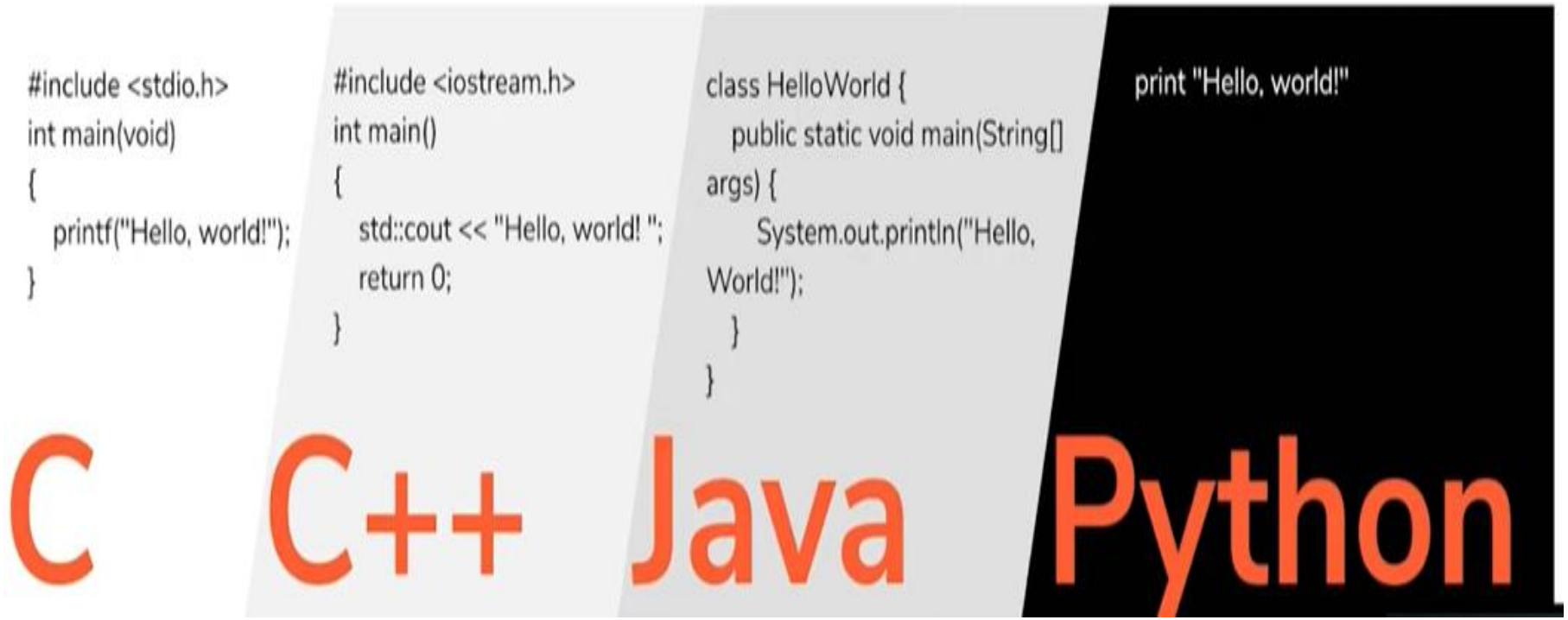
İleri Programlama Teknikleri

Doç.Dr.Derya AVCI

Giriş

- Python programlama dili 1990 yılında Guido van Rossum tarafından geliştirilen nesne tabanlı bir yazılım dilidir.
- Veri analizi, big data , derin öğrenme , makine öğrenmesi gibi alanlarda sıkça kullanılan , öğrenmesi diğer dillere nazaran daha basit olan bir dildir.
- C temelli
- Nesne yönelimli bir yazılımdır.
- Açık kaynak


#include <stdio.h>
int main(void)
{
printf("Hello, world!");
}
C
#include <iostream.h>
int main()
{
std::cout << "Hello, world! ";
return 0;
}
C++
class HelloWorld {
public static void main(String[]
args) {
System.out.println("Hello,
World!");
}
}
Java
print "Hello, world!"
Python

Python Kullanım Alanları Nelerdir?

- Python ile oyun kodlayabilirsiniz.
- Python ile mobil uygulama geliştirip yazabilirsiniz.
- **Python** ile veri analizi yapabilirsiniz.(excel, csv gibi dosyaları okuyabilirsiniz)
- Büyük verilerle hızlı işlemler yapabilirsiniz.
- Bazı yüksek matematiksel işlemleri kolaylıkla gerçekleştirebilirsiniz.
- Verileri görselleştirip, tablo ve grafiklerle yorumlayabilirsiniz.
- Hatta bir sudoku oyununun verilerini pythona okutup bu oyunu ona çözdürebilirsiniz.
- Çeşitli amaçlar için özelleşmiş python kütüphanelerini kullanarak (görüntü ses işleme, veri işleme ,yapay zeka ,makina öğrenmesi vb.alanlarda) bir çok işlevi kolaylıkla gerçekleştirebilirsiniz.

Python Nerelerde?

Kullananlar

- Google
- Yahoo
- Dropbox
- Reddit
- NASA
- CERN
- Wikipedia
- ILM
- ITA
- ...

Gömülü Ürünler

- 3dsMax
- Blender
- Cinema 4d
- Lightware
- Maya
- FreeCAD
- Nuke
- GIMP
- ...

İşletim Sistemleri

- Linux
- FreeBSD
- MacOS
- OpenBSD
- AmigaOS
- Anaconda

(Ubuntu, RedHat, Fedora)

- Portage

(Gentoo paket yönetim sistemi)

...

LibreOffice, Yapay Zeka, Raspberry Pİ,..

Kimler Kullanıyor

- Gömülü Sistemler
  - Aurdino, micropython
- Gerçek Dünya Uygulamaları
  - Google sürücüsüz araba
- Bilim Dünyası
  - Tensorflow, scipy
- Web
  - Instagram, Dropbox, Google
- Oyun/3B modelleme
  - Blender, Maya
- Standart Programlama Dili Olarak
  - Standford, mit

Neler Yapılır?

