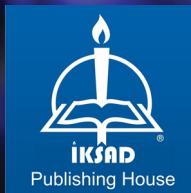


UYGULAMALARLA TEMEL SEVİYE

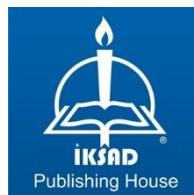


Öğr. Gör. Enes AÇIKGÖZÖĞLU & Öğr. Gör. Ziya DİRLİK



UYGULAMALARLA TEMEL SEVİYE PYTHON

Öğr. Gör. Enes AÇIKGÖZOGLU & Öğr. Gör. Ziya DİRLİK



Copyright © 2021 by iksad publishing house
All rights reserved. No part of this publication may be reproduced,
distributed or transmitted in any form or by
any means, including photocopying, recording or other electronic or
mechanical methods, without the prior written permission of the publisher,
except in the case of
brief quotations embodied in critical reviews and certain other
noncommercial uses permitted by copyright law. Institution of Economic
Development and Social
Researches Publications®
(The Licence Number of Publicator: 2014/31220)
TURKEY TR: +90 342 606 06 75
USA: +1 631 685 0 853
E mail: iksadyyayinevi@gmail.com
www.iksadyyayinevi.com

It is responsibility of the author to abide by the publishing ethics rules.
Iksad Publications – 2021©

ISBN: 978-625-8007-66-4
Cover Design: İbrahim KAYA
November / 2021
Ankara / Turkey
Size = 14,8 x 21 cm

ÖNSÖZ

İçinde yaşadığımız teknolojik döneminin son dönemde teknoloji literatürüne kattığı en büyük kavram şüphesiz endüstri 4.0 ve paydaşlarıdır. Endüstri 4.0'ın paydaşlarından olan yapay zekâ, nesnelerin interneti, robotik, simülasyon vb. teknolojiler yazılım desteği ile gelişen ve geliştirilen alanlardır. 1990'lı yıllarda ortaya çıkan Python programlama dili süreç içerisinde geliştiriciler tarafından yeni kabiliyetler kazandırılarak günümüze kadar gelmiştir. Nesne yönelimli, yorumlamalı ve modüler yapıda olan bu programlama dili birçok endüstri 4.0 paydaşının uygulamalarında yer almaktadır.

Bu kitabı yazmamızdaki motivasyon kaynağımız Python programlama diline yönelik uygulama tabanlı materyal eksikliği olarak gösterilebilir. Bu kitap Python öğrenmek isteyen yazılım gönüllüleri için temel seviyede programlama diline giriş ve uygulama örnekleri sunmaktadır.

Bu kitap çalışması esnasında destek ve anlayışını eksik etmeyen kıymetli eşim Senem AÇIKGÖZÖĞLU'na ve aynı zamanda bu çalışma sürecinde varlıklarıyla motivasyon kaynağımız olan kıymetli evlatlarım Ömer Asaf AÇIKGÖZÖĞLU ve Asel Hafsa AÇIKGÖZÖĞLU'na armağan edilmiştir.

İÇİNDEKİLER

ÖNSÖZ.....	i
İÇİNDEKİLER	iii
ŞEKİLLER DİZİNİ	vi
1. NEDEN PYTHON?	7
1.1. Python'un Kullanıcıya Sağladığı Avantajlar.....	7
2. PYTHON KURULUMU	9
2.1. Windows Kurulumu.....	9
2.2. Ubuntu Kurulumu	11
3. Python'a Giriş.....	12
3.1. Anaconda ve Spyder IDE kurulumu	12
3.1.1. Anaconda Nedir?	12
3.1.2. Anaconda Windows Kurulumu	13
3.1.3. Anaconda Linux Kurulumu.....	13
3.2. Python Spyder IDE nedir?.....	14
3.2.1. Spyder'in Özellikleri	14
3.2.2. Spyder IDE'de Python Projesi Oluşturma	15
4. İlk Python Uygulaması “Merhaba Python”	18
5. Temel Veri Türleri ve Değişkenler	19
5.1. Değişken Tanımlama ve Değer Atama	19
5.2. Python Veri Türleri	20
5.2.1. Sayısal Türler.....	21
5.2.2. Karakter Dizileri	23
5.2.3. Koşul İfadeleri	24
5.2.4. Döngüler.....	30

5.3. Listeler ve Demetler	41
5.3.1. Liste Tanımlamak	42
5.3.2. Liste Metotları	44
5.3.3. Listeye Öğe Eklemek	49
5.3.4. Listeden Öğe Çıkarmak	52
5.3.5. Listeleri Silmek.....	54
5.3.6. Listeleri Kopyalamak.....	54
5.3.7. Liste Üreteçleri (List Comprehensions)	55
5.3.8. Demetler (tuples)	57
5.3.9. Demet Tanımlamak.....	57
5.3.10.Tek Öğeli bir Demet Tanımlamak	58
5.3.11.Demetlerin Öğelerine Erişmek	59
5.3.12.Demetlerle Listelerin Birbirinden Farkı.....	60
5.3.13.Demetlerin Kullanım Alanı	62
6. Python Kütüphaneleri.....	62
6.1. Serial Kütüphanesi	64
6.2. Numpy Kütüphanesi	67
6.3. Turtle Kütüphanesi.....	78
7. Python Örnek Kodları.....	83
7.1. Ekrana “Merhaba Dünya” yazdırın Python örneği	83
7.2. Kullanıcının İşmini Alarak Ekrana “Merhaba (Kullanıcı İsmi)” Yazdırın Python Örneği	83
7.3. Girilen 2 Sayıyı Toplayan Python Örneği.....	85
7.4. Girilen 2 Sayının Ortalamasını Bulan Python Örneği	86
7.5. Girilen Sayının Asal Sayı Mı Değil Mi Olduğunu Bulan Python Örneği	86

7.6. Girilen Vize ve Final Notu Ortalaması Hesaplayan Python Örneği.....	88
7.7. Girilen 3 Sınav Notunun Ortalamasını Bulan Python Örneği	88
7.8. Kullanıcı Tarafından Girilmesi İstenen Sayının Tek ya da Çift Olduğunu Hesaplayan Örnek Program.....	89
7.9. Girilen İki Sayıya ve Operatöre (+,-,*,/) Göre Toplama, Çıkarma, Çarpma ya da Bölme İşlemlerini Yapan; Bu Operatörler Dışında Bir Değer Girildiğinde “yanlış işlem girdiniz.” Uyarısı Veren Python Örneği.	92
7.10. Daha Önceden Belirlenmiş Bir Liste İçerisinden Rastgele Bir Sayı Seçen ve Ekranda Gösteren Python Örneği.	93
7.11. Python Girilen Sayının Faktoriyelini Hesaplayan Python Örneği.	93
7.12. Girilen Sayıya Kadar Olan Çift Sayıları Listeleyen Python Örneği	94
7.13. 3×3 Bir Matrisin Eleman İndislerini Yazdırın Python Örneği.....	95
7.14. Kenarları Girilen Dikdörtgenin Alanı ve Çevresini Bulan Python Örneği	96
7.15. Maaşı ve Zam Oranı Girilen İşçinin Zamlı Maaşını Hesaplayarak Ekranda Gösteren Python Örneği.....	97
7.16. Sayı Tahmin Oyunu Python Örneği.....	98
7.17. Girilen metin içerisinde, girilen bir karakterin olup olmadığını kontrol eden Python programı kodları.....	103
7.18. Python Dizindeki İstenen Dosyaları Listeleme.....	104
7.19. Klavyeden Girilen Yıl ve O Yılın Ayının Takvimini Ekranda Gösteren Python Kodu.....	105
7.20. Serial Port üzerinden Veri göndererek Led Yakma/Söndürme Uygulaması	106
7.21. Serial Port üzerinden Veri Okuma Uygulaması.....	108
KAYNAKÇA.....	104

ŞEKİLLER DİZİNİ

Resim 1: Windows İşletim Sistemi Sürümü	9
Resim 2: Python Yükleme Ekranı	10
Resim 3: Python Yükleme Başarılı Ekranı	11
Resim 4: Anaconda Açılmış Ekranı	15
Resim 5: Python Karşılama Ekranı	16
Resim 6: Python Proje Dosyalarına Ulaşma Ekranı	16
Resim 7: Python Dosya Kaydetme Ekranı	17
Resim 8: Python Ekran Bölümleri	17
Resim 9: Pyhon'da Yazılan Uygulamnın Çalıştırılması	18
Resim 10: Koşul İşleçleri (Özgül, 2015)	25
Resim 11: While Döngüsü Adımları	33
Resim 12: Serial Port İşlevleri	66
Resim 13: Turtle kütüphanesi fonksiyonları	78

1. NEDEN PYTHON?

Python Windows, Linux, Unix, Mac, Symbian, Amiga vb. farklı platformlarca desteklenen bir programlama dilidir. Kullanım alanlarının çeşitliliği ve esnekliği sayesinde tercih sebebi olmaktadır. Kısaca Python programlama dilinin yazılım sektöründeki kullanım alanları: web geliştirme, bilimsel ve numerik hesaplama, yapay zekâ uygulamaları, IoT uygulamaları (nesnelerin interneti), veri madenciliği (Data Mining), masaüstü uygulamaları vb. şeklinde özetlenebilir. Belirtmiş olduğumuz alanların dışında da birçok alanda kullanılmaktadır (Kuhlman, 2009; beyaz.net. 10.09.2021).

NASA, Google, Yahoo, Wikipedia, Reddit, CERN, BitTorrent, YouTube vb. büyük şirketlerin Python dilini tercih etmeleri, Python'ın sektördeki popülerliğini artırmıştır.

1.1. Python'un Kullanıcıya Sağladığı Avantajlar

Python programlama dili yeni başlayanlar veya Python'da uzmanlaşanlar için önemli avantajlara sahiptir.

Kullanıcıya sağladığı avantajlardan bazıları:

- Python, geniş kütüphane portföyüne sahiptir. Bu sayede oldukça popüler duruma gelmiştir. Kullanım alanlarının oldukça fazla olması sayesinde yaygın bir kullanıcı kitlesine ulaşmıştır.
- Açık kaynaklı, ücretsiz ve sade olması ile diğer programlara dillerine göre kullanımını çok basittir. Bu sayede hızlıca öğrenilebilir.
- Python kodları derleme işlemi olmadan çalıştırılabildiği için hızlı bir şekilde program geliştirmeye imkân vermektedir.
- C veya C++ gibi dillere göre büyük yazılımların hızlı bir şekilde uygulama geliştirilmesi gerekiğinde daha kullanışlıdır.
- Python'ın birçok internet protokolünü destekleyen standart kütüphaneleri vardır.
- Python'da socket programlama kolaydır.
- Python'ın geniş bir kullanıcı ağına sahip olması sayesinde, örnek uygulamalara ve yardım dokümanlarına erişim kolay bir şekilde sağlanabilmektedir (Ceder, 2018; beyaz.net. 10.09.2021).

2. PYTHON KURULUMU

2.1. Windows Kurulumu

Öncelikle Windows başlat menüsünden ayarlar seçeneğine tıklanıp ayarlar menüsünde Sistem seçeneğine tıklanır. Açılan pencereden Hakkında sayfasına gidilir ve Windows işletim sistemimizin 32-bit ya da 64-bit versiyonu mu çalıştığı kontrol edilir.

Cihaz özellikleri

Cihaz adı

İşlemci Intel(R) Xeon(R) E-2136 CPU @ 3.30GHz 3.31 GHz

Takılı RAM 32,0 GB (kullanılabilir: 31,9 GB)

Cihaz Kimliği B3798

Ürün Kimliği 00330

Sistem türü 64 bit işletim sistemi, x64 tabanlı işlemci

Kalem ve dokunma Bu görüntü biriminde kalem girdisi veya dokunarak giriş yok

Resim 1: Windows İşletim Sistemi Sürümü

Windows için Python'ı indirmek için resmi siteyi ziyaret edebilirsiniz: <https://www.python.org/downloads/windows/>. "Son Python Sürümü - Python x.x.x" bağlantısına tıklayın.

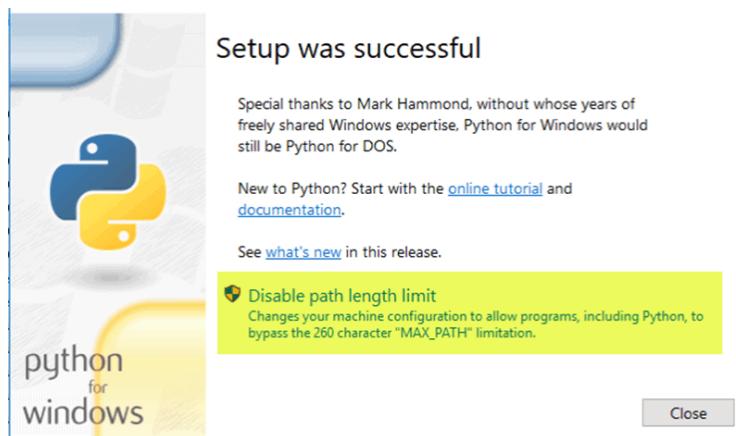
İşletim sisteminizin sürümüne (32-bit, 64-bit) uygun yükleyiciyi indirin (djangogirls.org. 05.09.2021).

İndirdikten sonra Python Installer'ı çalıştırın. **Install launcher for all users** ve **Add Python 3.10 to PATH** onay kutularını seçtiğinizden emin olun. **Install Now** seçeneğine tıklayınız.



Resim 2: Python Yükleme Ekranı

Sonraki pencerede, “**Disable path length limit**” seçeneğini belirlemeniz istenecektir. Bu seçeneğin seçilmesi Python'un 260 karakterlik “MAX_PATH” sınırını atlamasını sağlar. Etkili bir şekilde, Python'un uzun yol adlarını kullanmasını sağlayacaktır (djangogirls.org. 05.09.2021).



Resim 3: Python Yükleme Başarılı Ekranı

Python'un Windows'a Yüklediğini Doğrulamak için komut satırına “python” komutunu yazalarak Python sürümünüzü kontrol edebilirsiniz.

2.2. Ubuntu Kurulumu

Python ile program geliştirebilmeniz için bilgisayarınızda ilgili Python sürümünün yüklü olması gereklidir. Pek çok Linux dağıtıımı içerisinde Python yüklü olarak gelir. Ubuntu içerisinde Python 2 ve 3 yüklü olarak gelir. Ayrıca Mac işletim sisteminde de yüklüdür. Ancak Windows kullanıyorsanız Python kurulumunu kendiniz yapmanız gerekmektedir.

