



Data Science  
Çalışma Dökümanı  
Recep Aydoğdu

# İçindekiler

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
Integer .....	15
Float.....	15
String .....	15
Type .....	15
String Metodları .....	16
len() .....	16
upper() & lower().....	16
isupper() & islower().....	16
replace() .....	16
strip() .....	16
dir() .....	17
capitalize() .....	17
title() .....	17
Substring.....	17
Type Dönüşümleri .....	17
print() fonksiyonu .....	17
Python Programlama Alıştırmalar – 1 .....	18
Python Programlama Alıştırmalar – 2 .....	21
Python Programlama Alıştırmalar – 3 .....	26
Veri Yapıları (Data Types) .....	30
Listeler .....	30
Liste Elemanlarına Ulaşma .....	30
Liste İçi Type Sorgulama .....	30
Liste elemanlarını değiştirme .....	31
Listeye eleman ekleme.....	31
Listeden eleman silme.....	31
append ve remove metodları.....	31
insert metodu .....	31

pop metodu .....	31
count metodu .....	32
copy metodu .....	32
extend metodu .....	32
index metodu .....	32
reverse metodu .....	32
sort metodu .....	32
clear metodu .....	33
Tuple (Demet).....	33
Tuple Oluşturma.....	33
Eleman İşlemleri .....	33
Dictionary (Sözlük).....	33
Dictionary Nedir?.....	33
Dictionary Oluşturma .....	33
Eleman Seçme İşlemleri.....	34
Eleman Ekleme & Değiştirme .....	34
Sets (Kümeler) .....	35
Set Oluşturma.....	35
Set'lere eleman ekleme ve çıkarma işlemleri.....	36
Set'lerde Fark İşlemleri.....	38
Set'lerde Kesişim ve Birleşim İşlemleri .....	38
Set'lerde Sorgu İşlemleri .....	39
Veri Yapıları Özet .....	39
Python Programlama Alıştırmalar – 4 .....	40
Python Programlama Alıştırmalar – 5 .....	43
Python Programlama Alıştırmalar – 6 .....	48
Fonksiyonlar .....	53
Fonksiyon Nedir?.....	53
Matematiksel İşlemler .....	53
Üs Alma.....	53
Fonksiyon Nasıl Yazılır ?.....	53
Bilgi Notuyla Çıktı Üretmek .....	54
İki Argümanlı Fonksiyon Tanımlamak.....	54
Ön Tanımlı Argümanlar .....	55
Argümanların Sıralaması .....	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
<b>Uygulama:</b> if ve input ile kullanıcı etkileşimli program .....	60
Döngüler .....	60
For Döngüsü .....	60
Döngü ve Fonksiyonların Birlikte Kullanımı .....	61
<b>Uygulama:</b> 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 Nedir? .....	78
Class Özellikleri .....	78
Class Özelliklerine Erişmek .....	78
Class Özelliklerini Değiştirmek.....	78
Class Örneklemesi (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
Örnek-1: Bağımsızlık .....	81
Örnek-2: Ölümcül Yan Etkiler .....	82
İsimsiz Fonksiyonlar (Lambda) (Anonymous Functions) .....	83
Vektörel Operasyonlar (Vectorel Operations).....	83
OOP ile iki listeyi çarpmak .....	83
Functionel Programming ile .....	83