- Veri Analizi
- Bilimsel
- Sistem Yönetimi
- Web Uygulamaları
- Ağ ve Soket prog.
- Web Tarama (Örümcek)
- API oluşturma (REST)
- Machine Learning
- Yüz Tanıma
- Veri Madenciliği
- Parola Decrypt
- Oyun geliştirme
- Http Sunucu
- Yük Dengeleyici
- Plugin geliştirmek
- Modül ve Kütüphaneler

Ne Yapamazsınız?

- İşletim Sistemi yapamazsınız..

Pyhton Kütüphanesi Nedir?

- **Python** kütüphaneleri farklı fonksiyonları gerçekleştirebilen , yazılım geliştirme amaçlı kullanılan kaynaklarıdır.
- Python fazla sayıda kütüphaneye sahip çok fonksiyonlu bir yazılım dilidir.
- Her biri oldukça yetenekli olan bu kütüphaneleri kullanmak istediğinizde gerekli kodla çağrılarak işlem yapabilirsiniz.
- Birden fazla kütüphaneyi aynı anda çağırıp kullanmak da mümkündür.

IPython (interactive python)

- IPython bir python alt kabuğudur.
- **Python** ile yapılan tüm işlemleri IPython ile de gerçekleştirebilirsiniz.
- IPython geliştirilerek Jupyter notebook halini almıştır.

Pygame Kütüphanesi

- İsminden de anlaşılacağı gibi oyun üretiminde kullanılan kütüphanedir.
- Bu python kütüphanesi sayesinde interaktif oyunlar geliştirmek mümkündür.

Numpy Kütüphanesi (Numerical Python)

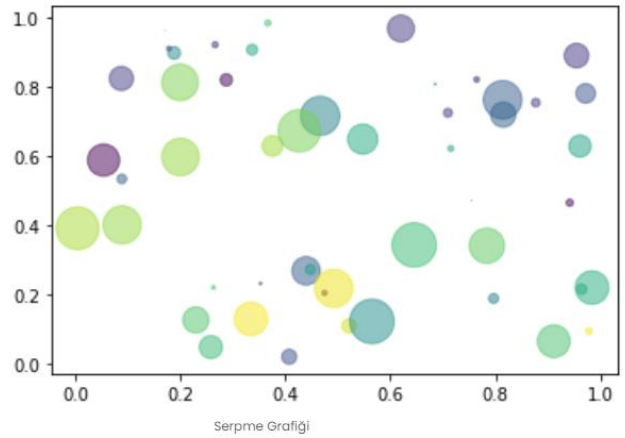
- Numpy pythonun meşhur kütüphanelerinden biridir.
- Math kütüphanesini içinde barındırır ve pythonun daha karmaşık matematiksel işlemleri de yapabilmesinde kullanılan kütüphanedir.
- Çok büyük matematiksel işlemleri hızlı ve esnek bir şekilde hesaplar.
- Veri analizi, veri madenciliği gibi alanlarda kullanılan bir python kütüphanesidir.

Matplotlib Kütüphanesi

- Data analizinde kullanılan bir kütüphanedir. Verileri farklı grafiksel şekillerde görselleştirip yorumlamayı kolaylaştırır.
-

```
In [4]: N=50
        x=np.random.rand(N)
        y=np.random.rand(N)
        colors=np.random.rand(N)
        area=np.pi*(15*np.random.rand(N))**2

        plt.scatter(x,y, s=area , c=colors ,alpha=0.5)
        plt.show()
```



Serpme Grafiği

Scrapy Kütüphanesi

- Web üzerindeki verileri içerikleri taramamızı sağlayan açık kaynaklı bir çerçevedir(python kütüphanesidir).
- Html ve Xml gibi yapısal içeriklerden verilerin ayıklanmasını sağlamaktadır.

Pytorch Kütüphanesi

- Grafik işlem birimlerini kullanan bir bilimsel bilgi-doğal işleme kütüphanesidir.

Caffe Kütüphanesi

- Uygulamalar oluşturmak için geliştirilmiş bir derin öğrenme çatısı (framework) kütüphanesidir.
- Sinir ağlarını metin ve kod yazmadan uygulamayı sağlar.

CatBoost Kütüphanesi

- Sınıflama ve regresyon kullanılan yüksek performanslı bir kütüphanedir.
- Makaine öğrenmesi biliminde yararlanılır.

Pybrain Kütüphanesi

- Modüler bir makine öğrenme kütüphanesidir.
- Amacı, makine öğrenimi görevleri için esnek, kullanımı kolay ancak hâlâ güçlü algoritmalarla algoritmalarınızı test etmek ve karşılaştırmak için çeşitli önceden tanımlanmış ortamlar sunmak olan bir python kütüphanesidir.

XGBoost Kütüphanesi

- XGBoost, yüksek verimli, esnek ve taşınabilir olarak tasarlanmış bir kütüphanedir.
- Optimum dağılım makine öğrenmesi alanında kullanılan bir python kütüphanesidir.

OpenCV Kütüphanesi

- Görüntü işleme için kullanılan en popüler python kütüphanelerinden biridir.
- Çok yaygın olarak kullanılan bu kütüphanenin kullanıcıları arasından Microsoft ,Yahoo, Google gibi büyük şirketler de vardır.
- Her hangi bir görüntü işleme değiştirme tanıma ve benzeri işler için fonksiyonlar barındıran güzel bir python kütüphanesidir.

Seaborn Kütüphanesi

- İstatiksel hesaplamalara farklı bir bakış açısı kazandırmak için kullanılan oldukça efektif ve bir o kadar popüler olan bir veri görselleştirme kütüphanesidir.

Speech Recognition Kütüphanesi

- Ses tanıma sistemlerinde kullanılan kütüphanedir. Apple siri, google asistan gibi uygulamalar bu ses tanıma teknolojilerinden yardım alarak oluşturulmuş bir python kütüphanesidir.

Spacy Kütüphanesi

- Doğal dil işleme için oluşturulmuş bir kütüphanedir. Uber, microsoft gibi büyük şirketlerin alt yapısında kullanılan bir python kütüphanesidir.

