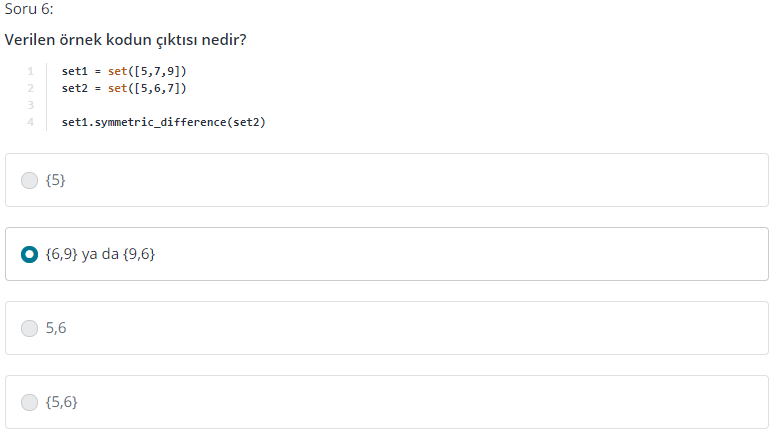
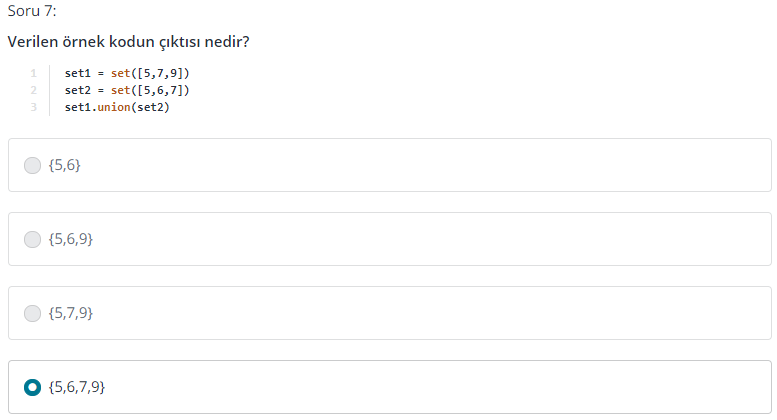


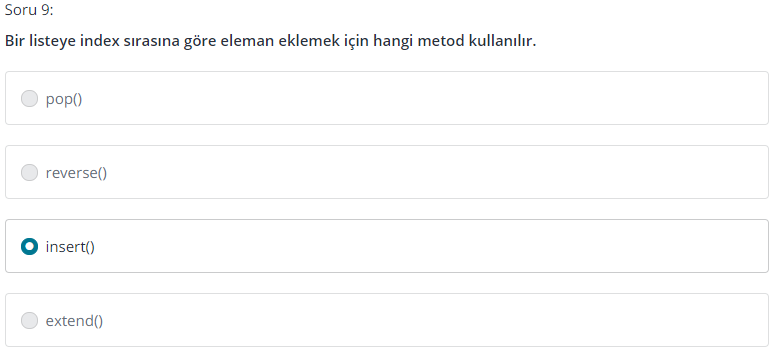
### Python Programlama Alıştırmalar – 5

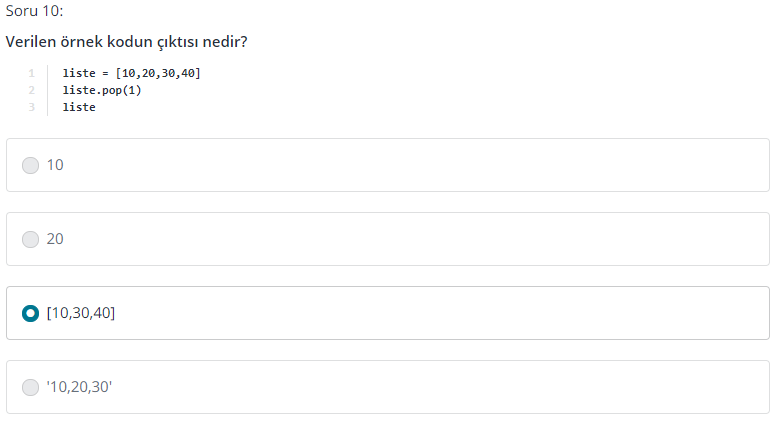


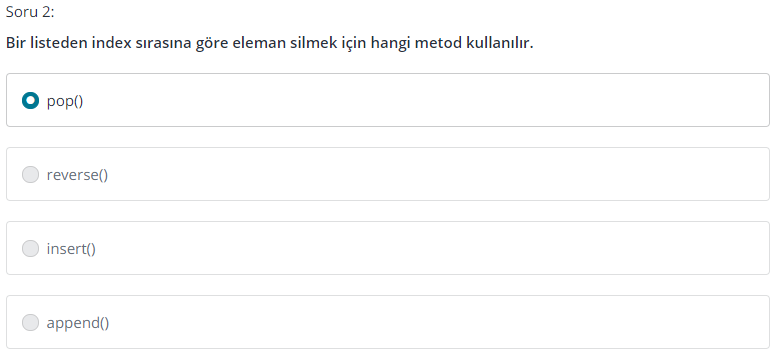
 

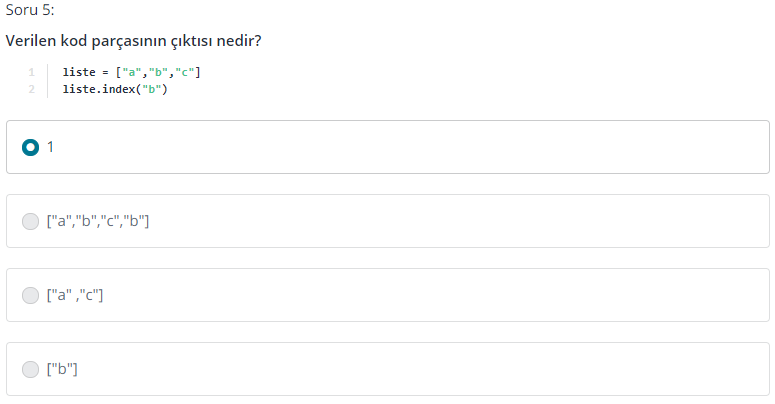


### Python Programlama Alıştırmalar – 6

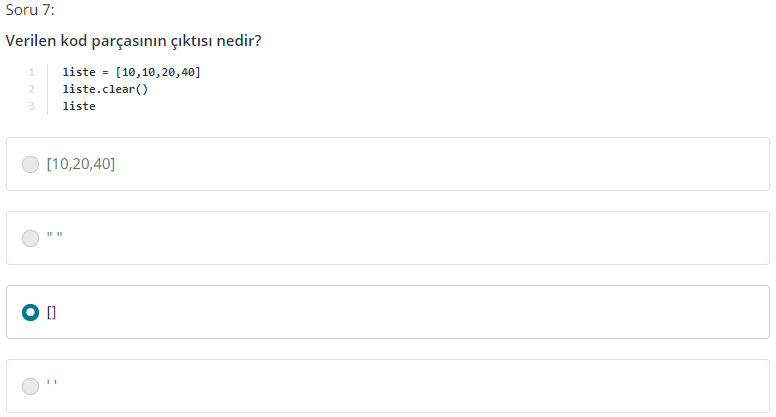
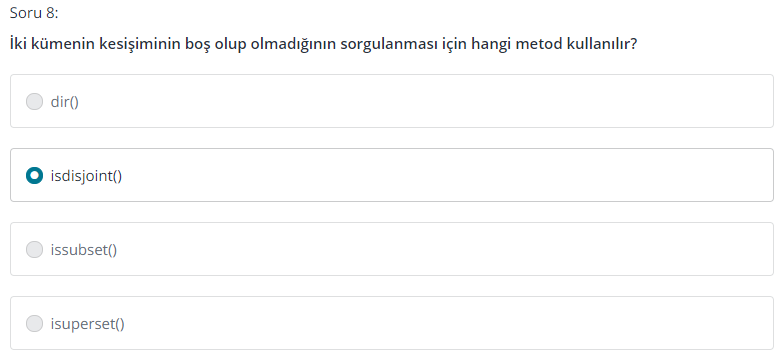
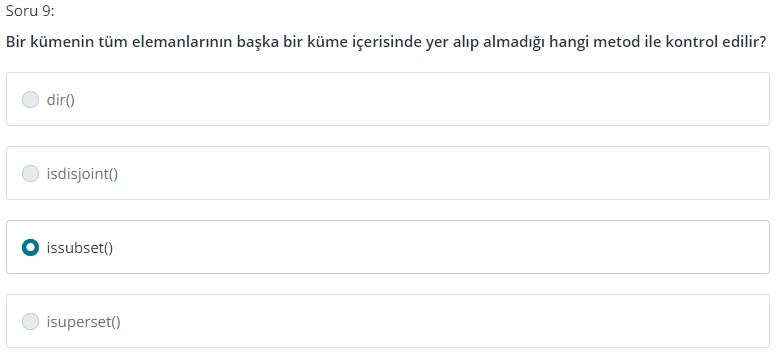


