



Data Science
Çalışma Dökümanı
Recep Aydoğdu

İçindekiler

Data Science	6
Data Science Kullanılan Alanlar	7
Data Science Proje Döngüsü.....	7
Veri Bilimine Giriş Alıştırmalar – 1.....	8
Veri Bilimine Giriş Alıştırmalar – 2.....	11
Python Programlama.....	15
Temel Hareketler.....	15
Integer, Float ve String	15
String Metodları	16
Python Programlama Alıştırmalar – 1	18
Python Programlama Alıştırmalar – 2	21
Python Programlama Alıştırmalar – 3	26
Veri Yapıları (Data Types)	30
Listeler	30
Tuple (Demet).....	33
Dictionary (Sözlük).....	33
Sets (Kümeler)	35
Veri Yapıları Özeti	39
Python Programlama Alıştırmalar – 4	40
Python Programlama Alıştırmalar – 5	43
Python Programlama Alıştırmalar – 6	48
Fonksiyonlar	53
Fonksiyon Nedir?	53
Matematiksel İşlemler.....	53
Fonksiyon Nasıl Yazılır ?.....	53
Bilgi Notuyla Çıktı Üretmek	54
İki Argümanlı Fonksiyon Tanımlamak.....	54
Ön Tanımlı Argümanlar	55
Ne Zaman Fonksiyon Yazılır?	55
Fonksiyon Çıktılarını Girdi Olarak Kullanmak	55
Local ve Global Değişkenler.....	56
Local Etki Alanından Global Etki Alanını Değiştirme.....	57
Karar-Kontrol Yapıları (Koşullar)	58

Koşul Nedir?	58
True – False Sorgulamaları (Boolean).....	58
if – else – elif.....	58
Uygulama: if ve input ile kullanıcı etkileşimli program	60
Döngüler	60
For Döngüsü	60
Döngü ve Fonksiyonların Birlikte Kullanımı	61
Uygulama: if, for ve fonksiyonların birlikte kullanımı	61
break & continue	61
while	62
Python Programlama Alıştırmalar - 7	63
Python Programlama Alıştırmalar - 8	67
Python Programlama Alıştırmalar – 9	72
Object Oriented Programming	78
Class'lara Giriş ve Class Tanımlamak	78
Class Özellikleri.....	78
Class Örneklendirmesi (instantiation)	78
Örnek Özellikleri	79
Örnek Metodları	80
Miras Yapıları (inheritance)	80
Functional Programming	81
Fonksiyonel Programlamaya Giriş	81
Yan Etkisiz Fonksiyonlar (Pure Functions)	81
İsimsiz Fonksiyonlar (Lambda) (Anonymous Functions)	83
Vektörel Operasyonlar (Vectorel Operations).....	83
Map & Filter & Reduce	84
Modül Oluşturma	84
Hatalar/İstisnalar (exception).....	85
Python Programlama Alıştırmalar – 10	86
Python Programlama Alıştırmalar – 11	90
Python Programlama Alıştırmalar – 12	95
Python ile Veri Manipülasyonu: NumPy & Pandas	100
NumPy (Numerical Python).....	100
NumPy Giriş	100
Neden NumPy?.....	101
NumPy Array'i Oluşturmak.....	101

zeros, ones, full, random, arange, linspace, random.normal, random.randint	102
NumPy Array Özellikleri.....	103
Matris Oluşturma	103
Reshaping (Array'i Yeniden Şekillendirme)	104
Concatenation (Array Birleştirme)	105
Splitting (Array Ayırma)	106
İki Boyutlu Array Ayırma.....	106
Sorting (Sıralama)	107
Matris sıralama.....	107
Index ile Elemana Erişmek.....	108
Matrislerde elemana erişme işlemleri.....	108
Slicing (Array Alt Küme İşlemleri)	109
Matrislerde Slicing İşlemleri	109
Alt Küme Üzerinde İşlem Yapmak	110
Fancy Index ile Elemanlara Erişmek	111
Matrislerde Fancy Index Kullanımı	111
Koşullu Eleman İşlemleri	112
Matematiksel İşlemler.....	113
Trigonometrik Fonksiyonlar	114
Logaritmik İşlemler	115
Numpy ile İki Bilinmeyenli Denklem Çözümü.....	115
NumPy Alıştırmalar – 1.....	116
NumPy Alıştırmalar – 2.....	120
NumPy Alıştırmalar – 3.....	125
Pandas	130
Pandas Giriş	130
Pandas Serisi Oluşturmak	131
Index İsimlendirmesi	132
Sözlük Üzerinden Seri Oluşturmak	132
İki Seriyi Birleştirerek Seri Oluşturma.....	133
Eleman İşlemleri	133
Eleman Sorgulama.....	134
Eleman Değiştirme	135
Pandas DataFrame Oluşturma	135
DataFrame İsimlendirme	136
DataFrame Özellikleri	136

DataFrame Eleman İşlemleri	137
Eleman Silme	139
Gözlem ve Değişken Seçimi: loc & iloc	142
Koşullu Eleman İşlemleri	144
Birleştirme (Join) İşlemleri.....	146
İleri Birleştirme İşlemleri	150
Birebir Birleştirme	150
Many to one (Çoktan teke)	151
Many to Many (Çoktan çoka)	152
Aggregation & Grouping (Toplulaştırma ve Gruplama)	153
Grouping.....	156
İleri Toplulaştırma İşlemleri(Aggregate, filter, transform, apply)	158
Aggregate	158
filter	159
transform.....	160
Apply.....	162
Pivot Tablolar.....	164
pivot_table	165
Dış Kaynaklı Veri Okuma.....	167
csv okuma	168
txt okuma	168
Excel dosyası okuma.....	169
Sıfırdan txt okuma	170
Pandas Alıştırmalar-1	171
Pandas Alıştırmalar-2	175
Pandas Alıştırmalar – 3	179

Data Science

VERİ BİLİMİNE GİRİŞ

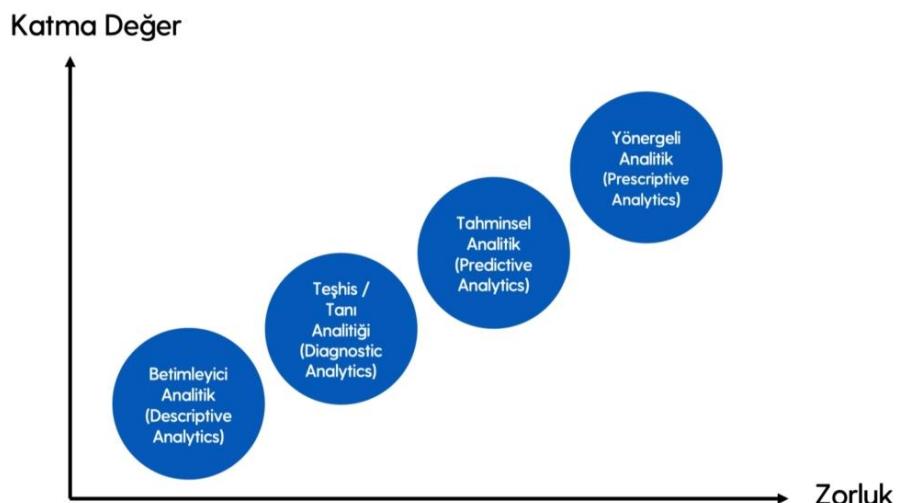


Veri Bilimci, veriden faydalı bilgi çıkarma sürecini yöneten kişidir.



VBO

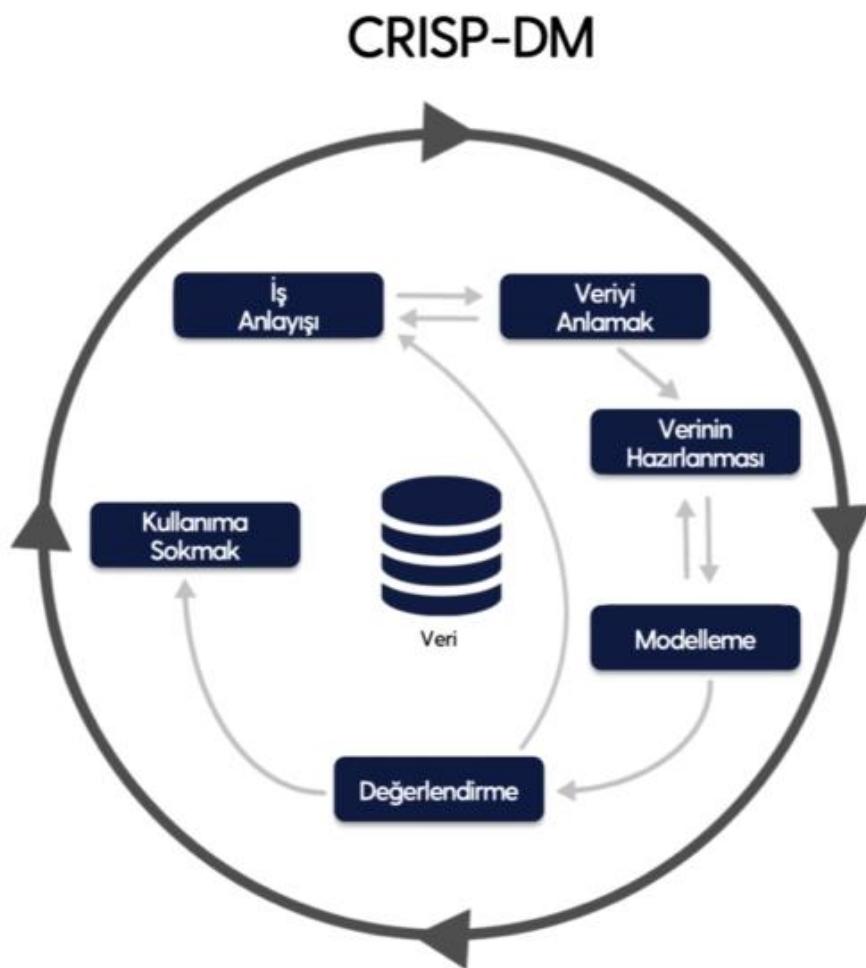
VERİDEN FAYDALI BİLGİ ÇIKARMAK



Data Science Kullanılan Alanlar

- Arkadaş önerileri
 - Otomatik fotoğraf etiketlemeleri
 - Hedefli içerik pazarlama
 - Otomatik mesaj tamamlama
 - Hedefli ürün pazarlama
 - Tavsiye sistemleri
 - Müşteri segmentasyonu
 - Kanser/Hastalık teşhisleri
 - Şirketlerin gelir tahmini ile strateji belirlemesi
 - Başvuru değerlendirme sistemleri
 - Akıllı portföy yönetimi
 - Doğal afet modelleme çalışmaları
 - E-Spor Analitiği
-
- Otonom araçlar
 - Nesne tanıma/takip uygulamaları
 - Sahte videolar
 - Eski resimlerin canlandırılması
 - Algoritmaların geliştirdiği resimler/var olmayan kişiler
 - Robotlar!

Data Science Proje Döngüsü



Veri Bilimine Giriş Alıştırmalar – 1

Soru 1:

Aşağıdakilerden hangisi günümüzün yeni petrolü olarak tanımlanmaktadır?

Sosyal medya

Veri

İstatistik

İnternet

Soru 2:

Aşağıdakilerden hangisi yapay zekayı besleyen temel kaynaktır?

Sosyal medya

İnternet

Veri

Algoritmalar

Soru 3:

Andrew Ng tarafından ifade edilen günümüzün yeni elektriği nedir?

Veri

Algoritmalar

Yapay Zeka

Veri Bilimi

Soru 4:

Veriden faydalı bilgi çıkarma sürecine ... denir? Boşluğa hangi ifade gelmelidir?

Makine öğrenmesi

Veri bilimi

a) Yapay zeka

İstatistik

Soru 5:

Aşağıdakilerden hangisi veri bilimi süreci bileşenlerinden değildir?

Veri kaynakları

Bilgi

Veri işleme

Aksiyon

Soru 6:

Bir bilgisayarın veya bilgisayar kontrolündeki bir robotun çeşitli faaliyetleri zeki canlılara benzer şekilde yerine getirme kabiliyetine ... denir.

Makine öğrenmesi

Veri bilimi

Yapay zeka

Derin öğrenme

Soru 7:

Aşağıdakilerden hangisi bir yapay zeka uygulaması değildir?

- Bir dizi matematiksel işlem gerçekleştiren program
- Belirli görevler için eğitilmiş robotlar
- Veri içerisindeki yapıları öğrenip genelleme yeteneği kazanmış bir fonksiyon
- Medikal görüntüler üzerinden hastalık tahmini yapan bir program

Soru 8:

Yapay zeka çağında hayatı kalmak ... ve ... yeteneklerine bağlıdır.

- Veri bilimi ve yapay zeka
- Veri analitiği ve analitik düşünce becerileri
- İstatistik ve programlama
- Programlama ve makine öğrenmesi

Soru 9:

Veri Bilimi çok disiplinli bir alan olarak ele alındığında aşağıdakilerden hangisi veri bilimini meydana getiren *ana unsurlardan* değildir.

Programlama

Bilgisayar Bilimleri

İş-Sektör Bilgisi

Matematik-İstatistik

Soru 10:

Belirli bir sektörde meydana gelen bilgi birikimine ne denir?

Veri Analitiği

Uzmanlık

İş Bilgisi

İş Dalı

Veri Bilimine Giriş Alistirmalar – 2

Soru 1:

Aşağıdakilerden hangisi veri analitiği türlerinden değildir?

Tahminsel Analitik

Betimleyici Analitik

Sektörel Analitik

Yonergeli Analitik

Soru 2:

"Neden olmuş" sorusuna yanıt arayan veri analitiği türü aşağıdakilerden hangisidir?

Teşhis/Tanı Analitiği

Betimleyici Analitik

Tahminsel Analitik

Yönergeli Analitik

Soru 3:

Aşağıdaki veri analitiği türlerinden hangisi diğerlerine göre daha kolay uygulanabilmektedir.

Betimleyici Analitik

Teşhis/Tanı Analitiği

Yönergeli Analitik

Tahminsel Analitik

Soru 4:

Aşağıdakilerden hangisi "ne olmalı" / "nasıl olmalı" sorusuna yanıt arar?

Betimleyici Analitik

Teşhis Analitiği

Yönergeli Analitik

Tanı Analitiği

Soru 5:

Aşağıdakilerden hangisi "ne olacak" sorusuna yanıt arar?

Betimleyici Analistik

Teşhis/Tanı Analitiği

Yönergeli Analistik

Tahminsel Analistik

Soru 6:

Bir ürünne ait satış sayılarının aylara göre görselleştirilmesi hangi veri analitiği türüne girer?

Normatif Analistik

Teşhis/Tanı Analitiği

Betimleyici Analistik

Yönergeli Analistik

Soru 7:

Yıl sonu elde edilecek gelirin ne olacağının araştırılması hangi veri analitiği türüne girer?

Tahminsel Analistik

Betimleyici Analistik

Teşhis/Tanı Analitiği

Yönergeli Analistik

Soru 8:

ABD başkanlık seçimlerinde en önemli rolü oynayan iki kavram aşağıdakilerden hangisi olabilir?

- Veri bilimi ve yapay zeka
- Veri analitiği ve analitik düşünce becerileri
- Veri ve tahminsel analitik
- Sosyal medya ve yapay zeka

Soru 9:

Aşağıdakilerden hangisi günümüz dünyasında veri bilimi ve yapay zekayı bu kadar önemli hale getiren sebepler birisi olamaz?

- Anlamlı hale getirilmeyi bekleyen verinin hızla artması
- Otonomlaştırılması gereken iş alanları
- Şirketlerin gelir ya da süreçlerinde iyileştirme ihtiyaçları
- Yeni istihdam alanlarının aranması

Soru 10:

Bir şirket gelirlerinde meydana gelen düşüşlerin nedenlerini veriye bakarak anlamak istiyor bu durumda hangi veri analitiğini kullanması gereklidir?

- Teşhis/Tanı Analitiği
- Betimleyici Analistik
- Normatif Analistik
- Tahminsel Analistik

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.

```
1  print("Hello AI Era")
2
3  #type komutu icerisine yazdigimiz nesnenin tipini verir.
4  type(9) #integer
5  type(9.2) #float
6  type("Recep Aydoğdu") #string
7
8  #####
9
10 type("123") #bunun da ciktisi str olacaktir.
11
12 "a"+"a"
13
14 "a" " a"
15
16 "a"*3
17
18 "a"/3 #type error hatali
19
20 "a " *5
21
```

- "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()`= içерisine yazılan değişkenin uzunluğunu verir.

```
1  # STRING METODLARI - len()
2
3  gel_yaz="gelecegi_yazanlar"
4
5  #del mvk #degiskeni silmek icin del kullaniriz. kullandiktan sonra
6  #      # yorum satiri haline getirilmelidir.
7
8  a=99
9  b=10
10
11 type(a/b) # a/b=9.9 olacagindan tipi float olur.
12
13 len(gel_yaz) # gel_yaz degiskenenin icersindeki string'in krktr uzunlugunu verir.
14
```

`upper() & lower() =`

```
17  #upper() & lower() fonksiyonlari
18
19  gel_yaz.upper() #stringi buyuk harflere cevirir.
20
21  gel_yaz.lower() #stringi kucuk harflere cevirir.
22
```

`isupper() & islower() =`

```
23  #isupper() & islower() fonksiyonlari
24
25  gel_yaz.isupper() #buyuk harf mi? sorusu sorar. T or F getirir.
26  gel_yaz.islower() #kucuk harf mi? sorusu sorar.
27
28  B = gel_yaz.upper() #B degiskenine buyuk harfli gel_yaz atadik.
29
30  B.isupper()
31
32  Dnm="AsDfGhGgGgG"
33
34  Dnm.isupper()
35  Dnm.islower()  #ikisi de false getirir.
```

`replace() =`

```
37  # replace() bir karakteri baska bir karakter ile degistirmek icin kullanilir.
38
39  gel_yaz.replace("a","i")
40
```

`replace("eski_karakter","yeni_karakter")`

`gelecegi_yazanlar → gelecegi_yizinler`

`strip()` = Karakter kırpma işlemleri

```
41  # strip() Karakter kırpma islemleri
42
43  gel_yaz= " gelecegi_yazanlar " #basinda ve sonunda bosluk var
44  gel_yaz.strip() #varsayılan olarak boslukları siler.
45
46  gel_yaz="*gelecegi_yazanlar*" # basina ve sonuna * ekledik.
47  gel_yaz.strip("*") # *(yıldız) arasındaki ifadeyi kırpar.
48
```

`dir() =`

```
49  # dir() icerisine yazdigimiz veri tipi icin kullanilabilir metodlari verir.
50
51  dir(gel_yaz)
52  | | | | #ikisi de ayni sonucu verir.
53  dir(str)
54
```

`capitalize()` = İlk harfi büyütür.

`gel_yaz.capitalize()`

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

`gel_yaz.title()`

`Substring` = Alt küme işlemleri

```
56
57  # Substring: string ifadeleri ile alt kume islemleri.
58
59  gel_yaz[0] # 0 index'li ifadeyi getirir.
60
61  gel_yaz[0:3] # 0'dan basla 3'e kadar getir.
62
```

Type Dönüşümleri

```
63  #TYPE DONUSMLERI
64
65  toplama_bir=input() #input ile kullaniciidan veri aliriz.
66  toplama_iki=input() #kullaniciidan aldigimiz veri str tipindedir.
67
68  toplama_bir+toplama_iki # 10+20 --> '1020' ciktisi verir.
69  | | | | # bunu engellemek icin type donusumunu yapmalıyız.
70  int(toplama_bir)+int(toplama_iki) #tip donusumlerini bu sekilde yapariz.
71
72  int(12.4) #float to int --> 12
73  float(12) #int to float --> 12.0
74  str(12)   #int to str   --> '12'
```

`print()` fonksiyonu

`print("gelecegi","yazanlar")` → gelecegi yazanlar

`print("gelecegi","yazanlar",sep = ("_"))` → gelecegi_yazanlar

```
76  #Print fonksiyonu
77
78  print("gelecegi","yazanlar")
79
80  print("gelecegi","yazanlar",sep = "_") #sep argumani araya gelecek degeri secmemize olanak saglar.
81
82  ?print #print fonksiyonu ile kullanabilecegimiz argumanlari verir.
83
```

Python Programlama Alıştırmalar – 1

Soru 1:

Kod bloğu içerisinde yapılan bir işlemin sonucunu ekrana bastırmak için hangi fonksiyon kullanılır?

len()

print

print()

def

Soru 2:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
print("uzaya", "git", sep = "***")
```

uzaya ** git

uzaya git

uzaya__git

uzaya**git

Soru 3:

Aşağıdaki ifadelerden hangisi sayı (float ya da integer) değildir?

64

2.3

"9"

2/10

Soru 4:

`type()` fonksiyonu ne için kullanılmaktadır?

Değişken dönüştürmek

Tip sorgulamak

Yazdırma

Fonksiyon tanımlamak

Soru 5:

`type(4)` kodunun çıktısı aşağıdakilerden hangisidir?

4

float

str(4)

int

Soru 6:

`type(3.14)` kodunun çıktısı aşağıdakilerden hangisidir?

3.14

int

float

str

Soru 7:

"**a**" + "**b**" kodunun çıktısı aşağıdakilerden hangisidir?

a + b

ab

'ab'

"a" + "b"

Soru 8:

"9" + "1" kodunun çıktısı aşağıdakilerden hangisidir?

10

9 + 1

"9" + "1"

'91'

Soru 9:

"**10**" + **2** kodunun çıktısı aşağıdakilerden hangisidir?

12

İşlem hata üretir

102

"102"

Soru 10:

Verilen örnek kodun çıktısı aşağıdakilerden hangisidir?

```
1 | a = 5  
2 | b = 10  
3 | c = a*b  
4 | c
```

5

10

15

50

Python Programlama Alıştırmalar – 2

Soru 1:

Verilen örnek kodun çıktısı aşağıdakilerden hangisidir?

```
1 | degisken = 4  
2 | print(degisken*degisken)
```

16

4

degisken

İşlem hata üretir

Soru 2:

Verilen örnek kodun çıktısı aşağıdakilerden hangisidir?

```
1 | sakla = 9  
2 | yeni_sakla = sakla*10
```

90

9

kod çalışır çıktı üretmez

yeni_sakla

Soru 3:

Verilen örnek kodun çıktısı aşağıdakilerden hangisidir?

```
1 | ifade = "selam"  
2 | type(ifade)
```

selam

int

ifade

str

Soru 4:

Aşağıdakilerden hangisi bir sayı (float ya da integer) değildir?

"3"

98

1/99

2.2

Soru 5:

Verilen örnek kodun çıktısı aşağıdakilerden hangisidir?

```
1 | ifade = "gelecegi yaziyoruz"  
2 | ifade[1]
```

'gelecegi yaziyoruz'

'g'

'e'

ifade

Soru 6:

Verilen örnek kodun çıktısı aşağıdakilerden hangisidir?

```
1 | ifade = "gelecegi yaziyoruz"  
2 | ifade[0:2]
```

'gelecegi yaziyoruz'

'ge'

'gel'

g

Soru 7:

Verilen örnek kodun çıktısı aşağıdakilerden hangisidir?

```
1 | a = "bu uzun bir metindir"  
2 | a[2:5]
```

'u uzun'

'uzun'

'uz'

'zun'

Soru 8:

Verilen örnek kodun çıktısı aşağıdakilerden hangisidir?