3. Python'a Giriş

Python, dünyanın en popüler ve en hızlı büyüyen programlama dilinden biridir (2019 ' de GitHub göre üçüncü kullanım). Web programlama ve veri analizi de dahil olmak üzere her türlü görev için kullanılan Python, makine öğrenimine yönelik öğrenme için en uygun dil olarak popülerlik kazandı. Bu popülerlik, python geliştiricilerine olan talebin arttığını ve python kullanarak programlama işlerinin kazançlı olabileceği işaret ediyor (Johansson vd., 2019).

3.1. Anaconda ve Spyder IDE kurulumu

3.1.1. Anaconda Nedir?

Anaconda, veri bilimi ve benzeri bilimsel uygulamalar için python kullanmak isteyenlere hazırlanmış tümleşik bir python dağıtımıdır. Veri bilimi, yapay zekâ vb konularında sıkça kullanılan kütüphanelerin yanı sıra jupiter notebook ve spyder gibi araçları da barındırır. Programı anaconda.org adresinden kendi işletim sisteminize uygun versiyonu indirerek kurabilirsiniz. Anacondayı kurduğunuzda sisteminizde python, jupiter notebook ve spyder da kurulmuş olacak (medium.com. 05.09.2021).

Anaconda tümleşik dağıtımını indirmek için [anaconda.org](https://www.anaconda.org) sitesine giderek ve bilgisayarınız için uygun olan (Windows & MacOS & Linux) kurulum dosyasını indirin.

3.1.2. Anaconda Windows Kurulumu

Ardından indirdiğiniz kurulum dosyasını açın ve basit olan kurulumu (Next → Next → Setup) şeklinde adımları izleyerek kurulumu tamamlayabiliriz (medium.com. 05.09.2021).

3.1.3. Anaconda Linux Kurulumu

1. <https://www.anaconda.com/products/individual> sitesinden uygun anaconda sürümü seçilerek indirilir. Not: Bilgisayarınızda Python sürümünün yüklü olması gerekmektedir.
2. İndirilen dizinde terminali açınız ve bash ile birlikte indirdiğiniz dosyanın adını yazınız. (bash Anaconda3-2021.5-Linux-x86_64.sh) gibi.
3. Lisans bilgilerinizi giriniz ve yazının sonuna gelin ve -yes diyin. Gelen bütün sorguları onaylayabilirsiniz.
4. Son olarak anaconda'nın yolunu. bashrc dosyasına eklemesi için onay veriyoruz. Farklı bir terminalde “conda

update –all” diyerek güncelleme işlemini gerçekleştiriyoruz.

3.2. Python Spyder IDE nedir?

Spyder, açık kaynaklı bir IDE’dir. Python Spyder IDE tamamen Python programlama dilinde yazılmıştır.

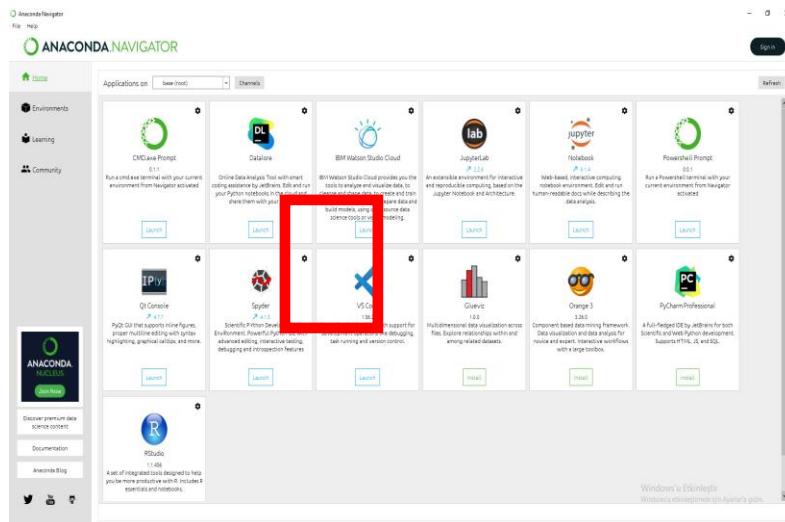
3.2.1. Spyder'in Özellikleri

Spyder’ın dikkat çekici özelliklerinden bazıları şunlardır:

- Özelleştirilebilir söz dizimi vurgulama.
- Kesme noktalarının kullanılabilirliği. (hata ayıklama ve koşullu kesme noktaları)
- Satır, dosya, hücre vb. çalışmalarınıza izin veren etkileşimli yürütme.
- Değişkenleri otomatik olarak temizleyebilir.
- İf, while, vb sonrasında otomatik iki nokta üst üste ekleme
- Yardım, dosya gezğini, dosya bulma vb. özellikleri sağlar (medium.com. 05.09.2021).

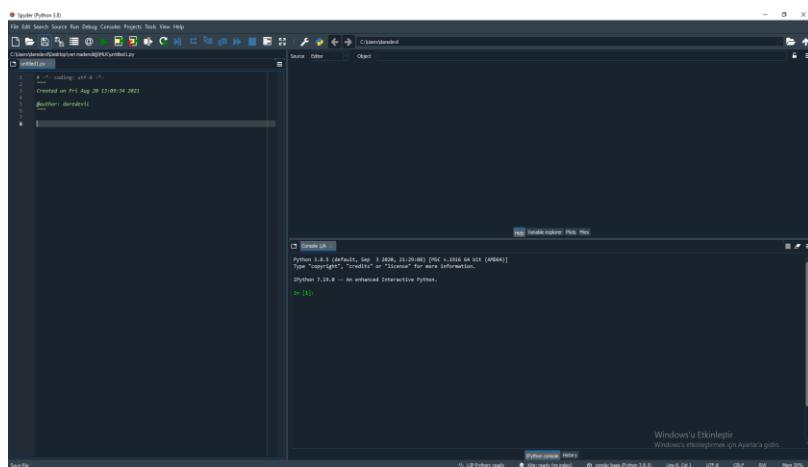
3.2.2. Spyder IDE’de Python Projesi Oluşturma

Python kodlaması yaparken kullanabileceğimiz editörlerden birisi Spyder’dir. Bunun için daha önceden kurulumunu anlattığımız Anaconda’nın bilgisayarınızda kurulu olması gerekmektedir. Anaconda Navigator’u açıyoruz.



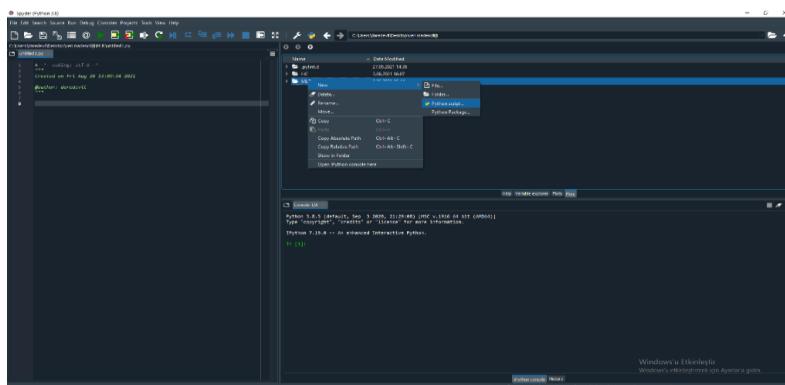
Resim 4: Anaconda Açılmış Ekranı

Spyder’ı şekilde gösterildiği gibi Launch (çalıştır) diyerek açıyoruz. Bizi aşağıdaki gibi bir ekran karşılayacaktır.

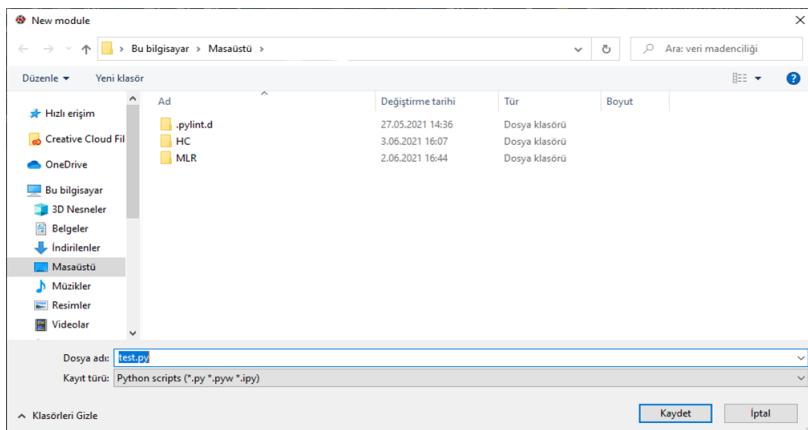


Resim 5: Python Karşılama Ekranı

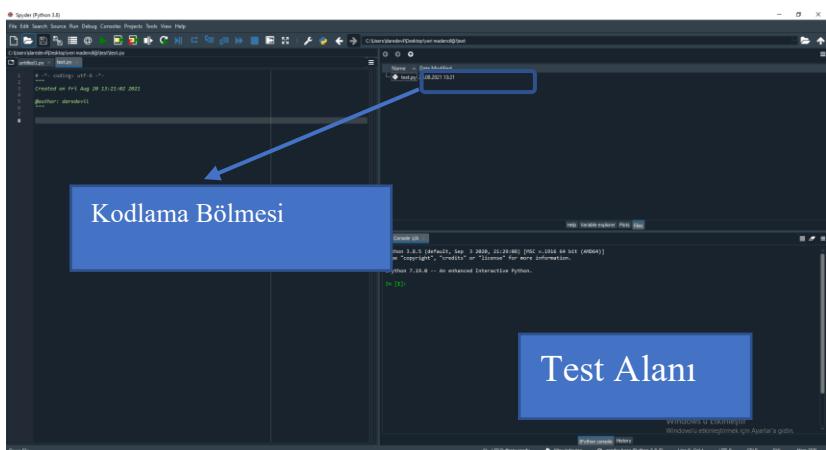
Sağ üst köşedeki bölümde dosya yönetimi bölümü bulunur. Buradan istediğimiz proje dosyasına ulaşıp var olan dosyaları açabilir yeni dosyalar oluşturabiliriz.



Resim 6: Python Proje Dosyalarına Ulaşma Ekranı



Resim 7: Python Dosya Kaydetme Ekranı

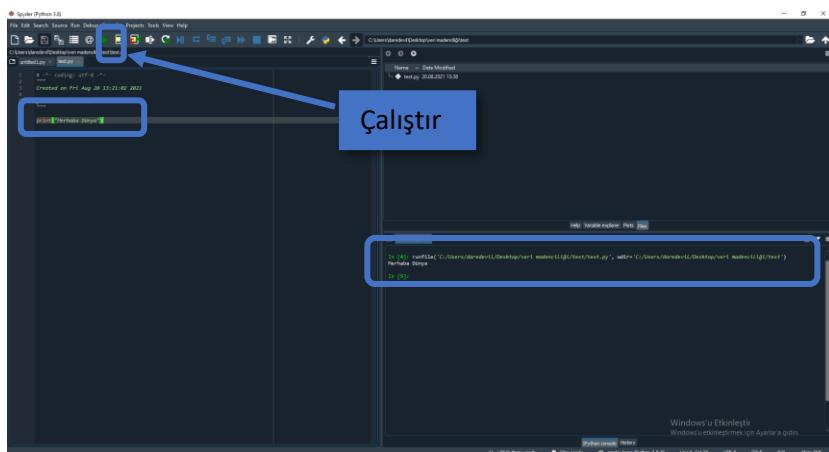


Resim 8: Python Ekran Bölümleri

Spyder genel olarak kullanışlı bir editördür. Şimdi Python ile ilk uygulamamızı geliştirelim.

4. İlk Python Uygulaması “Merhaba Python”

Tüm programlama dillerinde olduğu gibi Python'da da klasik olarak Merhaba dünya uygulaması ile işe başlıyoruz. En kısa kod Python ile yazıılır diyebiliriz. Hatta bunu aşağıdaki örnekle de pekiştirebiliriz.



Resim 9: Python'da Yazılan Uygulamnın Çalıştırılması

Not: Python'da String fonksiyonlarını çift tırnak ile kullanmak zorunlu değildir. Tek tırnak veya üç tırnakla da kullanabiliriz. Üç tırnak için, 3 adet tek tırnak ya da 3 adet çift tırnak kullanılabilir.

Örnek: Ekrana Yazdırma

```
print("'''Merhaba Dünya''')  
print('' ''Merhaba Dünya'' '')
```

Ekran Çıktısı:

```
Merhaba Dünya  
Merhaba Dünya
```

5. Temel Veri Türleri ve Değişkenler

Değişkenler, değerleri saklamak için ayrılmış bellek konumlarıdır. Bir değişken oluşturduğunuzda hafızada yer ayırmamanız anlamına gelir. Bir değişken değer türüne bağlı olarak belleğe yerleştirilir ve bellekten okunurken değişkenin değer türüne bağlı okuma yapılır. Bu nedenle, değişkenlere farklı veri türleri atayarak, bu değişkenlere tamsayı, ondalık veya karakter kaydedebilirsiniz (erdincuzun.com. 11.09.2021).

5.1. Değişken Tanımlama ve Değer Atama

Python değişkenleri, bellek alanını ayırmak için açık bildirime gerek duymaz. Değişken tanımlarında diğer diller de olduğu gibi tip belirtmeye gerek yoktur. Başka bir deyişle C, C# ve Java

dillerinde olduğu gibi değişkenin tipini önceden tanımlamaya gerek yoktur. Python atama yapıldığı anda değişkenin tipini otomatik olarak belirler. Bu açıdan Javascript ve PHP dillerindeki tanımlamaya benzerdir. Değer tipi belirleme programlama dili yorumlaması sırasında atanan ilk değere göre belirlenir. Başka bir deyişle tip belirleme atama işlemi olduğunda otomatik olarak belirlenir. Değer atamak için çoğu programlama dilindeki gibi eşittir (=) işaretini kullanılır (Severance, 2009; erdincuzun.com. 11.09.2021).

Örnek: Değer Atama

```
ay=12
kilometre=3000.0
sehir="ISPARTA"
print(ay)
print(kilometre)
print(sehir)
```

Ekran Çıktısı:



```
12
3000.0
ISPARTA
```

5.2. Python Veri Türleri

Bellekte saklanan veriler birçok türde olabilir. Örneğin, bir kişinin yaşı sayısal bir değer olarak saklanır ve onun adresi

karakter katarı olarak saklanır. Python'un beş standart veri türü vardır (erdincuzun.com. 11.09.2021).

5.2.1. Sayısal Türler

Python'da üç tür numerik yani sayısal veri tipi vardır; int, float, complex.

