

**Veri Bilimi
2023 2024
Vize Soruları
(Bahar Dönemi)**

1.
X

- A) Show()
- B) Clear()
- C) Shape()
- D) Print()**
- E) List()

büyük-küçük harf duyarlılığı
(Case sensitivity)

```
1 s = 'ümit'  
2 Print(s)  
[4] 0.0s
```

X

```
NameError  
Cell In[4], line 2  
  1 s = 'ümit'  
----> 2 Print(s)
```

Traceback (most recent call last)

```
NameError: name 'Print' is not defined
```

✓

```
1 s = 'ümit'  
2 print(s)  
[5] ✓ 0.0s
```

... ümit

2.

```
daire="3+1"
if daire=="1+1":
    print("(2+1) daire aranıyor")
elif daire=="2+1":
    print("(3+1) daire aranıyor")
else:
    print("Sitede (3+1)daire yok")
```

A) Sitede (3+1)daire yok

▷ v
1 "Ümit" == "ümİt"
[12] ✓ 0.0s

... False

1 "Ümit" == "Ümit "
[13] ✓ 0.0s

... False

3.

- A) Matematik bilgisi
- B)** HTML, Javascript bilgisi
- C) Makine öğrenimi bilgisi
- D) İstatistik bilgisi
- E) Program bilgisi (Python vb.)

Veri Biliminde Kullanılan Temel Kavramlar

Makine Öğrenimi

Modelleme

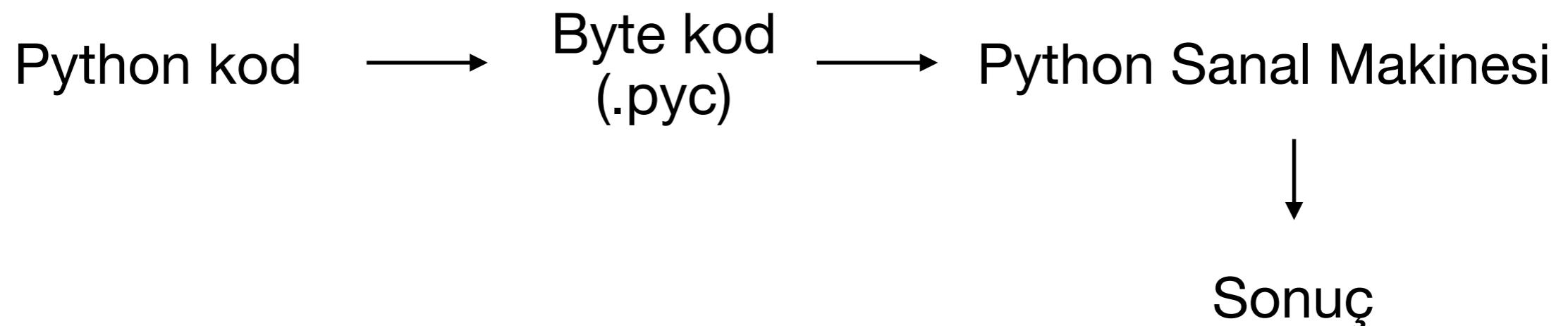
İstatistik

Programlama

Veri Tabanları

4. (i) Python, üst düzey bir programlama dilidir.
- (ii) Python yorumlanmış bir dildir.
- (iii) Python derlenmiş bir dildir.
- (iv) Python programı yorumlanmadan önce derlenir.

B) i, iv



5.

- A) Program başlamadan önce ve sona erdikten sonra ne olacağını belirler.
- B) Kontrol karakterlerinin giriş ve çıkışını yönetir.
- C) Python'a özgü veri yapılarını tanımlar.
- D) Programdaki satırları kontrol eder.
- E) Program içerisinde yer alan ifadelerin yürütme sırasını yönlendirir.

5.1. Programlama Kontrol Akış Şekilleri

Kontrol akışı (yapısı), bir yazılım programının bağımsız ifadelerinin, talimatlarının veya işlevlerinin yürütüldüğü ve değerlendirildiği sıradır. Basitçe ifade etmek gerekirse program akışı, kod satırlarının işlendiği sırayı tanımlayan genel bir terimdir. Bir bilgisayar programı temel kontrol yapıları kullanılarak yazılabilir. Bir kontrol yapısı değişkenleri analiz eden ve verilen parametrelere göre gidilecek yönü seçen bir programlama bloğudur. Başka bir ifade ile bir kontrol yapısı sadece bilgisayarın verdiği bir karardır. Bu nedenle, programlamada temel karar verme sürecidir ve kontrol akışı, bir bilgisayar programının belirli koşullar ve parametreler verildiğinde nasıl tepki vereceğini belirler.