```
1 | a = "bu uzun bir metindir"  
2 | a[8]
```

'm'

'e'

'b'

''

Soru 9:

"9" + 1 kodunun çıktısı aşağıdakilerden hangisini üretir?

TypeError

SyntaxError

Hata üretmez

20

Soru 10:

Aşağıdakilerden hangisi bir karakter dizisinin eleman sayısını verir?

print()

lenght()

len()

replace()

Python Programlama Alıştırmalar – 3

Soru 1:

Aşağıdakilerden hangisi bir karakter dizisinin tüm karakterlerini büyütmek için kullanılır?

len()

upper()

lower()

print()

Soru 2:

Bir karakter dizisi içerisinde yer alan karakterleri değiştirmek için aşağıdakilerden hangisi kullanılır?

lower()

upper()

replace()

len()

Soru 3:

Verilen örnek kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | ifade = "gelecek_geldi"  
2 | ifade.replace("i", "ı")
```

'gelecek_geldi'

Çıktı gelmez

'gelecek_geldi'

"gelecek_geldi"

Soru 4:

Verilen örnek kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | ifade = "Merhaba!"  
2 | ifade = ifade.lower()  
3 | ifade = ifade.replace("!", "")  
4 | ifade
```

MERHABA!

'merhaba! '

'merhaba'

MERHABA

Soru 5:

Verilen örnek kod parçasının çıktısı aşağıdakilerden hangisidir?

`"_Python_".strip(" ")`

Çalışmaz

'Python'

'_Python_'

"Python"

Soru 6:

Karakter dizilerinde sağ ve soldan "kırpma" işlemi yapmak için aşağıdakilerden hangisi kullanılır?

replace()

strip()

len()

lower()

Soru 7:

Verilen örnek kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | ifade = "Merhaba! "
2 | ifade.strip("")
```

Çalışmaz

Hata Üretir

Merhaba!

'Merhaba! '

Soru 8:

Veri yapılarına ilişkin metodlara erişmek için aşağıdakilerden hangisi kullanılır?

len()

dir()

print()

?print

Soru 9:

Verilen örnek kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | ifade = "1012340"
2 | ifade = ifade + "1"
3 | ifade.strip("1")
```

Hata üretir

'1012341'

ifade1

'012340'

Soru 10:

Aşağıdakilerden hangisi kullanıcıdan bilgi almak için kullanılır?

dir()

replace()

input()

put()

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.

```
notlar = [90,80,70,50] #liste olusturma
type(notlar) #--> list

liste=["a",19.5,3] #farkli tipleri barindiran liste

liste_genis=[ "a",19.5,3,notlar] #kapsayicidir. icersinde farkli veri tipleri hatta liste bile barindirabilir.
len(liste_genis) #boyutu 4 olur.
```

Liste Elemanlarına Ulaşma

```
#liste elemanlarina ulasma

liste_genis[0] #-->"a"
liste_genis[1] #-->19.5
liste_genis[2] #-->3
liste_genis[3] #-->[90,80,70,50]

liste_genis[0:2] #0'dan 2 indexli elemana kadar alir
liste_genis[:2] #0'dan 2 indexli elemana kadar alir
liste_genis[2:] # 2 indexli elemandan sona kadar alir

liste_genis

liste_genis[3][1] # liste_genis icersindeki notlar listesinin 1 indexli elemani
# --> 80

print(liste_genis[3][0]) #--> 90
```

Liste İçi Type Sorulama

```
#liste ici type sorgulama

type(liste_genis[0])
type(liste_genis[1])
type(liste_genis[2])
type(liste_genis[3])

tum_liste=[liste,liste_genis]
```

del liste → liste'yi siler

Liste elemanlarını değiştirme

```
# Liste elemanlarini degistirme

liste2=["ali","veli","berkcan","ayse"]
liste2

liste2[1]="velinin babasi" # 1 index'li elemani degistirdik
liste2

liste2[1]="veli"
liste2[:3]="alinin_babasi","velinin_babasi","berkcanin_babasi" #3 elemani degistirdik
liste2
```

Listeye eleman ekleme

```
#listeye eleman ekleme

liste2 + ["kemal"] # bu sekilde kaydetmez sadece goruntuler.

liste2 = liste2 + ["kemal"]
```

Listeden eleman silme

del liste2[5] → 5 index'li elamanı siler.

append ve remove metodları

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

liste2.remove("alinin_babasi") → silme yapar

liste2.remove("velinin_babasi")

insert metodu

index'e göre ekleme yapar.

```
#insert

liste2.insert(0,"ayca") #0 index'e ayca eklendi
liste2.insert(2,"recep") #2 index'e recep ekledi
liste2.insert(8,"asd") #fazla index girdik fakat sona ekledi
len(liste2)

liste2.insert(len(liste2),"son_eleman") #listenin sonuna ekledi
```

pop metodu

index'e göre silme yapar.

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

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

count metodu

```
#count  
  
liste=["ali","veli","ayca","veli","ali","ali"]  
  
liste.count("ali") # "ali" elemanın listede kaç kez yer aldığı gösterir.
```

→ 3

copy metodu

liste_yedek=liste.copy() → liste'yi liste_yedek'e kopyalar.

extend metodu

İki farklı listeyi birleştirir.

```
#extend  
  
liste.extend(liste2) # liste ile liste2'yi birleştirir.  
liste  
  
liste2.extend(["a",10]) # liste ile metodun içine yazılan elemları birleştirir.  
liste2
```

index metodu

```
#index  
  
liste.index("ali") # yazdığımız elemanın kaçinci index olduğunu verir.
```

reverse metodu

liste = [1,2,3]

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

liste = [3,2,1]

sort metodu

Elemanları küçükten büyüğe sıralar.

```
#sort  
  
liste3=[2,1,5,3,4]  
  
liste3.sort() # liste3'ü küçükten büyük sıralayıp kaydeder.  
liste3
```

clear metodu
liste'nin içini boşaltır.

```
#clear  
  
liste3.clear() #liste3'ün içini boşaltır  
  
del(liste3) #liste3'ü tamamen siler.
```

Tuple (Demet)

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

Tuple Oluşturma

```
#Tuple Olusturma  
  
t=(1,2,3,"eleman",[1,2,3,4])
```

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

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 tutıldığı veri yapısıdır.

Listelerde olduğu gibi index'leme yapılmaz.

Dictionary Oluşturma

```
# Sozluk Olusturma  
  
sozluk={"REG" : "regresyon modeli",  
        "LOJ" : "lojistik regresyon",  
        "CART" : "Classification And Reg"}  
  
sozluk  
len(sozluk) # --> 3  
  
{“key” : “key’in karşılığı”}
```

NOT= Sözlüklerde key'ler sadece sabit veri yapılarından oluşabilir. list gibi yapılarından 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 secme islemleri

sozluk={"REG" : "regresyon modeli",
        "LOJ" : "lojistik regresyon",
        "CART" : "Classification And Reg"}

sozluk["REG"] #REG key'inin karsiligini bu sekilde getiririz.

sozluk={"REG" : {"ASD" : 10,
                 "XXX" : 20,
                 "ZZZ" : 30},

         "LOJ" : {"ASD" : 10,
                 "XXX" : 20,      # Sozluk icinde sozluk olusturduk. Ic ice yapi.
                 "ZZZ" : 30},

         "CART" : {"ASD" : 10,
                    "XXX" : 20,
                    "ZZZ" : 30}
        }

sozluk["REG"]["XXX"] #ic ice bir yapida elemana erisim.
```

```
In [6]: sozluk[ "REG" ][ "XXX" ] #ic ice bir yapida elemana erisim.
Out[6]: 20
```

Eleman Ekleme & Değiştirme

```
In [14]: sozluk={"REG" : "regresyon modeli",
                  "LOJ" : "Lojistik regresyon",
                  "CART" : "Classification And Reg"}

In [15]: sozluk[ "GBM" ] = "Gradient Boosting Mac" #sozluk'e eleman ekleme.

In [16]: sozluk
Out[16]:
{'REG': 'regresyon modeli',
 'LOJ': 'lojistik regresyon',
 'CART': 'Classification And Reg',
 'GBM': 'Gradient Boosting Mac'}
```

```
In [17]: sozluk["REG"]= "REG'in yeni karsiligi" #REG Key'inin karsiligini degistirme.
...: sozluk
Out[17]:
{'REG': "REG'in yeni karsiligi",
 'LOJ': 'lojistik regresyon',
 'CART': 'Classification And Reg',
 'GBM': 'Gradient Boosting Mac'}
```

REG key'i olmasaydı yeni key oluşturulacaktı.

```
In [22]: t= ("tuple",) # t adinda tuple olusturduk.

In [23]: sozluk[t]="Tuple'dan key olusturuldu."
...: sozluk
Out[23]:
{'REG': "REG'in yeni karsiligi",
 'LOJ': 'lojistik regresyon',
 'CART': 'Classification And Reg',
 'GBM': 'Gradient Boosting Mac',
 ('tuple',): "Tuple'dan key olusturuldu."}
```

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.

```
In [1]: l= ["ali","ata","bakma","ali","uzaya","git"]

In [2]: s=set(l) # l listesindeki elemanlari birer kez alir.

In [3]: s #set'in elemanlari essiz olacaginden her eleman bir kez alınır.
Out[3]: {'ali', 'ata', 'bakma', 'git', 'uzaya'}
```

```
In [4]: ali="ali_ata_bakma_uzaya_git_lutfen"

In [5]: s=set(ali) #ali cumlesindeki her bir karakteri bir kez alir.

In [6]: s
Out[6]: {'_', 'a', 'b', 'e', 'f', 'g', 'i', 'k', 'l', 'm', 'n', 't', 'u', 'y', 'z'}
```

Set'lere eleman ekleme ve çıkarma işlemleri

add() fonksiyonu ile ekleme yaparız.

```
In [8]: s.add("ile") # ile stringini set'e ekledi
...: s
Out[8]:
{'_',
 'a',
 'b',
 'e',
 'f',
 'g',
 'i',
 'ile',
 'k',
 'l',
 'm',
 'n',
 't',
 'u',
 'y',
 'z'}
```

```
In [9]: t=("ali","bakma")

In [10]: s.add(t) # t isimli tuple'i set'e ekledi
...: s
Out[10]:
{('ali', 'bakma'),
 '_',
 'a',
 'b',
 'e',
 'f',
 'g',
 'i',
 'ile',
 'k',
 'l',
 'm',
 'n',
 't',
 'u',
 'y',
 'z'}
```

```
In [12]: s.add(ali) # ali elemanini set'e ekledi.  
...: s  
Out[12]:  
{('ali', 'bakma'),  
 ' ',  
 'a',  
 'ali_ata_bakma_uzaya_git_lutfen',  
 'b',  
 'e',  
 'f',  
 'g',  
 'i',  
 'ile',  
 'k',  
 'l',  
 'm',  
 'n',  
 't',  
 'u',  
 'y',  
 'z'}
```

remove() fonksiyonu ile set'lerden eleman silebiliriz.

```
In [13]: s.remove(ali) # ali elemanini sildi.  
...: s  
Out[13]:  
{('ali', 'bakma'),  
 ' ',  
 'a',  
 'b',  
 'e',  
 'f',  
 'g',  
 'i',  
 'ile',  
 'k',  
 'l',  
 'm',  
 'n',  
 't',  
 'u',  
 'y',  
 'z'}
```

```
s.remove(ali) # ali'yi tekrar silmek istedigimizde KeyError hatası verir.  
s.discard(t) # discard ile de silme islemi gerceklestirebiliriz  
s  
s.discard(t) # tekrar silmek istedigimizde discard hata uretmeyez.
```

Set'lerde Fark İşlemleri

difference & symmetric_difference

difference = kümelerin farkını verir.

```
#difference ve symmetric_difference  
  
set1= set([1,3,5])  
set2= set([1,2,3])
```

```
In [2]: set1.difference(set2) #set1'in set2'den farki  
Out[2]: {5}
```

```
In [3]: set2.difference(set1) #set2'in set1'den farki  
Out[3]: {2}
```

symmetric_difference = ikisinde de ortak olmayan elemanları verir.

```
In [4]: set1.symmetric_difference(set2) #ikisinde de ortak olmayan elemanlari verir  
Out[4]: {2, 5}
```

Set'lerde Kesişim ve Birleşim İşlemleri

intersection & union & intersection_update

intersection = kesişim

```
In [5]: set1.intersection(set2) # set1 ve set2'nin ortak elemanlari  
Out[5]: {1, 3}  
  
In [6]: set2.intersection(set1)  
Out[6]: {1, 3}
```

union = birleşim

```
In [7]: set1.union(set2) # set1 ve set2'nin birlesimi  
Out[7]: {1, 2, 3, 5}
```

intersection = set1'in değerini kesişim değerleri olarak değiştirir.

```
In [8]: set1.intersection_update(set2) #set1'in degerini kesisim degerleri olarak degistirir.  
In [9]: set1  
Out[9]: {1, 3}
```

Set'lerde Soru İşlemleri

isdisjoint & issubset & issuperset

isdisjoint = Ayrık küme mi?

İki kümenin kesişiminin boş olup olmadığını sorgular.

Bos ise True değil ise False döndürür.

```
In [10]: set1.isdisjoint(set2) #set1 ve set2'nin kesisimi bos mu? Ayrik kume mi?  
Out[10]: False
```

issubset = subset'i mi? Altkümesi mi? sorusunu yapar.

```
In [11]: set1.issubset(set2) #set1 set2'nin subset'i mi?  
Out[11]: True
```

issuperset = Kapsar mı?

```
In [13]: set2.issuperset(set1) #set2 set1'in superset'i mi? Kapsar mi?  
Out[13]: True
```

Veri Yapıları Özeti

Listeler	Tuple	Sözlük	Setler
Değiştirilebilir	Değiştirilemez	Değiştirilebilir	Değişebilir
Sıralı	Sıralı	Sırasız	Sırasız + Eşsizdir
Kapsayıcı	Kapsayıcı	Kapsayıcı	Kapsayıcıdır

Python Programlama Alıştırmalar – 4

Soru 1:

Aşağıdakilerden hangisi listelerin özelliklerinden değildir?

Kapsayıcıdır

Değiştirilemez

Sıralıdır

Index işlemleri yapılabilir

Soru 2:

Aşağıdakilerden hangisi tupleların özelliklerinden değildir?

Değiştirilemezdir

Değiştirilebilirdir

Kapsayıcıdır

Sıralıdır

Soru 3:

Aşağıdakilerden hangisi sözlük özelliklerinden değildir?

Kapsayıcıdır

Sıralıdır

Sırasızdır

Değiştirilebilirdir

Soru 4:

Aşağıdakilerden hangisi setlerin özelliklerinden değildir?

Sırasızdır

Değiştirilemezdir

Değerleri eşsizdir

Değiştirilebilirdir

Soru 5:

Bir liste tanımlanmak istendiğinde aşağıdakilerden hangisini kullanılır?

" "

()

{}

[]

Soru 6:

"()" ifadesi ile tanımlanan veri yapısı aşağıdakilerden hangisidir?

liste

tuple

vektör

sözlük

Soru 7:

"{}" ifadesi ile tanımlanan veri yapısı aşağıdakilerden hangisidir?

sözlük (dictionary)

liste

tuple

vektör

Soru 8:

`liste = ["A", "B", "C"]`

Yukarıdaki listeye "D" ifadesini eklemek için aşağıdakilerden kodlardan hangisini yazmak gereklidir?

`liste + "D"`

`liste["D"]`

`liste.append("D")`

`liste.insert("D")`

Soru 9:

`liste = ["A", "B", "C"]`

Yukarıdaki listeye "D" ifadesini 0. indekse eklemek için aşağıdakilerden hangisini yazmak gereklidir?

`liste[0] = "D"`

`liste.insert("D")`

`liste.append(0, "D")`

`liste.insert(0, "D")`

Soru 10:

Verilen "sozluk" ismindeki veri yapısının içerişine key ve value değerleri ile birlikte yeni bir eleman nasıl eklenir?

```
1 | sozluk = {"reg" : "regresyon modeli",
2 |   "loj" : "lojistik regresyon",
3 |   "cart" : "classification and regression trees"}
```

sozluk["gbm"] = "gradient boosting machines"

sozluk + "gbm"

sozluk[0] + "gbm"

sozluk[0] = "gradient boosting machines"

Python Programlama Alistirmalar – 5

Soru 1:

Verilen "sozluk" ismindeki nesne içerisinde LOJ ifadesinin MSE değerine nasıl ulaşılır?

```
1 | sozluk = {
2 |
3 |   "REG" : {"RMSE": 10,
4 |             "MSE": 11,
5 |             "SSE": 12},
6 |
7 |   "LOJ" : {"RMSE": 111,
8 |             "MSE": 2222,
9 |             "SSE": 333},
10 |
11 |   "CART" : {"RMSE": 99,
12 |              "MSE": 00,
13 |              "SSE": 66}}
```

sozluk["LOJ" = "MSE"]

sozluk["LOJ"]["MSE"]

sozluk["LOJ":"MSE"]

sozluk["LOJ","MSE"]

Soru 2:

Verilen örnek kodun çıktısı nedir?

```
1  sozluk = {"REG" : {"RMSE": 10,
2   "MSE": 11,
3   "SSE": 12},
4
5   "LOJ" : {"RMSE": 111,
6   "MSE": 2222,
7   "SSE": 333},
8
9   "CART" : {"RMSE": 99,
10  "MSE": 00,
11  "SSE": 66}
12
13
14  sozluk["CART"]["SSE"]
```

11

00

66

111

Soru 3:

Verilen örnek kod ile yapılan işlem nedir?

```
set([1,3,6,19])
```

liste oluşturulmuştur

tuple oluşturulmuştur

liste üzerinden set oluşturulmuştur

tuple üzerinden liste oluşturulmuştur

Soru 4:

Verilen kodun çıktısı nedir?

```
1 | set1 = set([5,7,9])
2 | set2 = set([5,6,7])
3 | set2.difference(set1)
```

{6,9}

6

5

{6}

Soru 5:

Verilen kodun çıktısı nedir?

```
1 | set1 = set([5,7,9])
2 | set2 = set([5,6,7])
3 |
4 | set1.difference(set2)
```

{9}

{6,9}

9

6

Soru 6:

Verilen örnek kodun çıktısı nedir?

```
1 | set1 = set([5,7,9])
2 | set2 = set([5,6,7])
3 |
4 | set1.symmetric_difference(set2)
```

{5}

{6,9} ya da {9,6}

5,6

{5,6}

Soru 7:

Verilen örnek kodun çıktısı nedir?

```
1 | set1 = set([5,7,9])
2 | set2 = set([5,6,7])
3 | set1.union(set2)
```

{5,6}

{5,6,9}

{5,7,9}

{5,6,7,9}

Soru 8:

Verilen örnek kodun çıktısı nedir?

```
1 | liste = [1,1,2,3,4,5,1,2,1]
2 | liste.count(1)
```



4



'1,1,1,1'



1111



Çalışmaz

Soru 9:

Bir listeye index sırasına göre eleman eklemek için hangi metod kullanılır.



pop()



reverse()



insert()



extend()

Soru 10:

Verilen örnek kodun çıktısı nedir?

```
1 | liste = [10,20,30,40]
2 | liste.pop(1)
3 | liste
```

10

20

[10,30,40]

'10,20,30'

Python Programlama Alistirmalar – 6

Soru 1:

Verilen örnek kodun çıktısı nedir?

```
1 | liste = ["a","b","c"]
2 | liste.extend(liste)
3 | liste
```

Hata üretir

['a', 'b', 'c']

['a', 'b', 'c', 'a', 'b', 'c']

['c', 'b', 'a']

Soru 2:

Bir listeden index sırasına göre eleman silmek için hangi metod kullanılır.

pop()

reverse()

insert()

append()

Soru 3:

Verilen örnek kodun çıktısı nedir?

```
1 | liste = ["a", "b", "c"]
2 | liste.reverse()
3 | liste
```

'abc'

['a', 'b', 'c']

['a', 'b', 'c', 'a', 'b', 'c']

['c', 'b', 'a']

Soru 4:

Verilen örnek kod parçasının çıktısı nedir?

```
1 | t = ("a", 10, "b")
2 | t[0] = 1
```

Hata üretir

('a', 10, 'b')

('1', 10, 'b')

('a', 1, 'b')

Soru 5:

Verilen kod parçasının çıktısı nedir?

```
1 | liste = ["a", "b", "c"]
2 | liste.index("b")
```

1

["a", "b", "c", "b"]

["a", "c"]

["b"]

Soru 6:

Verilen kod parçasının çıktısı nedir?

```
1 | liste = [50,10,30,40]
2 | liste.sort()
3 | liste
```

50

[10,30,40,50]

[50,10,30,40]

[50,40,30,10]

Soru 7:

Verilen kod parçasının çıktısı nedir?

```
1 | liste = [10,10,20,40]
2 | liste.clear()
3 | liste
```

[10,20,40]

''

[]

..

Soru 8:

İki kümenin kesişiminin boş olup olmadığı sorgulanması için hangi metod kullanılır?

dir()

isdisjoint()

issubset()

isuperset()

Soru 9:

Bir kümenin tüm elemanlarının başka bir küme içerisinde yer alıp olmadığı hangi metod ile kontrol edilir?

dir()

isdisjoint()

issubset()

isuperset()

Soru 10:

Bir kümenin bir diğer kümeyi tamamen kapsayıp kapsamadığını kontrol etmek için hangi metod kullanılır.

isuperset()

dir()

isdisjoint()

direction()

Fonksiyonlar

Fonksiyon Nedir?

Belirli amaçları yerine getiren işaretlerdir.

Matematiksel İşlemler

```
In [14]: 4*4
```

```
Out[14]: 16
```

```
In [15]: 4/4
```

```
Out[15]: 1.0
```

```
In [16]: 4-2
```

```
Out[16]: 2
```

```
In [17]: 4+2 # bunlar klasik matematiksel operatorlerdir.
```

```
Out[17]: 6
```

Üs Alma

$3^{**}2 \rightarrow 3^2$ anlamına gelir.

```
In [18]: 3**2 # 3'un 2'nci kuvveti
```

```
Out[18]: 9
```

```
In [19]: 3**3 # 3'un 3'ncu kuvveti
```

```
Out[19]: 27
```

Fonksiyon Nasıl Yazılır ?