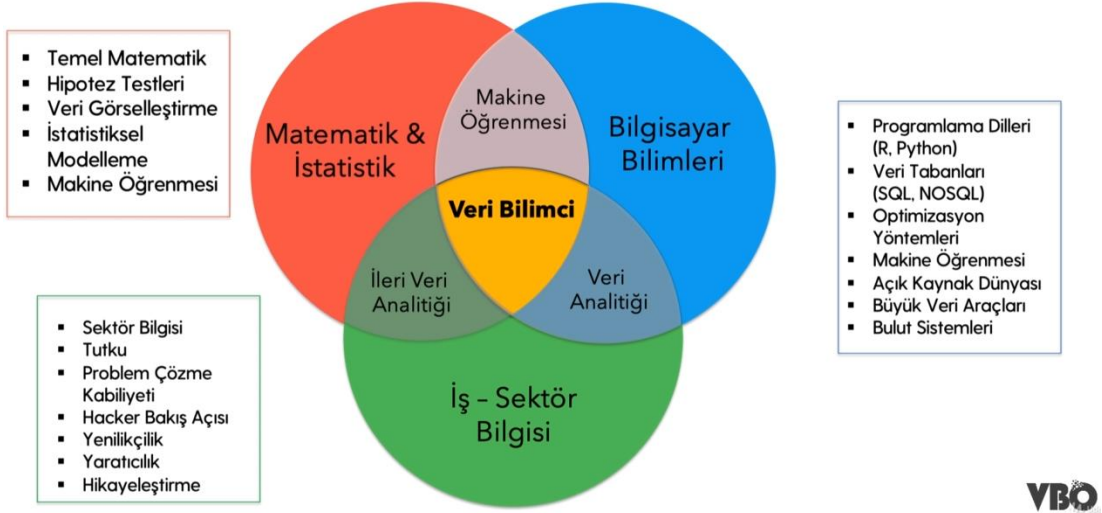
Map & Filter & Reduce .....	84
Map .....	84
Filter.....	84
Reduce .....	84
Modül Oluşturma .....	84
Hatalar/İstisnalar (exception).....	85
Python Programlama Alıştırmalar – 10 .....	86
Python Programlama Alıştırmalar – 11 .....	90
Python Programlama Alıştırmalar – 12 .....	95
Python ile Veri Manipülasyonu: 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
Basit Index ile Fancy kullanımı.....	112
Slice ile Fancy kullanımı .....	112

# Data Science

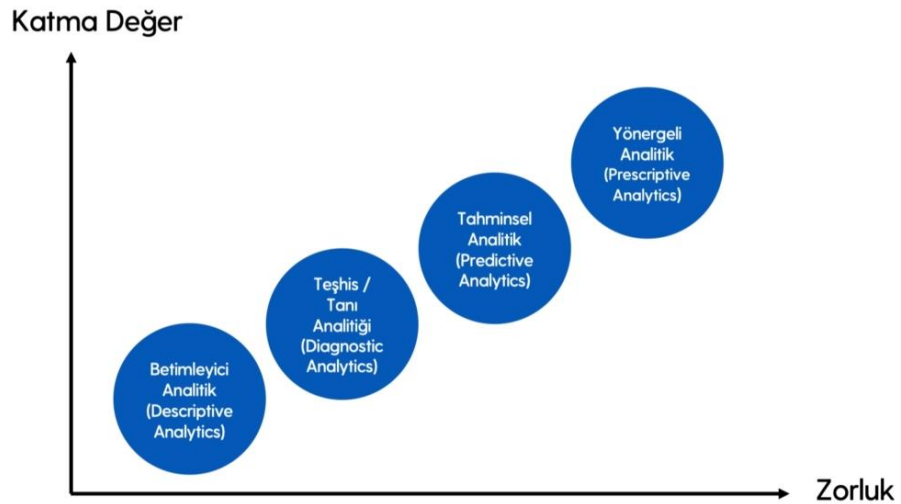
## VERİ BİLİMİNE GİRİŞ



Veri Bilimci, veriden faydalı bilgi çıkarma sürecini yöneten kişidir.