- int(Tamsayı)

Int(tamsayı) veri tipi, sınırsız uzunlukta pozitif veya negatif tam sayıları ifade etmektedir.

```
a=12      #int tanımlanır  
b=65.2    #float tanımlanır  
c=43j;    #complex tanımlanır
```

- Float (Ondalıklı Sayılar)

Float sayı veri tipi, noktadan sonra bir veya daha fazla ondalık sayı içeren pozitif veya negatif sayıları ifade etmektedir. Float, ayrıca içinde e veya E barındıran tüm bilimsel sayıları tanımlarken kullanılır.

```
a=0.10
b=200.0
c=34.3434
d=123E3
e=23423.4e34
```

- Complex (Karmaşık Sayılar)

Karmaşık sayılar, önceki sayısal veri tiplerinden de büyük sayılarından oluşurlar. O kadar büyüktürler ki iki parçadan oluşurlar; reel (gerçek) ve imajiner (sanal) isimli iki kısımdan oluşur ve complex veri tipi olarak ifade edilir (msoguz.com. 12.09.2021).

Normalde matematikte imajiner birimi "i" harfi ile gösterilir. i ise $\sqrt{-1}$ anlamındadır. Python'da ise i birimi yerine "j" harfi kullanılarak gösterilir. Yani $(3 + 5i)$ karmaşık sayı ifadesini Python'da yazmak isterseniz: $(3 + 5j)$ olarak yazmanız gerekiyor (msoguz.com. 12.09.2021).

```
karmasikSayi1=5+3j
karmasikSayi2=2+4j
sonuc=karmasikSayi1+karmasikSayi2

print("İşlem Sonucu= ",sonuc)
print("Veri Tipi= ",type(sonuc))
print("Gerçek Sayı= ",sonuc.real)
print("Sanal Sayı= ",sonuc.imag)
```

Ekran Çıktısı

```
In [33]: runfile('C:/Users/daredevil/...  
İşlem Sonucu= (7+7j)  
Veri Tipi= <class 'complex'>  
Gerçek Sayı= 7.0  
Sanal Sayı= 7.0
```

Karmaşık sayıları da normal sayılar gibi 4 işlemde (Toplama, Çıkarma, Çarpma ve Bölme) kullanabilirsiniz. Ayrıca karmaşık veri tipinin gerçek ve sanal sayılarını birbirinden ayırmak için 2 adet fonksiyonu vardır. Bunlar; real ve imag fonksiyonlardır. Real, karmaşık sayının gerçek kısmını ele alır. Imag ise karmaşık sayının sanal kısmını ele alır (Payne,2010; msoguz.com. 12.09.2021).

5.2.2. Karakter Dizileri

Karakter dizileri karakterlerden oluşan yapılardır. Örneğin “Isparta”, “güler diyarı” karakter dizileri tek tırnak (‘ ’), çift tırnak (“ ”) veya üç tırnak (“”” ”””) arasında yazılmaktadır.

```
Created on Thu Aug 26 16:53:55 2021

"""
deger1="Isparta"
deger2="Güller diyarı"
print (deger2, " ", deger1)
```

Ekran Çıktısı

```
In [7]: runfile('C:/Users/daredevil/untitled0.py')
Güller diyarı    Isparta
```

5.2.3. Koşul İfadeleri

Python koşul ifadelerini bir örnekle açıklamak gerekirse; bir hesabımıza giriş yapmaya çalıştığımız zaman bizden, kullanıcı adı ve şifre bilgilerimizi istemektedir. Eğer kullanıcı adı ve şifreyi doğru şekilde yazdığımızda giriş yapmamızı, yanlış yazdığımızda ise hesabımıza giriş yapamayız.

Başka bir örnek vermek gerekirse; bankacılık işlemlerini gerçekleştirmek istediğimizde oturum açma işlemlerini doğru şekilde yaptığımızda işlemleri yapmamıza izin vermektedir. Eğer oturum açma işlemini doğru yapamazsak işlemleri yapmamıza izin vermeyecektir.

İşte bu işlemleri Koşul ifadeleri sayesinde yapabiliyoruz. Python koşul ifadeleri; if, elif ve else şeklindedir. Bu deyimleri anlatmadan önce koşullu işlemlerin tablosuna bir bakalım.

İşleç	Anlamı
>	Büyük
<	Küçük
>=	Büyük eşittir
<=	Küçük eşittir
==	Eşittir
!=	Eşit değildir

Resim 10: Koşul İşleçleri (Özgül, 2015)

- İf Yapıları

if deyiminin İngilizcedeki anlamı “eğer” demektir. Hemen hemen bütün programlama dillerinde en çok kullanılan koşul ifadesi “if” yapılarıdır. If deyimindeki koşullu ifadesi *TRUE* ve *FALSE* döner. İfade doğruysa if yapısının içindeki kod çalışır. İfade yanlış ise if yapısının içerisindeki kod çalışmadan bir sonraki satırdaki kod çalışır.

Temel olarak kullanımı şu şekildedir:

```
if koşul:  
    buradaki kod çalışır
```

Yukarıdaki ifadenin anlamı şu şekildedir: Eğer if satırındaki koşul yazan yer *TRUE* dönüyorrsa yani doğru bir ifadeyse, “*buradaki kod çalışır*” yazan yerdeki kod çalışır. Eğer koşul sağlamıyorsa, eylem bloğu atlanır.

```
sayi1=200  
sayi2=50;  
if sayi1>sayi2:  
    print("sayi1 sayi2'den büyüktür.")
```

Ekran çıktısı:

```
sayi1 sayi2'den büyüktür.
```

- İf Else Yapısı

Sadece koşul sağlanmadığı taktirde başka bir takım işlemler yapılacaksa if-else yapısı kullanılır.

```
if şart
    şart True (Doğru) ise bu işlemi yap
else:
    şart False (Yanlış) ise bu işlemi yap
```

if koşulu sağlanırsa if bloğunun altındaki işlemler yapılır ve else bloğu atlanarak bir sonraki kod satırı çalışır. Eğer if koşulu sağlanmazsa, bu sefer if bloğu atlanır ve else bloğundaki işlemler yapılır.

```
"""
sayi1=-10
if sayi1>=0 :
    print("Sayı Sıfır ya da Pozitif")
else:
    print("Sayı Negatif")
```

Ekran Çıktısı:

```
Sayı Negatif
```

- if-elif-else Yapısı

Tek bir koşulun değil de, daha fazla koşulun sıra ile değerlendirilmesi gerekiyorsa if-elif-else koşul ifadesi kullanılır. Başka bir şekilde anlatmak gerekirse, elif ifadesi birden fazla if koşulunun doğruluğunu kontrol etmek ve doğru

olduğunda kod bloğunu çalıştırmak için kullanılır. Else kullanımının zorunluluğu olmadığı gibi, elif kullanımının da zorunluluğu yoktur. İhtiyaç duyulduğunda kullanılır. Else ifadesini bir defa kullanabiliyorken elif ifadesini istediğimiz kadar kullanabiliriz.

```
"""
sayi1=0
if sayi1>0 :
    print("Sayı Pozitif")
elif sayi1==0:
    print("Sayı Sıfır")
else:
    print("Sayı Negatif")
```

Ekran Çıktısı:

```
| Sayı Sıfır
```

- Koşul İfadelerinde And ve Or Operatörleri

Koşul ifadelerinde and ve or operatörleri yardımıyla birden fazla durumu aynı anda kontrol edilebilmektedir.

```
if(kullaniciadi=='admin' and sifre=='123456')  
    print("Hoş geldiniz.")  
else:  
    print("Hatalı kullanıcı adı veya şifre.")
```

kullanıcı adı ve şifrenin aynı anda doğru olduğu durumlarda ‘Hoş geldiniz’ yanlış olduğu durumlarda ise ‘Hatalı kullanıcı adı veya şifre.’ mesajını ekrana yazdırabilmektedir.

```
"""  
isim = input('isminiz: ')  
yas = int(input('yaşınız: '))  
egitimdurumu = input('eğitim: ')  
if (yas>=18):  
    if (egitimdurumu=='lise' or egitimdurumu=='üniversite'):  
        print(f'{isim} ehliyet alabilirsin.')  
    else:  
        print(f'{isim} ehliyet alamazsun eğitim durumun yetersiz.')  
else:  
    print(f'{isim} ehliyet alamazsun yaşın tutmuyor.')
```

Ekran Çıktısı:

```
isminiz: Ahmet  
yaşınız: 21  
eğitim: lise  
Ahmet ehliyet alabilirsin.
```

Yukarıdaki örnekte ehliyet alma şartının, 18 yaşından büyük olmasına ve eğitim durumunun lise ya da üniversite mezunu olmasına bağlıdır.

Eğer 18 yaşından küçük ise ekran “*ehliyet alamazsin yaşın tutmuyor.*” şeklinde mesaj yazacaktır. 18 yaşından büyük ve eğitim durumu lise ya da üniversite değilse “*ehliyet alamazsin eğitim durumun yetersiz.*” şeklinde mesaj yazacaktır. 18 yaşından büyük ve eğitim durumu lise ya da üniversite mezunu ise “*ehliyet alabilirsin.*” şeklinde mesaj yazacaktır.

5.2.4. Döngüler

Python'da döngüler(loop) bir kod dizisinin tekrar tekrar çalışmasını sağlayan yapılardır. Temiz bir kod oluşturmak için döngülerden sıklıkla yararlanılır. İki farklı döngü çeşidimiz bulunmaktadır. Bunlar **while** ve **for** döngüleridir.

- While Döngüsü

Türkçe karşılığı “-iken” anlamına gelen while kelimesi kodların her defasında tekrarlanmasılığını sağlar. Koşul sağlanıyorsa döngü devam eder, eğer koşul sağlanmıyorsa döngü sona erer. Yani kısaca; while döngüsü şart doğru olduğu sürece çalışmaktadır.

```
sayi1=1
sayi2=5
while sayi1<sayi2:
    print(sayi1, " ")
    sayi1+=1 #Sayi1 değişkenini 1 arttırır.
```

Ekrana Çıktısı:

```
1
2
3
4
```

Örnekte 1'den 5'e kadar olan sayıları while döngüsü ile ekrana yazan bir program yazdık. Programda sayi1 ve sayi2 değerlerine başlangıç değerlerini atadık. Ardından while döngüsü şartında, sayi1 değişkeninin sayi2'den küçük olma koşulunu ekledik. While döngüsü bloku bitince program başa döndü ve şartın doğru olup olmadığını kontrol etti. Döngünün bloklar üzerindeki her bir geçişine **iterasyon** denilmektedir.

While koşulu($sayi1 < sayi2$) koşulu sağlanmadığında döngüden çıkışacaktır. Döngünün bitiminde ekranda 5 sayısı yazılmadığını görüyoruz. Bunun nedeni döngünün 5 değerine ulaştığında “ $sayi1 < sayi2$ ” koşulunun (FALSE) sağlanmadığı ve buna buna bağlı olarak döngünün bir daha çalıştırılmamasıdır.

Başka bir örnekte; Bir ile On arasındaki sayıları toplayan programı while döngüsü ile yazalım.

```
baslangic=1
bitis=10
toplam=0
while baslangic<=bitis:
    toplam += baslangic
    baslangic += 1
print("Sayıların toplamı=",toplam)
```

Ekrana Çıktısı:

```
Sayıların toplamı= 55
```

```
In [46]:
```

Örnekte, döngünün her adımında başlangıç değerini toplam değişkenine ekliyoruz. While döngüsünün her bir adımını aşağıdaki tabloda gösterilmiştir.

<u>iterasyon</u>	<u>baslangic</u>	<u>baslangic <= bitis</u>	<u>toplam</u>
1	1	Doğru	1
2	2	Doğru	3
3	3	Doğru	6
4	4	Doğru	10
...
10	10	Doğru	55
11	11	Yanlış	55

Resim 11: While Döngüsü Adımları

- For Döngüsü

For döngüsünün çalışma mantığı temel olaraktan while döngüsüne benzer şekildedir. Ancak Python programlama dilinde, for döngüsünün while döngüsünden farkı sıralı nesneleri tek tek alıp işlemesidir. Ne demek istediğimizi kısa bir örnekle açıklayalım. Aradaki farklı daha iyi anlayabilmek için örneğimizi hem for döngüsü ile hem de while döngüsü ile yazalım.

```
"""
for a in ["Ankara", "İstanbul", "İzmir"]:
    print(a)
```

Ekran Çıktısı:

```
Ankara
İstanbul
İzmir
```

For döngüsü ile illerimizin isimlerini ekrana yazdırıldı. Şimdi de while döngüsü ile yazalım.

```
!
"""
iller= ["Ankara", "İstanbul", "İzmir"]
i=0
while i<len(iller): #iller dizisinin uzunluğu kadar döngü dönecek
    print(iller[i])
    i+=1
```

Ekran Çıktısı:

```
Ankara
İstanbul
İzmir
```

```
In [53]:
```

While döngüsü ile yazdığımızda programın daha karmaşık

olduğunu görmekteyiz. Ayrıca fazladan bir i değişkeni tanımlamamız gerekmektedir. For döngüsünde ise böyle bir duruma gerek yoktur.

Örnek: for döngüsü kullanımı

for döngüsü kullanılarak bir ile on arasındaki sayıları ekrana yazdırın python programını yazalım.

```
I
"""
for sayi in range(0,11):
    print(sayi)
```

Ekran Çıktısı:

```
0
1
2
3
4
5
6
7
8
9
10
In [54]:
```

Python döngülerde her ikisini de gördüğümüze göre, hangisini kullanmamız gereklili hususunda yazacağımız programın ihtiyacına göre karar vermemiz gerekmektedir. İki döngünün de birbirine göre üstünlükleri mevcuttur. Geliştirilen uygulamaya göre while ya da for döngüsü sizin için daha avantajlı olabilir.

- Len() Fonksiyonu

İngilizcedeki “length” kelimesinin kısaltması olan bu fonksiyon liste, karakter dizileri, demetler vb. değişkenlerin uzunluğunu verir (mobilhanem.com. 15.09.2021).

Örnek: len() fonksiyonu kullanımı

```
metin = "Python"  
print(len(metin))
```

Çıktı:

```
6
```

Örnek: len() fonksiyonu kullanımı 2

```
dizi =[20,6,"Python",12,25]  
print(len(dizi)) # dizinin uzunluğunu vermektedir.
```

Çıktı:

5

len fonksiyonunun kullanım alanları; örneğin bir uygulama için şifre üretildiği zaman en az 8 karakter ya da en fazla 10 karakter girmenizin istenildiği bu tür işlemlerde ya da uygulama içindeki bir verinin uzunluğunu öğrenmek için len fonksiyonu kullanılabilir. Tam sayılıarda kullanılmamaktadır.