`def` ile fonksiyon oluşturacağımızı belirtiriz.

```
# =====#
# #Fonksiyon Nasil Yazilir?

def kare_al(x):
    print(x**2) # def ile fonksiyon olusturacagimizi belirtiriz.

kare_al(5) #fonksiyonu bu sekilde calistiririz.

# =====#
```

```
In [21]: def kare_al(x):
...:     print(x**2) # def ile fonksiyon olusturacagimizi belirtiriz.
...:
...:
```

```
In [22]: kare_al(5) #fonksiyonu bu sekilde calistiririz.
```

```
25
```

Bilgi Notuyla Çıktı Üretmek

```
#Bilgi notuyla çıktı üretme
def kare_al(x):
    print("Girilen sayinin karesi : " + x**2) #str + int

kare_al(3) #hata aldık cunku str ifadeler sadece str ifadeler ile birleştirilebilir.
```

Bu fonksiyon çalıştırınca aldığımız hata :

```
In [17]: kare_al(3) #hata aldık cunku str ifadeler sadece str ifadeler ile birleştirilebilir.
Traceback (most recent call last):

  File "<ipython-input-17-31e075573f9a>", line 1, in <module>
    kare_al(3) #hata aldık cunku str ifadeler sadece str ifadeler ile birleştirilebilir.

  File "<ipython-input-16-4cc719a79d0b>", line 2, in kare_al
    print("Girilen sayinin karesi : " + x**2) #str + int

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

str ifadeler ile sadece str ifadeler birleştirilebilir!

type dönüşümü yapmalıyız.:

```
In [19]: def kare_al(x):
...:     print("Girilen sayinin karesi : " + str(x**2)) #str + str(type dönüşümü)
...:
...:

In [20]: kare_al(3) #bu kez hata almadan calisti.
Girilen sayinin karesi : 9
```

Başka bir örnek:

```
In [21]: def kare_al(x):
...:     print("Girilen sayi: " + str(x)
...:          +"\nKaresi: "+str(x**2)) #\n ile alt satira gectik.
...:
...:

In [22]: kare_al(4)
Girilen sayi: 4
Karesi: 16
```

İki Argümanlı Fonksiyon Tanımlamak

```
In [1]: def carpma_yap(x,y):
...:     print("Birinci sayi: "+ str(x)
...:          + "\nIkinci sayi: "+ str(y)
...:          + "\nCarpimlari: "+str(x*y))
...:
...:

In [2]: carpma_yap(3,4)
Birinci sayi: 3
Ikinci sayi: 4
Carpimlari: 12
```

Ön Tanımlı Argümanlar

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

```
In [8]: def carpma_yap(x,y=1): # y=1 demeseydik iki degeri de girmek zorunda kalirdik.  
...:     print(x*y)  
...:  
...:  
  
In [9]: carpma_yap(3) #Hata vermeden calisacak.  
3  
  
In [10]: carpma_yap(3,5) #yeni bir deger girdigimizde eski degeri ezeriz.  
15
```

y=1 yazarak ön tanımlı bir argüman oluşturmuş olduk.

Argümanların Sıralaması

Argümanların sırasını bilmediğimiz fakat isimlerini bildiğimiz zaman aşağıdaki şekilde çalıştırabiliriz.

```
# Argumanların Sıralaması  
  
def carpma_yap(x,y):  
    print(x*y)  
  
carpma_yap(y=2, x=4) # Argumanların sırasını bilmiyorsam ama isimlerini biliyorsam  
                      # Bu şekilde calistirabiliriz.
```

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.

```
#Fonksiyonlar ne zaman yaizlir?  
def direk_hesap(isi, nem, sarj):  
    print((isi+nem)/sarj)  
  
direk_hesap(25,40,70)
```

<pre>In [14]: direk_hesap(25,40,70)</pre>	<pre>0.9285714285714286</pre>
---	-------------------------------

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 Ciktilarini Girdi Olarak Kullanmak
#Fonksiyonun ciktisini baska bir yerde girdi olarak kullanmak icin
# return ifadesini kullanmaliyiz.

def direk_hesap(isi, nem, sarj):
    print((isi+nem)/sarj) #print ekrana ckti verir. Programlama anlaminda
                           #kullanilabilegi anlamina gelmez.

cikti = direk_hesap(25,40,70)
cikti #fonksiyonun sonucunu cikti'ya atayamadik
```

```
In [24]: def direk_hesap(isi, nem, sarj):
...:     return (isi+nem)/sarj #return ifadesini kullanırsak sonucu kullanabiliriz.
...:
...:

In [25]: cikti = direk_hesap(25,40,70)

In [26]: cikti
Out[26]: 0.9285714285714286
```

Fonksiyon **return** ifadesine gelince durur:

```
def direk_hesap(isi, nem, sarj):
    return
    (isi+nem)/sarj # bu sekilde calistirırsak fonksiyon islevini yapmaz.
                   # cunku fonksiyon return'un oldugu satira gelince durur.

direk_hesap(25,40,70)
```

Local ve Global Değişkenler

Ana çalışma alanımızdaki değişkenler **Global** değişkenlerdir.

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

```
#Local ve Global Degiskenler

x=10
y=10 #Ana calisma alanimizdaki degiskenler Global degiskenlerdir.

def carpma(x,y):
    return x*y #fonksiyon icerisindeki degiskenler Local degiskendir.

carpma(2,3)
```

Local Etki Alanından Global Etki Alanını Değiştirme

Yazmış olduğumuz bir döngü içerisinde ya da tanımlamış olduğumuz bir fonksiyon içerisinde 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 içerisinde (local'de) tarar, bulamazsa global alana çıkacak. Global alanda o değişkeni bulursa ona etki edecek (Orada da bulamazsa hata verecektir.). Aşağıdaki örnekte bu durumu gözlemlleyebiliriz.

```
In [1]: x=[] #bos bir liste olusturuldu

In [2]: def eleman_ekle(y):
...:     x.append(y) #x'e y'yi ekle.
...:     print(str(y)+" ifadesi eklendi."
...:           +"\nListenin yeni hali: "+str(x))
...:
...:

In [3]: eleman_ekle(4)
4 ifadesi eklendi.
Listenin yeni hali: [4]

In [4]: eleman_ekle(3)
3 ifadesi eklendi.
Listenin yeni hali: [4, 3]
```

NOT=

Argüman sayısı bilinmiyorsa argüman isminden önce ***** ekleyin

```
def my_function(*kids): #Arguman sayisi bilinmiyorsa arguman isminden once * ekleyin.
    print("The youngest child is " + kids[-1])

my_function("Emil", "Tobias","Linus")
```

```
The youngest child is Linus
```

Karar-Kontrol Yapıları (Koşullar)

Koşul Nedir?

Örneğin günlük hayatı 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.

```
In [3]: sinir = 5000 #sinir degiskene deger verdik

In [4]: sinir == 4000 #sinir=4000'mu? sorusu sorar. False
Out[4]: False

In [5]: sinir == 5000
Out[5]: True
```

if – else – elif

if eğer anlamındaki koşuldur.

Eğer yazdığımız sorgu true ise alt satırı geçer ve çalışır.

```
sinir = 50000
gelir = 40000

gelir < sinir #True

if gelir < sinir: #sorgu true ise if alt satira gecer ve calisir.
    print("Gelir sinirdan kucuk.")
```

if = eğer true ise if çalışır.

else= değilse else çalışır.

```
In [14]: sinir = 50000
...: gelir = 40000

In [15]: if gelir > sinir: #sorgu true ise if'i calistirir.
...:     print("gelir sinirdan buyuk")
...: else: # sorgu false ise else'i calistirir.
...:     print("gelir sinirdan kucuk")
...:
...:
gelir sinirdan kucuk
```

```

if gelir==sinir:
    print("gelir sinira esittir.")
else:
    print("gelir sinira esit degildir.")

```

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

Name	Type	Size	
gelir1	int	1	60000
gelir2	int	1	50000
gelir3	int	1	35000
sinir	int	1	50000

```

In [22]: if gelir1 < sinir:
...:     print("Geliriniz sinirdan kucuk!!")
...: elif gelir1 == sinir:
...:     print("Geliriniz sinirda.")
...: else:
...:     print("Tebrikler. Geliriniz sinirdan yukarida.")
...:
...:
Tebrikler. Geliriniz sinirdan yukarida.

In [23]: if gelir2 < sinir:
...:     print("Geliriniz sinirdan kucuk!!")
...: elif gelir2 == sinir: #if kosulu saglanmadıysa elif'e bakılır.
...:     print("Geliriniz sinirda.")
...: else: #hic bir kosul saglanmıyorsa else calisır.
...:     print("Tebrikler. Geliriniz sinirdan yukarida.")
...:
...:
Geliriniz sinirda.

In [24]: if gelir3 < sinir:
...:     print("Geliriniz sinirdan kucuk!!")
...: elif gelir3 == sinir: #if kosulu saglanmadıysa elif'e bakılır.
...:     print("Geliriniz sinirda.")
...: else: #hic bir kosul saglanmıyorsa else calisır.
...:     print("Tebrikler. Geliriniz sinirdan yukarida.")
...:
...:
Geliriniz sinirdan kucuk!!

```

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.

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

sinir = 50000
magaza_adi=input("Magaza adı nedir?\n ") #kullanicidan magaza_adi aldig
gelir = int(input("Gelirinizi giriniz: ")) #kullanicidan aldigimiz geliri int'e cevirdik.

if gelir > sinir:
    print("Tebrikler "+magaza_adi+ " Geliriniz sinirdan yüksek :)")
elif gelir == sinir:
    print(magaza_adi+" Geliriniz sinirda.")
else:
    print("Uyarı! "+magaza_adi +" Çok dusuk gelir: "+str(gelir))
```

Program çıktıları:

Magaza adı nedir? A Mağazası	Magaza adı nedir? B Mağazası
Gelirinizi giriniz: 35000 Uyarı! A Mağazası Çok dusuk gelir: 35000	Gelirinizi giriniz: 50000 B Mağazası Geliriniz sinirda.
Magaza adı nedir? C Mağazası	
Gelirinizi giriniz: 65000 Tebrikler C Mağazası Geliriniz sinirdan yüksek :)	

Döngüler

For Döngüsü

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

```
# For Dongusu

ogrenci = ["ali","veli","isik","berk"]

ogrenci[0]
ogrenci[1]

for i in ogrenci: #i gecici degiskendir.
    print(i) →
```

ali
veli
isik
berk

Döngü ve Fonksiyonların Birlikte Kullanımı

```
maaslar=[1000,2000,3000,4000,5000]
```

Maaşlara %20 zam yapılacak. Gerekli kodlar nelerdir?

```
#maaslara %20 zam yapılacak gerekli kodları yazınız.

def yeni_maas(x):
    print(x*1.20)

yeni_maas(1000) #fonksiyonun çalışmasına örnek.

for i in maaslar:
    yeni_maas(i)
```

```
1200.0
2400.0
3600.0
4800.0
6000.0
→
```

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.

```
#if, for ve fonksiyonların bir arada kullanımı

maaslar=[1000,2000,3000,4000,5000]

def maas_ust(x):
    print(x*1.10) # %10 zam

def maas_alt(x):
    print(x*1.20) # %20 zam

for i in maaslar:
    if i>=3000: #maas 3000'den fazla veya esit ise
        maas_ust(i) # %10 zam uygulanacak
    else:          #değilse
        maas_alt(i) # %20 zam uygulanacak
```

```
...
1200.0
2400.0
3300.0
4400.0
5500.0
→
```

break & continue

Döngüler içersinde belirli bir şartı sağlayan ifadeler yakalandığında (if döngüsü ile yakalıyoruz.) 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.

```
#break & continue

maaslar=[8000,5000,2000,1000,3000,7000,1000]

maaslar.sort() #Karışık yazılmış listeyi küçükten büyük'e sıraladık.
maaslar
```

```
In [7]: for i in maaslar:  
...:     if i ==3000: #1000,1000,2000 gecti 3000'e geldi if'e girdi. Durdu.  
...:         print("kesildi")  
...:         break  
...:     print(i)  
...:  
...:  
1000  
1000  
2000  
kesildi
```

Örneğin; 3000'i atlayıp devam etsin.

```
In [8]: for i in maaslar:  
...:     if i ==3000: #1000,1000,2000 gecti 3000'e geldi if'e girdi. Atladi.  
...:         print("atlandi.")  
...:         continue  
...:     print(i)  
...:  
...:  
1000  
1000  
2000  
atlandi.  
5000  
7000  
8000
```

while

Şart sağlandığı sürece devam eden bir döngüdür.

```
In [8]: sayi=1  
...:  
...: while sayi<10: #Sayi 10'a gelene kadar bu islemi devam ettir.  
...:     sayi += 1 #sayiyi 1 arttırır ve sayi degerine ata.  
...:     print(sayi)  
...:  
...:  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

Python Programlama Alıştırmalar - 7

Soru 1:

Aşağıdakilerden hangisi fonksiyon tanımlamak için kullanılır?

definition

func

def

function

Soru 2:

Aşağıdaki verilen kod ne işe yarar?

?print

print fonksiyonu çağırılır

print fonksiyonu hakkında bilgi alma imkanı sağlar

Böyle bir kod yoktur çalışmaz

Boş bir çıktı verir

Soru 3:

Verilen kod parçasında bir fonksiyon tanımlanmıştır. Tanımlanan fonksiyon işlevini yerine getirmek adına nasıl kullanılır?

```
1 | def kup_al(x):  
2 |     print(x**3)
```

kup_al

kup_al()

print(kup_al())

kup_al(2)

Soru 4:

Verilen kodun çıktısı nedir?

```
1 | def yazdir(metin):
2 |     print(metin, "yazanlar")
3 |
4 | yazdir("gelecegi")
```

gelecegi yazanlar

metin

yazanlar

gelecegi

Soru 5:

Verilen kodun çıktısı nedir?

```
1 | def islem(x, y):
2 |     print(x + y)
3 |
4 | islem(1,9)
```

1

10

0

9

Soru 6:

Verilen kodun çıktısı nedir?

```
1 | def islem(x, y):  
2 |     print(x - y)  
3 |  
4 | islem(3)
```

3

13

Kod çalışır ama çıktı üretmez

İşlem hata üretir

Soru 7:

Verilen kod parçası çalıştırıldığında hata verecektir. Bu hatanın önüne geçmek adına **fonksiyon tanımlama esnasında** ne yapmak gerekir.

```
1 | def islem(x, y):  
2 |     print(x - y)  
3 |  
4 |  
5 | islem(3)
```

İki argüman değeri de girilmelidir

y argümanına ön tanımlı değer verilmelidir

return eklenmelidir

print kaldırılmalıdır

Soru 8:

Verilen kodun çıktısı nedir?

```
1 | def harf_say(x):  
2 |     len(x)  
3 |  
4 |     harf_say("Merhaba!")
```

Kod çalışır ama çıktı üretmez

Merhaba!

8

7

Soru 9:

Verilen kod parçası çalışacak fakat çıktı üretmeyecektir. Kodun kullanılabilir bir çıktı üretmesi için ne yapmak gereklidir?

```
1 | def harf_say(x):  
2 |     len(x)  
3 |  
4 |     harf_say("Merhaba!")
```

Fonksiyon argümansız çalıştırılmalıdır

Fonksiyon tanımlama bölümüne ek argüman eklenmelidir

len yerine başka bir fonksiyon kullanılmalıdır

return ifadesi kullanılmalıdır

Soru 10:

Verilen kodun çıktısı nedir?

```
1 | def islem(x):
2 |     if (x<0):
3 |         return "NO"
4 |     else:
5 |         x*5
6 |
7 | islem(2)
```

Kod çalışır çıktı üretmez

10

YES

NO

Python Programlama Alıştırmalar - 8

Soru 1:

Verilen kodun çıktısı nedir?

```
1 | def islem(x):
2 |     if (x>10):
3 |         return "YES"
4 |     else:
5 |         return x*5
6 |
7 | islem(4)
```

Çalışmaz

NO

YES

20

Soru 2:

Verilen listenin her bir elemanını iteratif bir şekilde yakalayıp belirli bir işleme tabi tutmak için hangi yapı kullanılır?

Lambda yapısı

for yapısı

if

Index işlemleri

Soru 3:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | a = [2,4,6,8]
2 |
3 | for i in a:
4 |     print(i**2)
```

[2,4,6,8]

[4,8,12,16]

[4,16,36,64]

4
16
36
64

Soru 4:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | sayilar = [10,20,30]
2 |
3 | for i in sayilar:
4 |     if i > 20:
5 |         print(i/2)
```

Çalışmaz

15.0

20

5

Soru 5:

Verilen kodun çıktısı nedir?

```
1 | urun_fiyatlari = [19,29,39]
2 |
3 | for i in urun_fiyatlari:
4 |     if i >= 30:
5 |         print(i/2)
6 |     else:
7 |         print(i*0)
```

19

29

39

9.5

14.5

0

0

0

19.5

9

14

19

Soru 6:

Verilen kod parçasının çıktısı ne olacaktır?

```
1 | a = [1,2,3]
2 | b = []
3 | for i in a:
4 |     b.append(i**2)
5 |
6 | b
```

[1, 4, 9]

Çalışmaz

[1,2,3]

[2,4,6]

Soru 7:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | def mesaj():
2 |     print("Merhaba!")
3 |
4 | mesaj()
```

Hata üretir

Çalışır ama çıktı üretmez

Merhaba

Merhaba!

Soru 8:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | for i in ["a",11]:  
2 |     print(i)
```

11

a

Çalışmaz

a

11

Soru 9:

Verilen kod parçasının çıktısı ne olacaktır?

```
1 | def harf_say(x):  
2 |     return len(x)  
3 |  
4 | harf_say("Merhaba!")
```

7

8

Kod çalışmaz

Kod çalışır ama çıktı vermez

Soru 10:

break ifadesi ne için kullanılır?

Kod akşini kesmek için (Örneğin bir şart yakalandığında çalışmayı durdur demek gibi)

Bir şart yakalandığında ekrana yazdırınmak için

Bir şart yakalandığında ona bir işlem yapmak için

Yakalanan şartı atlayarak işleme devam etmek için

Python Programlama Alıştırmalar – 9

Soru 1:

continue ifadesi ne için kullanılır?

Bir şart yakalandığında ona bir işlem yapmak için

Yakalanan şartı atlayarak işleme devam etmek için

Bir şart yakalandığında ekrana yazdırınmak için

Yakalanan şartla gelindiğinde çalışmayı durdurmak için

Soru 2:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 sayilar = [10,20,30,40]
2
3 for i in sayilar:
4     if i == 30:
5         break
6     print(i)
```

10

10
20

10
 20
30

10
 20
40

Soru 3:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 A = []
2
3 for i in [1,2,3,4]:
4     A.append(i)
5
6
7 A[0]
```

1

3

4

[1]

Soru 4:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 sayilar = [10,20,30,40]
2
3 for i in sayilar:
4     if i == 30:
5         continue
6     print(i)
```

- 10
 20
 40

- 10

- 10
 20

- 30
 40

Soru 5:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 if [1,2,3,4][2] == 2:
2     print("YES")
3 else:
4     print("NO")
```

- NO

- YES

- 2

- 1

Soru 6:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | if [1,2,3,4][1] == 2:  
2 |     print("YES".lower())  
3 | else:  
4 |     print("NO")
```

no

yes

YES

NO

Soru 7:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | A = "*A*"  
2 | if type(A) == str:  
3 |     A = A.strip("*")  
4 |     print(A)
```

A

A

A

Hata üretir

Soru 8:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 A = 12
2
3 if type(A) == str:
4     A = A.strip("*")
5     print(A)
6 else:
7     A = "*"+str(A)+"*"
8     print(A.strip())
```

Hata üretir

A

A

12 Çünkü strip() argümanı sadece str ifadelerde çalışır.

Soru 9:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 A = []
2 B = []
3
4
5 for i in [1,"a",12,"b"]:
6     if type(i) == int:
7         B.append(i)
8     else:
9         A.append(i)
10
11 A[1]
```

1

12

'a'

'b'

Soru 10:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | def islem(x,y):
2 |     A = [x,y]
3 |     return A[0] + A[1]
4 |
5 | islem(1,3)
```

2

1

4

Hata üretir

Object Oriented Programming

Class'lara Giriş ve Class Tanımlamak

Class Nedir?

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

```
#Siniflar
class VeriBilimci(): #Class tanimlama
    print("Bu bir class'dir.")
```

Class Özellikleri

```
#Siniflarin Ozellikleri
class VeriBilimci(): #Class tanimlama
    bolum = ''
    sql = 'Evet'
    Deneyim_Yili = 0
    bildigi_diller=[]
```

Class Özelliklerine Erişmek

```
#Siniflarin Ozelliklerine Erismek
VeriBilimci.sql
VeriBilimci.Deneyim_Yili
```

Class Özelliklerini Değiştirmek

```
#Siniflarin Ozelliklerini Degistirmek
VeriBilimci.sql = 'Hayir'
VeriBilimci.sql #Ozellikin degeri degisti.
```

Class Örneklendirmesi (instantiation)

Sınıfın özelliklerini barındıran alt kümeler oluşturma işlemine sınıf örneklendirmesi denir.

```
#Sınıf Orneklendirmesi (instantiation)

ali = VeriBilimci() #VeriBilimci sınıfının özelliklerini taşıyan bir birim olustu.
                      #Yani orneklemeye yapmış oldum.

ali.sql
ali.bildigi_diller.append("Python") #ali'nin bildigi_diller'e Python ekledik.
                                      #Ancak bu class'in hepsini etkiledi.
ali.bildigi_diller

veli = VeriBilimci()
veli.bildigi_diller #ali'nin bildigi dillere python eklemistik ancak veli'nin
                    #bildigi dillerde de python oldu.
```

Örnek Özellikleri

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

Sınıflar için tanımlanan özellikler örnekler için değişimde bir formata getirilmedikçe bir örnekte yapılan değişiklik tüm örneklerde 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.

```
#Ornek Ozellikleri

class VeriBilimci(): #yeni bir sınıf tanımladık
    bildigi_diller = ["R","Python"] #Tüm class için özellik ataması.
    bolum = ''
    def __init__(self): #Örneklerde ayrı ayrı özellik ataması yapmak için.
        self.bildigi_diller = []
        self.bolum = ''

ali = VeriBilimci()
ali.bildigi_diller #bos

veli = VeriBilimci()
veli.bildigi_diller #bos

ali.bildigi_diller.append("Python") #ali'nin bildiği dillere ekleme yaptı.
ali.bildigi_diller #Bu kez python var.

veli.bildigi_diller.append("R") #veli'nin bildiği dillere ekleme yaptı.
veli.bildigi_diller #sadece veli'ye ekledigimiz R var.

VeriBilimci.bildigi_diller #Classın genelinde R ve Python var.

VeriBilimci.bolum # ''
ali.bolum = 'Istatistik'
veli.bolum = 'bil_sis_muh'
veli.bolum #bil_sis_muh
ali.bolum #istatistik
```

Örnek Metodları

Mesela her bir veri bilimci için bir yeni öğrenilen dili o veri bilimcinin bildiği dillere ekleme işlemi yapın.

Örnekler üzerinde çalışan fonksiyonlar yazmak istiyoruz.

```
# Ornek Metodlari

class VeriBilimci(): #Bir class tanimladik.
    calisanlar = [] # calisanlar adinda bir nesne
    def __init__(self): #orneklerin ozellikleri
        self.bildigi_diller = [] #orneklerin ozellikleri
        self.bolum = '' #orneklerin ozellikleri
    def dil_ekle(self, yeni_dil): #orneklere etki edecek bir fonksiyon yazdik
        self.bildigi_diller.append(yeni_dil)

ali = VeriBilimci()
ali.bildigi_diller #suan bos
ali.bolum

veli = VeriBilimci()
veli.bildigi_diller
veli.bolum

ali.dil_ekle("R") #dil_ekle fonksiyonunu calistirdik.

VeriBilimci.dil_ekle(ali,"Python") #dil_ekle fonksiyonunu calistirdik.
                                    #ali'nin bildigi dillere python eklendi.
                                    #dil_ekle fonksiyonu iki sekilde de calistirilabilir.

ali.bildigi_diller #Python ve R var.
```

Miras Yapıları (inheritance)

Başa yerde başka bir class tanımlarken, tanımlayacak olduğumuz bu class daha önceden tanımlamış olduğumuz başka bir class'ın özelliklerini barındırıyorsa ve biz bunları kullanmak istiyorsak eski class'ın özelliklerini miras olarak kullanabiliyoruz.