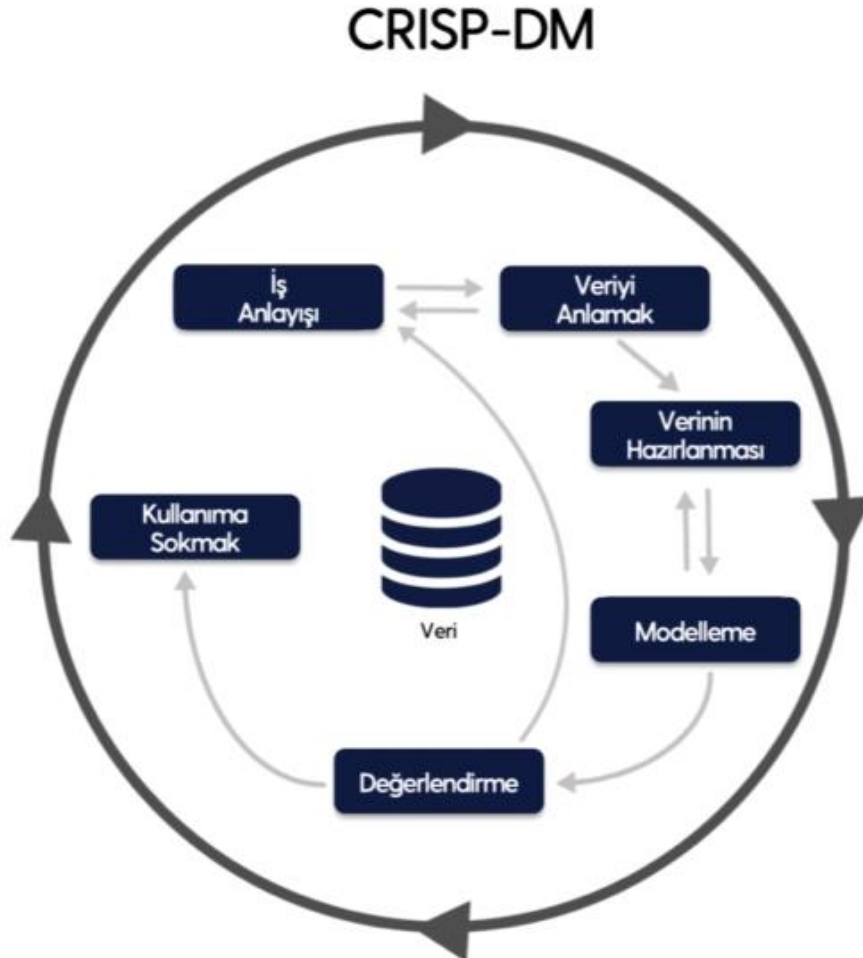
## VERİDEN FAYDALI BİLGİ ÇIKARMAK



## Data Science Kullanılan Alanlar

- Arkadaş önerileri
  - Otomatik fotoğraf etiketlemeleri
  - Hedefli içerik pazarlama
  - Otomatik mesaj tamamlama
  - Hedefli ürün pazarlama
  - Tavsiye sistemleri
  - Müşteri segmentasyonu
  - Kanseri/Hastalık teşhisi
  - 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 hayatta 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 Alıştırmalar – 2

Soru 1:

Aşağıdakilerden hangisi veri analitiği türlerinden değildir?

☐ Tahminsel Analitik

☐ Betimleyici Analitik

☒ Sektörel Analitik

☐ Yönergeli 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 Analitik

☐ Teşhis/Tanı Analitiği

☐ Yönergeli Analitik

☒ Tahminsel Analitik

Soru 6:

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

☐ Normatif Analitik

☐ Teşhis/Tanı Analitiği

☒ Betimleyici Analitik

☐ Yönergeli Analitik

Soru 7:

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

☒ Tahminsel Analitik

☐ Betimleyici Analitik

☐ Teşhis/Tanı Analitiği

☐ Yönergeli Analitik

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 gerekir?

- ☒ Teşhis/Tanı Analitiği
- ☐ Betimleyici Analitik
- ☐ Normatif Analitik
- ☐ Tahminsel Analitik

# 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 olacaktır.
11
12 "a"+"a"
13
14 "a" " a"
15
16 "a"*3
17
18 "a"/3 #type error hatasi
19
20 "a "*5
21
```

- "a"+"a" → aa
- "a""a" → aa
- "a"\*3 → aaa
- "a"-"b" → TypeError alırsınız. Bu operatör sadece numeric ifadelerde kullanılır.
- "a"/3 → TypeError

## String Metodları

`len()`= içerisinde yazılan değişkenin uzunluğunu verir.

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

`upper()` & `lower()` =

```
17 #upper() & lower() fonksiyonlari
18
19 gel_yaz.upper() #stringi buyuk harflere ceviris.
20
21 gel_yaz.lower() #stringi kucuk harflere ceviris.
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","ı")
40
```

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

`gelecegi_yazanlar` → `gelecegi_yızınlr`

`strip()` = Karakter kırpma işlemleri

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



dir() =