- Range Fonksiyonu

Türkçe olarak “aralık” anlamına gelmektedir. Python’da sayı aralıklarını belirtmek için kullanırız. Temel olarak çalışma mantığı şu şekildedir.

```
range(başlangıç değeri, bitiş değeri, değişim miktarı)
```

Birkaç tane örnek vermek gerekirse;

range(15): 0 ile 15 arasındaki sayıları belirtir. Başlangıç değerini belirtmediginde 0, değişim miktarını da yine belirtmediğimiz için 1 alırız. Bitiş değerini dahil etmediğimize dikkat edelim.

range(1,25,2): bir ile yirmi dört arasındaki sayıların, ikili artış şeklindeki sayıları kapsar.

range(25,1,-2): Yirmi dört ile bir arasındaki ikili azalma olayıyla oluşan sayıları kapsamaktadır.

Örnek: range() fonksiyonu kullanımı

```
for sayı in range(11):
```

Yukarıdaki verilen koddan çıktı olarak 0 ile 10 arasında bulunan sayıları alırız.

- Break Deyimi

Dilimizdeki karşılığı “terketmek” olan bu deyim Python dilinde döngüleri bitirmek için kullanılır. Yorumlayıcı “break” deyimini işlettiği zaman döngüyü bitirir ve diğer kodları yorumlamaz.

Örnek: break deyimi kullanımı

```
"""
i=0
while (i<30):
    print(i)
    if(i==5):
        break
    i+=1
|
```

Ekran Çıktısı:

```
0
1
2
3
4
5
```

Örnekte i'nin değeri 5 olunca sonuç sağlanıyor ve break ifadesiyle karşılaştığı için döngü sona eriyor.

Örnek: break deyimi kullanımı 2

```
while True:
    ad=input("isminizi giriniz ya da çıkmak için x tuşuna basınız: ")
    if(ad=="x"):
        print("x tuşuna basıldı. çıkış yapılıyor...")
        break
    print(ad)
```

Ekrان Çıktısı:

```
isminizi giriniz ya da çıkmak için x tuşuna basınız: Ahmet
Ahmet

isminizi giriniz ya da çıkmak için x tuşuna basınız: Mehmet
Mehmet

isminizi giriniz ya da çıkmak için x tuşuna basınız: x
x tuşuna basıldı. Çıkış yapılıyor...
In [64]: |
```

Örnekte sonsuz bir döngünün break komutu ile nasıl sonlandırılacağını gördük. Klavyeden “x” tuşuna basıldığında “if” şartı sağlanarak “break” komutu ile while döngüsünden çıkmaktadır.

- Continue Deyimi

Türkçe olarak “devam” anlamına gelen bu kelime Python’da yorumlayıcı tarafından görüldüğünde sonraki kodları işletmeden döngünün en başına gelinir.

Örnek: continue deyimi kullanımı

```
| for sayı in range(10):
|   if sayı%2==0:
|     continue
|   else:
|     print(sayı)
```

Ekran Çıktısı:

```
| 1
| 3
| 5
| 7
| 9
In [67]:
```

Yapılan örnekte eğer sayı değişkeninin 2'ye bölümünden kalan 0 ise continue deyimi çalışacak ve döngünün başına dönecektir.

5.3. Listeler ve Demetler

Bu kısımda listeler ve demetlerden söz edeceğiz. Ekstra olarak da liste ve demetlerin metodlarına da değineceğiz. Listeleri ve demetleri kavradıktan sonra, Python'daki bazı işlemleri kolaylıkla yapabildiğimizi göreceğiz. (Özgül. 2015)

5.3.1. Liste Tanımlamak

Listeler, Python'daki veri tiplerinden birisidir. Listeleri tanımlamak için köşeli parantez kullanırız. Karakter dizisini tanımlamak için aşağıdaki gibi yol izliyoruz:

```
karakterdizisi = "karakter dizisi"
```

Bir değeri karakter dizisi olarak tanımlamak için o değeri tırnak içine almamız yeterli olacaktır. Herhangi bir değeri “tek, çift veya üç” tırnak içine alındığında bir karakter dizisi oluşturulur.

```
liste = ["değer1", "değer2", "değer3"]
```

Yukarıdaki örnekte görüldüğü gibi liste oluşturmak için köşeli parantez içerisine alıp öğeleri virgülle ayıriyoruz.

Bir değerin karakter dizisi olduğunu anlamak için type() metodundan yararlanabiliriz. Eğer bir nesne type() fonksiyonu ile sorgulandığında, cevabı “<class ‘str’>” ise o nesne karakter dizisidir. Benzer bir sorgulama listeler için de yapılabilir. (Özgül. 2015)

Örnek: string karakter dizisi type() fonksiyonu kullanımı

```
karakterdizi = "karakter dizisi"  
print(type(karakterdizi))
```

Ekran Çıktısı:

```
<class 'str'>
```

Örnek: type() fonksiyonu kullanımı 2

```
liste = ["değer1", "değer2", "değer3"]  
print(type(liste))
```

Ekran Çıktısı:

```
<class 'list'>
```

Yukarıdaki çıktılardan anlaşılabileceği üzere karakter dizisi type() fonksiyonuna <class 'str'> cevabını veriyor. Liste veri tipi type() fonksiyonuna <class 'list'> cevabı veriyor.

Listeler, bir veya daha fazla veri tipini içinde barındıran kapsayıcı bir veri tipidir. Mesela şu listeye bir bakalım:

```
liste = ["Isparta", "Zonguldak", 13, 25, 23.26]
```

Yukarıdaki örnekte görüldüğü üzere, liste içerisinde hem karakter dizileri ("Isparta", "Zonguldak"), hem de sayılar (13,25, 23.26) yer alabilmektedir.

Listeler içerisinde başka listeler de bulunabilmektedir.

```
liste = ["Isparta", "Zonguldak", ["Ahmet", "Mehmet", "Zeynep"], 234, 365, 33, 15.6]
```

Bir de şuna bakalım:

Örnek: liste kullanımı

```
liste = ["Isparta", "Zonguldak", ["Ahmet", "Mehmet", "Zeynep"], 234, 365, 33, 15.6]
for item in liste:
    print("{} adlı ögenin veri tipi: {}".format(item, type(item)))
```

Ekran Çıktısı:

```
Isparta adlı ögenin veri tipi: <class 'str'>
Zonguldak adlı ögenin veri tipi: <class 'str'>
['Ahmet', 'Mehmet', 'Zeynep'] adlı ögenin veri tipi: <class 'list'>
234 adlı ögenin veri tipi: <class 'int'>
365 adlı ögenin veri tipi: <class 'int'>
33 adlı ögenin veri tipi: <class 'int'>
15.6 adlı ögenin veri tipi: <class 'float'>
```

5.3.2. Liste Metotları

Listenin Metotlarına geçmeden önce Metot kavramına biraz değinelim. Metot bir şey üzerinde; Ekleme, çıkarma, değiştirme

vs. gibi işlemler yapmak demektir. O zaman Listenin Metotları nelerdir hemen bakalım.

- dir fonksiyonu

`dir()` fonksiyonu sayesinde liste içerisinde hangi metotlar varsa hepsini görebiliyoruz.

```
print(dir(list) )
```

Ekrان Çıktısı:

```
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__',  
'__dir__', '__doc__', '__eq__', '__format__', '__ge__',  
'__getattribute__', '__getitem__', '__gt__', '__hash__', '__iadd__',  
'__imul__', '__init__', '__init_subclass__', '__iter__', '__le__',  
'__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__',  
'__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__',  
'__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append',  
'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove',  
'reverse', 'sort']
```

Yukarıda görüldüğü gibi, típkı karakter dizilerinde olduğu gibi, listelerin metotlarını görmek için de `dir()` fonksiyonuna parametre olarak veri tipinin teknik adını veriyoruz. Python'da listelerin teknik adı `list` olduğu için bu komutu `dir(list)` şeklinde kullanıyoruz.

- `split()` Fonksiyonu

Python'da `split()` metodu kendisine referans verilen bir ifade veya dizgeyi belirtilen parametrelere göre parçalayan ve bu parçalardan liste oluşturan bir metottur. Split metodu iki farklı parametre almaktadır. Bu parametrelerden birincisi parçalara ayırmak için kullanılacak referans ifadedir. Diğer parametre ise limit değeridir.

Örnek: `split()` fonksiyonu kullanımı

```
metin = "Merhaba Sayın Seyirciler"
kelime = metin.split()
print(kelime)
```

Ekran Çıktısı:

```
['Merhaba', 'Sayın', 'Seyirciler']
```

Yukarıdaki örnekte Üç kelimededen oluşan bir cümleyi `split` metodu kullanarak her kelimeyi ayrı ayrı bir eleman olarak değerlendiren bir liste yapısına dönüştürdük.

Örnek: split() fonksiyonu kullanımı 2

```
metin = "Test amaçlı şehir isimleri yazalım: Isparta,Ankara, Antalya, Zonguldak, Ankara, İstanbul"
sehirlerListesi = metin.split(',')
print(sehirlerListesi)
```

Ekran Çıktısı:

```
['Test amaçlı şehir isimleri yazalım: Isparta', 'Ankara', 'Antalya', 'Zonguldak', 'Ankara', 'İstanbul']
```

Bir önceki örneğimizde split metodunu kullanırken herhangi bir parametre vermeden kullanmıştık. Split metodu parametre verilmediği zaman default(varsayılan) olarak boşluk ifadesini parametre olarak kabul eder. Ancak bu örneğimizde parametre olarak virgül ifadesini kullanıyoruz ve metnimizi virgül ifadelerini dikkate alarak parçalara ayırmasını istiyoruz. Döngü kullanmadığımız için ayrılan parçalar bir liste yapısında belirli index değerleri kullanılarak bir araya getirilecektir.

Örnek: split() fonksiyonu kullanımı 3

```
metin = "Test amaçlı şehir isimleri yazalım: Isparta,Ankara, Antalya, Zonguldak, Ankara, İstanbul"
sehirlerListesi = metin.split(',')
for sehir in sehirlerListesi:
    print(sehir)
```

Ecran Çıktısı:

```
Test amaçlı şehir isimleri yazalım: Isparta
Ankara
Antalya
Zonguldak
Ankara
İstanbul
```

Yukarıdaki örnekte ise parçalara ayırdığımız metni bir liste yapısı olarak değil de bir string ifade olarak görmek istiyoruz.

- `list()` Fonksiyonu

Karakter dizisini listeye çevirmek için `list()` fonksiyonunu kullanıyoruz.

Örneğin: Elimizde alfabe karakter dizisi olsun. Bu karakter dizisi içerisinde her bir öğeyi birbirinden ayırmak istiyoruz. Yani listeye dönüştürmek istiyoruz. İşte bu tür durumlarda `list()` fonksiyonundan yararlanıyoruz.

```
alfabe = "abcçdefgğhıijklmnoöprsştuüvyz"  
harf_listesi = list(alfabe)  
print(harf_listesi)
```

Ekran Çıktısı:

```
['a', 'b', 'c', 'ç', 'd', 'e', 'f', 'g', 'ğ', 'h', 'ı', 'i', 'j', 'k', 'l',  
'm', 'n', 'o', 'ö', 'p', 'r', 's', 'ş', 't', 'u', 'ü', 'v', 'y', 'z']
```

Böylece list() fonksiyonu yardımıyla bu karakter dizisini tek hamlede listeye çevirmiş olduk.

5.3.3. Listeye Öğe Ekleme

- Append Metodu

Append Metodu, kendisine gönderilen değeri listenin sonuna ekler. Append metodu ile liste sonuna sadece bir elaman eklenebilmektedir.

Örnek: append() fonksiyonu kullanımı

```
sehirler = ["İsparta", "Ankara", "Antalya", "Zonguldak", "Ankara", "İstanbul"]  
print(sehirler)  
sehirler.append("İzmir")  
print(sehirler)
```

Ecran Çıktısı:

```
['Isparta', 'Ankara', 'Antalya', 'Zonguldak', 'Ankara', 'İstanbul']
['Isparta', 'Ankara', 'Antalya', 'Zonguldak', 'Ankara', 'İstanbul', 'İzmir']
```

- Insert Methodu

Append metodu ile listenin sonuna elaman eklenebilirken, insert metodu ile listenin belirlediğimiz indeksine eleman eklenebilmektedir.

Örnek: insert() fonksiyonu kullanımı

```
sehirler = ["Isparta", "Ankara", "Antalya", "Zonguldak", "Ankara", "İstanbul"]
print(sehirler)
sehirler.insert(2, "İzmir")
print(sehirler)
```

Ecran Çıktısı:

```
['Isparta', 'Ankara', 'Antalya', 'Zonguldak', 'Ankara', 'İstanbul']
['Isparta', 'Ankara', 'İzmir', 'Antalya', 'Zonguldak', 'Ankara', 'İstanbul']
```

Izmir şehirler dizinisinin 2. İndeksine eklendi.

Not: dizilerde indeks numarası 0 “sıfır” dan başlar.

- Extend Metodu

Dilimizdeki karşılığı “genişlemek” olan extend metodu sayesinde listeleri genişletmek mümkündür.

Örnek: extend() fonksiyonu kullanımı

```
sehirler = ["İsparta", "Ankara", "Antalya", "Zonguldak", "Ankara", "İstanbul"]
meyveler=[ "Portakal", "Elma"]
sehirler.extend(meyveler)
print(sehirler)
```

Ekran Çıktısı:

```
['İsparta', 'Ankara', 'Antalya', 'Zonguldak', 'Ankara', 'İstanbul',
'Portakal', 'Elma']
```

Yukarıdaki örnekte görüldüğü üzere, Extend() metodu meyveler listesini sehirler listesine ekledi. Dikkat ederseniz ikinci liste ilk listenin sonuna eklendi.

5.3.4. Listeden Öğe Çıkarmak

- Remove Metodu

Dilimzdeki karşılığı “atmak,silmek, kaldırırmak” olan Remove metodu listenin istenilen elamanını silmeye yarar. Eğer silinmek istenen eleman listede yoksa ValueError hatası verir.

Örnek: remove() fonksiyonu kullanımı

```
sehirler = ["İsparta", "Ankara", "Antalya", "Zonguldak", "Ankara", "İstanbul"]
print(sehirler)
sehirler.remove("Antalya")
print(sehirler)
```

Ekran Çıktısı:

```
['İsparta', 'Ankara', 'Antalya', 'Zonguldak', 'Ankara', 'İstanbul']
['İsparta', 'Ankara', 'Zonguldak', 'Ankara', 'İstanbul']
```

Yukarıdaki örnekte sehirler listesindeki “Antalya” elemanını silmiş olduk.