Bokeh Kütüphanesi

- Kolay ve hızlı bir şekilde verilerinizi görselleştirmeye yarayan bir python kütüphanesidir. **Matplotlib** ve seaborn kütüphanelerinin aksine verileri görselleştirirken Html ve Javascript kullanır.

NLTK Kütüphanesi

- Natural Language toolkit yani doğal dil işleme araç seti ismi verilen kütüphane sembolik matematik ve istatistiksel analizler için doğal dil işleme çözümleri sunan bir python kütüphanesidir. Bazı mesaj uygulamalarında konuştuklarımızı yazıya dökme (speech to text) teknolojilerinde kullanılmaktadır.

Tensorflow Kütüphanesi

- Bir dizi görev arasında veri akışı ve türevlenebilir programlama için kullanılan bir kütüphanedir. Ağırlıklı olarak sembolik matematik işlemleri ve sinir ağları gibi makine öğrenimi uygulamaları için kullanılır. Google'da hem araştırma hem de üretim için önemli bir yeri vardır.

Plotly Kütüphanesi

- Javascript, R gibi dillerle uyumlu çalışan veri görselleştirme ve dashboard oluşturmaya yarayan fonksiyonları barındıran bir python kütüphanesidir.

Theano Kütüphanesi

- Matematiksel ifadeleri, özellikle matris değerli ifadeleri işlemek ve değerlendirmek için optimize eden bir python kütüphanesidir.
- İleri düzey matematiksel araştırma ve çalışmalar (Etkili sembolik türev alma, Hız ve kararlılık optimizasyonları vb.) için sıkça kullanılmaktadır.

Beautiful Soup Kütüphanesi

- **HTML, XML belgelerini işlemek için** bu kütüphaneden yararlanılır. İlginç olan kısmı kütüphanenin ismini Alice Harikalar Diyarındaki kaplumbağanın söylediği şarkıdan aldığıdır.

SciKit-Learn Kütüphanesi

- Veri madenciliği ve veri analizinde gerekli küme analizi, regresyon, veri işleme gibi işlemleri gerçekleştirebilen çok yönlü bir kütüphanedir. Makine öğrenmesi ve Python deyince akla gelen ilk kütüphanedir Scikit-learn.

StatsModels Kütüphanesi

- İstatistiksel modelleri tahmin edip istatistiksel testler yapmayı sağlan bir python kütüphanesidir.
- Farklı veri türleri işlemesi, geniş tanımlayıcı istatistik yapısı, çizim fonksiyonu ve sonuç istatistikleri listesi gibi çözümlerinden dolayı çokça tercih edilmektedir.

Pillow Kütüphanesi

- **Pythonun açık kaynak kodlu görsel kütüphanesidir**. Grafik işlemleri için özelleşmiş hazır fonksiyonları sayesinde kullanıcıya üstün grafik işleme imkanları sunar.

- Grafik işlemeden kasıt özelleşmiş çizimler, boyutlandırma ölçeklendirme işlemleri, renk değerlerini düzenleme vb. olarak neitelendirilebilir.

MpMath Kütüphanesi

- Middle point mathematic yani orta nokta matematiği olarak isimlendirilen kütüphane adından anlaşılabileceği gibi Pythonun çeşitli matematiksel fonksiyonları barındıran kütüphanesidir.
- Reel ve kompleks sayılarla hassas hesaplamalar yapmaya olanak tanır.

Gensim Kütüphanesi

- Modern istatistiksel makine öğrenimi kullanarak modelleme , doğal dil işleme için oluşturulmuş açık kaynaklı bir kütüphanedir.

Requests Kütüphanesi

- Web üzerindeki isteklerinizi yönetmenizi sağlayan bir ***Python*** **http kütüphanesidir.**

Pyglet Kütüphanesi

- Oyun ve multimedya uygulamaların üretilmesinde kullanılan , resim, müzik, video derlemeyi destekleyen bir kütüphanedir.

Anaconda

- python kütüphanelerini içerisinde yüklü olarak barındıran ayrıca R istatistiksel kodlama programı,Orange,... gibi programları da içinde bulunduran bir programdır. Kodlamaya yeni giriş yapmış veya pythona başlamak isteyenler için ideal ve kullanışlı bir uygulamadır. Ayrıca ücretsiz ve kullanımı kolaydır.
- Eğer diğer uygulamalarda çalışıyorsanız bu python kütüphanelerini kullanabilmek için öncelikle kütüphanenin yüklü olması gerekir yüklü değilse;
- **pip install kütüphane adı**
- İfadesini ekrana yazıp run ederek yükleyebilirsiniz. Anaconda programı kullanıyorsanız kütüphaneler yüklü olarak programda mevcuttur.Yüklemeye gerek yoktur.

Python Programlama Dili

Doç.Dr. Derya AVCI

Genel özellikler

- Küme parantezi
- Begin-end benzeri ifadeler
- satır sonu ;
- Değişken tipi tanımlama yoktur
- Girintileme kuralı vardır
- Belirli bir blok’a ait kodlar girintilime yapılarak yazılmalıdır

Python Etkileşimli Kabuk

Python yüklü ise (veya **Anaconda** ortamı)

Komut satırından python denerek Shell çalıştırılır. **>>>**

Online python kabuğu https://www.python.org/shell/

```
>>> a=2    # Değişken tanımlama
>>> a**2  # Karesini alma
>>> dir(a) # Herhangi değişken yada metotla ilgili tüm seçenekler
>>> a.bit_length() # Değişkenin kaç bit kapladığı
>>> a.__abs__() # Sayının mutlak değerini alma

>>> type (değişken) # Değişkenin tipi nedir ?
```

Girintileme Kuralı

Metot/Şart/Döngü Kodlarından sonra (blok diye adlandıralım) : koymak girintileme yapmak kuraldır (editör ortamında tab tuşu ile)

Ana Program

Blok İfadesi:

    Bloka ait kodlar

    …

Ana Program Akışı

Yorum satırları

ile başlayarak tek satırlık yorum satırları oluşturulabilir.