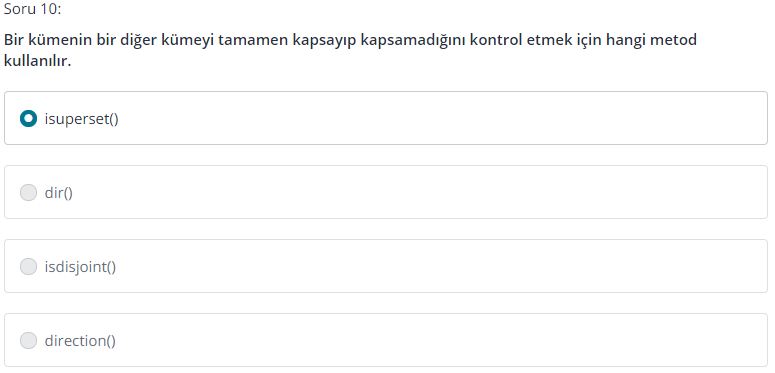
 







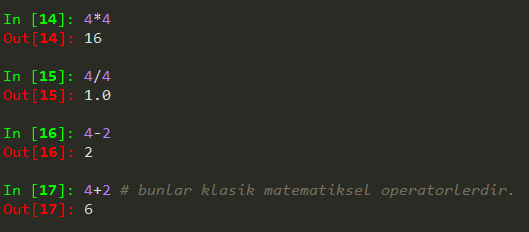


## Fonksiyonlar

### Fonksiyon Nedir?

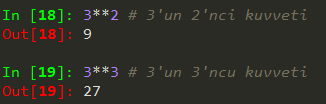
Belirli amaçları yerine getiren işleçlerdir.

### Matematiksel İşlemler



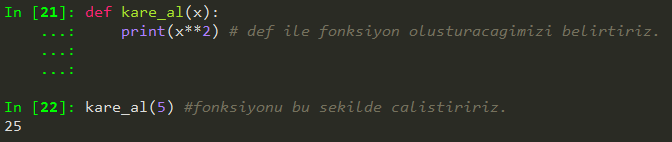
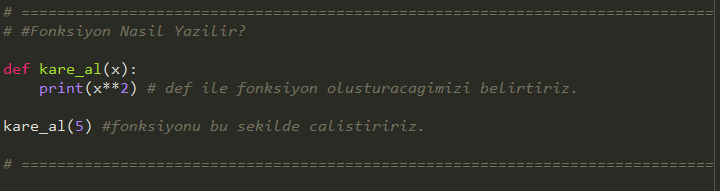
#### Üs Alma

3\*\*2 🡪 32 anlamına gelir.

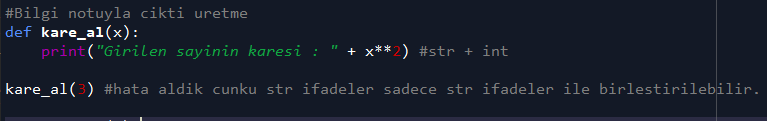


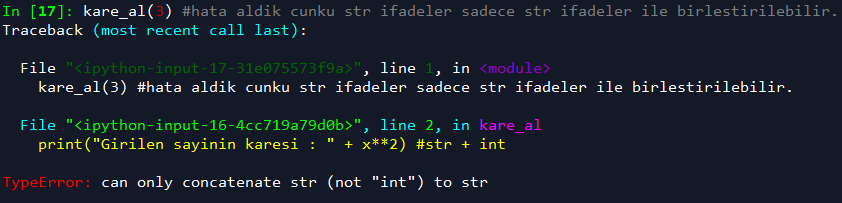
### Fonksiyon Nasıl Yazılır ?

def ile fonksiyon oluşturacağımızı belirtiriz.



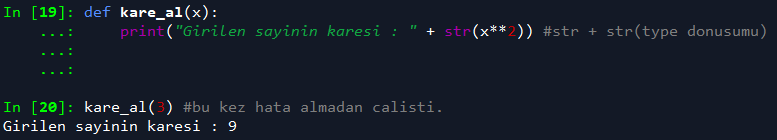
### Bilgi Notuyla Çıktı Üretmek

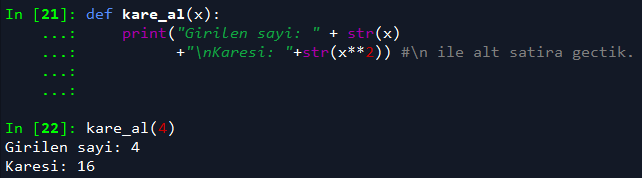


Bu fonksiyonu çalıştırınca aldığımız hata : 

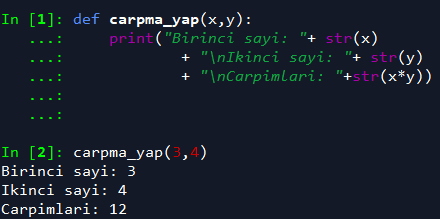
str ifadeler ile sadece str ifadeler birleştirilebilir!

type dönüşümü yapmalıyız.:



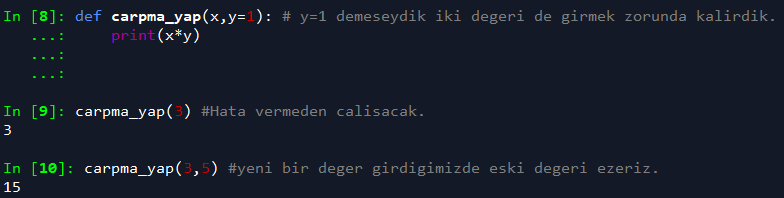
Başka bir örnek: 

### İki Argümanlı Fonksiyon Tanımlamak



### Ön Tanımlı Argümanlar

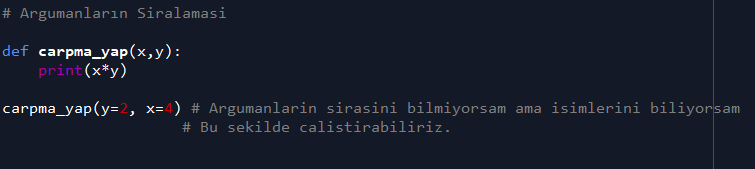
Print() fonksiyonundan hatırlayacağımız gibi sep() ve end() gibi argümanlardır.



y=1 yazarak ön tanımlı bir argüman oluşturmuş olduk.

#### Argümanların Sıralaması

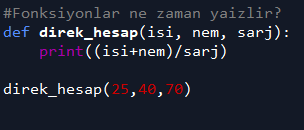
Argümanların sırasını bilmediğimiz fakat isimlerini bildiğimiz zaman aşağıdaki şekilde çalıştırabiliriz.



### Ne Zaman Fonksiyon Yazılır?

Fonksiyonlar programlama dilleri içerisinde tekrar eden görevleri yerine getirmek ve var olan işleri daha programatik bir şekilde gerçekleştirmek için kullanılır.

Örneğin bir şehirde binlerce sokak lambası var ve bu sokak lambaları için ısı, nem, şarj değerlerini kullanarak bir hesaplama yapmamız gerekiyor. Her lamba için tek tek hesap mı yapacağız?