```
49 # dir() icersine 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

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

Type Dönüşümleri

```
63 #TYPE DONUSUMLERI
64
65 toplama_bir=input() #input ile kullanıcıdan veri alırız.
66 toplama_iki=input() #kullanıcıdan aldığımız veri str tipindedir.
67
68 toplama_bir+toplama_iki # 10+20 --> '1020' çıktısı verir.
69 # bunu engellemek için type dönüşümü yapmalıyız.
70 int(toplama_bir)+int(toplama_iki) #tip donusumlerini bu sekilde yaparız.
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 Alıştırımlar – 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

☐ Yazdırmak

☐ 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 aşağıdaki hatalardan hangisini üretir?

☒ TypeError

☐ SyntaxError

☐ Hata üretmez

☐ 20

Soru 10:

Aşağıdakilerden hangisi bir karakter dizisinin eleman sayısını verir?

☐ print()

☐ lenght()

☒ len()

☐ replace()

### Python Programlama Alıştırımlar – 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", "1")
```

☒ '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 oluşturma
type(notlar) #--> list

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

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

### Liste Elemanlarına Ulaşma

```
#liste elamanlarına ulaşma

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

liste_genis

liste_genis[3][1] # liste_genis içersindeki notlar listesinin 1 indexli elemanı
# --> 80

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

### Liste İçi Type Sorgulama

```
#liste içi type sorgulama

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

tum_liste=[liste,list_genis]
```

**del liste** → liste'yi siler

### Liste elemanlarını değiştirme

```
# Liste elemanlarını değiştirme

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

liste2[1]="velinin babasi" # 1 index'li elemanı değiştirdik

liste2

liste2[1]="veli"
liste2[:3]="alinin_babasi","velinin_babasi","berkcanin_babasi" #3 elemanı değiştirdik
liste2
```

### Listeye eleman ekleme

```
#listeye eleman ekleme

liste2 + ["kemal"] # bu şekilde kaydetmez sadece görüntüler.

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

### Listeden eleman silme

**del** liste2[5] → 5 index'li elemanı 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 değerli elemanı siler

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

#### count metodu

```
#count  
  
liste=["ali","veli","ayca","veli","ali","ali"]  
  
liste.count("ali") # "ali" elemaninin listede kac kez yer aldigini gosterir.
```

→ 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 birlestirir.  
liste  
  
liste2.extend(["a",10]) #liste ile metodun icine yazilan elemanlari birlestirir.  
liste2
```

#### index metodu

```
#index  
  
liste.index("ali") #yazdigimiz elemanin kacinci index oldugunu 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 kucukten buyuge siralayip 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 Oluşturma  
  
t=(1,2,3,"eleman",[1,2,3,4])
```

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

Örneğin; t = ("eleman",)

### Eleman İşlemleri

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

t=(1,2,3,4)

t[0] → 1

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

### Dictionary (Sözlük)

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

### Dictionary Nedir?

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

Listelerde olduğu gibi index'leme yapılmaz.

### Dictionary Oluşturma

```
# Sozluk Oluşturma  
  
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 yapılardan olamaz. String ve sayılar sabit veri 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,
                 "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 adında tuple oluşturduk.

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

### 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 alır.

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 alır.

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 uretmez.
```

### 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 Sorgu İşlemleri

#### isdisjoint & issubset & issuperset

**isdisjoint** = Ayrık küme mi?

İki kümenin kesişiminin boş olup olmadığını sorgular.

Boş ise True değil ise False döndürür.

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

**issubset** = subset'i mi? Alt kümesi mi? sorgusunu 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 Özet

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

## Python Programlama Alıştırımlar – 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 gerekir?

☐ `liste + "D"`

☐ `liste["D"]`

☒ `liste.append("D")`

☐ `liste.insert("D")`

Soru 9:

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

Yukarıdaki listeye "D" ifadesini 0. indekse eklemek için aşağıdaki kodlardan hangisini yazmak gerekir?

☐ `liste[0] = "D"`

☐ `liste.insert("D")`

☐ `liste.append(0, "D")`

☒ `liste.insert(0, "D")`

Soru 10:

Verilen "sozluk" ismindeki veri yapısının içerisinde 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 Alıştırmalar – 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 Alıştırmalar – 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 olmadığının sorgulanması için hangi metod kullanılır?

☐ dir()

☒ isdisjoint()

☐ issubset()

☐ isuperset()

Soru 9:

Bir kümenin tüm elemanlarının başka bir küme içerisinde yer alıp almadığı 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 işleçlerdir.

### 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'ncü 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 çünkü str ifadeler sadece str ifadeler ile birleştirilebilir.
```