- Pop Metodu

Dilimizdeki karşılığı; “fırlatmak” olan Pop metodu Remove metodunda olduğu gibi eleman silmeye yarar. Pop metodu elamanın indeksi ile silmeye işlemini yapar, indeks

belirtildiğinde ise varsayılan olarak listenin son elemanını siler.

Örnek: pop() fonksiyonu kullanımı

```
sehirler = ["Isparta", "Ankara", "Antalya", "Zonguldak", "Ankara", "İstanbul"]
print(sehirler)
sehirler.pop()
print(sehirler)
```

Ekran Çıktısı:

```
['Isparta', 'Ankara', 'Antalya', 'Zonguldak', 'Ankara', 'İstanbul']
['Isparta', 'Ankara', 'Antalya', 'Zonguldak', 'Ankara']
```

- Del Metodu

del() metodu ile herhangi bir indeks numarasındaki elemanı silebiliriz.

Örnek: del() fonksiyonu kullanımı

```
sehirler = ["Isparta", "Ankara", "Antalya", "Zonguldak", "Ankara", "İstanbul"]
del sehirler[2]
print(sehirler)
```

Ekran Çıktısı:

```
['Isparta', 'Ankara', 'Zonguldak', 'Ankara', 'İstanbul']
```

5.3.5. Listeleri Silmek

Python'da listeleri tamamen silmek de mümkündür.

Örnek: liste silme

```
sehirler = ["Isparta", "Ankara", "Antalya", "Zonguldak", "Ankara", "İstanbul"]
del sehirler
print(sehirler)
```

Ekran Çıktısı:

```
In [42]: runfile('C:/data/untitled0.py', wdir='C:/data')
Traceback (most recent call last):

  File "C:\data\untitled0.py", line 10, in <module>
    print(sehirler)

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

Bu durumda liste objesine ulaşmak istediğimizde NameError alırız ve açıklama da ise sehirler tanımlanmadı şeklinde hata mesajı gelir.

5.3.6. Listeleri Kopyalamak

Diyelim ki yazmış olduğunuz programda var olan bir listeyi kopyalamak istiyorsunuz. Listelerde kopyalama işlemini Copy() methodu ile yapıyoruz.

Örnek: copy() fonksiyonu kullanımı

```
sehirler = ["İsparta", "Ankara", "Antalya", "Zonguldak", "Ankara", "İstanbul"]
yenidizi=[]
yenizdizi=sehirler.copy()
print(yenizdizi)
```

Ekran Çıktısı:

```
['İsparta', 'Ankara', 'Antalya', 'Zonguldak', 'Ankara', 'İstanbul']
```

5.3.7. Liste Üreteçleri (List Comprehensions)

Liste üreteçleri diğer liste tanımlama tiplerine göre daha kısa sözdizimi ile liste üretmemizi sağlar.

```
sehirler = [x for x in ("İsparta", "Ankara", "Antalya", "Zonguldak", "İstanbul")]
print(sehirler)
```

Ekran Çıktısı:

```
['İsparta', 'Ankara', 'Antalya', 'Zonguldak', 'İstanbul']
```

1 ile 100 arasındaki sayılardan oluşan bir liste üretmek için for döngüsü ve üreteç kullanımı ile liste oluşturmanın farkını aşağıdaki kod parçasında görebiliriz. Döngü ile liste doldurulmadan önce listenin boş olarak tanımlanması gerekmektedir.

```
sayılardongu=[]
for x in range(1,100):
    sayılardongu += [x]
print(sayılardongu)

sayılar=[x for x in range(1,100)]
print(sayılar)
```

Ekran Çıktısı:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48,
49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71,
72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94,
95, 96, 97, 98, 99]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48,
49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71,
72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94,
95, 96, 97, 98, 99]
```

Örnekte de görüldüğü üzere ekran çıktıları aynı ancak listenin tanımlama şekli üreteç ile daha kısa sözdiziminde oluşturulmuş oldu.

Liste üreteci ile liste tanımlanırken şart yazmakta mümkündür. 1 ile 100 arasında 5 ile tam bölünebilen sayılarından bir liste oluşturmak için;

```
sayılar=[x for x in range(1,100) if x % 5 == 0]
print(sayılar)
```

kod parçası kullanılabilir.

Ecran Çıktısı:

```
[5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95]
```

5.3.8. Demetler (tuples)

Demetler listelere çok benzeyen ve farklı türden verileri bir değişken içinde tutabileceğimiz bir veri türüdür. Listelerden en büyük farkı ise elemanlarının **değiştirilemez** olmasıdır.

5.3.9. Demet Tanımlamak

Demetler tanımlanırken normal parantez işaretini kullanılır.

Örnek: demet kullanımı

```
sehirDemeti=("Isparta","Ankara","Anatalya","Zonguldak","İstanbul")
print(sehirDemeti)
```

Ecran Çıktısı:

```
('Isparta', 'Ankara', 'Anatalya', 'Zonguldak', 'İstanbul')
```

Demet oluştururken liste oluşturmak için kullanılan list() fonksiyonu gibi, **tuple()** fonksiyonundan da yararlanılabilir.

Örnek: tuple() fonksiyonu kullanımı

```
rakamDemeti=tuple("0123456789")
print(rakamDemeti)
```

Ecran Çıktısı:

```
('0', '1', '2', '3', '4', '5', '6', '7', '8', '9')
```

tuple() fonksiyonu ile bir listeyi demet haline dönüştürmek mümkündür.

```
birdemet=tuple(["Isparta",32,"Burdur",15])
print(birdemet)
```

Ecran Çıktısı:

```
('Isparta', 32, 'Burdur', 15)
```

5.3.10. Tek Öğeli bir Demet Tanımlamak

Python'da tek öğeli bir demet tanımlamak biraz farklıdır. Tek öğeli bit liste tanımlarken tekListe=[“eleman”] kullanılabilir ancak aynı yazım şeklini tekDemet=(“eleman”) olarak

kullanırsak aslında parantez içine yazdığımız değere göre bir değişken tanımlamış oluruz.

```
tekListe=["eleman"]  
tekDemet=("eleman")
```

```
In [47]: type(tekListe)  
Out[47]: list  
  
In [48]: type(tekDemet)  
Out[48]: str
```

Tek öğeli bir demet tanımlamak için demetteki tek öğenin yanına virgül işaretini koymalıyız.

```
tekDemet=("eleman",)
```

```
In [50]: type(tekDemet)  
Out[50]: tuple
```

5.3.11. Demetlerin Öğelerine Erişmek

Tanımlanmış bir demetin öğelerine erişmek için tipki listelerde olduğu gibi köşeli parantez [] kullanılır. Örnekte sehirDemeti demetinin 1. indexli elemanını yazdırınmak için gerekli komutları görmekteyiz.

```
sehirDemeti=("Isparta", "Ankara", "Antalya", "Zonguldak", "İstanbul")
print(sehirDemeti[1])
```

Ecran Çıktısı:

```
Ankara
```

5.3.12. Demetlerle Listelerin Birbirinden Farkı

Demetler ve listeler birbirlerine çok benzerdirler. Aralarındaki en önemli fark demetlerin değiştirilemez olmasıdır. Bir demet tanımlandıktan sonra elemanlarına erişilebilir ama erişilen elemanlar üzerinde değişiklik yapılamaz ve hata alırız.

```
sehirDemeti=("Isparta", "Ankara", "Antalya", "Zonguldak", "İstanbul")
sehirDemeti[1]="Burdur"
print(sehirDemeti)
```

Ecran Çıktısı:

```
sehirDemeti[1]="Burdur"
TypeError: 'tuple' object does not support item assignment
```

Bir program yazarken değiştirilmesini istemediğimiz ya da değiştirmeyeceğimiz verileri demet olarak tanımlayabiliriz. Demetler üzerinde işlem yapmak listelere göre daha hızlı olduğundan demet kullanmak performans açısından da faydalı olacaktır.

Eğer bir demet elemanları üzerinde değişiklik yapmak istersek bunu doğrudan yapamayız. Bir demeti önce liste tipine çevirip istediğimiz değişiklikleri yaptıktan sonra tekrar demet haline dönüştürmek bu işlem için bir çözüm yoludur.

```
sehirDemeti=("Isparta","Ankara","Anatalya","Zonguldak","İstanbul")
print("SehirDemeti İlk Hali", sehirDemeti)
sehirListesi=list(sehirDemeti)
sehirListesi[4]="Burdur"
sehirDemeti=tuple(sehirListesi)
print("sehirDemeti Değiştirilmiş" ,sehirDemeti)
```

Ekran Çıktısı:

```
SehirDemeti İlk Hali ('Isparta', 'Ankara', 'Anatalya', 'Zonguldak', 'İstanbul')
sehirDemeti Değiştirilmiş ('Isparta', 'Ankara', 'Anatalya', 'Zonguldak', 'Burdur')
```

5.3.13. Demetlerin Kullanım Alanı

Demetler öğeleri değiştirilemediği için gerek görülmeyen bir veri tipi gibi gelebilir. Ancak kullanım alanı oldukça fazladır. Örneğin uygulamaların konfigürasyon dosyasında demetler ile bir çok tanımlama sabit olarak saklanır.

6. Python Kütüphaneleri

Python da kütüphane belli bir amacı gerçekleştirebilmek için içerisinde farklı hazır fonksiyonları barındıran kodların birleşimi ile oluşan kodlar topluluğu alarak tanımlanabilir. Belli başlı işler için oluşturulmuş kodları sürekli tekrar yazmak yerine hazır yazılmış kütüphaneleri uygulamamıza dâhil ederek hızlıca kullanabiliriz.

Python programlama dilinde kütüphane yüklemek ve kullanmak oldukça kolaydır. Eğer çevresel değişkenlere ekleme işlemi (Add to environmental variables) doğru şekilde yapıldı ise bilgisayarınızın komut penceresinde;

```
pip install kütüphane_adi
```

komutunu yazmak istenen kütüphanenin indirilip kurulması için yeterlidir. Farklı bir yöntem olarak Spyder editöründe test alanına aynı komutu girerek gerekli kütüphanenin yüklenmesini sağlayabiliriz.

Python da aynı uygulama içinde birden fazla kütüphane eklenip kullanılabilir. Python da kütüphane ekleme işlemi “import” deyimi ile gerçekleştirilir.

```
import kutuphane_adi  
import serial  
import numpy  
import turtle
```

Kütüphaneler import edildikten sonra kullanım aşamasında kendi anlayacağımız şekilde isimlendirip kullanabilmek için “as” ifadesinden yararlanabiliriz. Örneğimizde “serial” kütüphanesinin kod içerisinde “sr” olarak kullanılacağını belirtmiş olduk.

```
import serial as sr  
  
SeriCon = sr.Serial(port='COM5',baudrate=9600)
```

6.1. Serial Kütüphanesi

Python ile bilgisayarımızın seri portu üzerinden veri göndermek için “pyserial” kütüphanesinden faydalanzız. pyserial ile serial port tanımlayıp kullanılırken bazı ayarlar yapmak gereklidir (readthedocs.io. 05.09.2021).

Baud hızı (Baud Rate)	Serial port bağlantı noktamızın ne kadar hızla çalışacağını belirtir
Bağlantı Noktası (Port)	Serial port bağlantı noktasının adını belirtir
Eşlik Bitleri (Parity Bits)	Serial port hata düzeltmeleri için kullanılan bitlerdir, normal durumlarda kullanılmaz
Durdurma Bitleri (Stop Bits)	Zamanlama ile ilgili sorun bulanmadıkça bir durdurma biti kullanılır
Zaman Aşımı (Time Out)	Serial port bağlantı noktasının takılmasını önlemek için veri gönderme-alma sırasında zaman açımı süresi olarak kullanılır

Resim 122: Serial Port ayarları

Pyserial ile serial port kullanmak için öncelikle kütüphanemizi programımıza dahil etmemiz gereklidir. Ardından serial portumuzu tanımlayabiliriz.

```
import serial  
  
SeriCon = serial.Serial(port='COM5', baudrate=9600)
```

Serial port tanımlanırken sadece port adı ve hızı girilerek tanımlama yapılabilir. Zaman aşımı değeri belirtilmeden serial port tanımlanmasının yazılan kod parçasına göre uygulamayı sonsuz beklemeye sokabileceğinden zaman açımı değeri tanımlanması tavsiye edilir.

```
import serial

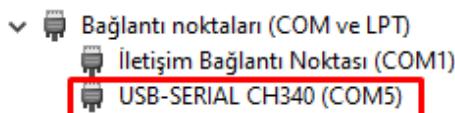
SeriCon = serial.Serial(port='COM5', \
    baudrate=9600, \
    parity=serial.PARITY_NONE, \
    stopbits=serial.STOPBITS_ONE, \
    bytesize=serial.EIGHTBITS, \
    timeout=1)
```

Yukarıdaki örnek kodlarımızda serial portun ilk tanımlanması esnasında ayarlarının yapılması gördük. Alternatif olarak serial port oluşturulduktan sonra da ayarlamaları yapılabilir.

```
import serial

SeriCon = serial.Serial()
SeriCon.port= 'COM5'
SeriCon.baudrate= 9600
SeriCon.timeout=1
SeriCon.open()
```

Bilgisayarımız bağlı seri portların isimlerini bilgisayarınızın aygit yöneticisinde bulunan bağlantı noktaları kısmından bakabilirsiniz.



Resim 133: Bilgisayara Bağlı Serial Portlar

Serial port noktamızı tanımladıktan sonra bağlantı noktamız üzerinde kullanacağımız işlevler aşağıda tabloda verilmiştir.

open()	Serial port bağlantı noktasını açmak için kullanılır
close()	Serial port bağlantı noktasını kapatmak için kullanılır
readline()	Serial port bağlantı noktasından bir satırı okumak için kullanılır
read(size)	Serial port bağlantı noktasından belirtilen 'size' kadar byte okumak için kullanılır
write(data)	Serial port bağlantı noktasına 'data' verisini gönderir
in_waiting	Serial port arabelleğindeki byte sayısını tutan değişkendir (veri olup olmadığını anlaşılmasıında kullanılır)

Resim 142: Serial Port İşlevleri

Yazılan programlarda serial port üzerinden veri gönderme ya da alma işlemlerinden önce bağlantı portunun açık olması gereği unutulmamalıdır. Eğer serial port kapalı ise serial portun open() fonksiyonu ile açılmalıdır.

6.2. Numpy Kütüphanesi

Numpy kütüphanesi python da kullanılan ve karmaşık matematiksel işlemleri yapabilen, çok büyük boyutlu dizileri ve matrisleri işleyebilen, veri madenciliği ve veri analizi alanlarında kullanılan bir kütüphanedir. Python listeleri ile benzer özelliklerdedir. Ancak python listelerine göre çok daha hızlıdır ve numpy listeleri üzerinde matris işlemleri yapılmaktadır (medium.com. 05.09.2021).