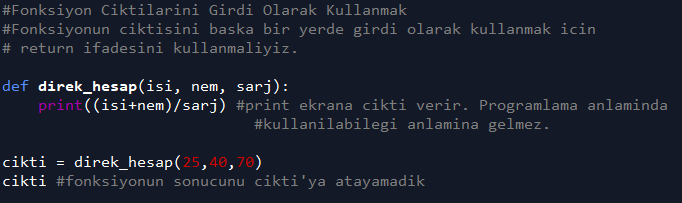
Hayır, fonksiyonu bir kez yazıp her lambada o fonksiyonu kullanacağız. 

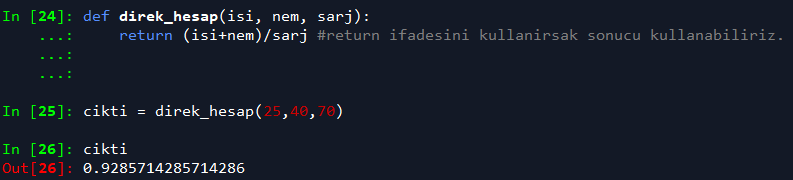
### Fonksiyon Çıktılarını Girdi Olarak Kullanmak

Yazdığımız bir fonksiyonun çıktısını başka bir yerde girdi olarak kullanmak istiyorsak return ifadesini kullanmalıyız.

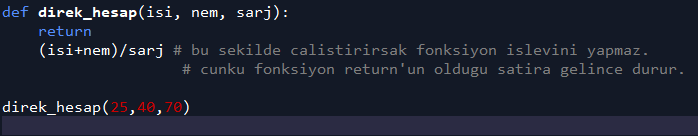
print() ekrana çıktı verir. Programlama anlamında kullanılabileceği anlamına gelmez.

Aşağıdaki örnekte görebiliriz.





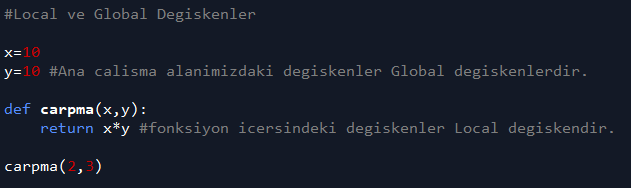
Fonksiyon return ifadesine gelince durur:



### Local ve Global Değişkenler

Ana çalışma alanımızdaki değişkenler Global değişkenlerdir.

Her hangi bir fonksiyonun ya da döngünün etkisindeki değişkenler ise Local değişkenlerdir.

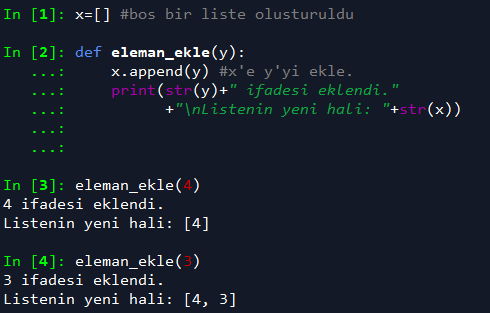


### Local Etki Alanından Global Etki Alanını Değiştirme

Yazmış olduğumuz bir döngü içerisinde ya da tanımlamış olduğumuz bir fonksiyon içerisinden global değişkenlerin değerlerinde değişiklik yapmak istediğimiz zaman ne yapmamız gerekiyor ?

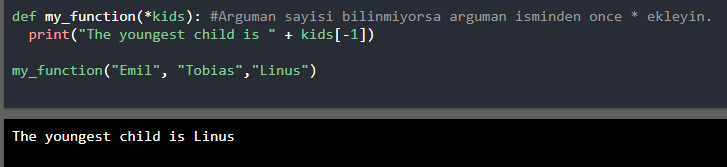
Python öncelikle local etki alanındaki değişkenleri tarar, arar ve bulmaya çalışır.

Örneğin bir fonksiyon yazdığımızda değişiklik yapmak istediğimiz değişkeni öncelikle kendi içersinde (local’de )arar, bulamazsa global alana çıkacak. Global alanda o değişkeni bulursa ona etki edecek (Orada da bulamazsa hata üretecek.). Aşağıdaki örnekte bu durumu gözlemleyebiliriz.



**NOT=**

Argüman sayısı bilinmiyorsa argüman isminden önce \* ekleyin



## Karar-Kontrol Yapıları (Koşullar)

### Koşul Nedir?

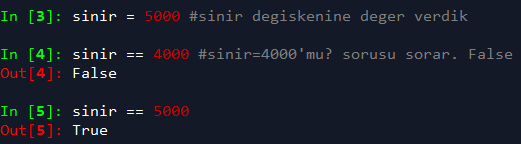
Örneğin günlük hayatta da kullandığımız gibi;

* Yağmur yağarsa şemsiye al
* Kar yağarsa zincir tak

gibi bazı olaylar gerçekleştiğinde bazı olayların gerçekleşmesi gerektiğini programlama diline ifade etmenin yollarıdır.

### True – False Sorgulamaları (Boolean)

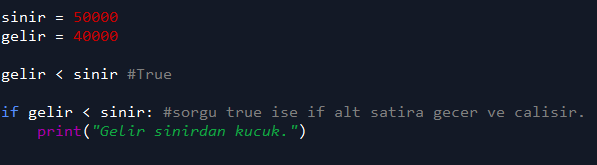
Doğru mu? sorusu sorar. == ile kullanırız.



### if – else – elif

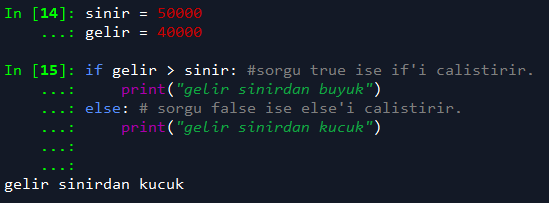
if eğer anlamındaki koşuldur.

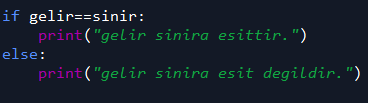
Eğer yazdığımız sorgu true ise alt satıra geçer ve çalışır.



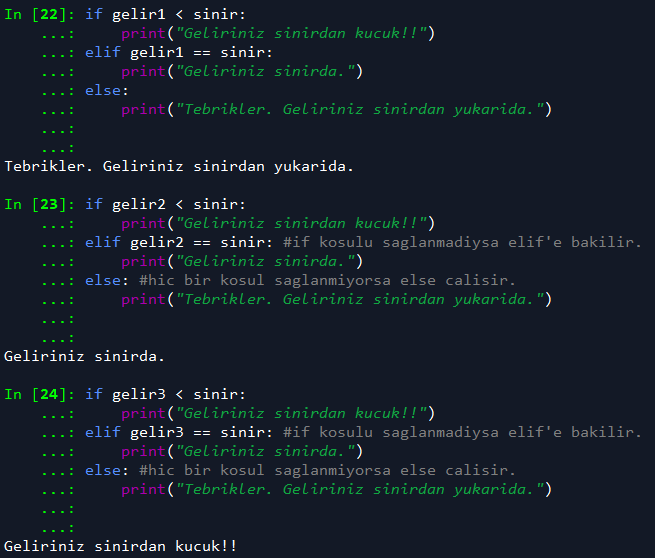
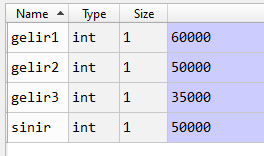
if = eğer true ise if çalışır.

else= değilse else çalışır.



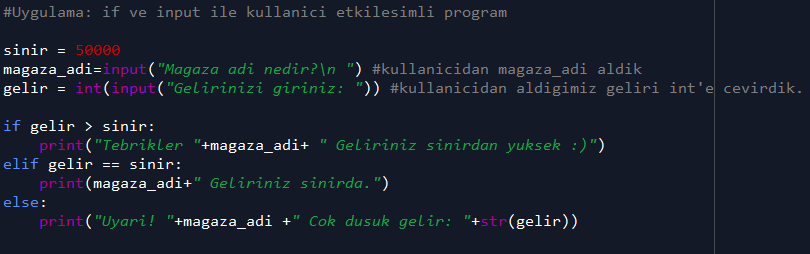


elif= if koşulu sağlanmazsa elif’e bakılır. elif koşulu da sağlanmazsa else çalışır.