```
# Miras Yapilari (inheritance)

class Employees():
    def __init__(self,FirstName, LastName, Address):#Ozellikleri fonksiyonel. Sabit degil.
        self.FirstName = FirstName
        self.LastName = LastName
        self.Address = Address

class DataScience(Employees): #Employees'den miras aliyor.
    def __init__(self,Programming):#Ozellikleri fonksiyonel. Sabit degil.
        self.Programming = Programming

class Marketing(Employees): #Employees'den miras aliyor.
    def __init__(self,StoryTelling):#Ozellikleri fonksiyonel. Sabit degil.
        self.StoryTelling = StoryTelling

veribilimci1 = DataScience() #Parametreyi bos bırakamayız. Hata verir.
veribilimci1 = DataScience("Python")
veribilimci1.Programming #Python

pazarlamaci = Marketing("Yes")
pazarlamaci.StoryTelling #Yes
```

Functional Programming

Fonksiyonel Programlamaya Giriş

Python dili ile bir program yazmak istediğimizde bunu OOP(Nesneye Dayalı Programlama) özellikleri ile de yazabiliyoruz FP(Fonksiyonel Programlama) özellikleri ile de yazabiliyoruz.

Fonksiyonlar dilin baştacıdır. (Birinci sınıf nesnelerdir.)

Yan etkisiz fonksiyonlar. (stateless(durumsuz), girdi-çıkıtı → Ancak bir girdi verdiğimde çıktı üretir. Ve bu çıktı hep aynı olur. Dışarıdan etkilenemez.)

Yüksek seviye fonksiyonlar.

Yan Etkisiz Fonksiyonlar (Pure Functions)

Fonksiyonun bir şekilde dışarı bağımlı olduğu durumlara yan etkili yani impure(saf olmayan) fonksiyon denir.

Örnek-1: Bağımsızlık

```
In [1]: #Yan Etkisiz Fonksiyonlar (Pure Functions) Örnek-1
In [2]: A = 5
In [3]: def impure_sum(b): #saf olmayan fonksiyon. Sonucu A degiskene bagimli.
...:     return b + A
In [4]: def pure_sum(a,b): #saf
...:     return a + b
In [5]: impure_sum(6) #A'yi degistirirsem sonucu degisir.
Out[5]: 11
In [6]: pure_sum(3,4) #Ne yaparsam yapayim sonucu girdilerden baska bir sey ile degismez.
Out[6]: 7
In [7]: A = 9 #eski deger 6 idi.
In [8]: impure_sum(6) #A'yi degistirirsem sonucu degisir. Girdi ayni, sonuc degisti.
Out[8]: 15
```

Örnek-2: Ölümcul Yan Etkiler

```
In [27]: #Örnek-2: Olumcul yan etkiler

In [28]: #00P

In [29]: class LineCounter:
....:     def __init__(self, filename):
....:         self.file = open(filename , 'r')
....:         self.lines = []
....:
....:     def read(self):
....:         self.lines = [line for line in self.file]
....:
....:     def count(self):
....:         return len(self.lines)

In [30]: lc = LineCounter('deneme.txt')

In [31]: print(lc.lines)
[]

In [32]: print(lc.count())
0

In [33]: lc.read()

In [34]: print(lc.lines)
['Bu bir denemedir.\n', '\n', 'asdasd\n', '\n', 'asdfd\n', 'dhhjfhhfg']

In [35]: print(lc.count())
6
```

```
In [36]: #FP

In [37]: def read(filename):
....:     with open(filename, 'r') as f:
....:         return [line for line in f]

In [38]: def count(lines):
....:     return len(lines)

In [39]: example_lines=read('deneme.txt')

In [40]: lines_count = count(example_lines)

In [41]: lines_count
Out[41]: 6

In [42]:
```

İsimsiz Fonksiyonlar (Lambda) (Anonymous Functions)

```
#İsimsiz Fonksiyonlar (Lambda) (Anonymous Functions)

def old_sum(a,b): #Eski tipte bir fonksiyon
    return a+b

new_sum = lambda a,b : a+b #Lambda ile fonksiyon- İsimli Fonksiyon
new_sum(4,5)

sirasiz_liste = [('b',3),('a',8),('d',12),('c',1)]
sirasiz_liste

sorted(sirasiz_liste, key=lambda x: x[1]) #Fonksiyon tanımladık.
#Out: [('c', 1), ('b', 3), ('a', 8), ('d', 12)]
```

Sorted bir fonksiyondu. Birinci argümanı bir nesneydi, listeydi. Elemanları da tuple idi. Bu listeye bir fonksiyon uygulamak istiyoruz. x'e bağlı bir fonksiyon, x olarak kendi içine girilen değerin 1. indexli elemanına ulaşın.

Vektörel Operasyonlar (Vector Operations)

OOP ile iki listeyi çarpmak

```
In [13]: #Vektörel Operasyonlar (Vector Operations)

In [14]: a = [1,2,3,4] #amacımız bu listeler icsindeki her bir elemani birbiriyle carpma
...: b = [2,3,4,5] #yani 1*2,2*3,3*4,4*5
...:           #Listelerimiz tek boyutlu oldugu icin bunlara 'vektor' denir

In [15]: ab = [] #Carpma islemi saklamak icin global alanda bos liste olusturduk.

In [16]: for i in range(0, len(a)): #a'nin uzunlugu kadar i degeri uretecek(0,1,2,3)
...:     ab.append(a[i]*b[i]) #a'nin i'nci elemani ile b'nin i'nci elemanini carp. ab'ye ekle.

In [17]: ab
Out[17]: [2, 6, 12, 20]
```

Functionel Programming ile

Fakat söz konusu matematik, istatistik, veri bilimi, makine öğrenmesi gibi konular olduğunda asla bu tip döngülere vs. girmiyoruz. Vektörel operasyonlara giriyoruz.

```
In [1]: import numpy as np #numpy kutuphanesini calisma ortamima dahil ettim. np kisayolu atadim.

In [2]: a = np.array([1,2,3,4])
...: b = np.array([2,3,4,5])

In [3]: a*b
Out[3]: array([ 2,  6, 12, 20])
```

Fonksiyonel Programlama ile daha az çaba ile aynı sonuca ulaşmış olduk.

Map & Filter & Reduce

Fonksiyona argüman olarak fonksiyon yazmamıza izin veren fonksiyonlara First Class fonksiyon denir.

Map

Verilen bir nesne üzerinde tanımlanacak bir fonksiyonu çalışma imkanı verir.(lambda yani isimsiz fonksiyonu)

```
In [4]: liste = [1,2,3,4,5]

In [5]: for i in liste:
...:     print(i+10) #her elemana 10 ekleyip yazdır
11
12
13
14
15

In [6]: list(map(lambda x:x+10, liste)) #map fonk. ile her elemana 10 ekleyip liste yap.
Out[6]: [11, 12, 13, 14, 15]
```

Filter

filter fonksiyonu iteratif bir nesne alır bu nesne üzerinden başka bir iteratif nesne oluşturulur. Ve iteratif nesne içerisinde aradığı şartın sağlandığı tüm elemanlar listelenir.

Cift sayıları bulan fonksiyonu yazalım.

```
In [9]: liste = [1,2,3,4,5,6,7,8,9,10]

In [10]: list(filter(lambda x:x % 2 == 0, liste)) #2'ye bölümünden kalani 0'a eşit olanları liste.
Out[10]: [2, 4, 6, 8, 10]
```

Reduce

Az önceki filter fonksiyonu bize aradığımız değerleri bulup getirdi. Yani değerler ile ilgili bir işlem yapmadı. Reduce fonksiyonu yine map ve filter'a benzerdir fakat indirgeme işlemi yapar.

```
In [17]: #reduce

In [18]: from functools import reduce

In [19]: liste = [1,2,3,4,5,6,7,8,9,10]

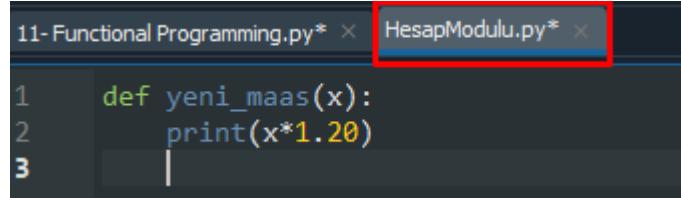
In [20]: reduce(lambda a,b:a+b , liste) #liste elemanlarını toplar.
Out[20]: 55
```

Modül Oluşturma

Bazen modül, bazen kütüphane, bazen de paket dendiğini görebiliriz, bunların üçü de doğrudur. Modüller belirli amaçları yerine getirmek için bir arada bulunan fonksiyonlar topluluğudur.

Maaşlarla ilgili işlemler gerçekleştiren birkaç tane fonksiyonumuz olduğunu düşünelim ve bunu paketleyip bir modül haline getirelim.

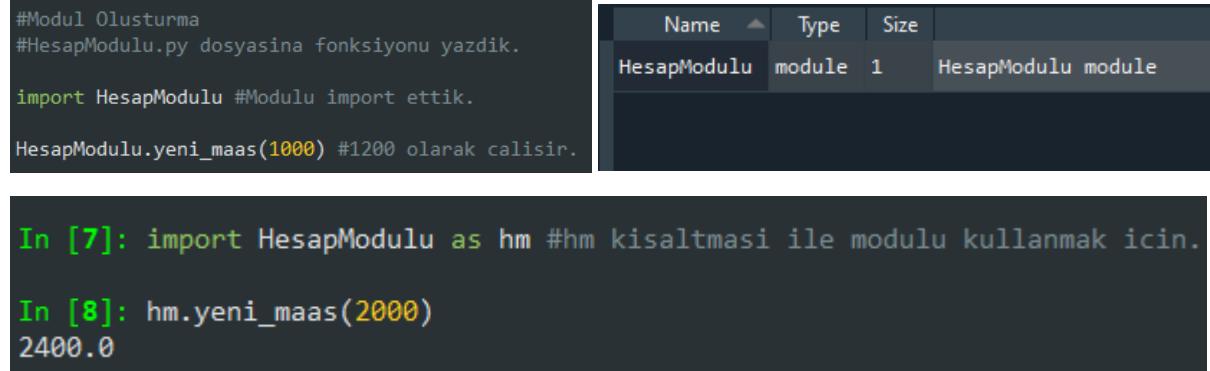
Yeni bir .py dosyası açalım ve ismi HesapModulu.py olsun.



```
11- Functional Programming.py* × HesapModulu.py* ×
1 def yeni_maas(x):
2     print(x*1.20)
3 
```

Modülün içine fonksiyonu yazıp kaydettik. Modülüümüz kullanıma hazır.

Başka bir .py dosyasından modüle erişmek:



```
#Modul Olusturma
#HesapModulu.py dosyasina fonksiyonu yazdik.

import HesapModulu #Modulu import ettik.

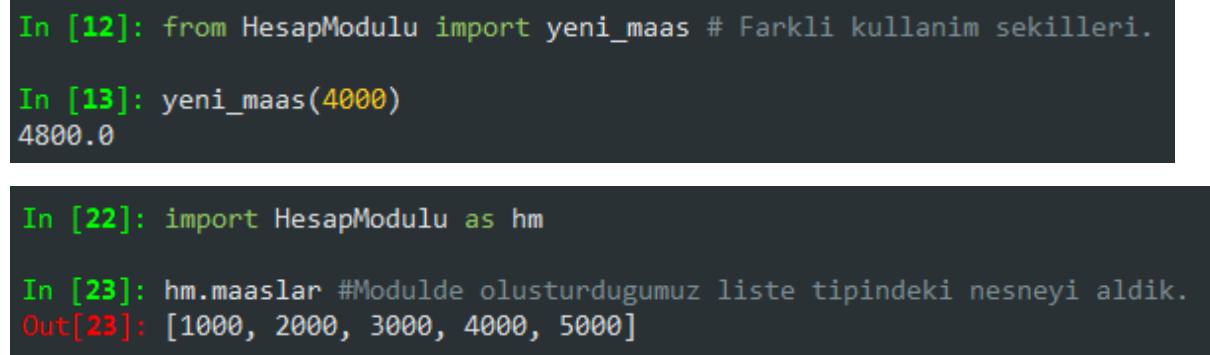
HesapModulu.yeni_maas(1000) #1200 olarak calisir.
```

Name	Type	Size
HesapModulu	module	1
		HesapModulu module

```
In [7]: import HesapModulu as hm #hm kisaltmasi ile modulu kullanmak icin.

In [8]: hm.yeni_maas(2000)
2400.0
```

Daha da kısa kullanımı için:



```
In [12]: from HesapModulu import yeni_maas # Farkli kullanım sekilleri.

In [13]: yeni_maas(4000)
4800.0
```



```
In [22]: import HesapModulu as hm

In [23]: hm.maaslar #Modulde olusturdugumuz liste tipindeki nesneyi aldig.
Out[23]: [1000, 2000, 3000, 4000, 5000]
```

Hatalar/İstisnalar (exception)

1-Programcı hataları: Bunlar basit hatalardır. Syntax hatası gibi.

2-Program hataları / bug: Bunlar kritik hatalardır çünkü program çalışmaya devam eder ancak çıktılar problemlidir. Çıktıların hatalı olmasının tespiti bile bazen zorlayıcı olabilir.

3-İstisnalar (exceptions): Programda bildiğimiz bazı hatalardır fakat bu hatalar gerçekleştiğinde programı durdurma, çalışmaya devam et demenin yoludur. Bunu **try except** yapısı ile sağlarız.

```
In [28]: a=10  
  
In [29]: b=0  
  
In [30]: a/b  
Traceback (most recent call last):  
  
  File "<ipython-input-30-aae42d317509>", line 1, in <module>  
    a/b  
  
ZeroDivisionError: division by zero
```

Gördüğümüz üzere `ZeroDivisionError` hatası ile karşılaştık. 0'a bölünemez.

```
In [28]: a=10  
  
In [29]: b=0  
  
In [30]: a/b  
Traceback (most recent call last):  
  
  File "<ipython-input-30-aae42d317509>", line 1, in <module>  
    a/b  
  
ZeroDivisionError: division by zero  
  
In [31]: try: #kodu dene  
...:     print(a/b)  
...: except ZeroDivisionError: #calismazsa bu hata ile karsilastiginda ne olacak  
...:     print("Payda sıfır olamaz.")  
Payda sıfır olamaz.
```

Python Programlama Alıştırmalar – 10

Soru 1:

Bir sınıf tanımlamak aşağıdakilerden hangisi kullanılır?

def

class

definition

function

Soru 2:

Kod parçasında yer alan “fonksiyonlar” ve “OOP” tanımlamaları ne ifade etmektedir?

```
1 | class BolumSorulari():
2 |     fonksiyonlar = []
3 |     OOP = []
```

Örnek tanımlama

Sınıf tanımlama

Sınıf özelliklerini tanımlama

Kod çalışmaz

Soru 3:

Verilen kod parçacığında yapılan işlem ne anlama gelmektedir?

```
1 | class BolumSorulari():
2 |     fonksiyonlar = []
3 |     OOP = []
4 |
5 |
6 | BolumSorulari.OOP
```

Bir sınıf özelliğine erişilmiştir

Sınıfa erişilmiştir

Özelliklere erişilmiştir

Fonksiyona erişilmiştir

Soru 4:

Verilen kod parçasına göre aşağıdakilerden hangisi bir sınıf örneklenmesidir?

```
1 | class BolumSorulari():
2 |     fonksiyonlar = []
3 |     OOP = []
```

- BolumSorulari.OOP
- BolumSorulari.fonksiyonlar
- BolumSorulari["fonksiyonlar"]
- donguler = BolumSorulari()

Soru 5:

Aşağıdaki fonksiyonel programlama ile ilgili ifadelerden hangisi yanlıştır?

- Fonksiyonlar dilin baş tacıdır
- Isimsiz fonksiyonlar kullanılabilir
- Yan etkili fonksiyonlar vardır
- Vektörel işlemlere imkan sağlanır

Soru 6:

"Ancak bir girdi verildiğinde çıktı üreten fonksiyonlar" ifadesi aşağıdaki fonksiyonel programlama özelliklerinden hangisini işaret etmektedir.

- Vektörel fonksiyonlar
- Döngüsel fonksiyonlar
- İç içe fonksiyonlar
- Yan etkisiz fonksiyonlar

Soru 7:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | fun = lambda x: x**2  
2 | fun(3)
```

Hata üretir

6

3

9

Soru 8:

Verilen kod parçasığının çıktısı aşağıdakilerden hangisidir?

```
list(map(lambda x: x*1, [2,7,4]))
```

7

[2, 7, 4]

2

4

Soru 9:

Verilen kod parçasığının çıktısı aşağıdakilerden hangisidir?

```
1 | a = [1,2,3]  
2 | list(map(lambda x: x*2, a))
```

[1,2,3]

[1,4,9]

[2,4,6]

Çalışmaz

Soru 10:

Var olan sınıfların özelliklerini başka sınıflar için kullanmak için aşağıdakilerden hangisi kullanılır?

Sınıf özellikleri

Miras yapıları

Örnek özellikleri

Örnek metodları

Python Programlama Alıştırmalar – 11

Soru 1:

Aşağıdakilerden hangisi bir modül import etmek için kullanılamaz.

from import modul_ismi

import modul_ismi

import modul_ismi as mi

import modul_ismi as modül

Soru 2:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
list(map(lambda x: x.upper(), ["Ali","Veli","isik"]))
```

[ali, veli, isik]

['ali', 'veli', 'isik']

['Ali', 'Veli', 'Isik']

['ALI', 'VELI', 'ISIK']

Soru 3:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | from functools import reduce
2 | a = [1,2,3,4]
3 | reduce(lambda a,b: a*b, a)
```

24

10

[1,2,3,4]

[1,4,9,16]

Soru 4:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | A = [[1,2],[3,4],[5,6]]
2 | list(map(lambda x: x[0]**3, A))
```

[1, 3, 5]

[2, 4, 6]

[3, 9, 15]

[3, 7, 1]

Soru 5:

Aşağıda verilen for döngüsünde ele alınan matematiksel *işlem* map() fonksiyonu ile nasıl gerçekleştirilir?

```
1 liste = [1,2,3,4]
2 A = []
3
4 for i in liste:
5     A.append(i**2)
6
7 print(A)
```

lambda x: x*2

list(map(lambda x: x**2))

list(map(lambda x: x**2, liste))

lambda x: x**2

Soru 6:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 A = [1,2,3,4,5]
2
3 if type(A) == ():
4     print("islem gecersiz")
5 else:
6     print(list(map(lambda x: x/1, A)))
```

islem geçersiz

Hata üretir

[1.0, 2.0, 3.0, 4.0, 5.0]

[1,1,1,1,1]

Soru 7:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | from functools import reduce  
2 | reduce(lambda a,b: a/b, [8,4,2])
```

1.0

[8,4,2]

[2,4,8]

64.0

Soru 8:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | def yap(x,y,z):  
2 |     try:  
3 |         print(x/y*z)  
4 |     except ZeroDivisionError:  
5 |         print("gecersiz islem")  
6 |  
7 | yap(1,2,0)
```

0.5

1.0

'gecersiz islem'

0.0

Soru 9:

Verilen kod parçası ve çıktı için yazılması gereken kod aşağıdakilerden hangisidir?

```
1 def islem(x,y,z):
2     if y == 0:
3         print("hatali islem")
4     else:
5         return x/y*z
```

Cıktı:

hatali islem

islem(1,2,3)

islem(1,0,2)

islem(1,2,0)

islem(1,1,1)

Soru 10:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 import numpy as np
2 a = np.array([1,1,1])
3 b = np.array([2])
4
5 a+b
```

[2]

[2,2,2]

[3,3,3]

array([3, 3, 3])

Python Programlama Alıştırmalar – 12

Soru 1:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 A = []
2
3 for i in ["ali","veli","isik"]:
4     A.append(i.replace("i","a"))
5
6 print(A)
```

Hata üretir

['ala', 'vela', 'asak']

['aala', 'avela', 'aasak']

['iala', 'ivela', 'iasak']

Soru 2:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
list(filter(lambda x: x < 2, [1,2,3,4,5]))
```

Çalışır ama çıktı üretmez

[1]

[1,2,3]

[]

Soru 3:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | liste = ["a",20,10,30,"b"]
2 | list(filter(lambda x: type(x) == int, liste))
```

[20, 10, 30]

["a","b"]

["a","20"]

["a",20,10,30,"b"]

Soru 4:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
list(filter(lambda x: len(x) > 8, ["pazartesi","sali","carsamba","persembe","cuma"]))
```

['pazartesi']

['sali']

['carsamba']

['persembe']

Soru 5:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
list(map(lambda x: x.capitalize(), ["abc","bcd","cde"]))
```

['bc', 'cd', 'de']

['Ab', 'Bc', 'Cd']

['Abc', 'Bcd', 'Cde']

['ABC', 'BCD', 'CDE']

Soru 6:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | from functools import reduce  
2 | reduce(lambda a,b: a+b, ["a","4","a"])
```

4

'a4a'

4a

a4a

Soru 7:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | A = ["ali","veli","isik"]  
2 | B = [1,2,3]  
3 | AB = [A,B]  
4 |  
5 |  
6 | for i in AB:  
7 |     if type(i[0]) == int:  
8 |         print(list(map(lambda x: x-3, i)))
```

[-2,-1,0]

[1,2,3]

["ali","veli","isik"]

Hata üretir

Soru 8:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
list(map(lambda x: x/10, filter(lambda x: x > 20, [10,20,30,40,50])))
```

[10.0, 20.0, 30.0, 40.0, 50.0]

[10.0, 20.0, 30.0, 40.0, 50.0]

[1.0, 2.0, 3.0, 4.0, 5.0]

[3.0, 4.0, 5.0]

Soru 9:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 A = ["ali","veli","isik"]
2 B = [1,2,3]
3 AB = [A, B]
4
5 for i in AB:
6     if type(i[0]) == str:
7         print(list(map(lambda x: x + " hi", i)))
```

['ali hi', 'veli hi', 'isik hi']

['ali', 'hi', 'veli', 'hi', 'isik', 'hi']

['ali', ' hi', 'veli', ' hi', 'isik', ' hi']

Çalışır ama çıktı üretmez

Soru 10:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | from functools import reduce
2 | A = ["Veri", "Bilimi", "Okulu"]
3 | reduce(lambda a,b: a+b, list(map(lambda x: x[0], A)))
```

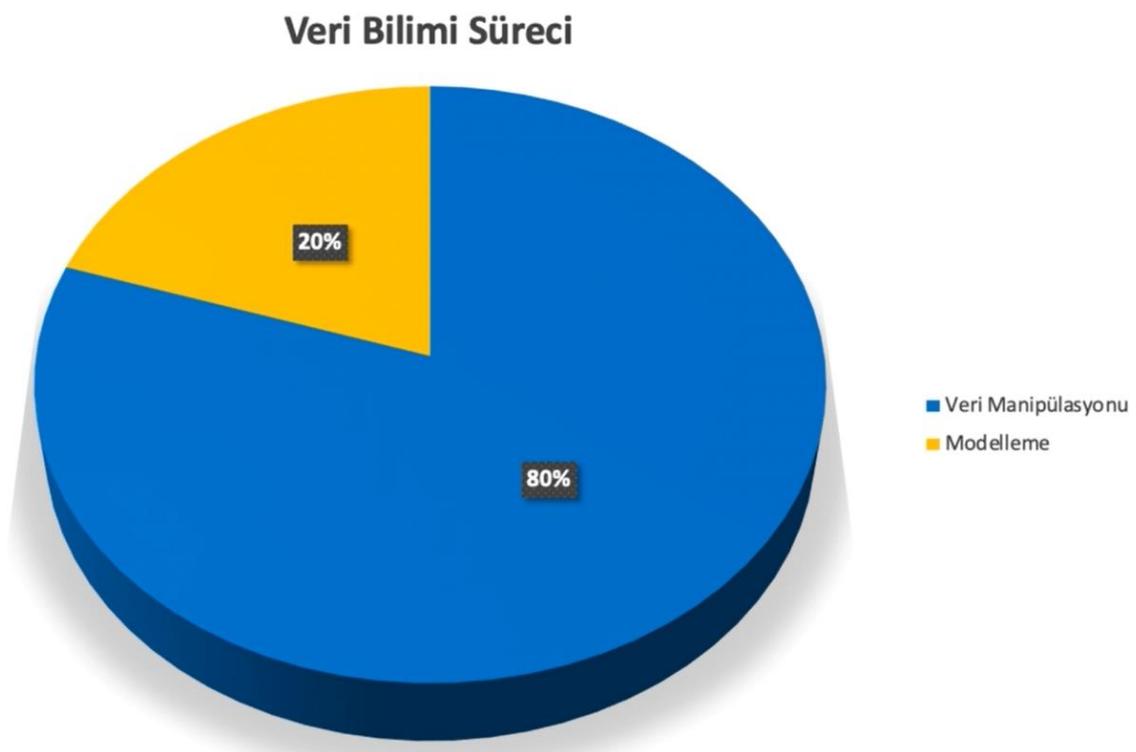
VBO

VeriBilimiOkulu

VeBiOk

Veri Bilimi

Python ile Veri Manipülasyonu: NumPy & Pandas



NumPy (Numerical Python)

NumPy Giriş

NumPy Python'ın bazı numerik işlemlerde yetersiz kaldığı noktalarda ihtiyaçlarımızı gidermek için ortaya çıkışmış bir kütüphanedir/modüldür.

Numpy Giriş

- Numerical Python
- Bilimsel hesaplamalar için kullanılır.
- Arrayler / çok boyutlu arrayler ve matrisler üzerinde yüksek performanslı çalışma imkanı sağlar.
- Temelleri 1995'te (matrix-sig, Guido Van Rossum) atılmış nihai olarak 2005 (Travis Oliphant) yılında hayatı geçmiştir.
- Listelere benzerdir, farkı; verimli veri saklama ve vektörel operasyonlardır.

Neden NumPy?

Daha üst seviyeden, daha az çabayla daha büyük işler yapma olanağı sağladığından dolayı kullanıyor olacağız.

Neden NumPy? sorusunun ikinci ve önemli yanı yer tutma maliyetlerini numpy çok azaltmaktadır. Örneğin listede 4 elemanın her biri için type=int bilgisi 4 kez tutulur. Numpy array'inde ise sadece bir kez array'in kendisi için tutulur.