Birden fazla satırı yorum satırı haline getirmek için

```
" " "
 Yorum satırı 1
 Yorum satırı 2
 devam...
" " "
```

Aritmetik İşleçler

+

-

*

**

/

//

%

+=

-=

*=

/=

Karşılaştırma İşleçleri

==	eşittir
!=	eşit değildir
>	büyüktür
<	küçüktür
>=	büyük eşittir
<=	küçük eşittir

Bool İşleçler

İki yada daha fazla şartı birleştirme

and

or

not

"p" in "Python"

a is 256

id(değişken) # Nesnenin bellekteki adresini gösterir

Şart İfadeleri

if *boolean_ifade*:

    if şartına ait kodlar

if boolean_ifade:

    şart ifadesi doğru ise

else:

    şart ifadesi yanlış ise

if-elif-if

```
if şart:
    şart1 doğru ise
elif şart:
    şart2 doğru ise
elif şart:
    şart3 doğru ise

…

else:
    default kısım
```

if-else örneği

```
a=2
b=3
if a<b:
        print "a b'den küçük"
else:
        print "a b'den büyük"
```

if-elif-else örneği

```
puan=36
if puan>85:
        print "A "
elif puan>70:
        print "B "
elif puan>50:
        print "C "
elif puan>40:
        print "D "
else:
        print "F "
print " \n "
```

Python’da Değişkenler

- **Python Değişkenleri** üzerinde veri depolamak için tanımladığımız sembolik isimler olarak adlandırabiliriz.

- Projelerimiz için oluşturduğumuz yapılarda ihtiyaç duyacağımız verileri birbirinden ayırt edebilmek için kullanırız. Çoğu programlama dilinin aksine tanımlayacağımız değişken yapısının veri tipini belirtmemize gerek yoktur.

- Bizim yerimize **python yorumlayıcısı** (interpreter) tanımlamış olduğumuz değişkenin veri tipini algılıyor ve ona göre işlemlerimizi gerçekleştirmemize yardımcı oluyor.

Python Değişken Tanımlama Nasıl Yapılır?

- Python projemizde değişken tanımlamak istediğimiz zaman vereceğimiz isimde **sayı** ile başlayamaz, **boşluk** bırakamaz ve **özel karakter** kullanamayız.
- Değişken tanımlarken Türkçe karakter kullanılmaz.

- Python diline ait özel anahtar kelimeler (**keyword**) yapıları kullanılarak değişken ismi belirtilemez.

- İsimlendirme esnasında küçük harfle başlanarak isimlendirilme yapılması önerilir.

Veri Tipleri

- Temel tipler (integer, string, boolean,double)

- Diğer veri tipleri
  - Liste (List)
  - Demet (Tuple)
  - Küme (Set)
  - Sözlük (Dictionary)

String (Metin) Veri Tipleri

- Değişkenlerimiz üzerinde **char** yapıların birleşmesinden meydana gelen **metinsel** verileri saklamak istediğimiz zaman "**String**" veri tipine ihtiyaç duyarız.

```
1.  strDegisken = "String Bir Metin"
2.  pythonString = "Merhaba Python!"
3.  # Değişkenimiz üzerinde metinsel bir karakter sakalyacağımız zaman çift tırnak kullanarak
tanımlamasını gerçekleştiririz.
```

Numerik (Sayısal) Veri Tipleri

Değişkenlerimiz üzerinde numerik bir veri saklamak istediğimiz zaman **dört adet** sayı tipi ile tanımlama gerçekleştirebiliriz. Bu sayı tipleri şu şekildedir:

- **Integer:** Belek üzerinde 4 Byte yer kaplar ve 32 bittir. İçerisinde -2^{31} ile $2^{31}-1$ arasında yer alan tamsayı değerlerini barındırır.
- **Long:** Bu veri tipinin uzunluğu 64 bittir. Tamsayı türünde 2^{63} ile $2^{63}-1$ arasında değerler tanımlanabilir.
- **Float:** Uzunluğu 32 bittir. Ondalıklı sayı türünde $-3.4*10^{38}$ ile $3.4*10^{38}$ arasında değerler alır.
- **Complex:** Karmaşık sayılar olarak bilinen bu tipler diğer veri tiplerinden daha büyük sayıları içerisinde barındırır. Bu değerler o kadar büyüktür ki iki parçadan oluşur. Reel (**gerçek**) ve imajiner (**sanal**) isimli iki kısımdan oluşur.

Veri Tipleri

```
1.  integerDegisken = 100
2.  # Değişkenimize 'Integer' türde veri tanımlaması gerçekleştirilmiştir.
3.  longDegisken = 535632149861
4.  # Değişkenimize 'Long' türünde veri tanımlaması gerçekleştirdik.
5.  floatDegisken = 128.65
6.  # Değişkenimize ondalıklı sayı eklemek için 'Float' türünde veri tanımlaması gerçekleştirdik.
7.  complexDegisken = 42j
8.  #Değişkenimize oldukça büyük bir sayı eklemek için 'Complex' türde veri tanımlaması gerçekleştirdik.
```

Listeler(List)

- Bir liste birden çok string yada sayı sabitini belirli bir sırada barındıran değişkenlerden veya sabitlerden oluşur ve oluştururken [] (köşeli parantez) ifadesi kullanılır.
- Liste içerisindeki elemanların indeks numarası 0 (sıfır) ile başlar.
- Listenin elemanlarına ulaşmak için *liste[indeks numarası]* şeklinde bir yazım kullanılırız.
- Dikkat edilmesi gereken diğer nokta ise indeks değerinin mutlaka bir tam sayı olması zorunluluğudur.
- Farklı nesneleri tanımlamakta görev yapar.
- İçeriği değiştirilebilir.