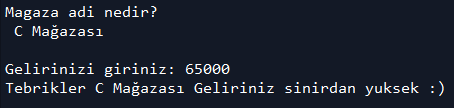
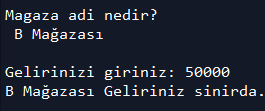
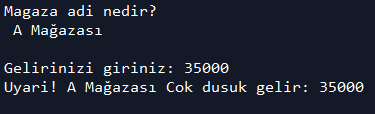


### **Uygulama:** if ve input ile kullanıcı etkileşimli program

Kullanıcıdan mağaza adı ve gelir bilgilerini alalım. Sınır değeri ile gelir değerini karşılaştıralım. Düşük, eşit, yüksek seviyelerine göre 3 farklı sonuç üretelim.



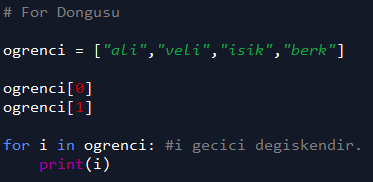
Program çıktıları:



## Döngüler

### For Döngüsü

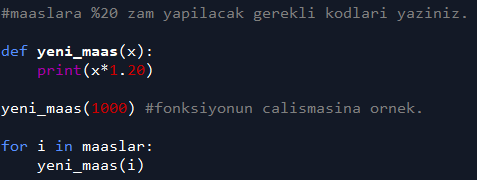
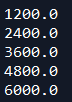
Örneğin bir liste içerisindeki elemanlara işlem yapmak istediğimizde o elemanlara tek tek gitme işlemini gerçekleştiren yapılara döngüler denir.

🡪

### Döngü ve Fonksiyonların Birlikte Kullanımı

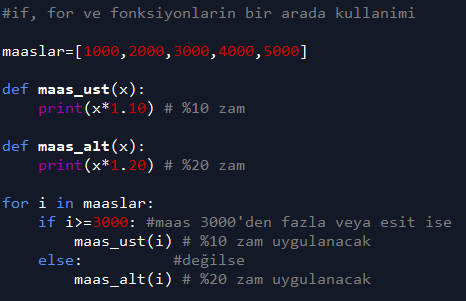


Maaşlara %20 zam yapılacak. Gerekli kodlar nelerdir?

🡪

### **Uygulama:** if, for ve fonksiyonların birlikte kullanımı

Az önceki uygulamadaki maaş listesi kullanılarak; maaşı 3000 tl’den yüksek olanlara %10 zam, maaşı 3000 tl’den az olanlara ise %20 zam yapılacak.

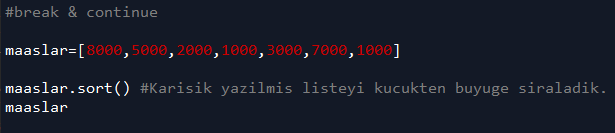
🡪

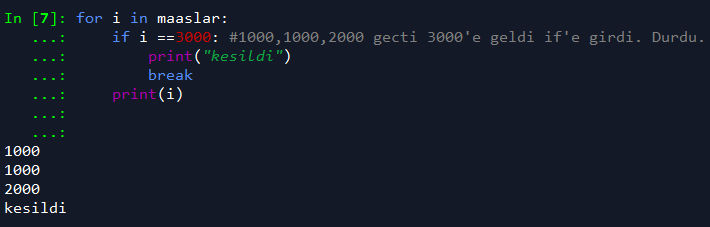
### break & continue

Döngüler içersinde belirli bir şartı sağlayan ifadeler yakalandığında (if döngüsü ile yakalıyorduk.) döngü bitirilmek istenebilir. Ya da bu şartı sağlayan eleman görmezden gelinmek istenebilir.

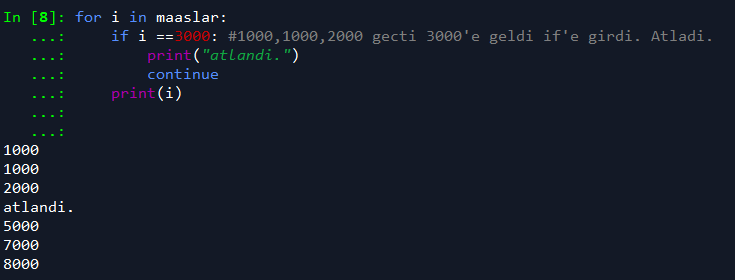
Bu gibi durumlarda break ve continue ifadeleri kullanılır.

Örneğin; maaşı 3000 tl’ye kadar olanlarla ilgilendiğimizi düşünelim.



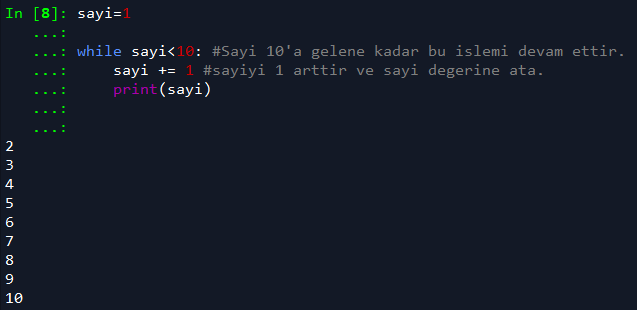


Örneğin; 3000’i atlayıp devam etsin.

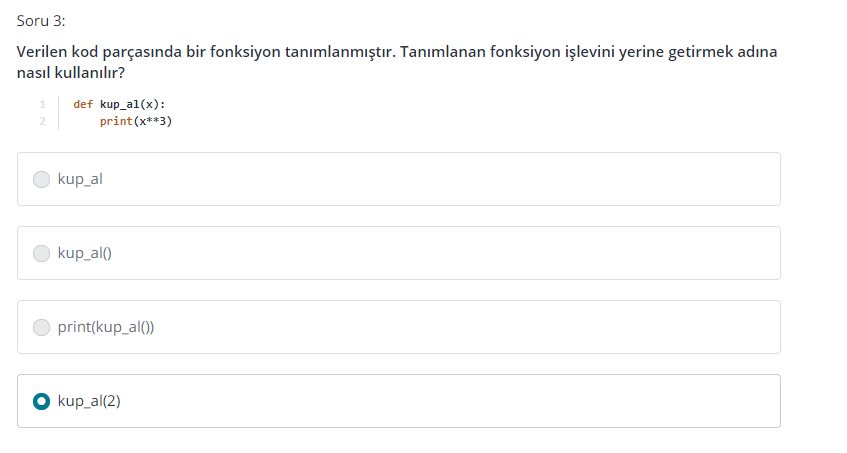
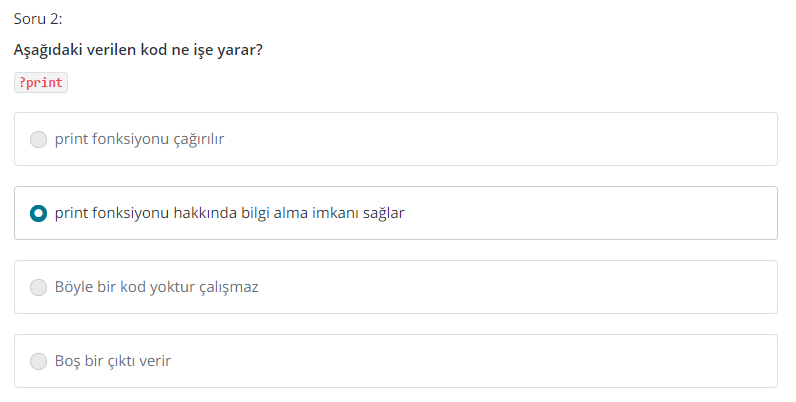
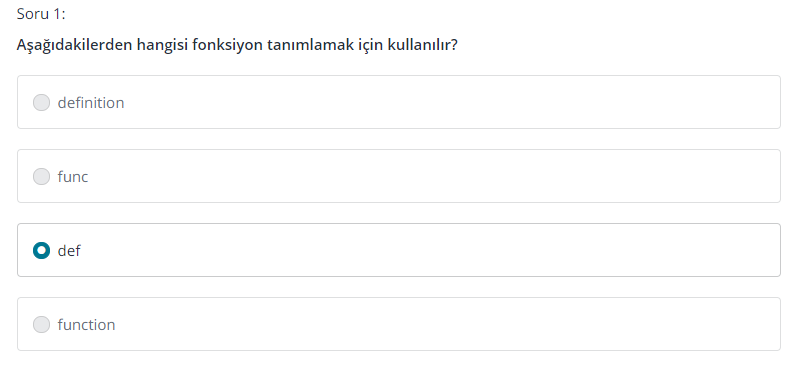