- Numpy Listesi Oluşturmak

Numpy kütüphanesinden bir liste oluşturmak aslında Python listesi oluşturmaya çok benzerdir. Numpy listesi oluşturmak için “numpy.array()” fonksiyonundan faydalanız.

Örnek:

```
import numpy  
  
np_listesi=numpy.array([0,1,2,3,4,5,6,7,8,9])  
  
print(np_listesi)
```

Ekran Çıktısı:

```
[0 1 2 3 4 5 6 7 8 9]
```

- Numpy Listesinin Boyutu

Numpy listesinin boyunu “ndim” fonksiyonu ile buluruz.

Örnek:

```
import numpy  
np_listesi=numpy.array([0,1,2,3,4,5,6,7,8,9])  
print("numpy liste boyutu")  
print(np_listesi.ndim)
```

Ekran Çıktısı:

```
numpy liste boyutu  
1
```

- Numpy Listesinin Satır ve Sütun Sayısı

Numpy listesinin satır ve sütun sayısını “shape” fonksiyonu ile buluruz.

Örnek:

```
import numpy  
  
np_listesi = numpy.array([[0, 1, 2, 3, 4], [5, 6, 7, 8, 9]])  
print(np_listesi)  
print("numpy Listesi satır sütün sayısı")  
print(np_listesi.shape)
```

Ekran Çıktısı:

```
[[0 1 2 3 4]
 [5 6 7 8 9]]
numpy listesi satır sütün sayısı
(2, 5)
```

- Arange fonksiyonu ile numpy listesi oluşturmak

Numpy “arange” fonksiyonu python listelerindeki “range” fonksiyonu gibi “başlangıç değeri, bitiş değeri, artış miktarı” değerlerine göre numpy dizisi oluşturur. Arange fonksiyonuna tek değer verilirse bu bitiş değeri olarak kabul edilir ve varsayılan olarak başlangıç değeri “0” ve artış değeri “1” alınır.

Örnek:

```
import numpy

dizi=numpy.arange(0,20,1)
print(dizi)
```

Ekran Çıktısı:

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14
15 16 17 18 19]
```

- Numpy dizisini yeniden boyutlandırmak

Numpy dizisini yeniden boyutlandırmak için “reshape” fonksiyonundan faydalanılır. “reshape” fonksiyonuna yeni satır ve sütun sayısını vererek boyut değiştirme işlemini gerçekleştirmiş oluruz. Dikkat edilmesi gereken durum ilk dizi ile boyutu değiştirilmiş dizinin aynı eleman sayısına sahip olması gerektigidir.

Örnek:

```
import numpy

dizi=numpy.arange(0,20,1)
print(dizi)
print()
print(dizi.reshape(2,10))
print()
print(dizi.reshape(10,2))
print()
print(dizi.reshape(4,5))
print()
```

Ekran Çıktısı:

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14  
15 16 17 18 19]  
  
[[ 0  1  2  3  4  5  6  7  8  9]  
 [10 11 12 13 14 15 16 17 18 19]]  
  
[[[ 0  1]  
 [ 2  3]  
 [ 4  5]  
 [ 6  7]  
 [ 8  9]  
 [10 11]  
 [12 13]  
 [14 15]  
 [16 17]  
 [18 19]]  
  
[[[ 0  1  2  3  4]  
 [ 5  6  7  8  9]  
 [10 11 12 13 14]  
 [15 16 17 18 19]]]
```

- Numpy dizisinden eleman seçmek

Numpy dizisinden eleman seçme işlemi köşeli parantezler “[ve ” ile yapılır. Genel yazılımı aşağıdaki gibidir;

[satır başlangıç : satır bitiş , sütun başlangıç : sütun bitiş]

Eğer köşeli parantez içine değer yazılmadan sadece : konulursa ([:,:]) bu tüm satır ve sütunlar anlamına gelir.

Örnek:

```
import numpy

dizi=numpy.arange(0,10,1).reshape(2,5)
print('dizi')
print(dizi)
satir1=dizi[0:1]
print('dizinin 0. satırı')
print(satir1)
satir12=dizi[0:2]
print('dizinin 0. ve 1. satırı')
print(satir12)
sutun1=dizi[:,2]
print('dizinin 0. sütunu')
print(sutun1)
sutun12=dizi[:,0:2]
print('dizinin 0. ve 1. sütunu')
print(sutun12)
eleman=dizi[1,3]
print('dizinin 1. satır 3. sütündeki elemanı')
print(eleman)
```

Ekran Çıktısı:

```
dizi
[[0 1 2 3 4]
 [5 6 7 8 9]]
dizinin 0. satırı
[[0 1 2 3 4]]
dizinin 0. ve 1. satırı
[[0 1 2 3 4]
 [5 6 7 8 9]]
dizinin 0. sütunu
[2 7]
dizinin 0. ve 1. sütunu
[[0 1]
 [5 6]]
dizinin 1. satır 3. sütündeki elemanı
8
```

- 0 ve 1 numpy matris dizisi oluşturmak

Numpy kütüphanesi ile tüm elemanları ‘0’ ya da ‘1’ olan matrisler oluşturmak için zeros ve ones fonksiyonlarından faydalanırız.

```
numpy.zeros((satır sayısı, sutun sayısı))
numpy.ones((satır sayısı, sutun sayısı))
```

Örnek:

```
import numpy  
print("0 Sifir matrisi")  
print(numpy.zeros((3,6)))  
print("1 Bir matisi")  
print(numpy.ones((2,4)))
```

Ekran Çıktısı:

```
0 Sifir matrisi  
[[0. 0. 0. 0. 0. 0.]  
 [0. 0. 0. 0. 0. 0.]  
 [0. 0. 0. 0. 0. 0.]]  
1 Bir matisi  
[[1. 1. 1. 1.]  
 [1. 1. 1. 1.]]
```

- Numpy ile birim matris oluşturma

Birim matris $n \times n$ (satır ve sütun sayısı eşit) boyutunda ve ana köşegeni ‘1’ lerden diğer elemanları ‘0’ lardan oluşan matrise denir. Numpy kütüphanesinde ‘eye’ fonksiyonu yardımıyla oluşturulabilir.

Örnek:

```
import numpy  
print("birim matris")  
print(numpy.eye(6))
```

Ekran Çıktısı:

```
birim matris  
[[1. 0. 0. 0. 0. 0.]  
 [0. 1. 0. 0. 0. 0.]  
 [0. 0. 1. 0. 0. 0.]  
 [0. 0. 0. 1. 0. 0.]  
 [0. 0. 0. 0. 1. 0.]  
 [0. 0. 0. 0. 0. 1.]]
```

- Numpy ile matris birleştirme işlemi

Numpy kütüphanesinde iki matrisi birleştirmek için ‘concatenate’ fonksiyonu kullanılır. Matris birleşimi satır ya da sütun tarafından yapılabilir.

Örnek:

```
import numpy

dizi1=numpy.arange(0,6,1).reshape(2,3)
dizi2=numpy.arange(6,12,1).reshape(2,3)
print("Dizi 1")
print(dizi1)
print("Dizi 2")
print(dizi2)

print("Satır bazlı birleştirme")
print(numpy.concatenate([dizi1,dizi2],axis=0))
print("Sütun bazlı birleştirme")
print(numpy.concatenate([dizi1,dizi2],axis=1))
```

Ekran Çıktısı:

```
Dizi 1
[[0 1 2]
 [3 4 5]]
Dizi 2
[[ 6  7  8]
 [ 9 10 11]]
Satır bazlı birleştirme
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
Sütun bazlı birleştirme
[[ 0  1  2  6  7  8]
 [ 3  4  5  9 10 11]]
```

- Numpy matrisindeki en büyük elemanını, en küçük elemanını ve matrisin elemanlarının toplamını bulma

Örnek:

```
import numpy

dizi=numpy.arange(0,12,1).reshape(4,3)
print("matris")
print(dizi)
print("matrisin en büyük elemani")
print(dizi.max())
print("matrisin en küçük elemani")
print(dizi.min())
print("matrisin elemanlarini toplami")
print(dizi.sum())
```

Ekran Çıktısı:

```
matris
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
matrisin en büyük elemani
11
matrisin en küçük elemani
0
matrisin elemanlarini toplami
66
```

6.3. Turtle Kütüphanesi

Python programlama dilinde “turtle” kütüphanesi bir pencere içerisinde çizim yapmamızı sağlar. Turtle kütüphanesinde kullanılan fonksiyonların bir kısmı aşağıdaki gibidir (bilisimkonulari.com. 15.09.2021).

Fonksiyon	Parametre	Anlam
Turtle()	-	Yeni bir turtle nesnesi oluşturup geri döndürür
forward()	deger	Verilen değer kadar turtle nesnesini ileri götürür
backward()	deger	Verilen değer kadar turtle nesnesini geri getirir
right()	derece	Turtle nesnesini saat yönünde derece kadar döndürür
left()	derece	Turtle nesnesini saat yönü tersinde derece kadar döndürür
color()	renk	Turtle kalemi rengini belirler
fillcolor()	renk	Turtle ile oluşturuluran poligonun dolgu rengini belirler
shape()	tip	Turtle kaleminin görüntüsünü belirler ('arrow', 'classic', 'turtle' ya da 'circle') değerlerini alabilir.

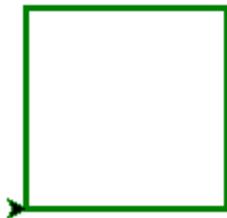
Resim 13: Turtle kütüphanesi fonksiyonları

Trutle kütüphanesini kullanarak geometrik şekillerin çizilmesi mümkündür. Geometrik şekilleri çizmeye başlamadan önce şeklin kenar sayısı ve kenarlar arası açıları belirlenmelidir. Örneğin bir kare çizmek istiyorsak kenar sayısını 4, açı değerini de 90 olarak belirlemeliyiz.

Örnek: Her bir kenarı 100 birim ve çizim kalınlığı 3 birim olan yeşil renkli kare

```
import turtle  
pen=turtle.Turtle()  
pen.pencolor("green")  
pen.pensize(3)  
for i in range(4):  
    pen.forward(100)  
    pen.left(90)  
turtle.done()
```

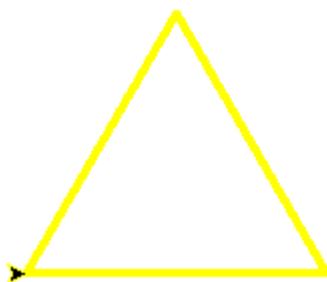
Ekran Çıktısı:



Örnek: Her bir kenarı 150 birim ve çizim kalınlığı 4 birim olan sarı renkli üçgen

```
import turtle  
pen=turtle.Turtle()  
pen.pencolor("yellow")  
pen.pensize(4)  
for i in range(3):  
    pen.forward(150)  
    pen.left(120)  
turtle.done()
```

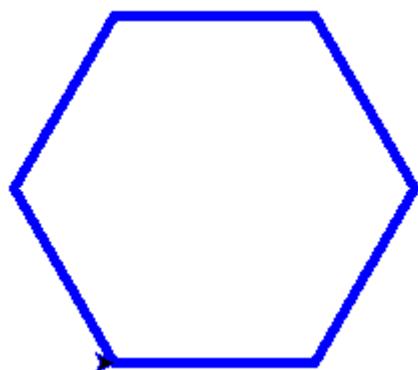
Ekran Çıktısı:



Örnek: Her bir kenarı 100 birim ve çizim kalınlığı 5 birim olan mavi renkli altıgen

```
import turtle  
pen=turtle.Turtle()  
pen.pencolor("blue")  
pen.pensize(5)  
for i in range(6):  
    pen.forward(100)  
    pen.left(60)  
turtle.done()
```

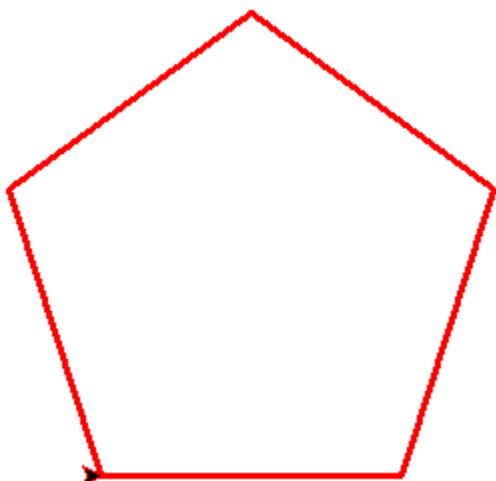
Ekran Çıktısı:



Örnek: Her bir kenarı 150 birim ve çizim kalınlığı 3 birim olan kırmızı renkli beşgen

```
import turtle  
pen=turtle.Turtle()  
pen.pencolor("red")  
pen.pensize(3)  
for i in range(5):  
    pen.forward(150)  
    pen.left(72)  
turtle.done()
```

Ekran Çıktısı:



7. Python Örnek Kodları

7.1. Ekrana “Merhaba Dünya” yazdırın Python örneği

- **print() Fonksiyonu Kullanımı:** print() fonksiyonu, konsola çıktı göndermek amacıyla kullanılır. Programların genellikle yapılan işlemler sonucunu kullanıcıya sunması gereklidir. Programda veri print() fonksiyonu ile Python'daki konsolda görüntülenebilir. (aktif.net. 10.09.2021)

Örnek:

```
print("Merhaba Dünya!")
```

Ecran Çıktısı:

```
Merhaba Dünya!
```

7.2. Kullanıcının İsmini Alarak Ekrana “Merhaba (Kullanıcı İsmi)” Yazdırın Python Örneği

- **input() Fonksiyonu Kullanımı:** input() fonksiyonu bir çok programlama dilinde kullanıcından yani klavyeden veri girişi yapmak için kullanılan hazır fonksiyondur. Kullanıcı klavyeden program içerisine bazı bilgiler

eklemesi gerekiyorsa input() fonksiyonunu kullanması gereklidir. Python programlama dilinde de kullanıcıdan bilgi almak için input() fonksiyonu kullanılır.

- input() fonksiyonu kullanılırken kullanıcıdan alınan değer bir değişkene atanmalıdır.
- input() fonksiyonu ile klavyeden girilen tüm bilgiler string olarak algılanır. Rakamlarda string olarak kabul edilir. Eğer girilen rakam ya da sayılar aritmetik işlemlerde kullanılacaksa herhangi bir sayısal tipe dönüştürülerek daha sonra işlemlere alınmalıdır.