Bu fonksiyonu çalıştırınca aldığımız hata :

```
In [17]: kare_al(3) #hata aldık çünkü 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 çünkü 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 donusumu)
...:
...:

In [20]: kare_al(3) #bu kez hata almadan çalıştı.
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 satıra geçtik.
...:
...:

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 Sıralamasi

def carpma_yap(x,y):
    print(x*y)

carpma_yap(y=2, x=4) # Argumanların sırasını bilmiyorsam ama isimlerini biliyorsam
                     # Bu şekilde çalıştırabiliriz.
```

### Ne Zaman Fonksiyon Yazılır?

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

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

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

```
#Fonksiyonlar ne zaman yazılır?
def direk_hesap(isi, nem, sarj):
    print((isi+nem)/sarj)

direk_hesap(25,40,70)
```

```
In [14]: direk_hesap(25,40,70)
0.9285714285714286
```

### 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 cikti 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 kullanirsak 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 calistirirsak 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 icersindeki 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) arar, bulamazsa global alana çıkacak. Global alanda o değişkeni bulursa ona etki edecek (Orada da bulamazsa hata üretecek.). Aşağıdaki örnekte bu durumu gözlemleyebiliriz.

```
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 hayatta da kullandığımız gibi;

- Yağmur yağarsa şemsiye al
- Kar yağarsa zincir tak

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

### True – False Sorgulamaları (Boolean)

Doğru mu? sorusu sorar. `==` ile kullanırız.

```
In [3]: sinir = 5000 #sinir degiskenine 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 satıra 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 saglanmadiysa elif'e bakilir.
...:     print("Geliriniz sinirda.")
...: else: #hic bir kosul saglanmiyorsa else calisir.
...:     print("Tebrikler. Geliriniz sinirdan yukarida.")
...:
...:
Geliriniz sinirda.

In [24]: if gelir3 < sinir:
...:     print("Geliriniz sinirdan kucuk!!")
...: elif gelir3 == sinir: #if kosulu saglanmadiysa elif'e bakilir.
...:     print("Geliriniz sinirda.")
...: else: #hic bir kosul saglanmiyorsa else calisir.
...:     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 adi nedir?\n ") #kullanıcıdan magaza_adi aldık
gelir = int(input("Gelirinizi giriniz: ")) #kullanıcıdan aldığımız geliri int'e cevirdik.

if gelir > sinir:
    print("Tebrikler "+magaza_adi+ " Geliriniz sinirdan yuksek :)")
elif gelir == sinir:
    print(magaza_adi+" Geliriniz sinirda.")
else:
    print("Uyari! "+magaza_adi + " Cok dusuk gelir: "+str(gelir))
```

Program çıktıları:

Magaza adi nedir? A Mağazası	Magaza adi nedir? B Mağazası
Gelirinizi giriniz: 35000 Uyari! A Mağazası Cok dusuk gelir: 35000	Gelirinizi giriniz: 50000 B Mağazası Geliriniz sinirda.
Magaza adi nedir? C Mağazası	
Gelirinizi giriniz: 65000 Tebrikler C Mağazası Geliriniz sinirdan yuksek :)	

### 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 kodlari yaziniz.

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

yeni_maas(1000) #fonksiyonun calismasina ornek.

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

Bu gibi durumlarda **break** ve **continue** ifadeleri kullanılır.

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

```
#break & continue

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

maaslar.sort() #Karisik yazilmis listeyi kucukten buyuge siraladik.
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 işlemi devam ettir.
...:     sayi += 1 #sayiyi 1 arttır 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 üretecektir. 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 gerekir?

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

☐ Fonksiyon argümentsiz 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 akışını kesmek için (Örneğin bir şart yakalandığında çalışmayı durdur demek gibi)

☐ Bir şart yakalandığında ekrana yazdırmak 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 yazdırmak için

☐ Yakalanan şarta 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 tanımlama
    print("Bu bir class'dir.")
```

#### Class Özellikleri

```
#Siniflarin Ozellikleri
class VeriBilimci(): #Class tanımlama
    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 #Ozelligin degeri degisti.
```

### Class Örneklendirmesi (instantiniation)

Sınıfın özelliklerini barındıran alt kümeler oluşturma işlemine sınıf örneklendirmesi denir.