Listeler

```
a=[1,2,"a",3,5]        # Veriler karışık türden olabilir, içeriği değiştirilebilir.

len(a)                 # Eleman sayısı
a.sort()               # listeyi sıralar
a.reverse()            # listeyi ters çevirir
a.pop()                # son elemanı siler
a.append("a")          # sonuna yeni eleman ekler
a.insert(indis, "a")   # yeni elemanı belirtilen indise ekler
```

Listeler devam

```
a.count(1)        # Bu eleman listede kaç tane var
a.index(1)        # Bu eleman kaçıncı indiste

print a[1]        # 1.indiste ki elemanı yazdır
a[1]=2            # 1.indisteki elemanın değerini değiştir
del a[2]          # 2.indisteki elemanı listeden sil

x=list()          # Boş liste oluşturur
x=[]              # Boş liste oluşturur
```

En Çok Kullanılan Liste Fonksiyonları

- **count ():** Listede bir elemanın kaç defa tekrarlandığını verir
  - **programlar.count('python')**

- **extend ():** İki listeyi toplar.
  - **programlar.extend(['Java'])**

- **index():** İstenilen bir elemanın liste içindeki indeksini verir.
  - **programlar.index('istenen değer')**

- **pop():** Listenin son elemanını çıkartır.
  - **programlar.pop()**
- **remove ():** Herhangi bir elemanı listeden çıkartmak için kullanılır.
  - **Programlar.remove('Java')**

- **reverse():** Listeyi tersten yazdırır.
- **sort():** Liste elamanlarını sıralamak için kullanılır.
- **len():** Listenin kaç elemandan oluştuğunu bulur.
  - **len(programlar)**

Listeler Devam

```
>>> dir(a)              # a ile başka ne yapabilirim
```

Kabukta belirli bir fonksiyonla ilgili yardım alma

```
>>> help(a.append)      # a.append nasıl çalışır
```

Demetler(Tuple)

Listelere benzerler!

a=(1,2,"a")

a.count(1) # 1 elemanı kaç tane var

a.index(2) # Bu elemanın index'i

print a[1] # 1.elemanını yazdır

Demetler tanımlandıktan sonra güncellenemezler!

Normal parantez içinde () gösterilirler.

Kümeler(set)

a={1,2,"a",5}

#Bu methodları desteklerler
add
revome
pop

#Ancak aşağıdaki methodlar çalışmaz, kümeler sıralıdır ve çift değer içermezler
a.index
a[1]
a.count(5) ?

Sözlükler(Dictionary)

```
x={"isim":"ali", "meslek":"muhendis","maas":1000, "ehliyet":True}
```

{key:value, key:value, key:value …}

```
print x[key]=value        # Değeri yazdırma
x[key]=new_value        # Değeri değiştirme
x[new_key]=value        # Yeni key:value çifti ekleme

x.keys()                    # x'in anahtarları
x.values()                  # x'in değerleri
```

Döngüler

For Döngüsü Yapısı:

```
for i in range(a,b,c):
    print i
```

a’dan b’ye (a dahil b değil), c artımlı döngü

range(10): 0…9

range(2,10): 2 3 4 5 6 7 8 9

range(2,10,3): 2 5 8

For Döngüsü

break

döngüyü kırıp bitirir, iç içe döngülerde sadece ait olduğu iç döngüyü bitirir

continue

döngüyü pas geçer (bir sonraki adımdan devam eder)

Liste, Demet, Set Üzerinde Döngüler

```
list=[1,2,5,10]
for i in range(0,len(list)):
        print list[i]
```

Python versiyonu

```
list=[1,2,5,10]
for i in list:
        print i
```

Sözlük Üzerinde Döngüler

```
x={"isim":"ali", "meslek":"muhendis","maas":1000, "ehliyet":True}

for (k,v) in x.items():
        print k,":",v
```

k: sözlük anahtarlarını

v: sözlük değerlerini simgeliyor

While Döngüsü

while *şart_ifadesi*:

    kodlar

Dikkat:

Blok içerisinde döngü artımı/döngüden çıkış şartı olmazsa sonsuz döngü yapmış oluruz

Metotlar

```
def metot_ismi(parametre_listesi):
        metot_kodları
            return değer
```

Metotlar devam

```
def hesapla(a,b):
    x=a*b
    return x


print hesapla(3,4)
```

Metotlar devam

```
def hesapla(a=2,b=3):
    x=a*b
    return x


print hesapla()          # Sonuç 6 olacak
print hesapla(5)        #  Sonuç ?
# Metot parametresiz çağrılırsa default değerler alınır
```

Metotlara belirsiz sayıda parametre göndermek

```
def hesapla(*liste):
        t=0
        for i in liste:
                t+=i
        return t


print hesapla(1,2,3,4,5,6)
```

Modul Kullanımı

Moduller kütüphanelerdir

```
import modül_ismi        # Modül programa dahil edilmiş olur
modül_ismi.method        # Modülün methodunu kullanmak

import string             # String modülü
import random             # Random modülü
```

Random Modülüne Giriş

```
random.randint(a,b)        # a-b aralığında bir tam sayı tutar
random.random()            # 0-1 aralığında bir rasyonel sayı tutar
```

```
x=[1,2,5,10]
print random.choice(x)          # x listesinden rastgele bir eleman seçer
print random.sapmle(x,3)        # x listesinden rastgele üç eleman seçer
```

String İfadeler

```
s="Merhaba Python"
print s
```

String ifadeler liste veri yapısına çok benzerler, örneğin len(s) karakter sayısını verir

```
s.upper()                 # Büyük harfe dönüştür
s.lower()                 # Küçük harfe dönüştür
s.count("python")    # python kelimesi kaç defa geçiyor
```

```
s[3].isupper()          # Stringin 3.indisteki karakteri büyük harf mi ?
s[3].islower()          # Stringin 3.indisteki karakteri küçük harf mi ?
s[7].isdigit()          # Stringin 7.indisteki karakteri rakam mı ?
```

```
x=s.split("")                                  # Boşluk karakterine göre ayırarak listeye atar
s1="".join(random.sample(s,3))          # s Stringinden rastgele 3 karakteri al s1 Stringinde birleştir
```

join ve split birbirlerinin tam tersi iş yaparlar

Ödev

1) Bir online sistem için 100 adet kullanıcıya 8 karakterden oluşan şifre belirlenecektir. Oluşturulan şifreler bir listeye atılacak
   - Şifre en az bir tane
     - büyük harf
     - (! % ? * #) özel karakter
     - rakam
     - küçük harf içermelidir

2) Yazacağınız 2.bir program 100 adet şifre içinde yan yana 2+ rakam içeren şifreleri bulup göstererek hata mesajı versin

3) Sayısal loto: program metoduna parametre olarak 6 tane sayı (tahmin edilen sayılar) gönderilecek, metot 1-49 arasında 6 adet sayı tutacak, kaç tane tahminin tuttuğunu yazacak

Not: ödevde **regex** yada benzeri kütüphane kullanılmayacak! random modülü kullanılabilir.