Örnek: input() fonksiyonu kullanımı

```
isim = input('İsminizi Girin : ')
print("Merhaba " + isim)
```

Ekran Çıktısı:

```
İsminizi Girin : Dirlik
Merhaba Dirlik
```

7.3. Girilen 2 Sayıyı Toplayan Python Örneği

Python format Nedir?

format() metodu Python (2.6) ile eklendi. format metodu ile stringleri biçimlendirme daha kolay hale getirilmiştir.

format() metodunda kullanıcı, bir değişkenin nerede ikame edileceğini işaretlemek için {} ‘yi kullanır ve ayrıntılı biçimlendirme yönergeleri sağlayabilir, ancak kullanıcının biçimlendirilecek bilgileri de sağlaması gereklidir.

Örnek: format() fonksiyonu kullanımı

```
sayi1 = input('1. Sayı giriniz : ')
sayi2 = input('2. Sayı giriniz : ')
sayilarinToplami = float(sayı1) + float(sayı2)
print("Sayıların ToplAMI :{0} ".format(sayılarinToplami))
```

Ekran Çıktısı:

```
1. Sayı giriniz : 12
2. Sayı giriniz : 21
Sayıların ToplAMI :33.0
```

7.4. Girilen 2 Sayının Ortalamasını Bulan Python Örneği

Ortalama hesaplaması 2 sayıyı istendikten sonra bu iki sayının toplanarak 2'ye bölünmesi ile elde edilecektir.

```
sayi1 = input('1. Sayı giriniz : ')
sayi2 = input('2. Sayı giriniz : ')
sayilarinOrtalaması = (float(sayi1) + float(sayi2))/2
print("Sayıların Toplamları :{0} ".format(sayilarinOrtalaması))
```

Ekrana Çıktısı:

```
1. Sayı giriniz : 43
2. Sayı giriniz : 22
Sayıların Toplamları :32.5
```

7.5. Girilen Sayının Asal Sayı Mı Değil Mi Olduğunu Bulan Python Örneği

Python ile kullanıcının girdiği sayının asal sayı olup olmadığını kontrol ederek ekranda gösteren örnek. Örneğimizde kullanıcından bir sayı isteyerek 2'den bu sayıya kadar bir döngü kuracağız. Daha sonra bu döngü vasıtasıyla kullanıcının girmiş olduğu sayının tam bölünüp bölünmediğini kontrol ederek

eklediğimiz sayacı arttıracğız. Sayacın artması durumunda diğer sayılarla bölünüp bölünmediğine bakmadan break komutu ile döngüyü kıracağız. Programımıza ait kodlar ve ekran çıktısı aşağıdaki gibidir.

```
sayac = 0
girilenSayi = input('Sayı Giriniz: ')
for i in range(2, int(girilenSayi)):
    if(int(girilenSayi) % i == 0):
        sayac += 1
        break
if(sayac != 0):
    print("Girilen Sayı Asal Değil")
else:
    print("Girilen Sayı Asal")
```

Ekran Çıktısı:

```
Sayı Giriniz: 43
Girilen Sayı Asal
```

```
Sayı Giriniz: 22
Girilen Sayı Asal Değil
```

7.6. Girilen Vize ve Final Notu Ortalaması Hesaplayan Python Örneği

Vize ve Final notu girilen öğrencinin sene sonu ortalamasını hesaplayan ve ekranda gösteren Python örneği. Vize notunun %30 ile final notunun %70'i toplanarak elde edilen ortalama ekranda gösterilecektir.

```
vizeNotu = input('Vize Notunu Giriniz : ')
finalNotu = input('Final Notunu Giriniz : ')
ortalama = (float(vizeNotu) * 0.3) + (float(finalNotu) * 0.7)
print("Vize Final Ortalaması :{} ".format(ortalama))
```

Ecran Çıktısı:

```
Vize Notunu Giriniz : 85
Final Notunu Giriniz : 70
Vize Final Ortalaması :74.5
```

7.7. Girilen 3 Sınav Notunun Ortalamasını Bulan Python Örneği

Kullanıcı tarafından girilen 3 adet sınav notunu alarak bu sınav notlarının ortalamasını hesaplayan ve ekrana yazdırın Python örneği.

```
sinav1 = input('1. Sınav Notunu Giriniz : ')
sinav2 = input('2. Sınav Notunu Giriniz : ')
sinav3 = input('3. Sınav Notunu Giriniz : ')
ortalama = (float(sinav1) + float(sinav2) + float(sinav3)) / 3
print("Ortalama :{0} ".format(ortalama))
```

Ekrان Çıktısı:

```
1. Sınav Notunu Giriniz : 80
2. Sınav Notunu Giriniz : 70
3. Sınav Notunu Giriniz : 90
Ortalama :80.0
```

7.8. Kullanıcı Tarafından Girilmesi İstenen Sayının Tek ya da Çift Olduğunu Hesaplayan Örnek Program

Klavyeden girilen sayının tek mi çift mi olduğu, sayının ikiye bölümünden kalan değer ile anlaşılabilmekteidir. Girilen sayının 2'ye bölümünden kalan sıfır ise çift sayı, kalan bir ise tek sayıdır.

Mod(%) fonksiyonu: Bir bölme işleminin kalanını hesaplamak için kullanılır. Python'da % simbolü olarak kullanılır. Belirtilen iki sayının bölünmesinden kalanı bulur.

Örnekte klavyeden girilen sayının mod’unu hesaplayarak, girilen sayının tek mi çift mi olduğu ekrana yazdırılmaktadır.

Örnek: mod kullanımı

```
girilenSayi = input('Sayı Giriniz : ')
if(int(girilenSayi) % 2 == 0):
    print("Girilen Sayı Çift")
else:
    print("Girilen Sayı Tek")
```

Ekran Çıktısı:

```
Sayı Giriniz : 22
Girilen Sayı Çift
```

```
Sayı Giriniz : 11
Girilen Sayı Tek
```

Kullanıcı Tarafından Girilen Sayının Pozitif, Negatif, Ya Da Sıfır Olduğunu Bulan Python Örneği

Bir sayının negatiflik ya da pozitiflik durumu o sayının 0 sayısına göre durumunu ifade eder. Eğer sayı sıfırdan küçükse negatif, sıfırdan büyükse pozitif olarak ifade edilir. Sıfıra göre konum aldığımız için sıfır sayısının negatiflik ya da pozitiflik durumu yoktur.

Örnek: Sayının NEGATIF, POZİTİF yada SIFIR olduğunu yazdırın python programı.

Örnekte, kullanıcıdan string formatında bir sayı girmesi istenir. Kullanıcının girdiği sayı int türüne çevrilerek if else bloklarında kontrol edilir. Kontrol edilen sayı sıfırdan küçükse negatif, sıfırdan büyükse pozitif olarak ekrana yazdırılır.

```
girilenSayi = input('Sayı Giriniz : ')
if(int(girilenSayi) < 0):
    print("Girilen Sayı Negatif")
elif(int(girilenSayi) > 0):
    print("Girilen Sayı Pozitif")
else:
    print("Girilen Sayı Sıfır")
```

Ekran Çıktısı:

```
Sayı Giriniz : 5
Girilen Sayı Pozitif
```

```
Sayı Giriniz : 0
Girilen Sayı Sıfır
```

```
Sayı Giriniz : -7
Girilen Sayı Negatif
```

7.9. Girilen İki Sayıya ve Operatöre (+,-,*,/) Göre Toplama, Çıkarma, Çarpma ya da Bölme İşlemlerini Yapan; Bu Operatörler Dışında Bir Değer Girildiğinde “yanlış işlem girdiniz.” Uyarısı Veren Python Örneği.

```
sayi1=int(input("Birinci sayıyı girin: "))
sayi2=int(input("İkinci sayıyı girin: "))
islem=input("İşlem seçin (+,-,*,/): ")
if islem=="+":
    sonuc=sayi1+sayi2
    print(sonuc)
elif islem=="-":
    sonuc=sayi1-sayi2
    print(sonuc)
elif islem=="*":
    sonuc=sayi1*sayi2
    print(sonuc)
elif islem=="/":
    sonuc=sayi1/sayi2
    print(sonuc)
else:
    print("Yanlış işlem girdiniz")
```

Ekran Çıktısı:

```
Birinci sayıyı girin: 50
İkinci sayıyı girin: 10
İşlem seçin (+,-,*,/): +
60
```

7.10. Daha Önceden Belirlenmiş Bir Liste İçerisinden Rastgele Bir Sayı Seçen ve Ekranda Gösteren Python Örneği.

random.choice(list) fonksiyonu, bir listenin içerisinde rastgele bir eleman çekmeye yarar. choice(list) fonksiyonunu kullanılmak için “random” kütüphanesini eklememiz gerekmektedir. Aşağıdaki örnekte kullanım örneği mevcuttur.

Örnek: choice() fonksiyonu kullanımı

```
import random
liste=[1,2,3,4,5,6,7,8,9,10,11,12,
       13,14,15,16,17,18,19,20,21,
       22,23,24,25,26,27,28,29,30]
sayi=random.choice(liste)
print (sayi)
```

Ekran Çıktısı:

22

7.11. Python Girilen Sayının Faktoriyelini Hesaplayan Python Örneği

Bir sayının faktöriyeli, 1'den o sayıya kadar olan tüm tam sayıların çarpımıdır.

Örneğin, 6'nın faktöriyeli $1 * 2 * 3 * 4 * 5 * 6 = 720$ dir.
Faktöriyel negatif sayılar için tanımlanmamıştır ve sıfırın faktöriyeli 1(bir) dir., $0! = 1$.

```
sayi=int(input("Sayıyı Girin : "))
factorial = 1
if sayi < 0:
    print("Negatif sayıların faktöriyeli hesaplanamaz.")
elif sayi == 0:
    print("0! = 1")
else:
    for i in range(1,sayi + 1):
        factorial = factorial*i
    print(sayı," sayısının faktöriyeli : ",factorial)
```

Ekran Çıktısı:

```
Sayıyı Girin : 5
5  sayısının faktöriyeli :  120
```

7.12. Girilen Sayıya Kadar Olan Çift Sayıları Listeleyen Python Örneği

Kullanıcıdan sayı isteyerek For Döngüsü kullanarak girilen sayıya kadar olan çift sayıları ekranda gösteren Python örneği:

Örnek: range() fonksiyonu kullanımı

```
girilenSayi = input('Sayı Giriniz : ')
for i in range(1,int(girilenSayi)):
    if i % 2 == 0:
        print(i)
```

Ekran Çıktısı:

```
Sayı Giriniz : 20
2
4
6
8
10
12
14
16
18
```

7.13. 3×3 Bir Matrisin Eleman İndislerini Yazdırın Python Örneği

```
for i in range(0,3):
    for j in range(0,3):
        print([i,j])
```

Ekran Çıktısı:

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

7.14. Kenarları Girilen Dikdörtgenin Alanı ve Çevresini Bulan Python Örneği

Alan hesabı: Bir dikdörtgenin alanını hesaplamak için kısa kenar ile uzun kenarın çarpılması gereklidir.

Çevre hesabı: Bir dikdörtgenin çevresini hesaplamak için kısa ve uzun kenar toplanıp iki katı alınır.

```
kisaKenar = input('Kısa Kenar : ')
uzunKenar = input('Uzun Kenar : ')
dikdortgenAlani = int(kisaKenar) * int(uzunKenar)
dikdortgencEVresi = 2 * (int(kisaKenar) + int(uzunKenar))
print("Dikdörtgen Alanı : {}".format(dikdortgenAlani))
print("Dikdörtgen Çevresi : {}".format(dikdortgencEVresi))
```

Ekran Çıktısı:

```
Kısa Kenar : 5  
Uzun Kenar : 12  
Dikdörtgen Alanı : 60  
Dikdörtgen Çevresi : 34
```

7.15. Maaşı ve Zam Oranı Girilen İşçinin Zamlı Maaşını Hesaplayarak Ekranda Gösteren Python Örneği

Klavyeden maaşı ve zam oranı girilen kişinin zamlı maaşını hesaplayarak ekranda gösteren Python örneği.

```
yeniMaas = 0  
maasiniz = input("Maaşınızı Giriniz : ")  
zamOrani = input("Zam Oranınızı Giriniz(%) : ")  
yeniMaas = int(maasiniz) + (int(maasiniz) * int(zamOrani)) / 100  
print("Zamlı Maaşınız :", yeniMaas)
```

Ekran Çıktısı:

```
Maaşınızı Giriniz : 5000  
Zam Oranınızı Giriniz(%) : 3  
Zamlı Maaşınız : 5150.0
```

7.16. Sayı Tahmin Oyunu Python Örneği

Geliştirilen sayı tahmin uygulamasının çalışma mekanizmasını açıklamak gerekirse; Python programını yüklerken gelen standart kütüphanelerde tanımlı olan **random** modülü ile uygulamamız 0 ile 50 arasındaki bir sayıyı rastgele seçecektir. Programımızın rastgele seçtiği sayıyı bulmaya çalışırken aynı zamanda program bize geri dönüşler verecek. Tahmin ettiğimiz sayı ile programın rastgele seçtiği sayı arasındaki duruma göre yüksek veya düşük sayı girişini yaptınız şeklinde geri dönüşler olacaktır.

Python'da belirli işleri yaparken işimizi kolaylaştırmak için yazılmış fonksiyonlar vardır. Bu fonksiyonların bir arada durduğu yapıya **modül** denir.

Random modülü de bulunan birçok modülden yalnızca birisidir. Programımızı geliştirmeye random modülünü dahil ederek başlıyoruz. Python'da bir modülü dahil etmek için import ifadesi kullanılır.

Bir sonraki aşamada ise tahmin adında bir değişken tanımlıyoruz. Bu değişkenin karşısındaki ifade bize 0 ile 50 arasında rastgele tam sayı üretmesi için randint fonksiyonunu kullanıyoruz ve üretilen sayıyı tahmin değişkenine atıyoruz. Bir

alt satırda `dogruTahminMi` değişkeni tanımlıyoruz ve `False` olarak değer atıyoruz.

Program geliştirirken aynı kodları tekrar tekrar yazmak sıkıcıdır. Python'da bizi bu sıkıcılıktan kurtaracak ifadeler vardır. Bunlardan birisi de döngü yapılarıdır. Biz bu programı geliştirirken `while` döngü yapısını kullandık. `While` döngülerini bizim tanımladığımız şart tamamlanana kadar içinde bulunan kodları çalıştırır. Bizim örneğimizde ise doğru sayıyı tahmin edene kadar programımız bize tuttuğu sayıyı soracak. Biz `while` döngüsünün çalışmasına başlaması için `dogruTahminMi` değişkenine `False` değerini vermişik hatırlarsınız. Bu değeri vermemiş olsaydık `while` döngüsü çalışmayaacaktı. Kullanıcı doğru sayıyı girdiğinde ise `while` döngüsü tamamlanmış olacak. `dogruTahminMi` değişkeni `False` olduğu sürece altında bulunan kodları çalıştır demektedir. `While` döngüsünün içinde bulunan kodları inceleyeceğimiz bölümme geçelim.