```
[1]: a = [1,2,3,4]
      b = [2,3,4,5]
      a
      b #sadece en sona ne yazdıysak onu yazdırır.

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

[9]: import numpy as np

[11]: a = np.array([1,2,3,4])
      b = np.array([2,3,4,5]) #listeleri numpy'da array olarak tanımlıyoruz.

[12]: a*b #uzun uzun işlemlere gerek kalmadan listeleri birbiri ile carpar.

[12]: array([ 2,  6, 12, 20])
```

NumPy Array'i Oluşturmak

NumPy Array tipki sözlükler gibi listeler gibi bir veri tipidir.

NumPy Array'i Oluşturmak

```
[1]: import numpy as np

[2]: np.array([1,2,3,4,5]) #array olusturma

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

[4]: a = np.array([1,2,3,4,5])

[5]: type(a)

[5]: numpy.ndarray

[6]: np.array([3.14,4,2,1,13]) #float ve int karışık array

[6]: array([ 3.14,  4. ,  2. ,  1. , 13. ])
```

Veri saklarken sadece bir veri tipi tutabilmek için bütün sayıları ondalıklı bir değere çevirdi.

```
[7]: np.array([3.14,4,2,1,13], dtype="int") #Veri tipini kendimiz belirledik.

[7]: array([ 3,  4,  2,  1, 13])
```

zeros, ones, full, random, arange, linspace, random.normal, random.randint

Sıfırdan Array Oluşturma

```
[1]: import numpy as np

[3]: np.zeros(10, dtype = int)

[3]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])

[6]: np.ones((3,5) , dtype=int) #1'Lerden oluşan 3'e 5'Lük 2 boyutlu array(matris)

[6]: array([[1, 1, 1, 1, 1],
           [1, 1, 1, 1, 1],
           [1, 1, 1, 1, 1]])

[7]: np.full((4,5) , 3) #3'Lerden oluşan 4x5'Lük matris

[7]: array([[3, 3, 3, 3, 3],
           [3, 3, 3, 3, 3],
           [3, 3, 3, 3, 3],
           [3, 3, 3, 3, 3]])

[8]: np.arange(0,31,3) #0'dan 31'e kadar 3'er 3'er artan doğrusal dizi.

[8]: array([ 0,  3,  6,  9, 12, 15, 18, 21, 24, 27, 30])

[9]: np.linspace(0,1,10) #0 ile 1 arasında 10 tane sayı oluştur.

[9]: array([0.          , 0.11111111, 0.22222222, 0.33333333, 0.44444444,
          0.55555556, 0.66666667, 0.77777778, 0.88888889, 1.        ])

[10]: np.random.normal(10, 4, (3,4)) #ortalaması=10, standart sapması=4 olan 3x4'Lük matris

[10]: array([[11.58826043, 11.68947875, 14.08713841, 10.49055712],
            [ 7.37205302, 11.91140801, 11.31641312, 12.05287956],
            [ 3.44476323,  9.28895348,  8.88684624,  6.07701004]])

[11]: np.random.randint(0, 10, (3,3)) #0 ile 10 aralığında rastgele int değerlerden 3x3'Lük matris

[11]: array([[8, 4, 3],
           [0, 3, 7],
           [1, 2, 3]])
```

NumPy Array Özellikleri

NumPy Array Özellikleri

- **ndim**: boyut sayısı
- **shape**: boyut bilgisi
- **size**: toplam eleman sayısı
- **dtype**: array veri tipi

```
[1]: import numpy as np

[4]: np.random.randint(10, size = 10)

[4]: array([3, 3, 9, 6, 7, 8, 5, 1, 6, 1])

[5]: a = np.random.randint(10, size = 10)

[6]: a.ndim #boyut sayisi-- Tek boyutlu bir array oldugundan 1 gelecek.

[6]: 1

[7]: a.shape #boyut bilgisi-- Elimizdeki array tek boyutlu oldugundan sadece tek boyutunun bilgisini verecek.

[7]: (10,)

[9]: a.size #eleman sayisi

[9]: 10

[10]: a.dtype #array'in veri tipi

[10]: dtype('int32')
```

Matris Oluşturma

İki boyutlu array oluşturalım

```
[11]: b = np.random.randint(10, size = (3,5)) #3x5'Lik 0 ile 10 arasındaki değerlerden oluşan matris.

[12]: b

[12]: array([[4, 8, 4, 6, 0],
   [9, 6, 3, 4, 0],
   [8, 8, 4, 5, 7]])

[13]: b.ndim

[13]: 2

[14]: b.shape

[14]: (3, 5)

[15]: b.size

[15]: 15

[16]: b.dtype

[16]: dtype('int32')
```

Reshaping (Array'i Yeniden Şekillendirme)

Elimizde var olan bir array'i yeniden şekillendirme işlemi yapacağız. Örneğin elimizde bir array olsun ve bunu yeniden boyutlandıralım.

Fonskiyonlarımızın ürettiği çıktılar tek bir boyutta, tek bir array formunda gerçekleşebiliyor.

Bunları bazen tek boyuttan 2 boyuta ya da 2 boyuttan tek boyuta indirgeme işlemi gerekebiliyor.

Bu ihtiyaçlarla **Reshape** fonksiyonu ile başa çıkmış oluyoruz.

```
[2]: import numpy as np  
[3]: np.arange(1, 10)  
[3]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])  
[4]: np.arange(1,10).reshape((3,3))  
[4]: array([[1, 2, 3],  
           [4, 5, 6],  
           [7, 8, 9]])
```

Not: Tek boyutlu array: Vektör, 2 boyutlu array: Matris.

```
[5]: a = np.arange(1,10)  
[6]: a  
[6]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Elimizdeki tek boyutlu array'i 2 boyutlu matris'e çevirmek istiyoruz ama tek boyuttaki bilgisi de olduğu şekilde kalsın.

```
[7]: a.ndim  
[7]: 1  
[9]: a.reshape((1,9)) #Artık bir matristir fakat tek boyutlu vektörün taşıdığı bilgiyi taşıır.  
[9]: array([[1, 2, 3, 4, 5, 6, 7, 8, 9]])  
[11]: b = a.reshape((1,9)) #b artık matris.  
[12]: b.ndim  
[12]: 2
```

Concatenation (Array Birleştirme)

`concatenate()` fonksiyonu ile array'leri birleştirebiliriz.

```
[1]: import numpy as np  
  
[2]: x = np.array([1,2,3])  
y = np.array([4,5,6])  
  
[3]: np.concatenate([x, y]) #İki adet tek boyutlu array birlestirme  
array([1, 2, 3, 4, 5, 6])  
  
[4]: z = np.array([7,8,9])  
  
[5]: np.concatenate([x,y,z]) #Üç adet tek boyutlu array birlestirme  
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

İki boyutlu matrislerde ise:

```
[6]: a = np.array([[1,2,3],  
                 [4,5,6]]) #el ile 2 boyutlu matris olusturma  
  
[7]: np.concatenate([a,a]) #standart olarak satir bazinda birlestirme yapar.  
array([[1, 2, 3],  
       [4, 5, 6],  
       [1, 2, 3],  
       [4, 5, 6]])  
  
[8]: np.concatenate([a,a], axis=1) #axis=0 satir, axis=1 sutun bazinda birlestirir.  
array([[1, 2, 3, 1, 2, 3],  
       [4, 5, 6, 4, 5, 6]])
```

Splitting (Array Ayırma)

split() fonksiyonu kullanılır.

```
[1]: import numpy as np  
[2]: x = np.array([1,2,3,99,99,3,2,1])  
[3]: np.split(x, [3,5]) #3. indis kadar ayır, sonra 5. indis kadar ayır, sonra sona kadar.  
[3]: [array([1, 2, 3]), array([99, 99]), array([3, 2, 1])]
```

split fonksiyonuna girilen indis sayısı **n** ise çıktı array sayısı **n+1** olur

```
[4]: a,b,c = np.split(x, [3,5])  
[5]: a  
[5]: array([1, 2, 3])  
[6]: b  
[6]: array([99, 99])  
[7]: c  
[7]: array([3, 2, 1])
```

İki Boyutlu Array Ayırma

vsplit() : dikey olarak ayırmak için kullanılır.

hsplit() : yatay olarak ayırmak için kullanılır.

```
[8]: m = np.arange(16).reshape(4,4) #0-16 arasında 4x4'Luk matris.  
[9]: m  
[9]: array([[ 0,  1,  2,  3],  
           [ 4,  5,  6,  7],  
           [ 8,  9, 10, 11],  
           [12, 13, 14, 15]])  
[11]: np.vsplit(m, [2]) # yataydaki 2. indis kadar ve sonrasını ayır.  
[11]: [array([[0, 1, 2, 3],  
           [4, 5, 6, 7]]),  
           array([[ 8,  9, 10, 11],  
           [12, 13, 14, 15]])]  
[13]: ust, alt = np.vsplit(m, [2])  
[14]: ust  
[14]: array([[0, 1, 2, 3],  
           [4, 5, 6, 7]])  
[15]: alt  
[15]: array([[ 8,  9, 10, 11],  
           [12, 13, 14, 15]])
```

```
[16]: np.hsplit(m,[2]) #dikeyde 2. indis'e kadar ve sonrasini ayir.
```

```
[16]: [array([[ 0,  1],
       [ 4,  5],
       [ 8,  9],
       [12, 13]]),
 array([[ 2,  3],
       [ 6,  7],
       [10, 11],
       [14, 15]])]
```

Sorting (Sıralama)

```
[17]: import numpy as np
```

```
[18]: v = np.array([2,1,4,3,5])
```

```
[19]: np.sort(v) #Kucukten buyuge siralar. Veri setinin orjinal yapisi bozulmadı.
```

```
[19]: array([1, 2, 3, 4, 5])
```

```
[20]: v.sort() #Veri setinin orjinal yapisini degistirdi.
```

```
[21]: v
```

```
[21]: array([1, 2, 3, 4, 5])
```

Matris sıralama

```
[23]: m = np.random.normal(20,5, (3,3))#ortalaması 20, standart sapması 3 olan 3x3 matris.
```

```
[24]: m
```

```
[24]: array([[14.72354718, 25.72515484, 13.24908455],
       [16.62938435, 22.16685623, 22.44070384],
       [22.05424029, 13.64292261, 21.38588038]])
```

```
[25]: np.sort(m, axis=1) #Her bir satiri kendi icinde siralar.
```

```
[25]: array([[13.24908455, 14.72354718, 25.72515484],
       [16.62938435, 22.16685623, 22.44070384],
       [13.64292261, 21.38588038, 22.05424029]])
```

```
[27]: np.sort(m , axis=0) #Sutunlara gore siralama yapar.
```

```
[27]: array([[14.72354718, 13.64292261, 13.24908455],
       [16.62938435, 22.16685623, 21.38588038],
       [22.05424029, 25.72515484, 22.44070384]])
```

Index ile Elemana Erişmek

Tek boyutlu array'lerde eleman yakalama işlemleri listeler ile aynıdır.

```
[2]: import numpy as np  
a = np.array([1,2,3,4,5,6,7,8])  
a  
  
[2]: array([1, 2, 3, 4, 5, 6, 7, 8])  
  
[3]: a[0] #0 index'li eleman  
[3]: 1  
  
[5]: a[-1] #Sondan birinci eleman  
[5]: 8  
  
[6]: a[0] = 100 #eleman degerini degistirmek.  
  
[7]: a  
  
[7]: array([100, 2, 3, 4, 5, 6, 7, 8])
```

Matrislerde elemana erişme işlemleri

```
[14]: m = np.random.randint(10, size = (3,5))  
m  
  
[14]: array([[3, 1, 4, 6, 3],  
           [4, 9, 6, 7, 1],  
           [9, 4, 1, 4, 7]])  
  
[15]: m[0,0] #0'a 0 koordinatindaki eleman (index'e gore)  
[15]: 3  
  
[16]: m[1,1] #1'e 1 koordinatli eleman  
[16]: 9  
  
[18]: m[1,4]  
[18]: 1  
  
[19]: m[1,4] = 99  
m  
  
[19]: array([[ 3,  1,  4,  6,  3],  
           [ 4,  9,  6,  7, 99],  
           [ 9,  4,  1,  4,  7]])  
  
[20]: m[1,4] = 2.2 #float eklemek istiyoruz ancak ondalik kismini keserek ekleyecekt.  
m  
#Daha onceki olusturulan bir array'in tipi sonradan ekleme ile degismez.  
  
[20]: array([[3, 1, 4, 6, 3],  
           [4, 9, 6, 7, 2],  
           [9, 4, 1, 4, 7]])
```

Slicing (Array Alt Küme İşlemleri)

Tek boyutlu array'lerde slicing işlemleri

```
[1]: import numpy as np  
  
[4]: a = np.arange(20,30)  
a  
  
[4]: array([20, 21, 22, 23, 24, 25, 26, 27, 28, 29])  
  
[5]: a[0:3]  
  
[5]: array([20, 21, 22])  
  
[6]: a[:3]  
  
[6]: array([20, 21, 22])  
  
[7]: a[3:]  
  
[7]: array([23, 24, 25, 26, 27, 28, 29])  
  
[8]: a[1::2] #1 index'den baslayarak 2'ser 2'ser artar.  
  
[8]: array([21, 23, 25, 27, 29])  
  
[11]: a[0::3] #0'dan baslar 3'er 3'er artar.  
  
[11]: array([20, 23, 26, 29])
```

Matrislerde Slicing İşlemleri

Matrislerde Slicing İşlemleri

```
[12]: m = np.random.randint(10, size=(5,5))  
  
[13]: m  
  
[13]: array([[1, 9, 0, 0, 4],  
           [9, 3, 3, 7, 3],  
           [5, 2, 6, 8, 7],  
           [3, 7, 2, 0, 9],  
           [2, 1, 3, 4, 0]])  
  
[15]: m[:,0] #Butun satirlar, 0. sutun  
[15]: array([1, 9, 5, 3, 2])  
  
[17]: m[:,1] #Butun satirlar, 1. sutun  
[17]: array([9, 3, 2, 7, 1])  
  
[19]: m[0,:] #0. satir, butun sutunlar  
[19]: array([1, 9, 0, 0, 4])  
  
[23]: m  
  
[23]: array([[1, 9, 0, 0, 4],  
           [9, 3, 3, 7, 3],  
           [5, 2, 6, 8, 7],  
           [3, 7, 2, 0, 9],  
           [2, 1, 3, 4, 0]])  
  
[24]: m[1:3,1:2] #1. ve 2. satirlar, 1. sutun  
[24]: array([[3],  
           [2]])  
  
[29]: m[:,2] #butun satirlar, ilk 2 sutun  
[29]: array([[1, 9],  
           [9, 3],  
           [5, 2],  
           [3, 7],  
           [2, 1]])
```

Alt Küme Üzerinde İşlem Yapmak

Önceki bölümde array'lerin alt kümelerine erişik fakat burada söyle bir durum söz konusu;

Örneğin bir array'in alt kümese erişikten sonra bunu isimlendirip kaydettiğimizi düşünelim.

Bu kaydetmiş olduğumuz isimlendirme üzerinde bir değişiklik yaptığımızda array'in orjinali de değişiyordu.

Fakat bazen seçilen array'in alt kümese o alt kümeye özel işlemler yapılmak istenebilir.

İşte bu yüzden alt kümeleri bağımsızlaştırmak isimli bir işlem yapılması gerekiyor.

```
[30]: #Bir örnek ile yukarıdaki durumu daha iyi anlayalim:  
import numpy as np  
a = np.random.randint(10, size=(5,5)) #5x5 matris olusturduk.  
  
[30]: array([[0, 0, 0, 1, 0],  
           [8, 4, 5, 2, 9],  
           [5, 2, 7, 4, 1],  
           [7, 6, 2, 2, 6],  
           [3, 6, 2, 1, 0]])  
  
[34]: alt_a = a[0:3,0:2] #alt kume olusturduk  
alt_a  
  
[34]: array([[999, 0],  
           [8, 888],  
           [5, 2]])  
  
[32]: alt_a[0,0]=999 #alt kume elemanlarında degisiklik yaptik.  
alt_a[1,1]=888  
alt_a  
  
[32]: array([[999, 0],  
           [8, 888],  
           [5, 2]])  
  
[33]: a #orjinal matrisimiz de etkilendi.  
  
[33]: array([[999, 0, 0, 1, 0],  
           [8, 888, 5, 2, 9],  
           [5, 2, 7, 4, 1],  
           [7, 6, 2, 2, 6],  
           [3, 6, 2, 1, 0]])
```

Bu durum bazen çok iş görebilmekte.

Çok büyük boyutta array'ler elimizde olduğunda onların bazı parçalarını seçip spesifik olarak onların üzerinde çalışıp ana parçanın üzerinde değişiklik yapmak açısından çok işe yarar.

copy() metodunu kullanarak bu durumdan vazgeçebiliriz.

```
[38]: alt_b=m[0:3,0:2].copy() #bu islemden sonraki islemler ana array'den bagimsiz olacak.

[40]: alt_b[0,0]=9999
      alt_b #alt kume etkilendi

[40]: array([[9999,      9],
       [      9,      3],
       [      5,      2]])

[41]: m #orjinal array etkilenmedi.

[41]: array([[1, 9, 0, 0, 4],
       [9, 3, 3, 7, 3],
       [5, 2, 6, 8, 7],
       [3, 7, 2, 0, 9],
       [2, 1, 3, 4, 0]])
```

Fancy Index ile Elemanlara Erişmek

Fancy Index kavramı ilerleyen bölümlerde bizim için en önemli kavramlardan birisi olacak.

Bize hem Pandas data frame'lerinde hem de NumPy array'lerinde ileri düzey eleman seçme imkanları vermektedir.

```
[1]: import numpy as np
v = np.arange(0,30,3)
v

[1]: array([ 0,  3,  6,  9, 12, 15, 18, 21, 24, 27])

[2]: [v[1], v[2], v[3]] #eski yontemle elemanlara eristik.

[2]: [3, 6, 9]

[3]: #Ancak elimizde 100lerce elemanli bir array oldugunda bunu yapmak zor olacak.

[4]: al_getir = [1,3,5]

[6]: v[al_getir] #Iste buna Fancy Index denir.

[6]: array([ 3,  9, 15])
```

Matrislerde Fancy Index Kullanımı

Matrislerde Fancy Index Kullanımı

```
[8]: m = np.arange(9).reshape((3,3))
m

[8]: array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])

[9]: satir = np.array([0,1])
      sutun = np.array([1,2])

[10]: m[satir, sutun]

[10]: array([1, 5])
```

Basit Index ile Fancy kullanımı

```
[11]: #basit index ile fancy index  
  
[12]: m  
  
[12]: array([[0, 1, 2],  
           [3, 4, 5],  
           [6, 7, 8]])  
  
[13]: m[0, [1,2]] #basit index ile fancy'i aynı anda kullandık.  
  
[13]: array([1, 2])
```

Slice ile Fancy kullanımı

```
[14]: #slice ile fancy  
  
[15]: m[0:, [1,2]] #basit index ile fancy'i aynı anda kullandık.  
  
[15]: array([[1, 2],  
           [4, 5],  
           [7, 8]])  
  
[ ]: #Buradaki işlemlerin teknik olarak farklı olduğunu anlamamız gereklidir.
```

Koşullu Eleman İşlemleri

Koşullu Eleman İşlemleri

```
[2]: import numpy as np  
  
[3]: v = np.array([1,2,3,4,5])  
  
[4]: v > 5  
  
[4]: array([False, False, False, False, False])  
  
[5]: v < 3  
  
[5]: array([ True,  True, False, False, False])  
  
[6]: v[v < 3] #Fancy  
  
[6]: array([1, 2])  
  
[7]: v[v > 3] #Fancy  
  
[7]: array([4, 5])
```

```

[8]: v[v >= 3] #Fancy
[8]: array([3, 4, 5])

[9]: v[v == 3] #Fancy
[9]: array([3])

[10]: v[v != 3] #Fancy
[10]: array([1, 2, 4, 5])

[11]: v
[11]: array([1, 2, 3, 4, 5])

[12]: v*2
[12]: array([ 2,  4,  6,  8, 10])

[13]: v/5
[13]: array([0.2, 0.4, 0.6, 0.8, 1. ])

[14]: v*5/10
[14]: array([0.5, 1. , 1.5, 2. , 2.5])

[15]: v**2
[15]: array([ 1,  4,   9, 16, 25], dtype=int32)

```

Matematiksel İşlemler

Matematiksel İşlemler

```

[1]: import numpy as np
v = np.array([1,2,3,4,5])
v

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

[2]: v*5
[2]: array([ 5, 10, 15, 20, 25])

```

Biz çarpma işlemi yapsak da arka tarafta bu işlemler bir dönüştürmeye tabi tutulup NumPy içerisindeki spesifik fonksiyonlar çalıştırılıyor.