İleri Programlama Teknikleri

Doç.Dr. Derya AVCI

Python Ekrana Yazdırma

- Python programlama dilinde diğer dillerde olduğu gibi kullanıcı tarafından sonuçların görüntülenmesi gerekmektedir.
- Python programlama dilinde kullanıcıların ekranda sonuçları görebilmesi için kullanılan ekrana yazdırma komutu **print** komutudur.
- Bu komut ile **konsol** üzerinde verileri gösterme işlemi yapılmaktadır.

Print Fonksiyonu Kullanımı:

- **1-** Herhangi bir metni yazmak için aşağıdaki yöntemler kullanılır.

```
print("Ekrana Yazdırılacak Metin Girilir")

#veya

print('Ekrana Yazdırılacak Metin Girilir')

#veya

print("""Ekrana Yazdırılacak Metin Girilir""")

#yukarıdaki üç komut aynı işlemi yapmaktadır.
```

- **2-** Herhangi bir değişkeni ekrana yazdırmak için aşağıdaki yöntem kullanılır.

Değişkenler ekrana yazdırılırken tırnak işareti kullanılmaz. Değişken ismi parantez içerisine aynen yazılmaktadır.

```
sayi=3
print(sayi)
```

- **Örneğin:** Birden fazla elemanı print paremetresi kullanarak yan yana yazdıralım.

```
print("Ocak","Şubat","Mart","Nisan","Mayıs","Haziran","Temmuz","Ağustos","Eylül","Ekim","Kasım","Aralık")

""" Ekrana

Ocak Şubat Mart Nisan Mayıs Haziran Temmuz Ağustos Eylül Ekim Kasım Aralık
```

Değişken ile Belirli Bir Metni Ekrana Yazdırma

- Değişken ile herhangi bir metni ekrana yazdırmak için **, (virgül)** işareti kullanılmaktadır. Örneğin: Kullanıcıdan bir sayı istedik ve bu sayının ne olduğunu "Girilen sayı = 5 " şeklinde göstermek için kullanılmaktadır.

```
sayi=5
print("Girilen Sayı= ",sayi)

#Ekrana
#Girilen Sayı= 5
#yazacaktır.
```

Print Fonksiyonu ile Alt Satıra Geçme

- Print fonksiyonu içinde alt satıra geçmek için **\n** kullanılmaktadır. En çok kullanılan parametrelerden biridir.

```
print("Madde 1 \nMadde 2 \nMadde 3")

""" Alt alta

Madde 1
Madde 2
Madde 3

yazacaktır """
```

Print Fonksiyonu End Parametresi

- Print fonksiyonu içerisinde end parametresini bulundurmaktadır.
- Varsayılan olarak bulundurulan end parametresi alt satıra geçme işlemi için kullanılmaktadır.
- Yani hiçbir değer yazmasak bile bizim görmediğimiz bir end parametresi bulunmakta ve alt satıra geçme işlemi yapmaktadır.
- Biz end parametresini kullanarak varsayılan değeri değiştirebiliriz. Örneğin boşluk bırakarak yazıları yan yana yazdırma işlemi yapabiliriz.

```
print("Merhaba",end=" ")
print("Azkod.com")

#Ekrana Merhaba Azkod.com yazacaktır.

#end parametresini kullanmadan çalıştırsaydık aşağıdaki örnekteki gibi olacaktı.

print("Merhaba")
print("Azkod.com")

""" Ekrana

Merhaba
Azkod.com

yazacaktı. Biz alt alta yazmasını engelledik yan yana yazmasını sağladık.
"""
```

Sep Parametresi

- Print fonksiyonu içerisinde birden fazla eleman olabileceği ayları yazdırılarak gösterilmişti. Bu parametre ile birden fazla elemanı yazdırırken aralarına istediğimiz karakteri koyma işlemi yapabileceğiz.
- Örneğin yukarıdaki örneğimizde ayları yazdırdığımızda aralarında birer boşluk atarak ekrana yazma işlemi yapmıştı. Biz aralarına virgül veya istediğimiz herhangi bir karakteri yazdırmak ister isek sep parametresini kullanırız.

```
print("Ocak","Şubat","Mart","Nisan","Mayıs","Haziran","Temmuz","Ağustos","Eylül","Ekim","Kasım","Aralık",sep=",")

""" Ekrana

Ocak,Şubat,Mart,Nisan,Mayıs,Haziran,Temmuz,Ağustos,Eylül,Ekim,Kasım,Aralık

şeklinde yazacaktır."""
```

Format Metodu

- Format metodu **biçimli** olarak yazdırmak için kullanılmaktadır. Daha doğrusu yazdırma işlemlerinde **karmaşayı önlemek** içinde kullanılır.
- Formatlı kullanımda ekrana yazdırılacak değerler **{0}** 'dan başlayarak artarak eleman sayısına göre devam etmektedir. Çift tırnaktan sonra **.format** yazılır ve içerisine sırasıyla ya değişken ya da değerler girilerek **0. indisten** itibaren başlayarak değerleri yazma işlemi yapar.

```
print("Birinci sayı= {0} İkinci Sayı= {1}".format(2,5))

#Ekrana
#Birinci sayı= 2 İkinci Sayı= 5
#yazacaktır.

""" Bu kodun format kullanmadan halini yazalım """

print("Birinci sayı=",2,"İkinci Sayı=",5)
```

print komutu ve kullanımı

In [7]:
```
print("Mehaba \nYakin Kampüs")
print("Benim adim \tMesut")
```

```
Mehaba
Yakin Kampüs
Benim adim      Mesut
```