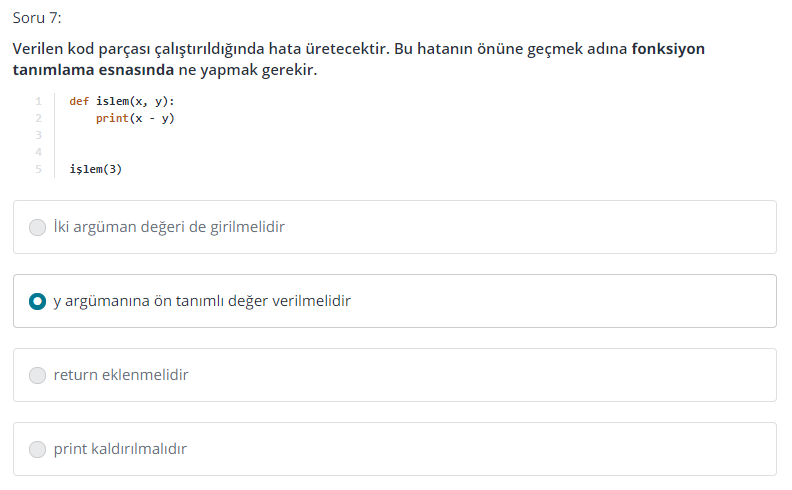
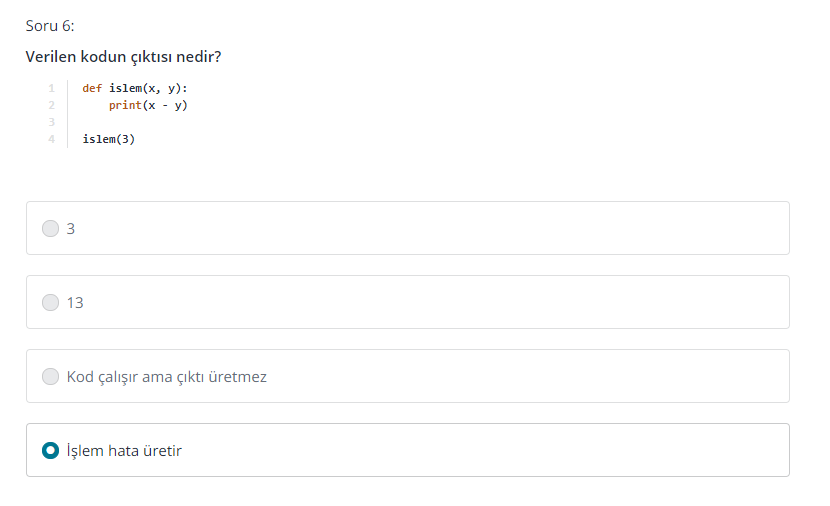
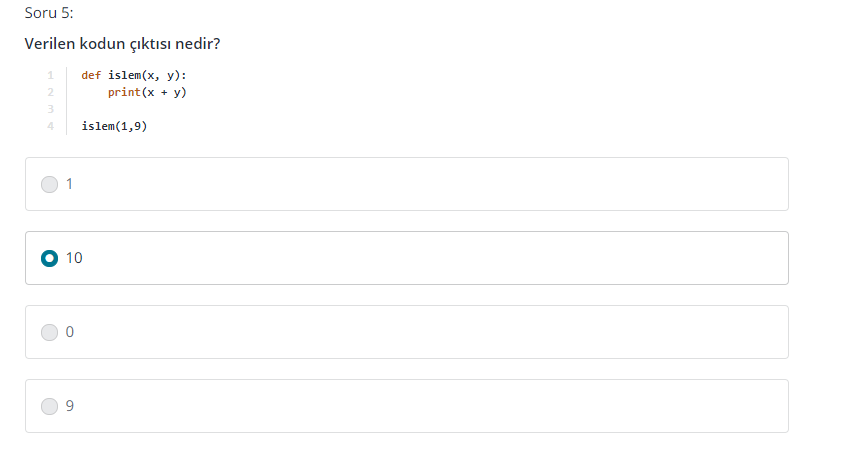
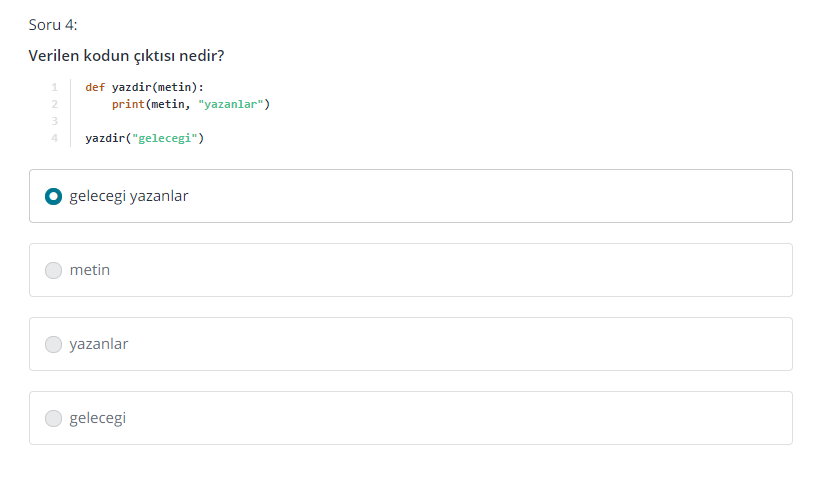
### while

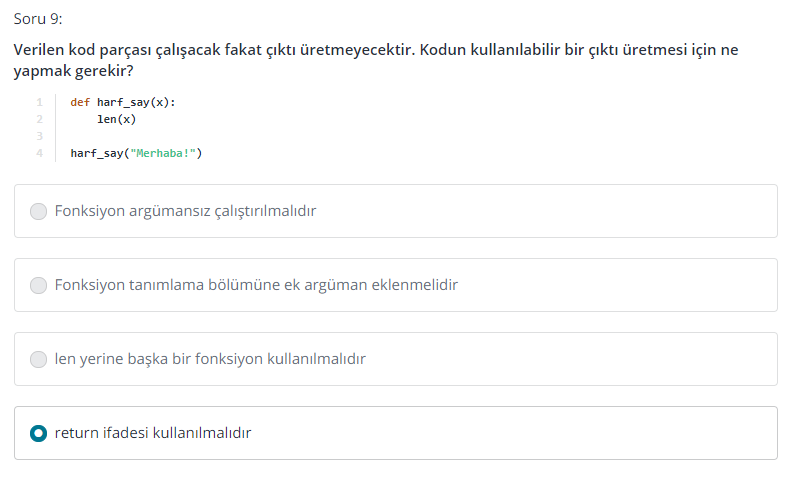
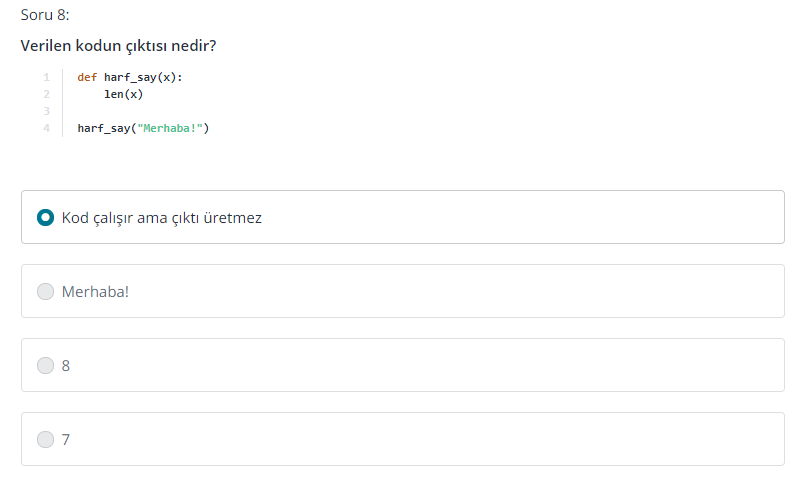
Şart sağlandığı sürece devam eden bir döngüdür.