```
[3]: #bunlara ufunc denir.

[5]: np.subtract(v, 1) # v-1 isleminin arka planinda calisan fonksiyon

[5]: array([0, 1, 2, 3, 4])

[6]: np.add(v, 1) #v+1

[6]: array([2, 3, 4, 5, 6])

[7]: np.multiply(v, 4) #v*4

[7]: array([ 4,  8, 12, 16, 20])

[8]: np.divide(v, 3) #v/3

[8]: array([0.33333333, 0.66666667, 1.          , 1.33333333, 1.66666667])

[9]: np.power(v, 3) #v**3

[9]: array([ 1,   8,  27,  64, 125], dtype=int32)

[10]: np.mod(v, 2) #v%2

[10]: array([1, 0, 1, 0, 1], dtype=int32)

[11]: np.absolute(np.array([-3])) #Mutlak deger

[11]: array([3])
```

Trigonometrik Fonksiyonlar

Trigonometrik Fonksiyonlar

```
[12]: np.sin(360)

[12]: 0.9589157234143065

[13]: np.cos(180)

[13]: -0.5984600690578581
```

Logaritmik İşlemler

Logaritmik İşlemler

```
[14]: v = np.array([1,2,3])  
  
[15]: np.log(v)  
  
[15]: array([0.       , 0.69314718, 1.09861229])  
  
[16]: np.log2(v)  
  
[16]: array([0.       , 1.       , 1.5849625])  
  
[17]: np.log10(v)  
  
[17]: array([0.       , 0.30103  , 0.47712125])
```

Numpy ile İki Bilinmeyenli Denklem Çözümü

Numpy ile İki Bilinmeyenli Denklem Çözümü

NumPy'i daha çok matematiğin alt dalı olan Lineer Cebir alanında düşünmeliyiz.

```
[3]: import numpy as np  
  
5 * x0 + x1 = 12  
x0 + 3 * x1 = 10
```

Bu denklemdeki bilinmeyenlerin katsayılarını array'ler cinsinden ifade ederek numpy'in altında yer alan bir fonksiyon aracılığı ile bilinmeyen değerleri çözmüş olacağız.

Bu matematiksel problemi python'in anlayacağı formata getirmemiz gerekiyor.

Bunun yolu da bilinmeyen ifadelerin katsayılarını bir vektöre koymak, **(a)**

bu denklemler sonucunda oluşan değerleri bir vektöre koymak, **(b)**

ve son olarak, **linalg** paketi içinde geliştirilmiş **solve** isimli fonksiyonu çalışırmak. **(x)**

```
[5]: a = np.array([[5,1], [1,3]])  
b = np.array([12,10])  
  
[6]: a  
  
[6]: array([[5, 1],  
           [1, 3]])  
  
[7]: b  
  
[7]: array([12, 10])  
  
[8]: x = np.linalg.solve(a,b)  
x  
  
[8]: array([1.85714286, 2.71428571])
```

x0 ve **x1** değerlerlerini solve fonksiyonu ile bulduk.

NumPy Alıştırmalar– 1

Soru 1:

Aşağıdakilerden hangisi NumPy özelliklerinden değildir?

- Bilimsel hesaplamalar için kullanılır
- Array'ler üzerinde yüksek performanslı çalışma imkanı sağlar**
- Temelleri 1995'te atılmış ve nihai olarak 2005 yılında hayatı geçmiştir
- Daha iyi döngüler yazmaya yardımcı olur

Soru 2:

Verilen kodun çıktısı aşağıdakilerden hangisidir?

```
1 | import numpy as np  
2 | np.array([3.14, 4, 6, 1.2])
```

- Çıktı yoktur
- Kod çalışmaz
- array([3.14, 4., 6., 1.2])
- array([3.14, 4, 6, 1.2])

Soru 3:

Aşağıda bir kod parçası ve çıktısı verilmiştir. Buna çıktıının bu şekilde (kod bölümünde integer, çıktı bölümünde float tip gözlenmesi) olmasının sebebi nedir?

Kod:

```
1 | import numpy as np  
2 | np.array([3.14, 4, 6, 1.2])
```

Cıktı:

```
array([3.14, 4., 6., 1.2])
```

Kütüphane yüklemesi ile ilgilidir

Numpy array'lerinin **sabitlenmiş tip** özelliği ile ilgilidir

Çıktının bir özelliği

Numpy array'lerinin vektörel olmasından

Soru 4:

Aşağıdaki çıktıyı üretmek için hangi kod yazılmalıdır?

```
1 | array([[1., 1., 1.],  
2 |         [1., 1., 1.]])
```

1 | import numpy as np
2 | np.ones((3,2))

1 | import numpy as np
2 | np.ones((2,3))

1 | import numpy as np
2 | np.eye((2,3))

1 | import numpy as np
2 | np.eye((2,1))

Soru 5:

Bir NumPy array'i için satır ve sütun bilgisine nasıl erişilir?

ndim

shape

Cevap shape olabilir.

dtype

dir

Soru 6:

Bir NumPy array'i için toplam eleman sayısı bilgisine nasıl erişilir?

shape

dtype

size

dir

Soru 7:

Bir NumPy array'i için veri tipi bilgisine nasıl erişilir?

dtype

ndim

shape

size

Soru 8:

Aşağıda verilen çıktıının kodu hangisidir?

```
1 | array([[[7, 9],  
2 |     [4, 0],  
3 |     [5, 9]],  
4 |     [[4, 8],  
5 |     [6, 4],  
6 |     [4, 5]],  
7 |     [[2, 2],  
8 |     [8, 2],  
9 |     [0, 2]]])
```

1 | import numpy as np
2 | np.random.randint(10, size = (1,3,2))

1 | import numpy as np
2 | np.random.randint(10, size = (2,3,2))

1 | import numpy as np
2 | np.random.randint(10, size = (3,2,2))

1 | import numpy as np
2 | np.random.randint(10, size = (3,3,2))

Soru 9:

Aşağıda verilen çıktıının kodu hangisidir?

array([1, 2, 3, 4, 5, 6, 7, 8, 9])

1 | import numpy as np
2 | np.arange(0,10)

1 | import numpy as np
2 | np.arange(2,11)

1 | import numpy as np
2 | np.arange(1,10)

1 | import numpy as np
2 | np.arange(1,9)

Soru 10:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | import numpy as np
2 | x = np.array([1, 2, 3])
3 | y = np.array([4, 5, 6])
4 | np.concatenate([x,y])
```



```
1 | array([[1, 2, 3],
2 |         [4, 5, 6]])
```



```
1 | array([[1, 2, 3, 1, 2, 3],
2 |         [4, 5, 6, 4, 5, 6]])
```



```
array([1, 2, 3, 4, 5, 6])
```



```
array([14, 5, 6, 1, 2, 3])
```

NumPy Alıştırmalar – 2

Soru 1:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
5*np.array([1, 2, 3])
```



Çalışmaz çünkü gerekli import işlemi yapılmamıştır



```
array([ 11111, 22222, 33333])
```



```
array([ 5, 10, 15])
```



5

Soru 2:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | import numpy as np  
2 | 5*np.array([1,2,3])
```

Çalışmaz

array([5, 10, 15])

array([11111,22222,33333])

5

Soru 3:

Verilen kodun çıktısı aşağıdakilerden hangisidir?

```
1 | import numpy as np  
2 | np.arange(0,10, 2)
```

array([0,10,0,10])

array([3,8])

array([0,2,4,6,8])

array([0,2,4,6,8,10])

Soru 4:

Verilen kodun çıktısı aşağıdakilerden hangisidir?

```
1 import numpy as np  
2 v = np.array([2, 1, 4, 3, 5])  
3 np.sort(v)
```

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

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

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

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

Soru 5:

Aşağıda verilen array'de yer alan 9 değerine erişmek için hangi kod yazılmalıdır?

```
array([7, 3, 4, 7, 0, 9, 3, 2, 2])
```

v[4]

v[5]

v[9]

v[6]

Soru 6:

Verilen kod parçasının çıktısı aşağıdakilerden hangisidir?

```
1 | import numpy as np
2 | v = np.array([7, 3, 4, 7, 0, 9, 3, 2, 9, 2])
3 | v[-2]
```

4

3

2

9

Soru 7:

Aşağıda "a" ismindeki bir array'in çıktısı verilmiştir. Buna göre yazılan kodun çıktısı hangisidir?

Array çıktısı:

```
1 | array([[4, 7, 4, 5, 9],
2 | [2, 5, 0, 7, 7],
3 | [1, 9, 0, 8, 2]])
```

Kod:

a[1,1]

4

2

7

5

Soru 8:

Aşağıda "a" ismindeki bir array'in çıktısı verilmiştir. Buna göre yazılan kodun çıktısı hangisidir?

Çıktı:

```
1 | array([[4, 0, 3, 0, 1],  
2 |         [9, 6, 1, 5, 9],  
3 |         [1, 9, 0, 8, 2]])
```

Kod:

`a[0:1]`

array([[4, 0, 3, 0, 1],
[9, 6, 1, 5, 9]])

array([4, 0, 3, 0, 1])

array([[4, 0, 3, 0, 1]])

array([[1, 9, 0, 8, 2]])

Soru 9:

Aşağıda "a" ismindeki bir array'in çıktısı verilmiştir. Buna göre yazılan kodun çıktısı hangisidir?

Çıktı:

```
1 | array([[4, 7, 4, 5, 9],  
2 |         [2, 5, 0, 7, 7],  
3 |         [1, 9, 0, 8, 2]])
```

Kod:

`a[2,3]`

5

8

7

2

Soru 10:

Aşağıda "a" ismindeki bir array'in çıktısı verilmiştir. Buna göre yazılan kodun çıktısı hangisidir?

Çıktı:

```
1 | array([[4, 7, 4, 5, 9],  
2 | [2, 5, 0, 7, 7],  
3 | [1, 9, 0, 8, 2]])
```

Kod:

a[3,2]

0

8

7

Çalışmaz çünkü index hayatı yok

NumPy Alıştırmalar – 3

Soru 1:

Bir NumPy array'i için boyut sayısı bilgisine nasıl erişilir?

ndim

shape

dtype

dir

Soru 2:

Aşağıda "a" ismindeki bir array'in çıktısı verilmiştir. Buna göre yazılan kodun çıktısı nedir?

Çıktı:

```
1 | array([[4, 7, 4, 5, 9],  
2 | [2, 5, 0, 7, 7],  
3 | [1, 9, 0, 8, 2]])
```

Kod:

a[:,2]

array([4, 0, 0])

array([[2, 5, 0, 7, 7],
[1, 9, 0, 8, 2]])

array([7, 5, 9])

array([[4, 7, 4, 5, 9]
[2, 5, 0, 7, 7]])

Soru 3:

Aşağıda "a" ismindeki bir array'in çıktısı verilmiştir. Buna göre yazılan kodun çıktısı nedir?

Çıktı:

```
1 | array([[4, 7, 4, 5, 9],  
2 | [2, 5, 0, 7, 7],  
3 | [1, 9, 0, 8, 2]])
```

Kod:

a[:2, :3]

array([1, 9, 0, 8, 2])

array([[4,7,4],
[2,5,0]])

array([5,7,8])

array([4, 7, 4, 5, 9])

Soru 4:

Aşağıda "v" ismindeki bir array'in çıktısı verilmiştir. Buna göre yazılan kodun çıktısı nedir?

Çıktı:

```
array([ 0, 3, 6, 9, 12, 15, 18, 21, 24, 27])
```

Kod:

```
[v[1], v[3]]
```

Çalışmaz

3,9

[3,9]

[0,6]

Soru 5:

Aşağıda "v" ismindeki bir array'in çıktısı verilmiştir. Buna göre yazılan kodun çıktısı nedir?

Çıktı:

```
array([ 0, 3, 6, 9, 12, 15, 18, 21, 24, 27])
```

Kod:

```
[v[9], v[0]]
```

Çalışmaz

0,9

[9,0]

[27,0]

Soru 6:

Aşağıdaki seçim işleminin teknik ismi nedir?

```
1 import numpy as np  
2 v = np.array([ 0, 3, 6, 9, 12, 15, 18, 21, 24, 27])  
3  
4 v[[1,2,3]]
```

Index seçimi

Fancy index seçimi

Slice index seçimi

Vektör seçimi

Soru 7:

Aşağıda "m" ismindeki bir array'in çıktısı verilmiştir. Buna göre yazılan kodun çıktısı nedir?

Çıktı:

```
1 array([[0, 1, 2],  
2 [3, 4, 5],  
3 [6, 7, 8]])
```

Kod:

`m[0, [1,2]]`

`array([0, 1])`

`array([1, 2])`

Çalışmaz

`array([0,3])`

Soru 8:

Bir numpy array'nin alt kümesi üzerinde işlem yaparken alt küme üzerinde yapılan değişikliklerin array'in ilk halinden bağımsız olması için hangi fonksiyon kullanılır?

multiply()

divide()

copy()

dir

Soru 9:

Aşağıda verilen fonksiyon ile aynı işlevi gören kod aşağıdakilerden hangisidir?

```
1 | import numpy as np  
2 | np.power(v, 3)
```

$3*v$

v^3

$v^{***}3$

$v^{**}3$

Soru 10:

Aşağıda verilen fonksiyon ile aynı işlevi gören kod aşağıdakilerden hangisidir?

```
1 | import numpy as np  
2 | np.subtract(v, 2)
```

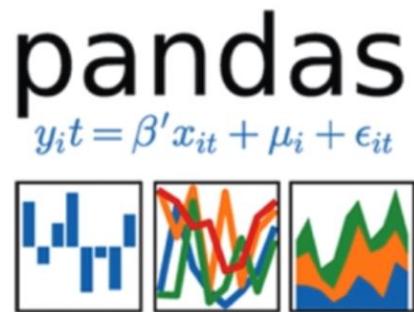
$v/2$

$v^{**}2$

$v + 2$

$v-2$

Pandas



Pandas Giriş

- Panel Data
- Veri manipülasyonu ve veri analizi için yazılmış açık kaynak kodlu bir Python kütüphanesidir.
- Ekonometrik ve finansal çalışmalar için doğmuştur.
- Temeli 2008 yılında atılmıştır.
- R DataFrame yapısını Python dünyasına taşımış ve DataFrame'ler üzerinde hızlı ve etkili çalışabilme imkanı sağlamıştır.
- Bir çok farklı veri tipini okuma ve yazma imkanı sağlar.

Pandas Serisi Oluşturmak

Pandas Serisi Oluşturmak

Pandas içerisinde yer alan veri tipleri değerleri, indeksleri ile beraber tutar.

```
[2]: import pandas as pd

[3]: pd.Series([10,88,3,4,5]) #pandas serisi olusturmak.

[3]: 0    10
     1    88
     2     3
     3     4
     4     5
    dtype: int64

[4]: seri = pd.Series([10,88,3,4,5])

[5]: type(seri)

[5]: pandas.core.series.Series

[6]: seri.axes #Serinin index bilgisine ulasiriz.

[6]: [RangeIndex(start=0, stop=5, step=1)]

[7]: seri.dtype

[8]: dtype('int64')

[9]: seri.size #eleman sayisi

[9]: 5

[10]: seri.ndim #boyutu

[10]: 1

[11]: seri.values #vektor formunda sadece degerlere ulasiriz.

[11]: array([10, 88, 3, 4, 5], dtype=int64)

[7]: seri.head() #ilk 5 eleman
[7]: 0    10
     1    88
     2     3
     3     4
     4     5
    dtype: int64
[7]:          seri.head(3) #ilk 3 eleman  seri.tail(3) #son 3 eleman
[7]:          0    10          2     3
          1    88          3     4
          2     3          4     5
          dtype: int64          dtype: int64
```

Index İsimlendirmesi

Index İsimlendirmesi

```
[10]: pd.Series([23,24,25,26,27], index = [2,4,6,8,10])  
[10]: 2    23  
      4    24  
      6    25  
      8    26  
     10   27  
dtype: int64  
[11]: seri = pd.Series([23,24,25,26,27], index = ["a","b","c","d","e"])  
[13]: seri  
[13]: a    23  
      b    24  
      c    25  
      d    26  
      e    27  
dtype: int64  
[14]: seri["a"] #elemana erisme  
[14]: 23  
[15]: seri["a":"c"] #serilerde slice islemi  
[15]: a    23  
      b    24  
      c    25  
dtype: int64
```

Sözlük Üzerinden Seri Oluşturmak

Sözlük Üzerinden seri oluşturmak

```
[16]: sozluk={"reg":10, "log":11, "cart":12}  
[18]: seri = pd.Series(sozluk)  
[19]: seri  
[19]: reg    10  
      log    11  
      cart   12  
dtype: int64
```

İki Seriyi Birleştirerek Seri Oluşturma

İki Seriyi Birleştirerek Seri Oluşturma

```
[20]: pd.concat([seri, seri])
```

```
[20]: reg      10
      log      11
      cart     12
      reg      10
      log      11
      cart     12
      dtype: int64
```

Eleman İşlemleri

Eleman İşlemleri

```
[23]: import numpy as np
      a = np.array([15,233,34,52,64])
      seri = pd.Series(a) #NumPy Array'i üzerinden seri oluşturalım
      seri
```

```
[23]: 0    15
      1   233
      2    34
      3    52
      4    64
      dtype: int32
```

```
[24]: seri[0] #0 indexli eleman
```

```
[24]: 15
```

```
[25]: seri[0:3] #3'e kadar olan elemanlar
```

```
[25]: 0    15
      1   233
      2    34
      dtype: int32
```

```
[27]: seri = pd.Series([133,244,355,467,234], index = ["reg","log","cart","pcv","rf"])
      seri
```

```
[27]: reg      133
      log      244
      cart     355
      pcv      467
      rf       234
      dtype: int64
```

```
[29]: seri.index #sadece indexler
```

```
[29]: Index(['reg', 'log', 'cart', 'pcv', 'rf'], dtype='object')
```

```
[30]: seri.keys #seri'nin key'lerini gösterir

[30]: <bound method Series.keys of reg      133
      log    244
      cart   355
      pcv    467
      rf     234
      dtype: int64>

[31]: list(seri.items()) #key degerine karsilik gelen value'lari bir araya getirerek list olusturur.

[31]: [('reg', 133), ('log', 244), ('cart', 355), ('pcv', 467), ('rf', 234)]

[32]: seri.values #seri'nin sadece degerlerini gösterir

[32]: array([133, 244, 355, 467, 234], dtype=int64)
```

Eleman Sorğulama

Eleman Sorğulama

```
[33]: "reg" in seri

[33]: True

[34]: "a" in seri

[34]: False

[35]: seri["reg"]

[35]: 133
```

Fancy Eleman

Fancy Eleman

```
[37]: seri[["rf","reg"]] #fancy ile eleman secme

[37]: rf    234
      reg   133
      dtype: int64
```

Eleman Değiştirme

Eleman Değiştirme

```
[39]: seri["reg"] = 111
seri #atama yontemi ile tekrardan eleman atayabiliriz.
```

```
[39]: reg      111
      log      244
      cart     355
      pcv      467
      rf       234
      dtype: int64
```

Pandas DataFrame Oluşturma

Pandas DataFrame Oluşturma

Pandas DataFrame yapısal bir veri tipidir.

```
[2]: import pandas as pd
l = [5,12,37,62,14] #list olusturduk
l
```

```
[2]: [5, 12, 37, 62, 14]
```

```
[3]: pd.DataFrame(l, columns = ["degisken_ismi"]) #DataFrame olusturma
```

	degisken_ismi
0	5
1	12
2	37
3	62
4	14

```
[5]: import numpy as np
m = np.arange(1,10).reshape(3,3)
m #3x3'Luk bir matris
```

```
[5]: array([[1, 2, 3],
           [4, 5, 6],
           [7, 8, 9]])
```

```
[6]: pd.DataFrame(m, columns=["var1","var2","var3"]) #2 boyutlu DataFrame
```

	var1	var2	var3
0	1	2	3
1	4	5	6
2	7	8	9

Yapay zeka ve Veri Biliminde en çok kullanacağımız veri tipi DataFrame'dır.

DataFrame İsimlendirme

DataFrame İsimlendirme

```
[7]: df = pd.DataFrame(m, columns=["var1","var2","var3"])
df.head(2)
```

```
[7]:   var1  var2  var3
  0      1      2      3
  1      4      5      6
```

```
[8]: df.columns = ("col1","col2","col3") #Sutunları yeniden isimlendirme
df
```

```
[8]:   col1  col2  col3
  0      1      2      3
  1      4      5      6
  2      7      8      9
```

DataFrame Özellikleri

DataFrame Özellikleri

```
[9]: type(df)
```

```
[9]: pandas.core.frame.DataFrame
```

```
[10]: df.axes #Satır ve sutun bilgisi
```

```
[10]: [RangeIndex(start=0, stop=3, step=1),
       Index(['col1', 'col2', 'col3'], dtype='object')]
```

```
[11]: df.shape #boyut bilgisi
```

```
[11]: (3, 3)
```

```
[12]: df.ndim #boyut sayısı
```

```
[12]: 2
```

```
[13]: df.size #eleman sayısı
```

```
[13]: 9
```

```
[14]: df.values #DataFrame tipindeki veri yapisinin icersinden
       #Degerleri array tipinde aliyor.

[14]: array([[1, 2, 3],
           [4, 5, 6],
           [7, 8, 9]])

[15]: type(df.values)

[15]: numpy.ndarray      Çok önemli!

[17]: df.tail(1) #sondan 1. index

[17]:   col1  col2  col3
      2      7      8      9
```

Diğer veri tiplerinde veri oluşturmak için çeşitli formatlar kullandık.