In [13]:
```
print("Benim adim {}".format('Mesut'))
print("Benim adim {}, yasim {}".format('Mesut', 32))

print("Benim adim {0}, yasim {1}".format('Mesut', 32))
print("Benim adim {1}, yasim {0}".format('Mesut', 32))

print("Benim adim {ad}, yasim {yas}".format(ad='Mesut', yas=32))
print("Benim adim {ad}, yasim {yas}".format(yas=32, ad='Mesut'))
```

```
Benim adim Mesut
Benim adim Mesut, yasim 32
Benim adim Mesut, yasim 32
Benim adim 32, yasim Mesut
Benim adim Mesut, yasim 32
Benim adim Mesut, yasim 32
```

Python Veri Girişi

- **Python** programlama dilinde tüm programlama dillerinde olduğu gibi **veri girişi** kullanıcıyı etkin kılmak ve kullanıcı ile etkileşim halinde olmak için veri girişi yapılabilmektedir.
- Python programlama dilinde veri girişi **input()** fonksiyonu ile yapılmaktadır. Bu fonksiyon ile kullanıcıdan veri alırken aldığımız veri bir değişkene atanmak zorundadır.
- İnput fonksiyonu kullanıcıdan **metinsel** ve **sayısal** veriler almak için kullanılabilmektedir.

İnput Fonksiyonu Kullanımı

- İnput fonksiyonunu kullanırken bir değişkene atanması gerekir.

```
isim=input("Lütfen İsminizi Giriniz")

print("Hoşgeldin ",isim)
```

Python Sayısal Veri Girişi

- İnput fonksiyonuna herhangi bir işlem yapmaz isek metinsel veri girişi şeklinde alacaktır.
- Biz metinsel verilerin türünü int() fonksiyonu kullanarak sayısal veri türüne çevireceğiz. Burada kullanıcıdan veriyi alırken input() fonksiyonu'nu int() fonksiyonu içerisinde alarak kullanacağız ve aldığımız veri sayısal ifadeye dönüşmüş olacak.

```
degiskenIsmi=int(input("Lütfen sayısal veri giriniz"))
```

- Örneğin:

Aşağıdaki örnekte iki sayı girdirilmesi istenmektedir. **Örneğin**: Kullanıcı sayıyı:**10**, 2. sayıyı **20** girerse ekrana **Sayıların Toplamı=30** yazacaktır.

```
birinciSayi=int(input("Lütfen Birinci Sayıyı Giriniz"))
ikinciSayi=int(input("Lütfen Birinci Sayıyı Giriniz"))

sayilarinToplami=birinciSayi+ikinciSayi

print("Sayıların Toplamı=",sayilarinToplami)
```

Python Programlama Dili

```
"""
a*b işlemini döngü ile yapmak
"""
a=7
b=5
c=0
for x in range(a):
    c+=b

Print(c)
```

```
"""
Bu program verilen n sayısına göre
f=1+2+4+...+2^n serisini hesaplar
"""


f=0
n=3
for i in range (n):
    f+=2**i
    print (f)

Print("sonuc=",f)
```

```
"""
Bu program verilen n sayısına göre
f=1/1+1/2+1/4+...+1/(2^n) serisini hesaplar
f toplamında 1./(2**i) işlemde yuvarlatma yapılmaması
double olarak işlem yapması içindir
"""


f=0.0
n=10
for i in range (n):
    f+=1./(2**i)
    print(f)

print ("sonuc=",f)
```

```
"""
ilk 100 fibonacci sayısını yazdırma
"""
a=1
b=1 #sıradaki fibonacci sayısı
c=1
for x in range(100):
    a,b=b,c  #çoklu atama
    print(c)
    c=a+b
```

```
liste=[3,1,"b",5,"a",10,2,3,5]    #Tamamen dinamiktirler
demet=(3,1,5,"a",10,2,3,5)         #Sonradan değiştirilemezler
kume={3,1,5,10,2,3,5}              #Çift eleman içermezler


liste.pop()     #Son elemanı çek
liste.append(20)  #20 yi ekle
liste.sort()    #Listeyi sırala

# Liste üzerinde döngü
print ("Liste")
for i in liste:
    print (i)
```

```
demet=(3,1,5,"a",10,2,3,5)        #Sonradan değiştirilemezler
# demet.append(3) Hata verecektir çünkü demetlerde günceleme

# Demet üzerinde döngü
print ("Demet")
for i in demet:
    print (i)
```

```
kume={3,1,5,10,2,3,5}              #Çift eleman içermezler
#Kümeler üzerinde güncelleme yapabiliriz ama çift değer içermezler
kume.pop()
kume.add("b")   #Kümeye add ile eleman ekliyoruz

# Küme üzerinde döngü
print ("Küme")
for i in kume:
    print( i)
```

Veri Tipleri

- Listeler

  [] # Dinamiktirler
- Demetler

  () # Güncelleme yapılmıyor
- Kümeler

  {} # İkili değer içermezler
- Sözlükler

  {key:value}

Listeler

```
a=[1,2,"a",3,5]        # Veriler karışık türden olabilir

len(a)                  # Eleman sayısı
a.sort()                # listeyi sıralar
a.reverse()             # listeyi ters çevirir
a.pop()                 # son elemanı siler
a.append("a")           # sonuna yeni eleman ekler
a.insert(indis, "a")    # yeni elemanı belirtilen indise ekler
```

Listeler devam

```
a.count(1)        # Bu eleman listede kaç tane var
a.index(1)        # Bu eleman kaçıncı indiste

print a[1]        # 1.indiste ki elemanı yazdır
a[1]=2            # 1.indisteki elemanın değerini değiştir
del a[2]          # 2.indisteki elemanı listeden sil

x=list()          # Boş liste oluşturur
x=[]              # Boş liste oluşturur
```

Liste üzerinde döngüler

```
liste=["a","b","c"]
for i in liste:
    print(i)
```

Listeler devam

Soru: Döngülerde indislere de erişmek istersek ?