```
#Sinif Orneklendirmesi (instantiation)

ali = VeriBilimci() #VeriBilimci sinifinin ozelliklerini tasiyan bir birim olustu.
                    #Yani ornekleme yapmis 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 değişebilen özelliklerden oluşabildiği bilgisini vermek. Yani her bir ayrı örneklendirme için özellik tutma bilgisini sağlıyor.

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

```
def __init__(self):
```

```
    self.bildigi_diller = "
```

```
    self.bolum = " → fonksiyonunu kullanacağız. Buradaki self temsilci anlamındadır. Her bir örnekleme 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 sinif tanımladık
    bildigi_diller = ["R","Python"] #Tum class için özellik ataması.
    bolum = ''
    def __init__(self): #Orneklere 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 bildigi dillere ekleme yaptık.
ali.bildigi_diller #Bu kez python var.

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

VeriBilimci.bildigi_diller #Classin 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 yapsın.

Örnekler üzerinde çalışan fonksiyonlar yazmak istiyoruz.

```
# Örnek Metodları

class VeriBilimci(): #Bir class tanımladık.
    çalışanlar = [] # çalışanlar adında bir nesne
    def __init__(self): #örneklerin özellikleri
        self.bildigi_diller = [] #örneklerin özellikleri
        self.bolum = '' #örneklerin özellikleri
    def dil_ekle(self, yeni_dil): #örneklere etki edecek bir fonksiyon yazdık
        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)

Başka 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 Yapıları (inheritance)

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

class DataScience(Employees): #Employees'den miras alıyor.
    def __init__(self,Programming):#Özellikleri fonksiyonel. Sabit değil.
        self.Programming = Programming

class Marketing(Employees): #Employees'den miras alıyor.
    def __init__(self,StoryTelling):#Özellikleri fonksiyonel. Sabit değil.
        self.StoryTelling = StoryTelling

veribilimci1 = DataScience() #Parametreyi bos birakamayiz. 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 yazabiliriz FP(Fonksiyonel Programlama) özellikleri ile de yazabiliriz.

Fonksiyonlar dilin baştaçdır. (Birinci sınıf nesnelerdir.)

Yan etkisiz fonksiyonlar. (stateless(durumsuz), girdi-çıkı →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 degiskenine 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: Ölümcül Yan Etkiler

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

In [28]: #OOP

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', 'dhhjfhfg']

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- İsimsiz 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 fonksiyondur. 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 ulaşsın.

## Vektörel Operasyonlar (Vectorel Operations)

### OOP ile iki listeyi çarpmak

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

In [14]: a = [1,2,3,4] #amacımız bu listeler icersindeki her bir elemani birbiriyle carpmak
...: 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 islemini 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 çalıştırma 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.

Çift 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 bolumunden kalanı 0'a esit olanları listele.
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]

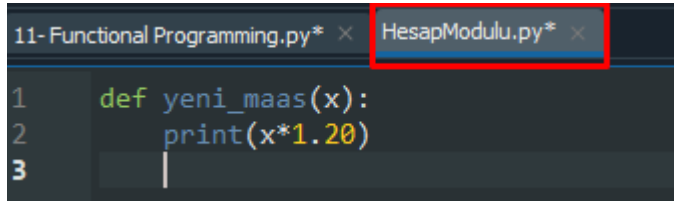
In [20]: reduce(lambda a,b:a+b , liste) #liste elemanlarını toplar.
Out[20]: 55
```

### 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.

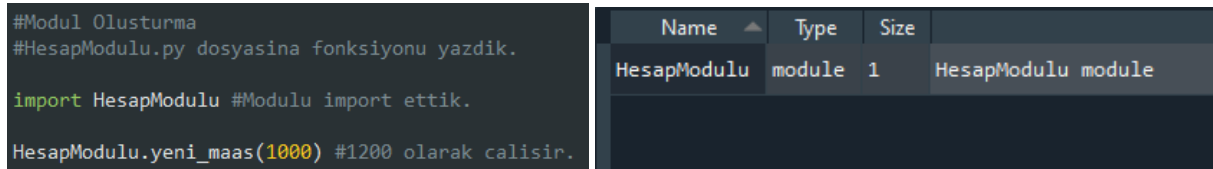
Yeni bir .py dosyası açalım ve ismi HesapModulu.py olsun.



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

Modülün içine fonksiyonu yazıp kaydettik. Modülümüz kullanıma hazır.

Başka bir .py dosyasından modüle erişmek:

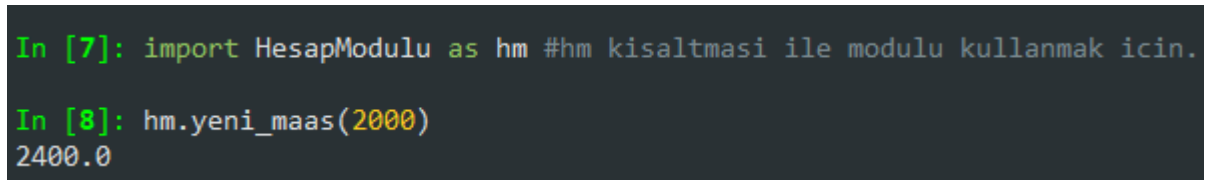