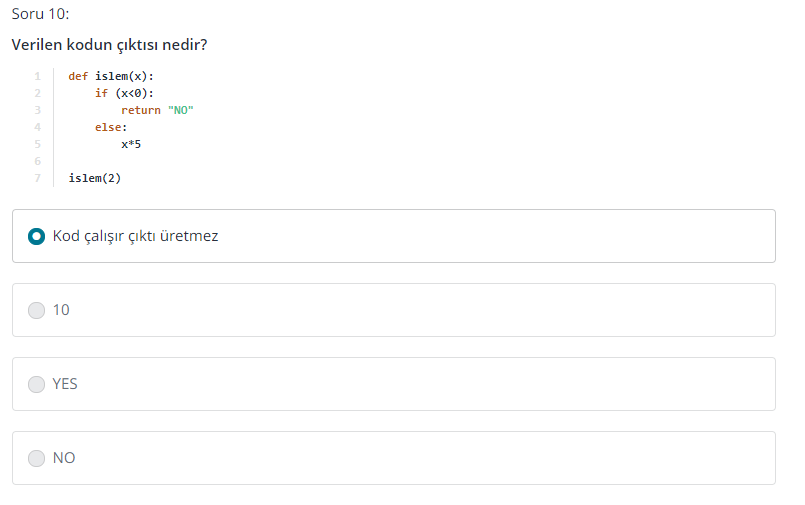


### Python Programlama Alıştırmalar - 7

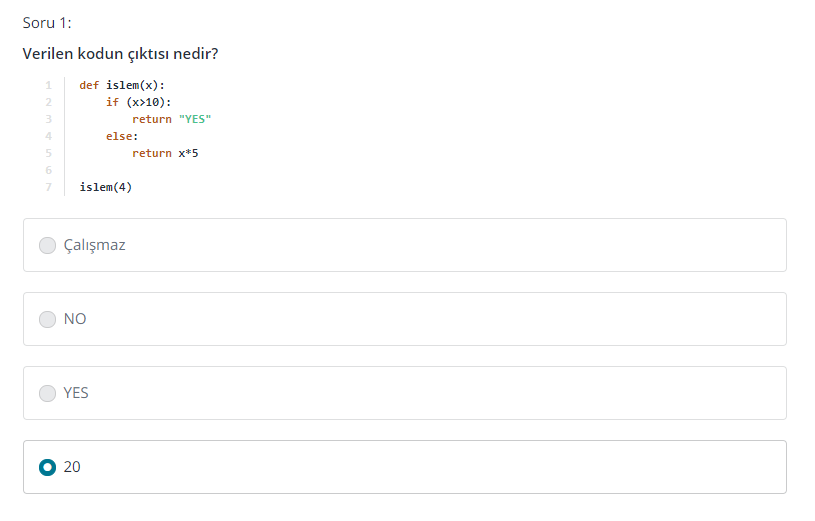


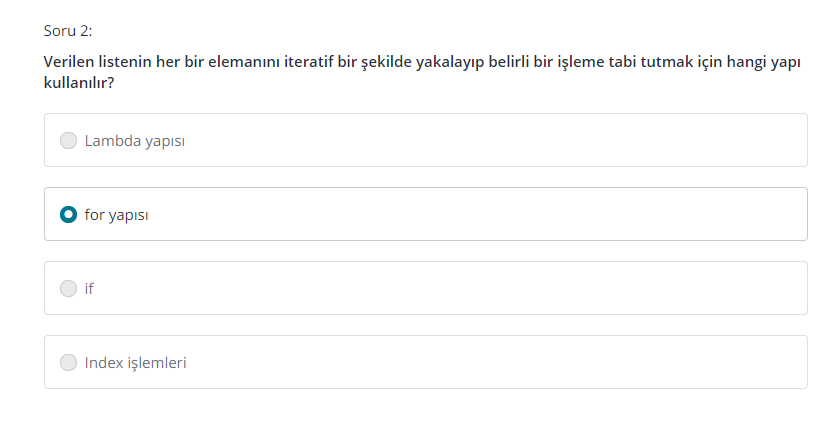




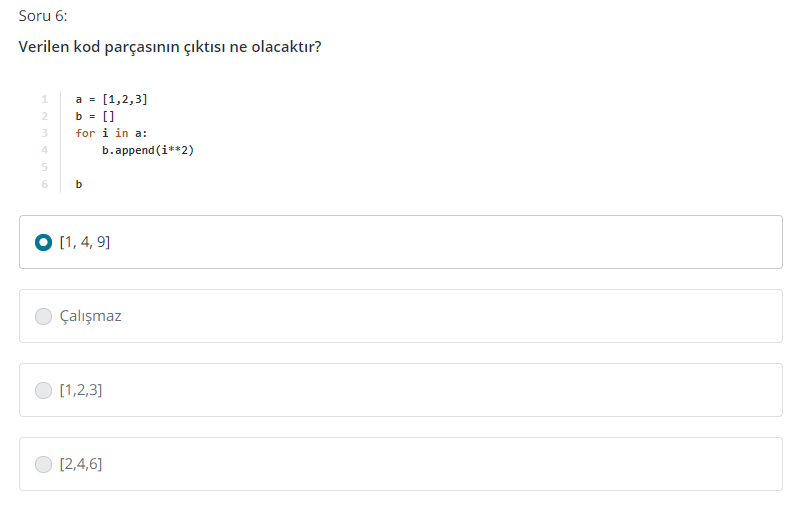


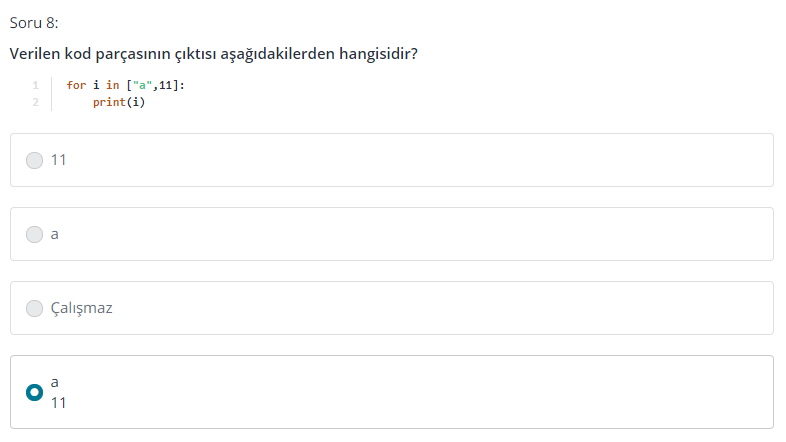
### Python Programlama Alıştırmalar - 8



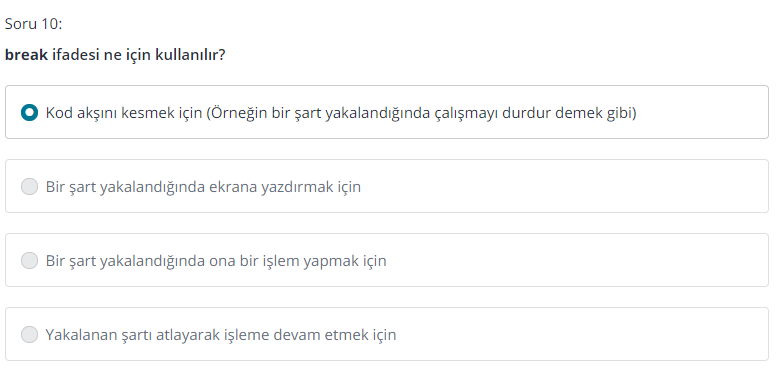