6.

```
for i in range(1, 5):
    print(i)
```

B) 1, 2, 3, 4

```
1 for i in "Ümit":
2 |   print(i)
```

ü
m
i
t

```
1 for i in ["ü","m","i","t"]:
2 |   print(i)
```

ü
m
i
t

7.

- A) Kümeleme işlemi denetimsiz algoritmalarla yapılır.
- B) Denetimli algoritmalarla veri eğitim ve test verisi olarak ayrılır.
- C) Denetimli algoritmalarla eğitim işlemi yapılır.
- D) Kümeleme işleminde etiketli veri kullanılmaz.
- E) Sınıflama işleminde etiketlenmiş veri kullanılmaz.

Denetimli Öğrenme ve Denetimsiz Öğrenme

- Denetimli öğrenme
 - Bir etiketleme ve öğretici mekanizması var. Amaç tahmin yapmak
 - Örneğin :
 - spam mailleri spam diye işaretlemek
 - sosyal medyada gönderilerini dolandırıcı, şiddet, istismar vs bildirmek, hava durumu tahmini gibi
- Denetimsiz Öğrenme
 - Etiketleme yok. Amaç verilerden bir örüntü / kalıp çıkarma değerler arası ilişkileri ortaya çıkarma.
 - Örneğin :
 - Bir kişinin alışveriş alışkanlıklarına göre reklam ya da ürün önerisi

1. Regresyon

Regresyon, denetimli öğrenme tekniklerine dayalı bir makine öğrenmesi algoritmasıdır. Girdi özniteliklerinin bir fonksiyon ile temsil edilerek, çıktı özniteliklerinin tahmin edilmesine imkân veren bir modeldir. Regresyon çıktısı gerçek veya sürekli bir değer olabilir. Örneğin, bir odanın sıcaklığını tahmin etmek gibi.

2. Kümeleme

Kümeleme, denetimsiz öğrenme tekniklerine dayalı bir makine öğrenimi algoritmasıdır. Bir dizi etiketlenmemiş veri noktası üzerinde çalışır ve her veri noktasını bir kümede grupperendir.

3. Karar Ağacı

Karar ağacı, öncelikle sınıflandırma için kullanılan denetimli bir öğrenme yöntemini ifade eder. Algoritma, çeşitli girdileri belirli bir parametreye göre sınıflandırır. Bir karar ağacının en önemli avantajı, anlaşılmasının kolay olması ve sınıflandırılmasının nedenini açıkça göstermesidir.

4. Destek Vektör Makineleri

Destek vektör makineleri (SVM'ler) öncelikle sınıflandırma ve regresyon modelleri için kullanılan denetimli bir öğrenme yöntemidir. Ancak çoğunlukla sınıflandırma problemlerinde kullanılır. SVM'ler ile hem doğrusal hem de doğrusal olmayan sınıflandırmalar gerçekleştirilebilir.

5. Naive Bayes

Naive Bayes, ikili ve çok sınıflı sınıflandırma problemleri için en iyi kullanılan istatistiksel olasılık tabanlı bir sınıflandırma yöntemidir. Elinizde eğitilmiş bulunan verileri kendi formülüne göre işler ve her durum için yüzdelik bir oran çıkarır. Daha sonra girilen test verisini bu çıkan olasılıklara göre sınıflandırır.

8.

- A) Liste'lerin içerikleri değiştirilebilirken tuple'ların içerikleri değiştirilemez.
- B) Hem tuple hem de liste içerikleri değiştirilemez.
- C) Tuple ve liste'lerin tüm özellikleri aynıdır.
- D) Liste'lerin içerikleri değiştirilemezken tuple'ların içerikleri değiştirilebilir.
- E) Hem tuple'ların hem de liste'lerin içerikleri değiştirilebilir.

(100, 150, 200)

(150, 100, 200)

1 (1,2) == (2,1)

✓ 0.0s

False

9.