```
#Modul Olusturma
#HesapModulu.py dosyasina fonksiyonu yazdik.

import HesapModulu #Modulu import ettik.

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

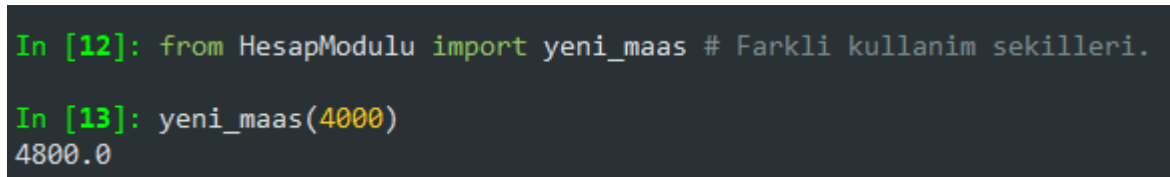
Name	Type	Size	
HesapModulu	module	1	HesapModulu module



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

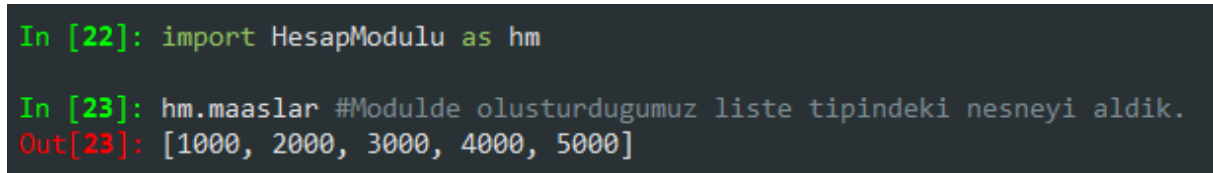
In [8]: hm.yeni_maas(2000)
2400.0
```

Daha da kısa kullanımı için:



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

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



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

In [23]: hm.maaslar #Modulde olusturdugumuz liste tipindeki nesneyi aldik.
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 karstilastiginda 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 özellikleri 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 parçacığına göre aşağıdakilerden hangisi bir sınıf örneklendirmesidir?

```
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

☐ İsimsiz fonksiyonlar kullanılabilir

☒ Yan etkili fonksiyonlar vardır

☐ Vektörel işlemlere imkan sağlar

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 parçasının çıktısı aşağıdakilerden hangisidir?

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

☐ 7

☒ [2, 7, 4]

☐ 2

☐ 4

Soru 9:

Verilen kod parçasının 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
```

Çıktı:

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", "salı", "carsamba", "persembe", "cuma"]))
```

☒ ['pazartesi']

☐ ['salı']