Örneğin; NumPy array'i üzerinden oluşturduk list üzerinden oluşturduk ve buna benzer farklı formatlardan oluşturduk. Bu işlemler DataFrame için de geçerlidir.

```
[18]: a = np.array([1,2,3,4,5])

[21]: pd.DataFrame(a, columns = ["deg1"]) #numpy array'i ile df olusturduk.

[21]:   deg1
      0    1
      1    2
      2    3
      3    4
      4    5
```

DataFrame Eleman İşlemleri

DataFrame Eleman İşlemleri

```
[1]: import numpy as np
s1 = np.random.randint(10, size=5)
s2 = np.random.randint(10, size=5)
s3 = np.random.randint(10, size=5)

[2]: sozluk={"var1":s1,"var2":s2,"var3":s3} #array'Lerden sozluk
sozluk

[2]: {'var1': array([0, 6, 2, 5, 7]),
      'var2': array([2, 1, 7, 6, 2]),
      'var3': array([3, 2, 3, 2, 3])}
```

```
[4]: import pandas as pd  
df = pd.DataFrame(sozluk) #sozluk'den df  
df
```

```
[4]:   var1  var2  var3  
0     0     2     3  
1     6     1     2  
2     2     7     3  
3     5     6     2  
4     7     2     3
```

```
[7]: df[0:2] #0'dan 2'ye kadar
```

```
[7]:   var1  var2  var3  
0     0     2     3  
1     6     1     2
```

```
[8]: df.index
```

```
[8]: RangeIndex(start=0, stop=5, step=1)
```

```
[10]: df.index = ["a","b","c","d","e"]
```

```
[11]: df
```

```
[11]:   var1  var2  var3  
a     0     2     3  
b     6     1     2  
c     2     7     3  
d     5     6     2  
e     7     2     3
```

```
[12]: df.index
```

```
[12]: Index(['a', 'b', 'c', 'd', 'e'], dtype='object')
```

Eleman Silme

Eleman Silme

```
[22]: df.drop("a", axis=0) #0 ekseninden "a" indexli satiri sil.
```

```
[22]:   var1  var2  var3
```

b	6	1	2
c	2	7	3
d	5	6	2
e	7	2	3

```
[17]: df #sildi ancak kaydetmedi.
```

```
[17]:   var1  var2  var3
```

a	0	2	3
b	6	1	2
c	2	7	3
d	5	6	2
e	7	2	3

```
[23]: df.drop("a", axis=0, inplace=True) #inplace argumani kalici olsun mu? anlamindadir.
```

```
[24]: df #kalici olarak silindi.
```

```
[24]:   var1  var2  var3
```

b	6	1	2
c	2	7	3
d	5	6	2
e	7	2	3

Fancy ile eleman silme

```
[25]: #fancy
```

```
[26]: l = ["c","e"]
```

```
[27]: df.drop(l, axis=0) #c ve e silindi
```

```
[27]:   var1  var2  var3
```

b	6	1	2
d	5	6	2

Değişkenler için eleman işlemleri

```
[30]: #Degisenler icin
```

```
[31]: df
```

```
[31]:   var1  var2  var3
      b      6      1      2
      c      2      7      3
      d      5      6      2
      e      7      2      3
```

```
[32]: "var1" in df
```

```
[32]: True
```

```
[35]: l = ["var1","var4","var2"]
```

```
[37]: for i in l:
      print(i in df)
```

```
True
False
True
```

Bir değişken oluşturmak isteyelim fakat bu değişkenimizi DataFrame içinde var olan değişkenlerden yapmak istediğimizi düşünelim.

```
[38]: df
```

```
[38]:   var1  var2  var3
      b      6      1      2
      c      2      7      3
      d      5      6      2
      e      7      2      3
```

```
[39]: df["var4"] = df["var1"] / df["var2"]
```

```
[40]: df
```

```
[40]:   var1  var2  var3      var4
      b      6      1      2  6.000000
      c      2      7      3  0.285714
      d      5      6      2  0.833333
      e      7      2      3  3.500000
```

Değişken Silme

Değişken Silme

```
[43]: df
```

```
[43]:   var1  var2  var3      var4
      b      6      1      2  6.000000
      c      2      7      3  0.285714
      d      5      6      2  0.833333
      e      7      2      3  3.500000
```

```
[44]: df.drop("var4", axis=1, inplace=True)
df
```

```
[44]:   var1  var2  var3
      b      6      1      2
      c      2      7      3
      d      5      6      2
      e      7      2      3
```

```
[47]: l = ["var1","var2"]
df.drop(l, axis=1) #Fancy ile silme
```

```
[47]:   var3
      b      2
      c      3
      d      2
      e      3
```

Gözlem ve Değişken Seçimi: loc & iloc

```
[1]: import numpy as np
import pandas as pd
m = np.random.randint(1,30, size=(10,3))
df = pd.DataFrame(m, columns=["var1","var2","var3"])
df
```

```
[1]:   var1  var2  var3
 0      9     23    14
 1     17     12     5
 2      4     14     8
 3     14     27     6
 4     21     28    25
 5      3      9    20
 6     15      3    18
 7     16     27    14
 8      9     23    24
 9     19     12    14
```

loc: tanımlandığı şekliyle seçim yapmak için kullanılır

```
[2]: df.loc[0:3] #veri setinin ilk halindeki indexlere sadık kalacak şekilde secim imkani verir.
```

```
[2]:   var1  var2  var3
 0      9     23    14
 1     17     12     5
 2      4     14     8
 3     14     27     6
```

iloc: alışık olduğumuz index'leme mantığıyla seçim yapar.

```
[4]: df.iloc[0:3]
```

```
[4]:   var1  var2  var3
 0      9     23    14
 1     17     12     5
 2      4     14     8
```

```
[5]: df.iloc[0,0]
[5]: 9

[10]: df.iloc[:3,:2]
[10]:
  var1  var2
0     9    23
1    17    12
2     4    14

[11]: df.loc[:3,"var3"]
[11]: 0    14
      1    5
      2    8
      3    6
      Name: var3, dtype: int32

[ ]: df.iloc[:3,"var3"] # hata verir.
```

Eğer değişken ya da satırlar ile ilgili mutlak bir değer işaretlemesi yapacaksak bu durumda **loc** kullanmamız gerekiyor.

Yani değişken ismi ile işaretleme yapacaksak **loc** kullanmalıyız.
Index'lere göre işaretleme yapacaksak **iloc** kullanmalıyız.

```
[13]: df.iloc[:3,1:3]
[13]:
  var2  var3
0    23   14
1    12    5
2    14    8

[14]: df.iloc[:3]["var3"]
[14]: 0    14
      1    5
      2    8
      Name: var3, dtype: int32
```

Koşullu Eleman İşlemleri

Koşullu Eleman İşlemleri

```
[1]: import numpy as np
import pandas as pd
m = np.random.randint(1,30, size=(10,3))
df = pd.DataFrame(m, columns=["var1","var2","var3"])
df
```

```
[1]:      var1  var2  var3
```

	var1	var2	var3
0	8	14	15
1	5	10	11
2	8	26	10
3	21	8	10
4	24	21	2
5	2	12	9
6	21	28	21
7	25	17	14
8	4	14	11
9	11	19	18

```
[2]: df.var1
```

```
[2]: 0     8
1     5
2     8
3    21
4    24
5     2
6    21
7    25
8     4
9    11
Name: var1, dtype: int32
```

```
[4]: df[df.var1 > 15]["var1"]
```

```
[4]: 3    21  
4    24  
6    21  
7    25  
Name: var1, dtype: int32
```

```
[5]: df[(df.var1 > 15) & (df.var3 < 5)]
```

```
[5]:   var1  var2  var3  
4    24    21     2
```

```
[8]: df.loc[(df.var1 > 15),["var1","var2"]]
```

```
[8]:   var1  var2  
3    21    8  
4    24    21  
6    21    28  
7    25    17
```

```
[9]: df[(df.var1 > 15)][["var1","var2"]]
```

```
[9]:   var1  var2  
3    21    8  
4    24    21  
6    21    28  
7    25    17
```

Birleştirme (Join) İşlemleri

Birleştirme (Join) İşlemleri

```
[3]: import pandas as pd
import numpy as np
m = np.random.randint(1,30, size=(5,3))
df1 = pd.DataFrame(m, columns=["var1","var2","var3"])
df1
```

```
[3]:   var1  var2  var3
  0    12     9     5
  1     5    15    12
  2     2    28     2
  3    20    12     6
  4    14    25    19
```

```
[5]: df2 = df1 + 99
df2
```

```
[5]:   var1  var2  var3
  0   111   108   104
  1   104   114   111
  2   101   127   101
  3   119   111   105
  4   113   124   118
```

```
[6]: pd.concat([df1,df2])
```

```
[6]:   var1  var2  var3
```

0	12	9	5
1	5	15	12
2	2	28	2
3	20	12	6
4	14	25	19
0	111	108	104
1	104	114	111
2	101	127	101
3	119	111	105
4	113	124	118

Birleştirme işlemi yaptıktan fakat indexlerde bir karmaşıklık oldu.

```
[9]: pd.concat([df1,df2],ignore_index = True)
```

```
[9]:   var1  var2  var3
```

0	12	9	5
1	5	15	12
2	2	28	2
3	20	12	6
4	14	25	19
5	111	108	104
6	104	114	111
7	101	127	101
8	119	111	105
9	113	124	118

```
[12]: df2.columns
```

```
[12]: Index(['var1', 'var2', 'var3'], dtype='object')
```

```
[14]: df2.columns = ["var1","var2","deg3"]
```

```
[14]: df2
```

```
[14]:   var1  var2  deg3
```

	var1	var2	deg3
0	111	108	104
1	104	114	111
2	101	127	101
3	119	111	105
4	113	124	118

```
[15]: df1
```

```
[15]:   var1  var2  var3
```

	var1	var2	var3
0	12	9	5
1	5	15	12
2	2	28	2
3	20	12	6
4	14	25	19

```
[16]: pd.concat([df1, df2])
```

```
[16]:   var1  var2  var3  deg3
```

	var1	var2	var3	deg3
0	12	9	5.0	NaN
1	5	15	12.0	NaN
2	2	28	2.0	NaN
3	20	12	6.0	NaN
4	14	25	19.0	NaN
0	111	108	NaN	104.0
1	104	114	NaN	111.0
2	101	127	NaN	101.0
3	119	111	NaN	105.0
4	113	124	NaN	118.0

Bir veri setinin diğerinde karşılığı olmadığı için böyle bir sorun yaşıyoruz.

Bu sorunu kısmi olarak aşabiliyoruz.

join = "inner" argümanı ile veri setlerinin kesişimlerini alabiliyoruz.

```
[17]: pd.concat([df1, df2], join="inner") #kesisimlerini alır.
```

```
[17]:   var1  var2
```

	var1	var2
0	12	9
1	5	15
2	2	28
3	20	12
4	14	25
0	111	108
1	104	114
2	101	127
3	119	111
4	113	124

```
[57]: pd.concat([df1, df2], axis=1).reindex(df1.index)
```

```
[57]:   var1  var2  var3  var1  var2  deg3
```

	var1	var2	var3	var1	var2	deg3
0	12	9	5	111	108	104
1	5	15	12	104	114	111
2	2	28	2	101	127	101
3	20	12	6	119	111	105
4	14	25	19	113	124	118

İleri Birleştirme İşlemleri

Birebir Birleştirme

Tüm elemanların iki veri setinde de birebir yer alması durumudur.

```
[1]: import pandas as pd  
df1 = pd.DataFrame({"calisanlar":["Ali","Veli","Ayse","Fatma"],  
                     "grup":["Muhasebe","Muhendislik","Muhendislik","IK"]})  
df1
```

```
[1]:    calisanlar      grup  
0        Ali    Muhasebe  
1       Veli  Muhendislik  
2       Ayse  Muhendislik  
3     Fatma         IK
```

```
[2]: df2 = pd.DataFrame({"calisanlar":["Ali","Veli","Ayse","Fatma"],  
                     "ilk_giris":[2010,2009,2014,2019]})  
df2
```

```
[2]:    calisanlar  ilk_giris  
0        Ali      2010  
1       Veli      2009  
2       Ayse      2014  
3     Fatma      2019
```

```
[3]: pd.merge(df1,df2)
```

```
[3]:    calisanlar      grup  ilk_giris  
0        Ali    Muhasebe    2010  
1       Veli  Muhendislik    2009  
2       Ayse  Muhendislik    2014  
3     Fatma         IK      2019
```

Merge() Fonksiyonu birleştirme işleminin hangi değişkene göre yapılacağını kendisi anlıyor. Eğer bunu belirtmek istersek **on** argümanı aracılığı ile belirtebiliriz.

Her iki veri setinde de calisanlar olduğu için bu veri setlerini calisanlar'a göre birleştirdi.

```
[8]: pd.merge(df1, df2, on="calisanlar")
```

	calisanlar	grup	ilk_giris
0	Ali	Muhasebe	2010
1	Veli	Muhendislik	2009
2	Ayse	Muhendislik	2014
3	Fatma	IK	2019

Many to one (Çoktan teke)

Many to one (Çoktan teke)

```
[9]: df3 = pd.merge(df1,df2)  
df3
```

	calisanlar	grup	ilk_giris
0	Ali	Muhasebe	2010
1	Veli	Muhendislik	2009
2	Ayse	Muhendislik	2014
3	Fatma	IK	2019

```
[10]: df4 = pd.DataFrame({"grup": ["Muhasebe", "Muhendislik", "IK"],  
                         "mudur": ["Caner", "Mustafa", "Berkcan"]})  
df4
```

	grup	mudur
0	Muhasebe	Caner
1	Muhendislik	Mustafa
2	IK	Berkcan

```
[14]: pd.merge(df3,df4) #Many to one birlestirme.
```

	calisanlar	grup	ilk_giris	mudur
0	Ali	Muhasebe	2010	Caner
1	Veli	Muhendislik	2009	Mustafa
2	Ayse	Muhendislik	2014	Mustafa
3	Fatma	IK	2019	Berkcan

Many to Many (Çoktan çoka)

Many to Many (Çoktan çoka)

```
[19]: df5 = pd.DataFrame({'grup' : ['Muhasebe','Muhasebe','Muhendislik','Muhendislik','IK','IK'],
                       'yetenekler' : ['Matematik','Excel','Kodlama','Linux','Excel','Yonetim']})
df5
```

```
[19]:      grup  yetenekler
0    Muhasebe  Matematik
1    Muhasebe        Excel
2   Muhendislik     Kodlama
3   Muhendislik       Linux
4            IK        Excel
5            IK      Yonetim
```

```
[17]: df1
```

```
[17]:      calisanlar      grup
0          Ali  Muhasebe
1         Veli  Muhendislik
2         Ayse  Muhendislik
3        Fatma        IK
```

```
[21]: pd.merge(df1,df5) #Many to Many
```

```
[21]:      calisanlar      grup  yetenekler
0          Ali  Muhasebe  Matematik
1          Ali  Muhasebe        Excel
2         Veli  Muhendislik     Kodlama
3         Veli  Muhendislik       Linux
4         Ayse  Muhendislik     Kodlama
5         Ayse  Muhendislik       Linux
6        Fatma        IK        Excel
7        Fatma        IK      Yonetim
```

Aggregation & Grouping (Toplulaştırma ve Gruplama)

Basit toplulaştırma fonksiyonları:

- count()
- first()
- last()
- mean()
- median()
- min()
- max()
- std()
- var()
- sum()

```
[1]: import seaborn as sns #Bu kütüphanemiz içerisindeki bazı veri setlerini kullanıcaz.
```

```
[6]: df = sns.load_dataset("planets") #planets isimli dataset'i kullandık.  
df
```

```
[6]:
```

	method	number	orbital_period	mass	distance	year
0	Radial Velocity	1	269.300000	7.10	77.40	2006
1	Radial Velocity	1	874.774000	2.21	56.95	2008
2	Radial Velocity	1	763.000000	2.60	19.84	2011
3	Radial Velocity	1	326.030000	19.40	110.62	2007
4	Radial Velocity	1	516.220000	10.50	119.47	2009
...
1030	Transit	1	3.941507	NaN	172.00	2006
1031	Transit	1	2.615864	NaN	148.00	2007
1032	Transit	1	3.191524	NaN	174.00	2007
1033	Transit	1	4.125083	NaN	293.00	2008
1034	Transit	1	4.187757	NaN	260.00	2008

1035 rows × 6 columns

```
[7]: df.head()
```

```
[7]:
```

	method	number	orbital_period	mass	distance	year
0	Radial Velocity	1	269.300	7.10	77.40	2006
1	Radial Velocity	1	874.774	2.21	56.95	2008
2	Radial Velocity	1	763.000	2.60	19.84	2011
3	Radial Velocity	1	326.030	19.40	110.62	2007
4	Radial Velocity	1	516.220	10.50	119.47	2009

Artık satırlara **gözlem**, sütunlara ise **değişken** demeye alışmamızı.

```
[8]: df.shape
```

```
[8]: (1035, 6)
```

dataset'in 1035 gözlem, 6 değişkenden oluştuğunu gözlemeğemekteyiz.

```
[9]: df.mean() #Tüm değişkenlerin ortalamaları
```

```
[9]: number           1.785507
      orbital_period  2002.917596
      mass            2.638161
      distance        264.069282
      year            2009.070531
      dtype: float64
```

```
[10]: df["mass"].mean() # mass değişkeninin ortalaması.
```

```
[10]: 2.6381605847953216
```

```
[12]: df.count() #Degiskenlerdeki gözlem sayıları.
```

```
[12]: method          1035
      number          1035
      orbital_period  992
      mass            513
      distance        808
      year            1035
      dtype: int64
```

```
[13]: df.min() #Minimum değerler
```

```
[13]: method          Astrometry
      number          1
      orbital_period  0.0907063
      mass            0.0036
      distance        1.35
      year            1989
      dtype: object
```

```
[14]: df.max() #Maximum değerler
```

```
[14]: method          Transit Timing Variations
      number          7
      orbital_period  730000
      mass            25
      distance        8500
      year            2014
      dtype: object
```

```
[15]: df.sum() # Değişkenlerin değerlerinin toplamı
```

method	Radial Velocity	Radial Velocity	Radial Velocity	R...
number				1848
orbital_period				1.98689e+06
mass				1353.38
distance				213368
year				2079388
dtype:	object			

```
[16]: df.std() #Değişkenlerin standart sapması
```

number	1.240976
orbital_period	26014.728304
mass	3.818617
distance	733.116493
year	3.972567
dtype:	float64

```
[17]: df.var() #Değişkenlerin varyansı
```

number	1.540022e+00
orbital_period	6.767661e+08
mass	1.458183e+01
distance	5.374598e+05
year	1.578129e+01
dtype:	float64

```
[18]: df.describe() #Verisetindeki tüm değişkenleri betimsel istatistikleri anlamında görebiliyoruz.
```

	number	orbital_period	mass	distance	year
count	1035.000000	992.000000	513.000000	808.000000	1035.000000
mean	1.785507	2002.917596	2.638161	264.069282	2009.070531
std	1.240976	26014.728304	3.818617	733.116493	3.972567
min	1.000000	0.090706	0.003600	1.350000	1989.000000
25%	1.000000	5.442540	0.229000	32.560000	2007.000000
50%	1.000000	39.979500	1.260000	55.250000	2010.000000
75%	2.000000	526.005000	3.040000	178.500000	2012.000000
max	7.000000	730000.000000	25.000000	8500.000000	2014.000000

```
[19]: df.describe().T #Transpozu alındığında
```

	count	mean	std	min	25%	50%	75%	max
number	1035.0	1.785507	1.240976	1.000000	1.00000	1.0000	2.000	7.0
orbital_period	992.0	2002.917596	26014.728304	0.090706	5.44254	39.9795	526.005	730000.0
mass	513.0	2.638161	3.818617	0.003600	0.22900	1.2600	3.040	25.0
distance	808.0	264.069282	733.116493	1.350000	32.56000	55.2500	178.500	8500.0
year	1035.0	2009.070531	3.972567	1989.000000	2007.00000	2010.0000	2012.000	2014.0

Elimizdeki verisetinin eksik gözlemleri silip describe yapmak istediğimizde **dropna()** fonksiyonunu kullanırız.

```
[20]: df.dropna().describe().T
```

	count	mean	std	min	25%	50%	75%	max
number	498.0	1.734940	1.175720	1.0000	1.00000	1.000	2.0000	6.0
orbital_period	498.0	835.778671	1469.128259	1.3283	38.27225	357.000	999.6000	17337.5
mass	498.0	2.509320	3.636274	0.0036	0.21250	1.245	2.8675	25.0
distance	498.0	52.068213	46.596041	1.3500	24.49750	39.940	59.3325	354.0
year	498.0	2007.377510	4.167284	1989.0000	2005.00000	2009.000	2011.0000	2014.0

Grouping

Grouping

```
[26]: import pandas as pd
df = pd.DataFrame({'gruplar' : ['A','B','C','A','B','C'],
                   'veri' : [10,11,52,23,43,55]}, columns=['gruplar','veri'])
df
```

	gruplar	veri
0	A	10
1	B	11
2	C	52
3	A	23
4	B	43
5	C	55

Genelde graplama işlemleri ile toplulaştırma(Aggregation) işlemleri bir arada kullanılır.

```
[27]: df.groupby("gruplar") #gruplar içerisindeki grupları yakaladı.
```

```
[27]: <pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000018B7EA5D648>
```

```
[28]: df.groupby("gruplar").mean() # Ortalamasını aldı.
```

	veri
gruplar	
A	16.5
B	27.0
C	53.5

```
[30]: df.groupby("gruplar").sum()
```

```
[30]: veri
```

gruplar

A	33
B	54
C	107

```
[33]: import seaborn as sns
df = sns.load_dataset("planets")
df.head()
```

```
[33]: method  number  orbital_period  mass  distance  year
```

0	Radial Velocity	1	269.300	7.10	77.40	2006
1	Radial Velocity	1	874.774	2.21	56.95	2008
2	Radial Velocity	1	763.000	2.60	19.84	2011
3	Radial Velocity	1	326.030	19.40	110.62	2007
4	Radial Velocity	1	516.220	10.50	119.47	2009

```
[37]: df.groupby("method")["orbital_period"].describe()
#method'a göre grupla. orbital_period değişkeninin istatistikleri al.
```

method	count	mean	std	min	25%	50%	75%	max
Astrometry	2.0	631.180000	544.217663	246.360000	438.770000	631.180000	823.590000	1016.000000
Eclipse Timing Variations	9.0	4751.644444	2499.130945	1916.250000	2900.000000	4343.500000	5767.000000	10220.000000
Imaging	12.0	118247.737500	213978.177277	4639.150000	8343.900000	27500.000000	94250.000000	730000.000000
Microlensing	7.0	3153.571429	1113.166333	1825.000000	2375.000000	3300.000000	3550.000000	5100.000000
Orbital Brightness Modulation	3.0	0.709307	0.725493	0.240104	0.291496	0.342887	0.943908	1.544929
Pulsar Timing	5.0	7343.021201	16313.265573	0.090706	25.262000	66.541900	98.211400	36525.000000
Pulsation Timing Variations	1.0	1170.000000	NaN	1170.000000	1170.000000	1170.000000	1170.000000	1170.000000
Radial Velocity	553.0	823.354680	1454.926210	0.736540	38.021000	360.200000	982.000000	17337.500000
Transit	397.0	21.102073	46.185893	0.355000	3.160630	5.714932	16.145700	331.600590
Transit Timing Variations	3.0	79.783500	71.599884	22.339500	39.675250	57.011000	108.505500	160.000000

İleri Toplulaştırma İşlemleri(Aggregate, filter, transform, apply)

Aggregate

Aggregate

```
[38]: import pandas as pd
df = pd.DataFrame({"gruplar" : ["A","B","C","A","B","C"],
                    "degisken1" : [10,23,33,22,11,99],
                    "degisken2" : [100,253,333,262,111,969]},
                    columns = ["gruplar","degisken1","degisken2"])
df
```

```
[38]:   gruplar  degisken1  degisken2
      0        A         10        100
      1        B         23        253
      2        C         33        333
      3        A         22        262
      4        B         11        111
      5        C         99        969
```

```
[40]: df.groupby("gruplar").mean()
```

```
[40]:    degisken1  degisken2
gruplar
      A         16        181
      B         17        182
      C         66        651
```

```
[45]: import numpy as np
df.groupby("gruplar").aggregate(["min",np.median, max])

#Yaptığımız gruptama içerisinde kendi istediğimiz istatistikî değerleri
#bir arada görmek için agggregate kullanırız.
```

```
[45]:           degisken1           degisken2
              min  median  max  min  median  max
gruplar
      A    10     16    22   100     181    262
      B    11     17    23   111     182    253
      C    33     66    99   333     651    969
```