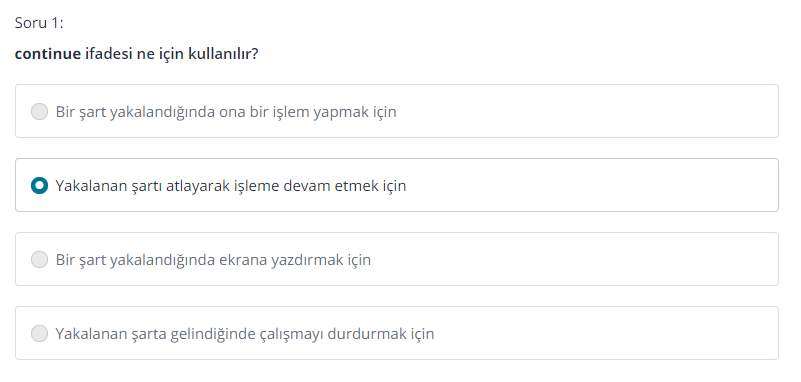


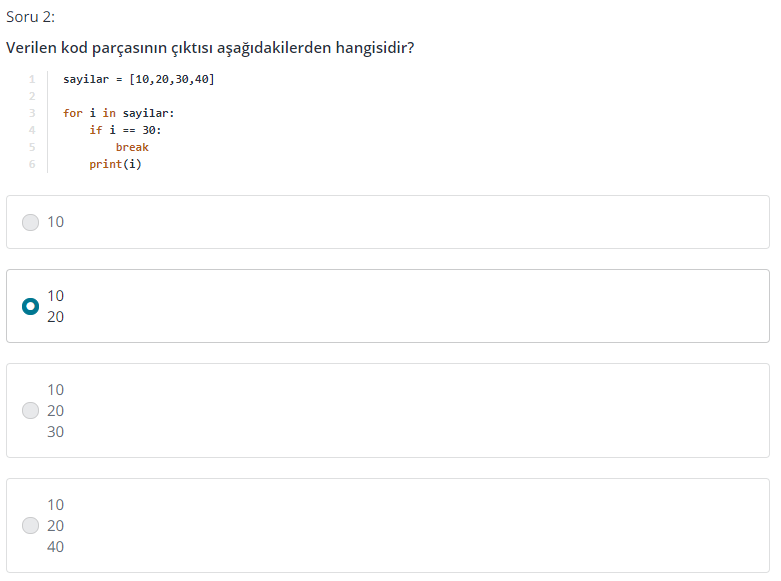
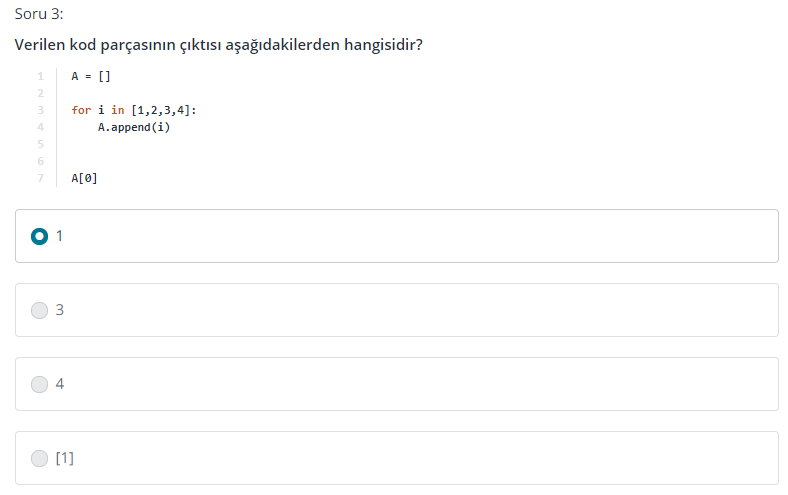


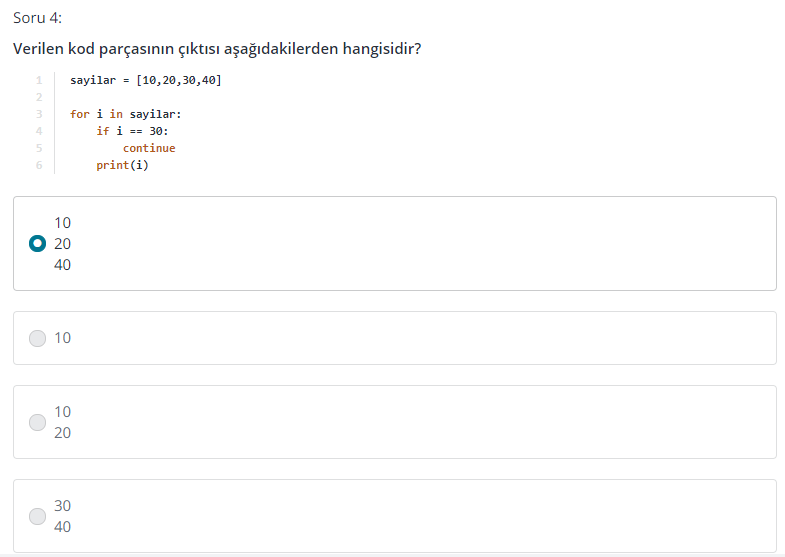
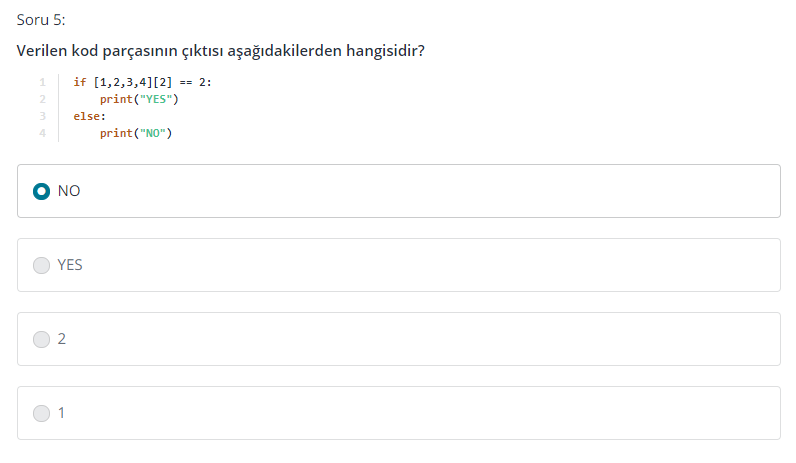


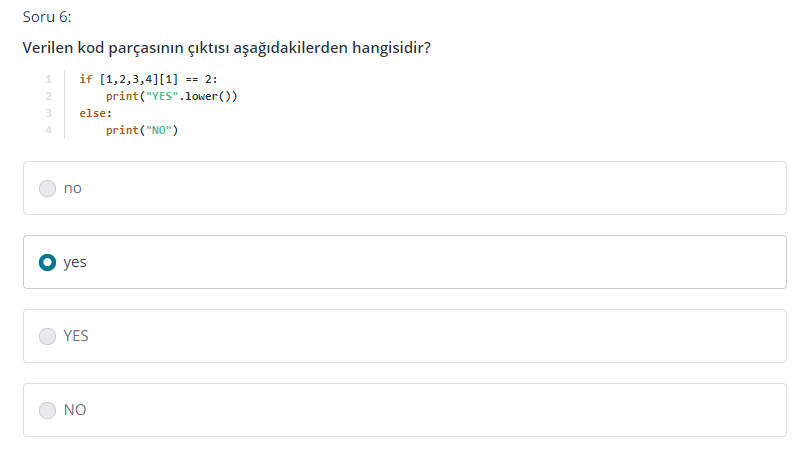
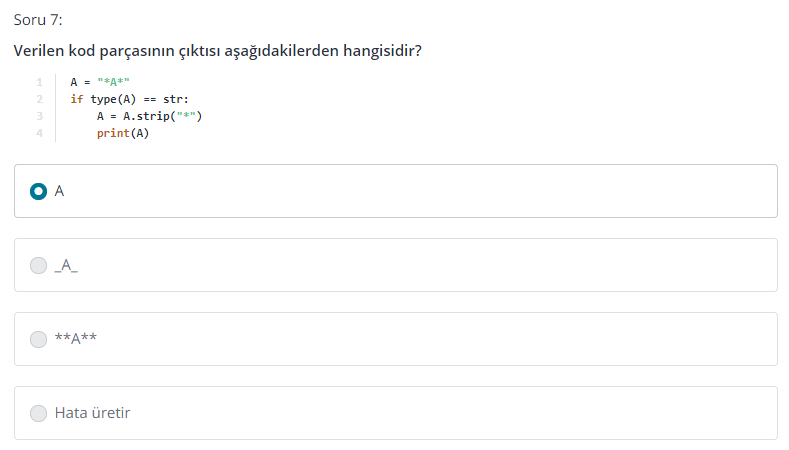