☐ ['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

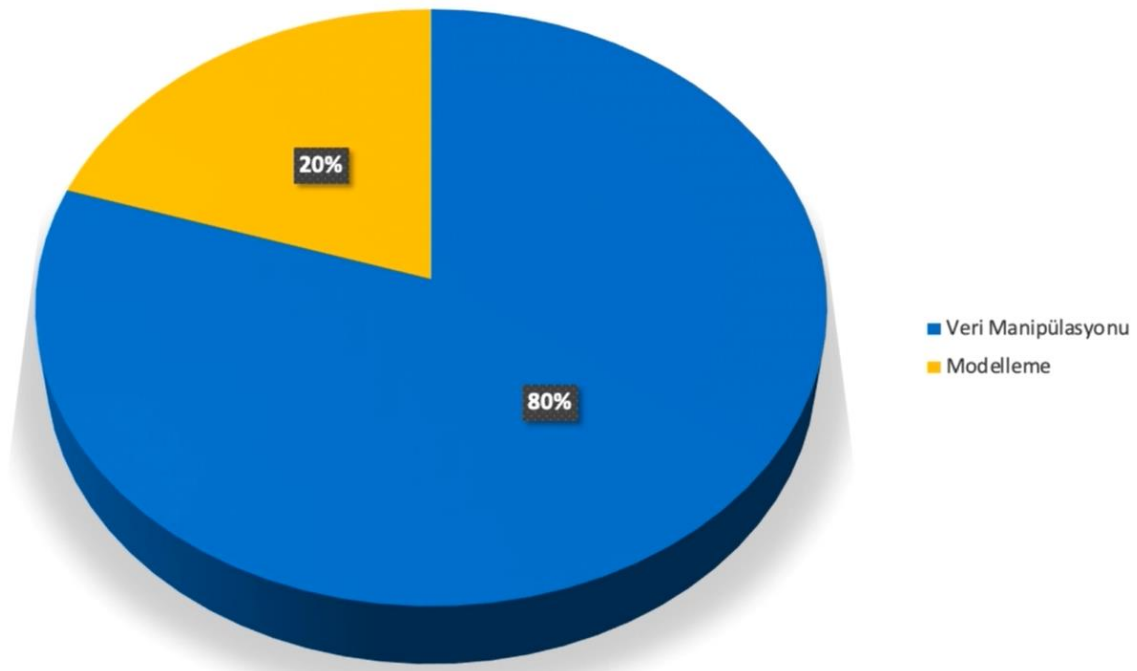
☐ VeriBilimiOkulu

☐ VeBiOk

☐ Veri Bilimi

## Python ile Veri Manipölasyonu: NumPy & Pandas

### Veri Bilimi Süreci



## NumPy (Numerical Python)

### NumPy Giriş

NumPy Python'ın bazı numerik işlemlerde yetersiz kaldığı noktalarda ihtiyaçlarımızı gidermek için ortaya çıkmış 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 hayata 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 tıpkı 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 oluşturma
```

```
[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'lik 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'lik 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 dogrusal 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 sayi olustur.

[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)) #ortalamasi=10, standart sapmasi=4 olan 3x4'luk 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 araliginde rastgele int degerlerden 3x3'luk 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 arasindaki degerlerden olusan 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.

Fonksiyonlarımızın 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 taşır.

[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 birleştirme

[3]: 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 birleştirme

[5]: 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 oluşturma

[7]: np.concatenate([a,a]) #standart olarak satir bazinda birlestirme yapar.

[7]: 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.

[8]: 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. indise kadar ayır, sonra 5. indise 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. indise 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. indise 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 bozulmadi.
```

```
[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))#ortalamasi 20, standart sapmasi 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 koordinatındaki 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 anca ondalik kismini keserek ekleyecek.  
m #Daha onceden 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 başlayarak 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 eriştik fakat burada şöyle bir durum söz konusu;

Örneğin bir array'in alt kümesine eriştikten sonra bunu isimlendirip kaydettiğimizi düşünelim.

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

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

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

```
[30]: #Bir ornek ile yukaridaki durumu daha iyi anlayalım:
import numpy as np
a = np.random.randint(10, size=(5,5)) #5x5 matris olusturduk.
a
```

```
[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 elemanlarinda 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 ayni anda kullandik.  
  
[13]: array([1, 2])
```

### Slice ile Fancy kullanımı

```
[14]: #slice ile fancy  
  
[15]: m[0:, [1,2]] #basit index ile fancy'i ayni anda kullandik.  
  
[15]: array([[1, 2],  
           [4, 5],  
           [7, 8]])  
  
[ ]: #Buradaki islemlerin teknik olarak farkli oldugunu anlamamiz gerekir.
```