İki değişken için iki ayrı istatistiki hesaplama yapmak istiyoruz.

```
[49]: df.groupby("gruplar").aggregate({"degisken1" : "min", "degisken2" : np.median})
```

```
[49]: degisken1  degisken2
```

gruplar	degisken1	degisken2
A	10	181
B	11	182
C	33	651

filter

Filter

Pandas'ın sunduğu özelliklerden daha karmaşık bir isteğimiz olduğunda kendi fonksiyonumuzu yazıp ona göre filtreleyebiliriz.

```
[1]: import pandas as pd
df = pd.DataFrame({"gruplar" : ["A","B","C","A","B","C"],
                   "degisken1" : [10,23,33,22,11,99],
                   "degisken2" : [100,253,333,262,111,969]},
                   columns = ["gruplar","degisken1","degisken2"])
df
```

```
[1]: gruplar  degisken1  degisken2
      0       A        10       100
      1       B        23       253
      2       C        33       333
      3       A        22       262
      4       B        11       111
      5       C        99       969
```

```
[6]: def filter_func(x):
      return x["degisken1"].std() > 9
#degisken1'e göre standart sapması 9'dan büyük olan değerler
```

```
[7]: df.groupby("gruplar").std() #standart sapmalar
```

```
[7]:      degisken1  degisken2
```

gruplar

A	8.485281	114.551299
B	8.485281	100.409163
C	46.669048	449.719913

```
[8]: df.groupby("gruplar").filter(filter_func)
```

```
[8]:      gruplar  degisken1  degisken2
```

2	C	33	333
5	C	99	969

Aynı işlemin **lambda** ile çözümü:

```
[9]: df.groupby("gruplar").filter(lambda x : x["degisken1"].std() > 9)
```

```
[9]:      gruplar  degisken1  degisken2
```

2	C	33	333
5	C	99	969

transform

transform

Kendi tanımladığımız bir fonksiyonu değişkenler üzerinde uygulayabiliyoruz.

```
[2]: import pandas as pd
df = pd.DataFrame({"gruplar" : ["A","B","C","A","B","C"],
                   "degisken1" : [10,23,33,22,11,99],
                   "degisken2" : [100,253,333,262,111,969]},
                   columns = ["gruplar","degisken1","degisken2"])
df
```

```
[2]:      gruplar  degisken1  degisken2
```

0	A	10	100
1	B	23	253
2	C	33	333
3	A	22	262
4	B	11	111
5	C	99	969

```
[3]: df["degisken1"]*9
```

```
[3]: 0      90
1     207
2     297
3     198
4      99
5    891
Name: degisken1, dtype: int64
```

```
[4]: df_a = df.iloc[:, 1:3]
df_a
```

```
[4]:   degisken1  degisken2
```

	degisken1	degisken2
0	10	100
1	23	253
2	33	333
3	22	262
4	11	111
5	99	969

Birazdan yapacağımız işlem sayısal bir işlem olduğundan gruplar değişkenine uygulamak istediğimizde hata ile karşılaşmamak için, yukarıdaki gibi grupları ayrı tutmamız gerekiyor.

```
[5]: df_a.transform(lambda x : x-x.mean())
#yakaladı olsuğu bütün elemanlardan o değişkenin ortalamasını çıkartacak.
```

```
[5]:   degisken1  degisken2
```

	degisken1	degisken2
0	-23.0	-238.0
1	-10.0	-85.0
2	0.0	-5.0
3	-11.0	-76.0
4	-22.0	-227.0
5	66.0	631.0

Apply

Apply

```
[4]: import pandas as pd
import numpy as np
df = pd.DataFrame({"degisken1" : [10,23,33,22,11,99],
                   "degisken2" : [100,253,333,262,111,969]},
                   columns = ["degisken1","degisken2"])
df
```

```
[4]:   degisken1  degisken2
 0         10      100
 1         23      253
 2         33      333
 3         22      262
 4         11      111
 5         99      969
```

apply() fonksiyonu tipki transform fonksiyonu ve filter fonksiyonu gibi değişkenlerin üzerinde gezinme yeteneği olan ve **aggregation**(toplulaştırma) amacıyla kullanılacak olan bir fonksiyondur.

```
[9]: df.apply(np.sum)
```

```
[9]: degisken1    198
      degisken2   2028
      dtype: int64
```

```
[8]: df.apply(np.mean)
```

```
[8]: degisken1    33.0
      degisken2   338.0
      dtype: float64
```

```
[12]: df = pd.DataFrame({"gruplar" : ["A","B","C","A","B","C"],
                         "degisken1" : [10,23,33,22,11,99],
                         "degisken2" : [100,253,333,262,111,969]},
                         columns = ["gruplar","degisken1","degisken2"])
df
```

```
[12]:    gruplar  degisken1  degisken2
0         A        10       100
1         B        23       253
2         C        33       333
3         A        22       262
4         B        11       111
5         C        99       969
```

```
[14]: df.groupby("gruplar").apply(np.sum)
```

```
[14]:    gruplar  degisken1  degisken2
gruplar
A      AA        32       362
B      BB        34       364
C      CC       132      1302
```

```
[18]: df
```

```
[18]:    gruplar  degisken1  degisken2
0         A        10       100
1         B        23       253
2         C        33       333
3         A        22       262
4         B        11       111
5         C        99       969
```

```
[26]: df.groupby("gruplar").apply(lambda x : (x["degisken1"]-x["degisken2"]))
```

```
[26]: gruplar
A      0     -90
      3    -240
B      1    -230
      4    -100
C      2    -300
      5   -870
dtype: int64
```

Bu işlemi apply ile yapabiliyoruz ancak transform hata verir.

Pivot Tablolar

Veri setleri üzerinde bazı satır ve sütun işlemleri yaparak, veri setini amaca uygun hale getirmek için kullanılan yapılardır.

`groupby()`'ın çok boyutlu versiyonu olarak düşünülebilir.

```
[27]: import pandas as pd
import seaborn as sns
titanic = sns.load_dataset("titanic")
titanic
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

891 rows × 15 columns

Cinsiyete göre gruplayıp hayatta olma ortalamalarına bakalım.

```
[30]: titanic.groupby("sex")["survived"].mean()
```

```
[30]: sex
      female    0.742038
      male      0.188908
      Name: survived, dtype: float64
```

```
[31]: titanic.groupby("sex")[["survived"]].mean() #Basit bir pivot işlemi.
      # değişken etrafına köşeli parantez ekleyerek dataset olarak gözlemleyelim.
```

```
[31]:      survived
```

sex	survived
female	0.742038
male	0.188908

Cinsiyete ve class'a göre ölüm ortalamaları:

```
[37]: titanic.groupby(["sex","class"])["survived"].aggregate("mean")
```

```
[37]:          survived
```

sex	class	survived
female	First	0.968085
	Second	0.921053
	Third	0.500000
male	First	0.368852
	Second	0.157407
	Third	0.135447

```
[39]: titanic.groupby(["sex","class"])["survived"].aggregate("mean").unstack()  
#unstack bizi hiyerarsik görünümden kurtarır.,
```

```
[39]:      class    First   Second   Third
```

	sex	First	Second	Third
	female	0.968085	0.921053	0.500000
	male	0.368852	0.157407	0.135447

Bir boyut daha ekleyerek pivot table oluşturduk.

pivot_table

```
[40]: #Pivot ile table
```

```
[41]: titanic.pivot_table("survived", index = "sex", columns = "class")
```

```
[41]:      class    First   Second   Third
```

	sex	First	Second	Third
	female	0.968085	0.921053	0.500000
	male	0.368852	0.157407	0.135447

```
[45]: titanic.age
```

```
[45]: 0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
...
886    27.0
887    19.0
888    NaN
889    26.0
890    32.0
Name: age, Length: 891, dtype: float64
```

Bu sürekli değişkeni bir kategorik değişkene çevirip, bu kategorik değişkenin sınıflarını da pivot table'a boyut olarak ekleyelim.

```
[51]: age = pd.cut(titanic["age"], [0,18,90])
age.head(15)
```

```
[51]: 0      (18.0, 90.0]
1      (18.0, 90.0]
2      (18.0, 90.0]
3      (18.0, 90.0]
4      (18.0, 90.0]
5          NaN
6      (18.0, 90.0]
7      (0.0, 18.0]
8      (18.0, 90.0]
9      (0.0, 18.0]
10     (0.0, 18.0]
11     (18.0, 90.0]
12     (18.0, 90.0]
13     (18.0, 90.0]
14     (0.0, 18.0]
Name: age, dtype: category
Categories (2, interval[int64]): [(0, 18] < (18, 90]]
```

Sürekli bir değişken olan age değişkenini kategorik değişken haline getirdik.

```
[58]: titanic.pivot_table("survived", ["sex",age], "class")
```

		class	First	Second	Third
	sex	age			
female	(0, 18]	0.909091	1.000000	0.511628	
	(18, 90]	0.972973	0.900000	0.423729	
male	(0, 18]	0.800000	0.600000	0.215686	
	(18, 90]	0.375000	0.071429	0.133663	

Dış Kaynaklı Veri Okuma

Dış Kaynaklı Veri Okuma

.txt formunu ve .csv formunu aynı fonksiyon ile okuyabiliyoruz.

```
[1]: import pandas as pd  
[2]: pd.read_csv("reading_data/ornekcsv.csv")  
[2]:      a;b;c  
0    78;12;1  
1    78;12;2  
2    78;324;3  
3    7;2;4  
4    88;23;5  
5    6;2;  
6    56;11;6  
7    7;12;7  
8    56;21;7  
9    346;2;8  
10   5;1;8  
11   456;21;8  
12   3;12;88
```

Veri düzgün biçimde gelmedi. Veri okuma işlemlerinde en sık karşılaşılan problem budur. Paylaşılan veri seti genellikle csv ya da txt formunda olduğunda değişkenler birbirinden bazı ayraçlar ile ayrılır.

Ayraç olarak öntanımlı değer ",","dür. Bize gelen veride ise ";" kullanılmış. **sep** argümanı ile bu sorunu çözebiliriz.

c
c
cc

csv okuma

```
[4]: #csv okuma  
pd.read_csv("reading_data/ornekcsv.csv", sep=";")
```

```
[4]:      a    b    c  
0     78   12   1.0  
1     78   12   2.0  
2     78  324   3.0  
3      7    2   4.0  
4     88   23   5.0  
5      6    2   NaN  
6     56   11   6.0  
7      7   12   7.0  
8     56   21   7.0  
9    346    2   8.0  
10     5    1   8.0  
11   456   21   8.0  
12     3   12  88.0
```

txt okuma

```
[6]: #txt okuma  
pd.read_csv("reading_data/duz_metin.txt")
```

```
[6]:      1 2  
0     2 2  
1     3 2  
2     4 2  
3     5 2  
4     6 2  
5     7 2  
6     8 2  
7     9 2  
8    10 2
```

sep argümanını kullanmadık fakat veriler düzgün biçimde geldi.

Düz metinlerde arada boşluk olsa dahi bunu fonksiyon görebiliyor.

Excel dosyası okuma

Excel Dosyası Okuma

```
[9]: pd.read_excel("reading_data/ornekx.xlsx")
```

```
[9]:
```

	a	b	c
0	78	12	1.0
1	78	12	2.0
2	78	324	3.0
3	7	2	4.0
4	88	23	5.0
5	6	2	NaN
6	56	11	6.0
7	7	12	7.0
8	56	21	7.0
9	346	2	8.0
10	5	1	8.0
11	456	21	8.0
12	3	12	88.0

```
[10]: df = pd.read_excel("reading_data/ornekx.xlsx")
```

```
[11]: type(df)
```

```
[11]: pandas.core.frame.DataFrame
```

DataFrame'lere yaptığımiz tüm işlemleri artık burada da yapabiliyoruz.

```
[12]: df.head()
```

```
[12]:
```

	a	b	c
0	78	12	1.0
1	78	12	2.0
2	78	324	3.0
3	7	2	4.0
4	88	23	5.0

```
[14]: df.columns = ("A", "B", "C") #column isimlerini değiştirdik.  
df
```

```
[14]:      A      B      C  
0    78     12    1.0  
1    78     12    2.0  
2    78    324    3.0  
3     7      2    4.0  
4    88     23    5.0  
5     6      2    NaN  
6    56     11    6.0  
7     7     12    7.0  
8    56     21    7.0  
9   346      2    8.0  
10    5      1    8.0  
11  456     21    8.0  
12    3     12  88.0
```

Sıfırdan txt okuma

GitHub'da tek bir veri setini almak istediğimizde, veri setinin olduğu sayfada **Raw** butonuna tıklayıp verisetinin ham haline ulaşabiliyoruz. Buradaki verileri txt dosyasına kopyalayıp kullanabiliriz.

```
[15]: #sıfırdan txt okuma  
tips = pd.read_csv("reading_data/data.txt")
```

```
[19]: tips.head()
```

```
[19]:  total_bill  tip      sex  smoker  day    time  size  
0    16.99  1.01  Female      No  Sun Dinner    2  
1    10.34  1.66    Male      No  Sun Dinner    3  
2    21.01  3.50    Male      No  Sun Dinner    3  
3    23.68  3.31    Male      No  Sun Dinner    2  
4    24.59  3.61  Female      No  Sun Dinner    4
```

Pandas Alıştırmalar-1

Question 1:

"df" isimli bir Pandas DataFrame için ilk 2 gözleme erişmek istenilirse aşağıdaki kodlardan hangisi kullanılır?

df.head()

df.tail()

df.describe()

df.head(2)

Question 2:

"df" isimli bir Pandas DataFrame için son 3 gözleme erişmek istenilirse aşağıdaki kodlardan hangisi kullanılır?

df.head(3)

df.tail(3)

df.describe()

df.head()

Question 3:

```
seri = pd.Series([121,200,150,99], argüman_ismi = ["reg","loj","cart","rf"])
```

Yukarıda "argüman_ismi" yazan bölüme aşağıdakilerden hangisi gelmelidir?

columns

column

indexes

index

Question 4:

Bir Pandas DataFrame oluştururken *değişken isimlerini belirtmek için* hangi arguman kullanılır?

variable

variables

column

columns

Question 5:

Solda verilen "df" isimli Pandas DataFrame için aşağıdaki seçeneklerden hangisi uygulanırsa sağdaki çıktıya ulaşılır?

	col1	col2	col3
a	9	2	7
b	3	3	4
c	2	9	8
d	1	7	0
e	0	5	6

	col1	col2	col3
c	2	9	8
d	1	7	0
e	0	5	6

df[1:3]

df[1:3,:]

df["c":"e"]

df["col1", "col3"]

Question 7:

Pandas DataFrame üzerinde **indeks isimlendirmelerine bağlı kalarak** (label based) gözlem ve değişken seçimi yapmak için kullanılır. Boşluğa aşağıdakilerden hangisi gelmelidir?

loc

iloc

slice

fancy index

Question 6:

Pandas DataFrame üzerinde hem gözlem hem değişken seçimi için **indeks isimlendirmelerinden (labelardan) bağımsız** seçim yapmak üzere kullanılır. Boşluğa aşağıdakilerden hangisi gelmelidir?

loc

iloc

slice

fancy index

Question 8:

```
1 | seri = pd.Series([121,200,150,99])
2 | seri.values
```

Yukarıda verilen kodun çıktısı aşağıdakilerden hangisidir?

pd.DataFrame([121, 200, 150, 99])

array(["reg","loj","cart","rf"])

pd.Series([121, 200, 150, 99])

array([121, 200, 150, 99])

Question 9:

```
1 import numpy as np
2 m = np.arange(1,7).reshape((3,2))
3 pd.DataFrame(m, columns = ["var1","var2"])
```

Yukarıda kodu verilen kodun çıktısı hangisidir?

1 var1 var2
2
3 var1 1 2
4
5 var1 3 4
6
7 var1 5 6

1 var1 var2
2
3 0 1 2
4
5 1 3 4
6
7 2 5 6

Question 10:

Aşağıda bir kod ve çıktısı verilmiştir. Buna göre hangisi bir Pandas DataFrame oluşturur?

Kod:

`type(sozluk)`

Çıktı:

`dict`

1 import numpy as np
2 np.DataFrame(dict)

1 import numpy as np
2 np.DataFrame(sozluk)

1 import pandas as pd
2 pd.DataFrame(dict)

1 import pandas as pd
2 pd.DataFrame(sozluk)

Pandas Alıştırmalar-2

Question 1:

Verilen kod parçasına göre aşağıdakilerden hangisi yanlıştır?

```
1 import numpy as np
2 import pandas as pd
3
4 m = np.random.randint(1,30, size = (10,3))
5 df = pd.DataFrame(m, columns = ["var1","var2","var3"])
```

- Yukarıdaki kod parçasının 3. satırında , 10x3'lük rastgele integer değerlerden numpy array oluşturma işlemi yapılmıştır.
- `df[df.var1 > 15]` ile df'nin 1. kolonuna bir filtreleme yapılabilir
- Pandas DataFrame oluşturmayı tamamlamak için numpy array yapısı sözlük yapısına çevrilmelidir
- 'pd', 'pandas' kütüphanesine verilen takma isimdir. 'pan' şeklinde de kullanılabilir

Question 2:

Verilen kod parçası ile ilgili aşağıdakilerden hangisi yanlıştır?

```
df[(df.var1 > 15)][["var1","var2"]]
```

- `[["var1","var2"]]` ifadesinde çift köşeli parantez kullanılmasının sebebi, çıktıının tablo şeklinde gösterilmesi içindir
- Çıktının türü Pandas DataFrame olacaktır
- `df[(df.var1 > 15)]["var1"]` kodu ile bir değişken kolonu seçilebilir ve türü pandas.Series olur
- Çıktıda değişken isimleri görünmez

Question 3:

df1 Pandas DataFrame olarak tanımlanmıştır. Verilen koda göre hangisi yanlıştır?

`df2 = df1 + 99`

Çalışmaz

İlk satırın her elemanına 99 eklenir

İlk sütunun her elemanına 99 eklenir

Her elemana 99 eklenir Cevapta yanlışlık olabilir.

Question 4:

df1 ve df2 Pandas DataFrame olarak tanımlanmıştır. Verilen kod parçası ile ilgili aşağıdakilerden hangisi yanlıştır?

`pd.concat([df1,df2])`

Kolon adları aynı ise satır bazında alçalta birleşirler

"ignore_index = True" argümanı ile oluşan DataFrame indeksleri gösterilmmez

Kolon adları aynı ise sütun bazında alçalta birleşirler

Kolon adları farklı ise uyarı hatası verir

Question 5:

Verilen df1 ve df2 ile ilgili aşağıdakilerden hangisi yanlıştır?

df1:

	calisanlar		grup
0	Ali	Muhasebe	
1	Veli	Muhendislik	
2	Ayse	Muhendislik	
3	Fatma		IK

df2:

	calisanlar	ise_giris
0	Ayse	2010
1	Ali	2009
2	Veli	2014
3	Fatma	2019

- pd.merge(df1, df2) kodu, ortak olan 'calisanlar' değişkeni üzerinden dataframe'leri sütun bazında birleştirir
- pd.merge(df1, df2) ile Indexlere göre değil, ortak verilere göre birleştirilir
- pd.merge(df1, df2) sonucu toplam 3 sütun oluştur
- pd.merge(df1, df2) sonucu toplam 8 satır oluştur

Question 6:

Pandas kütüphanesinin merge fonksiyonu için yapılan genellemelerden hangisi yanlıştır?

- Dataframe'leri birleştirir
- Dataframelerdeki ortak kolon varsa bu kolon bir defa yazılır
- Dataframe'leri kolon bazında (yanyana) birleştirir
- on='colon' argümanı kullanmaksızın çalışmaz

Question 7:

Aşağıdakilerden hangisi toplulaştırma (aggregation) fonksiyonlarından biri değildir?

count()

top()

last()

min()

Question 8:

Aşağıdakilerden hangisi yanlıştır?

var() varyansı hesaplar.

median() en çok tekrar eden veriyi gösterir.

mean() ortalamayı hesaplar.

std() standart sapmayı hesaplar.

Question 10:

Dataframe'lere uygulanan "describe()" metodunun çıktısında hangi bilgi yoktur?

Sum

Mean

Count

Median

Pandas Alıştırmalar – 3

Question 1:

```
df.groupby("gruplar").aggregate([min, np.median, max])
```

yukarıdaki kod ile ne amaçlanmıştır?

- df dataframe'i grüplamak, sırasıyla her satırda min, np.median ve max fonksiyonu uygulamak
- df dataframe'i grüplamak, 1. gruba min, 2. gruba np.median ve 3. gruba max fonksiyonu uygulamak
- df dataframe'i grüplamak, sırasıyla her sütun için min, np.median ve max fonksiyonlarını uygulamak ve her sütun için bu 3 sonucu birer sütun olarak göstermek
- df dataframe'i grüplamak, sırasıyla her sütuna min, np.median ve max fonksiyonlarını uygulamak, her sütun için nihai sonuç tek sütun olacak şekilde göstermek

Question 2:

"gruplar" dışında 2 kolona sahip olan "df" isimli Pandas DataFrame önce grüplamak sonra da 1. kolona min fonksiyonu, 2.kolona max fonksiyonu uygulanmak isteniyor. Hangi seçenek doğrudur?

- df.groupby("gruplar").aggregate({"degisken1": "min", "degisken2": "max"})
- df.groupby("gruplar").agg(["degisken1": "min", "degisken2": "max"])
- df.groupby("gruplar").agg({"degisken1": "max", "degisken2": "min"})
- df.groupby("gruplar").aggregate(["degisken1": "min", "degisken2": "max"])

Question 3:

Aşağıda çıktısı verilen kod aşağıdakilerden hangisi olabilir?

Çıktı:

class	First	Second	Third
sex			
female	0.968085	0.921053	0.500000
male	0.368852	0.157407	0.135447

titanic.pivot_table("survived", columns = "sex", index = "class")

titanic.pivot_table("survived", index = "sex", columns = "class")

Question 4:

Aşağıdakilerden hangileri doğrudur?

I. unstack() fonksiyonu çoklu indeks yapısındaki dataframe'i enine genişletir.

II. unstack() fonksiyonu çoklu indeks yapısındaki dataframe'in bir indeksini kolon başlığı olarak yer değiştirir.

III. stack() fonksiyonu unstack() fonksiyonunun tersidir.

I,II,III

Question 5:

Aşağıdakilerden hangisi doğrudur?

import pandas as np
pd.read_csv("ornekcsv.csv")

import pandas as pd
np.read_csv("ornekcsv.csv")

import pandas as pd
pd.read_csv("ornekcsv.csv")

import pandas as np
np.readcsv("ornekcsv.csv")

Question 6:

Aşağıdakilerden hangisi yanlıştır? (pandas'ın import edildiğini varsayıınız)

pd.read_csv("reading_data/ornekcsv.csv", sep = ";")

pd.read_txt("reading_data/duz_metin.txt")

pd.read_excel("reading_data/ornekx.xlsx")

pd.read_csv("reading_data/duz_metin.txt")

Question 7:

"numbers" değişkeninin bir Pandas Serisi olduğu bilindiğine göre aşağıdaki kodun çıktısı hangisi olabilir?

Kod:

`numbers.dtype`

`dtype('int64')`

`str`

`dtype(string)`

`dtype(boolean)`

Question 8:

Bir Pandas Serisine "ndim" metodu uygulanırsa ne bilgisini almış oluruz?

Eleman sayısı

Uzunluğu

Boyutu

Hata verir

Question 9:

Bir Pandas Serisini array olarak almak istersek hangi metodu uygulamak gereklidir?

column

value

value()

values

Question 10:

"seri" değişkeninin bir Pandas Serisi olduğu bilindiğine göre aşağıdaki kod ile ne amaçlanmıştır?

Kod:

`"KNN" in seri`

- "KNN" ifadesini seriyeye dönüştürmek
- "KNN" ifadesini seriyeye eklemek
- "KNN" ifadesinin "seri" içinde olup olmadığını sorgulamak
- "KNN" ifadesinin türünü değiştirmek