- A) Veri Tabanları
- B) Sayısal Analiz**
- C) İstatistik
- D) Makine Öğrenimi
- E) Programlama

Veri Biliminde Kullanılan Temel Kavramlar

Makine Öğrenimi

Modelleme

İstatistik

Programlama

Veri Tabanları

10. `list1=[6,23,3,2,0,9,8,75]` → [23,2,9,75]

D) `print(list1[1:8:2])`

[6,23,3,2,0,9,8,75]

Index 0 1 2 3 4 5 6 7

[Başlangıç : Bitiş : Adım Sayısı]

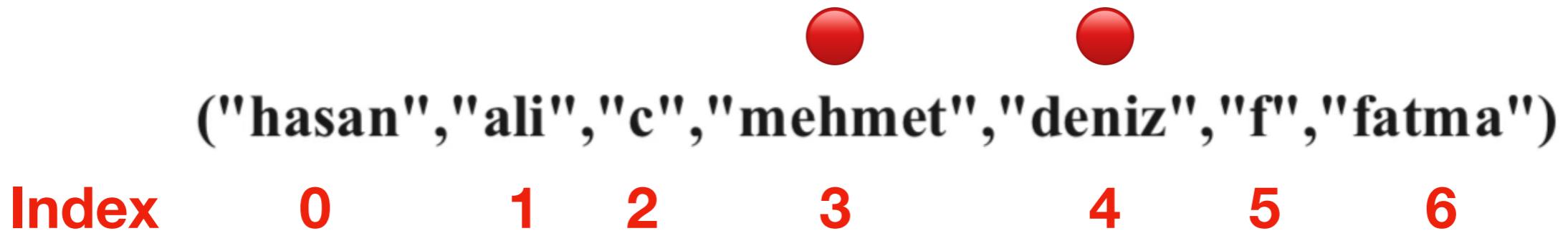


Dahil Değil

!!!

11. tuple = ("hasan","ali","c","mehmet","deniz","f","fatma") → tuple[3:5]

C) ("mehmet","deniz")



12. **set1 = set ([4, 5, (6, 7)])**
set1.update ([10, 11])

A) {4, 5, (6, 7), 10, 11}

```
1 set1 = set ([4, 5, 5, (6, 7), 6, 7])
2 set1.update ([10, 11, 11])
3 print (set1)
```

✓ 0.0s

{4, 5, 6, 7, 10, 11, (6, 7)}

```
1 set1 = set ("ümit sen, python programlama dili.")
2 print (set1)
```

✓ 0.0s

{'r', 'e', 'h', 'l', ' ', ',', 'a', 'n', 'ü', 'o', '.', 'm', 'i', 'y', 't', 'd', 'p', 's', 'g'}

13.

- A) Program koduna hatırlatma satırı eklemek için kullanılır.
- B)** Program kodunu daha hızlı çalıştırılmak için kullanılır.
- C) Program veya kodlarla ilgili açıklama yapmak için kullanılır.
- D) Ekran çıktısında çeşitli süsleme yapmak amaçlı kullanılır.
- E) Kullanılmayan bir kod satırını pasif hale getirmek için kullanılır.

14.

ogr=["Ahmet", "İşıl", "Mehmet", "Ayşe", "Kazım", "Zehra", "Turgay"]
Z → A

B) sorted(ogr, reverse=True)

```
1 ogr=["Ahmet", "İşıl", "Mehmet", "Ayşe", "Kazım", "Zehra", "Turgay"]
2 sorted(ogr)
```

✓ 0.0s

```
['Ahmet', 'Ayşe', 'İşıl', 'Kazım', 'Mehmet', 'Turgay', 'Zehra']
```

```
1 ogr=["Ahmet", "İşıl", "Mehmet", "Ayşe", "Kazım", "Zehra", "Turgay"]
2 sorted(ogr,reverse=True)
```

✓ 0.0s

```
['Zehra', 'Turgay', 'Mehmet', 'Kazım', 'İşıl', 'Ayşe', 'Ahmet']
```

```
1 nums = [1,2,3,4,5]
2 sorted(nums,reverse=True)
```

✓ 0.0s

```
[5, 4, 3, 2, 1]
```

```
1 nums = [1,2,3,4,5,"a","b"]
2 sorted(nums,reverse=True)
```

⊗ 0.0s

TypeError

Cell In[32], line 2

```
1 nums = [1,2,3,4,5,"a","b"]
----> 2 sorted(nums,reverse=True)
```

Traceback (most recent call last)

TypeError: '<' not supported between instances of 'int' and 'str'

15.