Her programlama dilinde olduğu gibi Python'da da koşul deyimleri vardır. Bu koşul deyimleri sayesinde daha etkili programlar geliştirebiliriz. Python'da bulunan `if/elif/else` ifadeleri koşul deyimleridir. Gelin bunları bir örnek üzerinden anlatalım.

```
sayı=int(input('Bir sayı giriniz: '))
if sayı>0:
    print('Girdığınız sayı pozitif')
elif sayı == 0:
    print('Girdiniz sayı sıfırdır')
else:
    print('Girdığınız sayı negatif tam sayıdır')
```

Ekran Çıktısı:

```
Bir sayı giriniz: 25
Girdığınız sayı pozitif
```

İlk olarak kullanıcidan bir sayı girmesini istiyoruz ve bu sayıyı if/elif/else bloklarının içindeki deyimlere göre değerlendiriyoruz. If deyimi eğer anlamına gelir. Yani if deyiminin yanında bulunan koşul doğru ise, bizim örneğimizde $\text{sayı} > 0$ ifadesi, program altında bulunan kod parçalarını çalıştırır. If deyiminin yanında bulunan koşul yanlış ise program kodu okumaya devam eder ve bir alt satıra geçer. Elif deyimi ise if deyiminin çalışmadığı durumlarda yanında bulundan koşul doğru ise çalışır. Bizim örneğimizde elif deyimi kullanıcının girdiği sayının sıfır olup olmadığı durumu sorguluyor. Kullanıcı 0 girmiş ise kodumuz elif deyiminin bir alt satırında bulunan print() fonksiyonunu çalıştıracaktır. Kullanıcı sıfır veya sıfırdan büyük bir sayı girmediği zaman ise

`else` deyimi çalışacaktır. `Else` deyimi koşul almaz. `If` ve `elif` deyimlerinin çalışmadığı durumlarda çalışır. Örneğin kullanıcı negatif bir sayı girmiş olsun. O zaman programımız ‘Girdığınız sayı negatif bir tam sayıdır.’ çıktısını konsola yazdıracaktır.

Ekran Çıktısı:

```
Bir sayı giriniz: -15
Girdığınız sayı negatif tam sayıdır
```

Sayı tahmin oyununa donecek olursak, kullanıcı programın seçtiği sayıdan daha düşük bir sayı tahmin ederse ‘**Düşük Sayı Girişi Yaptınız. Daha Yüksek Bir Sayı Giriniz.**’ çıktısı olacaktır. Ancak kullanıcı programın seçtiği sayıdan daha yüksek bir sayı tahmin ederse bu durumda programımız ‘**Yüksek Bir Sayı Girişi Yaptınız. Daha Düşük Bir Sayı Girin**’ çıktısı verecektir. Kullanıcı doğru sayıyı tahmin ettiyse `else` deyiminin altında bulunan ‘**Doğru Tahmin.**’ Çıktısını ve Tahmin edilen sayı ile kaçinci seferde tahmin ettiğinin çıktısını olacaktır.

Python sayı tahmin oyununun kodunun yazımını genel olarak inceledik. Kodların tamamı ise aşağıdadır.

```
from random import randint
tahmin = randint(1, 50)
sayac = 0
dogruTahminMi=False
while (dogruTahminMi==False):
    sayac += 1
    girilenSayi = int(input("1 ile 50 arasında bir sayı giriniz. +\n    \"(Çıkmak İçin 0(sıfır) ) :\""))
    if(girilenSayi == 0):
        print("Oyundan Çıktınız.")
        break
    elif girilenSayi < tahmin:
        print("Düşük Sayı Girişi Yaptınız. Daha Yüksek Bir Sayı Giriniz.")
        continue
    elif girilenSayi > tahmin:
        print("Yüksek Bir Sayı Girişi Yaptınız. Daha Düşük Bir Sayı Girin.")
        continue
    else:
        dogruTahminMi=True
        print("Doğru Tahmin. Rastgele Tahmin Edilen Sayı {0}!".format(tahmin))
        print("Tahmin sayınız {0}".format(sayac))
```

Ekran Çıktısı:

```
1 ile 50 arasında bir sayı giriniz. (Çıkmak İçin 0(sıfır) ) :25
Düşük Sayı Girişi Yaptınız. Daha Yüksek Bir Sayı Giriniz.

1 ile 50 arasında bir sayı giriniz. (Çıkmak İçin 0(sıfır) ) :45
Yüksek Bir Sayı Girişi Yaptınız. Daha Düşük Bir Sayı Girin.

1 ile 50 arasında bir sayı giriniz. (Çıkmak İçin 0(sıfır) ) :35
Düşük Sayı Girişi Yaptınız. Daha Yüksek Bir Sayı Giriniz.

1 ile 50 arasında bir sayı giriniz. (Çıkmak İçin 0(sıfır) ) :40
Yüksek Bir Sayı Girişi Yaptınız. Daha Düşük Bir Sayı Girin.

1 ile 50 arasında bir sayı giriniz. (Çıkmak İçin 0(sıfır) ) :37
Düşük Sayı Girişi Yaptınız. Daha Yüksek Bir Sayı Giriniz.

1 ile 50 arasında bir sayı giriniz. (Çıkmak İçin 0(sıfır) ) :38
Doğru Tahmin. Rastgele Tahmin Edilen Sayı 38!
Tahmin sayınız 6
```

Else ifadesinin altında bulunan print() fonksiyonlarının içinde bulunan format ifadesi ilginizi çekmiştir. format() metodu ile içine aldığı parametreleri karakter dizisinin içinde bulunan süslü parantezlerin içine yerleştirir. Bizim örneğimizde kullanıcının girdiği ismi ve tahmin ettiği sayıyı karakter dizisinin içine yazdırır.

7.17. Girilen metin içerisinde, girilen bir karakterin olup olmadığını kontrol eden Python programı kodları.

Kullanıcı tarafından girilen bir metnin içerisinde, yine kullanıcı tarafından girilen karakterin olup olmadığını kontrol edip sonucu ekrana yazan Python örneği. Kontrol etme işlemi fonksiyon içinde yapılmıştır.

```
def kontrol(str,karakter):
    sayac = 0
    for ch in str:
        if ch ==karakter:
            sayac = sayac + 1
            return True
            break
metin=input('Metin : ')
karakter=input('Aranacak karakter: ')[0]
if(kontrol(metin,karakter)==True):
    print('{0} karakteri metin içinde var' .format(karakter))
else:
    print('{0} karakteri metin içinde yok'.format(karakter))
```

Ekran Çıktısı:

```
Metin : test metin içerik  
Aranacak karakter: e  
e karakteri metin içinde var
```

Ekran Çıktısı:

```
Metin : test metin içerik  
Aranacak karakter: a  
a karakteri metin içinde yok
```

7.18. Python Dizinindeki İstenen Dosyaları Listeleme

Örnek Python Kodları, (“C:\\OrnekData\\”) dizinin altındaki “.txt” ile biten tüm dosyaları aramak için “os.listdir” fonksiyonunu (işletim sistemi modülüne ait olan) kullanır.

For döngüsü bir eşleşme bulduğunda, append işlevini kullanarak, bulunan dosyanın adını “liste” listesine ekler.

```
import os  
dizindekiDosyalar = os.listdir("C:\\OrnekData\\")  
liste= []  
for dosya in dizindekiDosyalar:  
    if dosya.endswith(".txt"):  
        liste.append(dosya)  
print(liste)
```

Ekran Çıktısı:

```
[ 'Yeni Metin Belgesi (1).txt', 'Yeni Metin Belgesi (3).txt', 'Yeni Metin Belgesi (4).txt', 'Yeni Metin Belgesi (5).txt', 'Yeni Metin Belgesi (6).txt', 'Yeni Metin Belgesi (7).txt', 'Yeni Metin Belgesi (8).txt', 'Yeni Metin Belgesi (9).txt' ]
```

7.19. Klavyeden Girilen Yıl ve O Yılın Ayının Takvimini Ekranda Gösteren Python Kodu.

Kullanıcı tarafından girilen Yıl ve Ay bilgisini alıp, o ayın takvimini gösteren Python örneği;

Örnek: calendar kütüphanesi kullanımı

```
print("Oluşturmak istediğiniz yılı ve kaçinci ayı olduğunu giriniz")
yıl = int(input("Yılı Giriniz: "))
ay = int(input("Ayı Giriniz: "))
#Takvim modülünün kütüphanesini ekliyoruz.
import calendar
#Takvimi görüntülemek için kodu ekliyoruz
print(calendar.month(yıl, ay))
```

Ekran Çıktısı:

```
Oluşturmak istediğiniz yılı ve kaçinci ayı olduğunu giriniz
Yılı Giriniz: 2021
Ayı Giriniz: 7
July 2021
Mo Tu We Th Fr Sa Su
      1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

7.20. Serial Port üzerinden Veri göndererek Led Yakma/Söndürme Uygulaması

Kütüphanemiz kurulduktan sonra aşağıdaki örnek kodumuzu seri port üzerinden veri göndermek için kullanabiliriz.

```
import serial

SeriCon = serial.Serial(
    port='COM5', \
    baudrate=9600, \
    parity=serial.PARITY_NONE, \
    stopbits=serial.STOPBITS_ONE, \
    bytesize=serial.EIGHTBITS, \
    timeout=0)

while True:
    durum=input('Led Durumu 0 ya da 1 giriniz (2 çıkış): ')
    if (durum=="1"):
        print("Led Yakıldı")
        SeriCon.write(b"1")
    elif (durum=="0"):
        print("Led Söndürüldü")
        SeriCon.write(b"0")
    else:
        break

SeriCon.close()
```

Örneğimizde serial kütüphanesi import edildikten sonra SeriCon adında bir seri port tanımlaması yapılmıştır. ‘COM5’ olarak belirtilen port adı değişiklik gösterebilir. Veri yollayacağınız seri port numarasına bilgisayarınızın aygit yöneticisinde bulunan bağlantı noktaları kısmından bakabilirisiniz.

Yapılan örnekte kullanıcı tarafından girilen sayısal değere göre bilgisayarın COM5 portuna bağlı Arduino devresinde 13 numaralı porta bağlı LED ışığın yakılıp söndürülmesi sağlanmıştır. Örneğimizde “0” gönderilince led söndürülmüş, “1” gönderilince led yakılmıştır. 0 ve 1 dışında herhangi bir değer girilirse program sonlandırılmıştır. Dikkat edilmesi gereken bir husus seri port üzerinden verilerin byte türünden gönderilmesidir. Python kodlarımıza karşılık seri port üzerinde çalışan arduino uno kodları aşağıda verilmiştir.

```
void setup() {
    pinMode(13, OUTPUT);
    Serial.begin(9600);
    while (!Serial);
    digitalWrite(13, HIGH);
}
void loop() {
    if (Serial.available())
    {
        char veri=Serial.read();
        if (veri=='1')
        {
            digitalWrite(13, LOW);
        }
        else if (veri=='0')
        {
            digitalWrite(13, HIGH);
        }
    }
}
```

7.21. Serial Port üzerinden Veri Okuma Uygulaması

Bu örneğimizde bilgisayarımızın COM5 seri portuna bağlı arduino devresinde bulunan bir butonun basılı olup olmadığını ekrana yazdırılmış olacağız. Arduino devresi üzerine bağlı buton basılı olduğunda “buton basılı”, butona basılı olmadığına ise “buton basılı değil” verisini serial port üzerinden göndermektedir.

Yazdığımız python kodu ile serial port üzerinden gelen bilgiyi alarak ekrana yazdırma işlemini gerçekleştiriyoruz. Örnekte okunan verinin boş olup olmadığına veri!=b'' kod parçası ile karar verdik.

```
import serial

SeriCon = serial.Serial(
    port='COM5',
    baudrate=9600,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS,
    timeout=0)

while True:
    veri = SeriCon.readline()
    if(veri!=b''):
        print(veri)

SeriCon.close()
```

Python uygulamamıza veri gönderecek olan arduino uno devresinin kodları aşağıda verilmiştir.

```
int deger=0;
void setup() {
    pinMode(8, INPUT);
    Serial.begin(9600);
    while (!Serial);
}
void loop() {
    deger=digitalRead(8);
    if(deger==HIGH)
    {
        Serial.println("Buton basili");
    }
    else if(deger==LOW)
    {
        Serial.println("Buton basili degil");
    }
    delay(100);
}
```

KAYNAKÇA

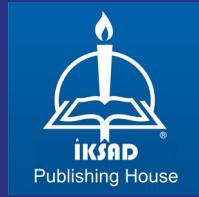
- aktif.net. <https://aktif.net/python-programlama-ve-uygulama-ornekleri/>. Erişim Tarihi: 10.09.2021
- beyaz.net. https://www.beyaz.net/tr/yazilim/makaleler/neden_python.html. Erişim Tarihi: 10.09.2021
- bilisimkonulari.com. <https://www.bilisimkonulari.com/python-turtle-ile-geometrik-sekiller-cizme.html>. Erişim Tarihi: 15.09.2021
- Ceder, N. (2018). The quick Python book. Simon and Schuster.
- djangogirls.org. https://tutorial.djangogirls.org/tr/python_installation/. Erişim Tarihi: 05.09.2021
- erdincuzun.com. <https://erdincuzun.com/python/3-degisken-tipleri/>. Erişim Tarihi: 11.09.2021
- Johansson, R., Johansson, R., & John, S. (2019). Numerical Python (Vol. 1). Apress.
- Kuhlman, D. (2009). A python book: Beginning python, advanced python, and python exercises (pp. 1-227). Lutz: Dave Kuhlman.
- medium.com. <https://medium.com/kodcular/python-ve-anacon-dakurulumu-b8931bd80e64>. Erişim Tarihi: 05.09.2021
- mobilhanem.com. <https://www.mobilhanem.com/python-don-guler/>. Erişim Tarihi: 15.09.2021
- msoguz.com. <https://www.msoguz.com/2019/06/python-saysal-veri-tipleri.html>. Erişim Tarihi: 12.09.2021
- Özgül, F. (2015). Her Yönüyle Python. Kodlab.

Severance, C. R. (2009). Python for everybody. Charles Severance.

Payne, J. (2010). Beginning Python: Using Python 2.6 and Python 3.1.
John Wiley & Sons Incorporated.

readthedocs.io <https://pyserial.readthedocs.io/en/latest/pyserial.html>.

Erişim Tarihi: 17.09.2021



ISBN: 978-625-8007-66-4