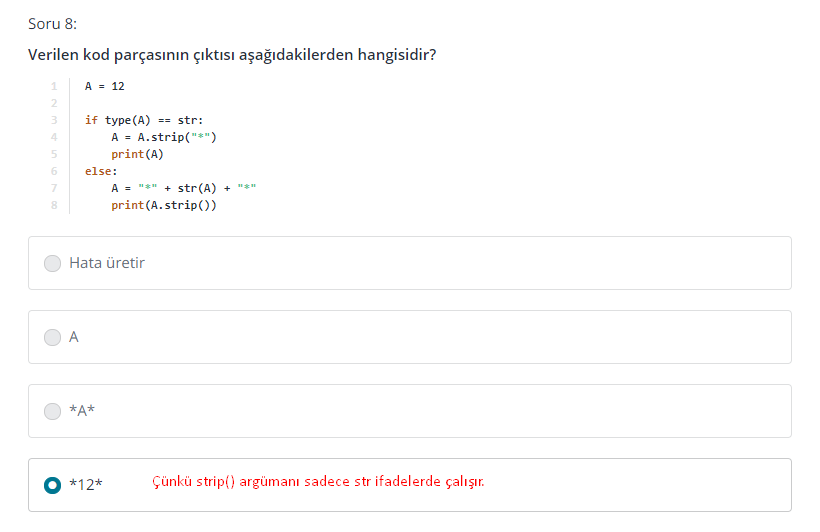
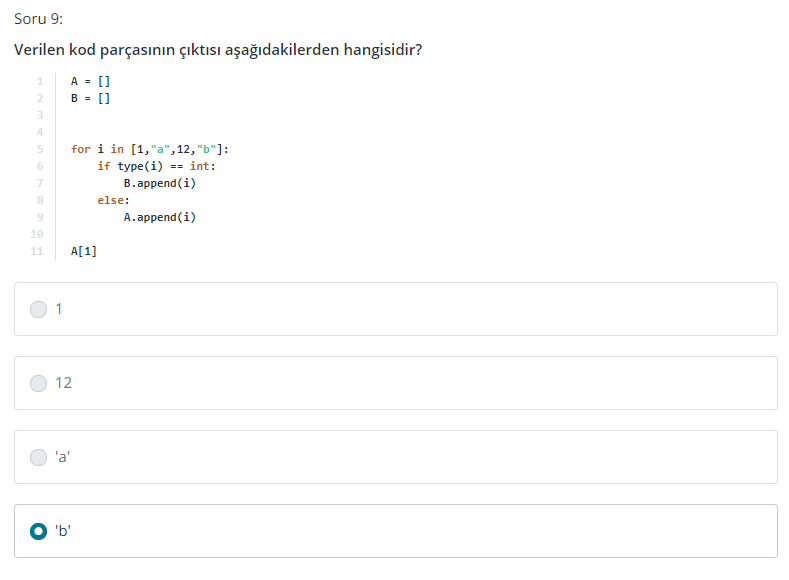
### Python Programlama Alıştırmalar – 9

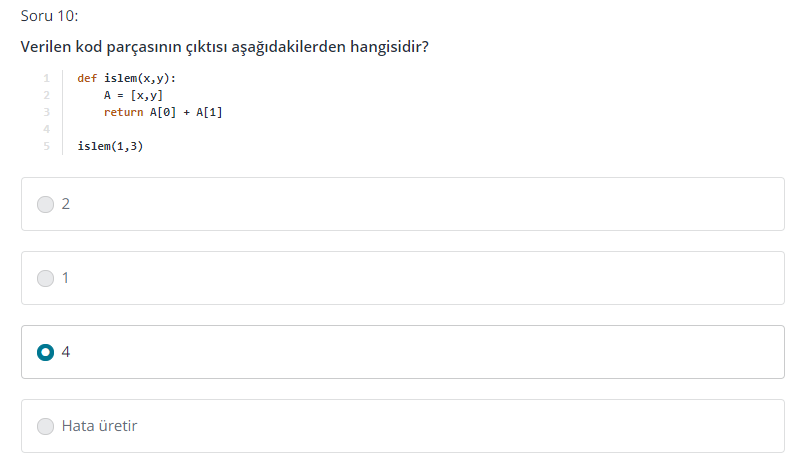


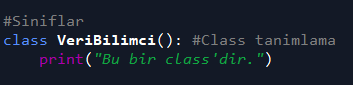


## Nesne Yönelimli Programlama

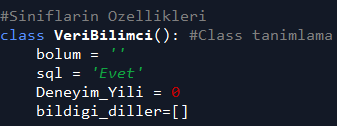
### Sınıflara Giriş ve Sınıf(Class) Tanımlamak

#### Sınıf Nedir?

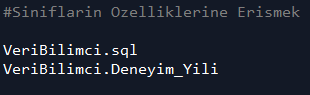
Sınıflar; benzer özellikler, ortak amaçlar taşıyan, içerisinde metod ve değişkenler olan yapılardır.



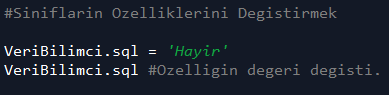
### Sınıfların Özellikleri



#### Sınıfların Özelliklerine Erişmek

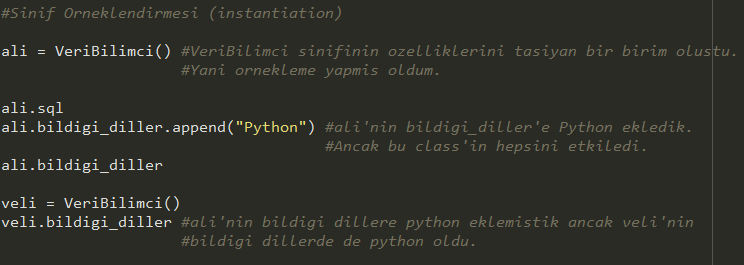


#### Sınıfların Özelliklerini Değiştirmek



### Sınıf Örneklendirmesi (instantiniation)

Sınıfın özelliklerini barındıran alt kümeler oluşturma işlemine sınıf örneklendirmesi denir.



### Örnek Özellikleri

Şuan yapmış olduğumuz işlem her bir örneğin kendi içinde değişebilen özelliklerden oluşabildiği bilgisini vermek. Yani her bir ayrı örneklendirme için özellik tutma bilgisini sağlıyor.

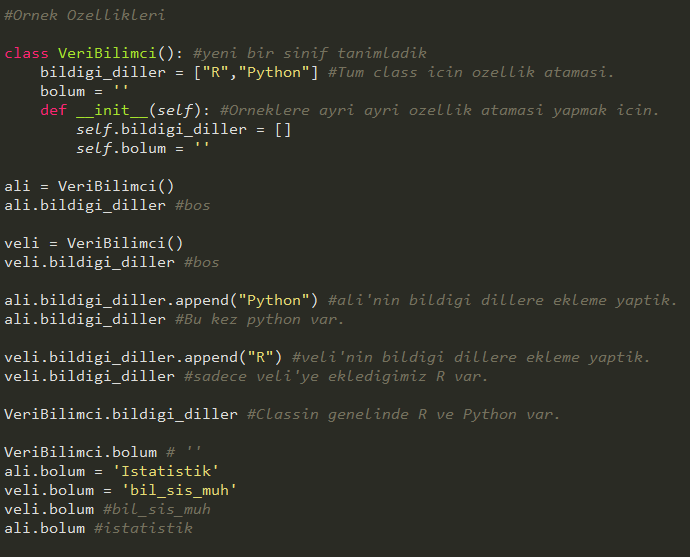
Sınıflar için tanımlanan özellikler örnekler içn değişebilir bir formata getirilmedikçe bir örnekte yapılan değişiklik tüm örneklere etki ediyor.

def \_\_init\_\_(self):

self.bildigi\_diller = ‘’

self.bolum = ‘’ 🡪 fonksiyonunu kullanacağız. Buradaki self temsilci anlamındadır. Her bir örneklemi temsil eder (ali, veli gibi).

Genelde sınıf özelliklerinin isimleri ve örnek niteliklerinin isimleri aynı olmamalıdır. Örneğimizde anlaşılır olması açısından aynı kullandık.



### Örnek Metodları