- A) Denetimli bir algoritmadır.
- B) Çıktılarının yorumlanması kolay olan bir algoritmadır.
- C) Girdileri, belli bir parametreye göre sınıflandıran bir algoritmadır.
- D) Makine öğrenmesi algoritmaları içerisinde yer alan bir algoritmadır.
- E) Sadece Kümeleme amaçlı kullanılan bir algoritmadır.

3. Karar Ağacı

Karar ağacı, öncelikle sınıflandırma için kullanılan denetimli bir öğrenme yöntemini ifade eder. Algoritma, çeşitli girdileri belirli bir parametreye göre sınıflandırır. Bir karar ağacının en önemli avantajı, anlaşılmasının kolay olması ve sınıflandırılmasının nedenini açıkça göstermesidir.

16.

- A) Talimatların uygulandığı sırayı takip etmek için
- B) Programın çalıştığından emin olmak için
- C) Programın daha iyi çalışması için
- D) Bir programdaki karar noktasını takip etmek için
- E) Program içindeki belli bir işlemin tekrarı için

6.1. While Kontrol Yapısı

Yineleme, aynı kod bloğunu defalarca, potansiyel olarak birçok kez çalıştırma anlamına gelir. Yinelemeyi uygulayan bir programlama yapısına döngü denir. Programlamada, belirsiz ve belirli olmak üzere iki tür yineleme vardır: Belirsiz yinelemeyle, döngünün çalıştırılma sayısı önceden açıkça belirtilmemiştir. Aksine belirtilen blok, bazı koşullar karşılandığı sürece tekrar yürütülür (Şekil 7).

17.

```
if 4 + 5 == 10:  
    print("TRUE")  
else:  
    print("FALSE")  
print("TRUE")
```

- B) FALSE
TRUE

```
1 if 4 + 5 == 10:  
2     print("TRUE")  
3 else:  
4     print("FALSE")  
5 print("TRUE")
```

✓ 0.0s

FALSE
TRUE

18.

```
rakam = [1, 2, 3] harf = ["a", "b", "c"]
```

```
[(1, 'a'), (2, 'b'), (3, 'c')]
```

E) `x = zip(rakam, harf)
list(x)`

```
1 a = zip([1,2],"ümit","sen")  
2 print(list(a))
```

✓ 0.0s

```
[(1, 'ü', 's'), (2, 'm', 'e')]
```

```
1 a = zip([1,2,3],"ümit","sen")  
2 print(list(a))
```

✓ 0.0s

```
[(1, 'ü', 's'), (2, 'm', 'e'), (3, 'i', 'n')]
```

```
1 a = zip("sen",[1,2,3],"ümit",)  
2 print(list(a))
```

✓ 0.0s

```
[('s', 1, 'ü'), ('e', 2, 'm'), ('n', 3, 'i')]
```

19.

```
import numpy as np  
a = np.array([1,2,3])
```

B) [1, 2, 3]

```
1 a = [1,2,3]  
2 print(a[0:2]) #A  
3 print(a[0:3:2]) #C  
4 print([a[0]]) #D  
5 print([a[2]]) #E
```

✓ 0.0s

```
[1, 2]  
[1, 3]  
[1]  
[3]
```

20. `dizi2 = np.array([1,2,3,4,5,6,7])` → [3,4,5]

X

A) `print(dizi2[2:5])`

```
1 dizi2 = np.array([1,2,3,4,5,6,7])
```

⊗ 0.0s

NameError

Traceback (most recent call last)

Cell In[49], line 1

----> 1 dizi2 = np.array([1,2,3,4,5,6,7])

NameError: name 'np' is not defined

[1, 2, 3, 4, 5, 6, 7]
Index 0 1 2 3 4 5 6



```
1 import numpy as elma
2 dizi2 = elma.array([1,2,3,4,5])
3 dizi2
```

✓ 0.0s

array([1, 2, 3, 4, 5])

[Başlangıç : Bitiş : Adım Sayısı]



Dahil Değil
!!!