Cevap: enumerate

```
for i in enumerate(liste):
    print(i)
```

Listeler üzerinde oynamak

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

liste[1]          #listenin 1. elemanı=2
liste[-1]         #listenin -1. elemanı=9
liste[1:3]        #listenin 1. elemanından 3. elemanına kadar [2,3]
liste[:3]         #listenin ilk elemanından 3. elemanına kadar[1,2,3]
liste[3:]         #listenin 3. elemanından listenin sonuna kadar[3,4,5,6,7,8,9]
```

Listeleri Birleştirmek

```
a=[1,2,3]
b=[4,5,6]
c=a+b
```

Çok Boyutlu Listeler

```
x=[
      [1,0,0],
      [0,1,0],
      [0,0,1]
    ]

len(x)          # satır sayısını verir
len(x[0])    # ilk satırdaki sütun sayını verir
```

Çok Boyutlu Listeler

```
x=[
      [1,0,0],
      [0,1,0],
      [0,0,1]
    ]

for i in x:
       print i
```

Çok Boyutlu Diziler

```
import random
x=[]
for i in range(5):
        x.append([random.randint(1,5) for c in range(4)])


print x
```

Döngülerle Birlikte Else Kullanımı

```
while şart:
    komutlar1
else:
    komutlar2
```

Döngü yanlış ise else kısmı çalışacak

```
while şart:
    komutlar1
else:
    komutlar2
```

Klasik yaklaşımda ki boolean tipte bir kontrol değişkeni kullanımı gereksiz olur

For-Else Örneği

- **Klasik yöntem**

```
x=25
asal=True
for i in range(2,x/2+1):
        if x%i==0:
                asal=False
                break
if asal:
        print x,"asal bir sayidir»
else:
        print x,"asal degildir"
```

- **For-Else ile**

```
x=25
for i in range(2,x/2+1):
        if x%i==0:
                print x,"asal sayi değildir"
        break
else:
        print x,"asal bir sayidir"
```

While-Else Örneği

- **Klasik Yöntem**

```
x=23
i=1
asal=True
while i<(x/2)+1:
        i+=1
        if x%i==0:
                asal=False
                break

if asal:
        print x,"asal bir sayidir»
else:
        print x,"asal bir sayi degildir"
```

- **While - Else ile**

```
x=17
i=1
while i<(x/2)+1:
        i+=1
        if x%i==0:
                print x,"asal bir sayi degildir"
        break

else:
        print x,"asal bir sayidir"
```

Liste Üreteçleri

- **Klasik yaklaşım**

```
x=[]
for i in range(100):
    x.append(i)

print x
```

- **Veya**

```
x=range(100)

print x
```

Liste Üreteçleri

- **Klasik yaklaşım**

```
x=[]
for i in range(100):
    x.append(i**2)

print x
```

- **Veya**

```
x=[i**2 for i in range(100)]

print x
```

Her iki programda

1,4,9,25,36 … n^2

den oluşan bir liste üretir

Satır İçi Fonksiyonlar

```
x=list()
x=[i**2 for i in range(100)]
```

- **Biraz Daha İleri Seviye**

```
str="Hayat Kısa Python Ogrenin"
x=[ i for i in str if i.isupper() ]
```

X ne olur ?

İstisna İşleme

- **Tipine Göre Hatalar**
  - Programcı Hataları (Error)
    - Syntax hataları
  - Program Kusurları (Bug)
    - Dilden kaynaklı hatalar (Update yada fix ile çözülebilir)
  - İstisnalar (Exception)
    - Programcıdan kaynaklı hatalar (Çalışma zamanında oluşur)
- **Çalışma Zamanına Göre Hatalar**
  - Derleme zamanı hataları
  - Çalışma zamanı hataları

Temel İstisna İşleme Mekanizması

```
a=10
b=0
try:
    c=a/b
    print c
except ZeroDivisionError as e:
    print "hata"
finally:
    print "Son"
```

try:

İstisna oluşturabilecek kodlar

except:

İstisna durumunda yapılacaklar

finally:

Son İşlemler (Her iki durumdada çalışır)

İstisna Durumları

- 0'a bölme
- Aritmetik işlem yaparken kullanıcının rakam yerine harf girmesi
- Olmayan bir dosyayı okumaya çalışmak
- Yazma izni olmayan bir dosyaya yazmaya çalışmak
- Veritabanına bağlanamadan tablo okumaya/yazmaya çalışmak
- Vb…

İstisnalar programların kilitlenmesine/yarıda kesilmesine neden olur.

Ödev

- Elemanları 1-9 arası rakamlar içeren 100 elemanlı bir random bir liste üreterek, bu liste üzerinde her bir rakamın kaç defa geçtiğini gösteren bir sözlük yapısı kurun.

  (not: Hazır fonksiyonlar kullanılmadan yapılacak)

- random.random() ile üretilecek 100 tane sayı içinde birbirine en yakın iki sayıyı bulan programı kodlayın.

  (not: Hazır fonksiyonlar kullanılmadan yapılacak)

Python Programlama Dili

Dosya İşlemlerine Giriş

nxn Birim matris oluşturma

```
n=5
x=list()
for i in range(n):
    y=list()
    for j in range(n):
        y.append(1) if i==j else y.append(0)  # if-else satır içi kullanımı
    x.append(y)


for i in x:
    print i
```

Köşegen matrisi oluşturma

```
n=10
x=list()
for i in range(n):
    y=list()
    for j in range(n):
        y.append(1) if (i+j)==n-1 or (i==j) else y.append(0)  # if-else satır içi kullanımı
    x.append(y)

for i in x:
    print i
```

nxn boyutunda 2 matrisi toplama

matrisler metot ile elemanları random olacak şekilde oluşturulacak

```
n=3
import random

def matris_olustur():
    x=[ [random.randint(1,9) for c in range(n)] for i in range(n)] #Satır içi üreteçle iki boyutlu matris üretme
    return x



def topla(a,b):
    for i in range(n):
        for j in range(n):
            c[i][j]=a[i][j]+b[i][j]
    return c

a= matris_olustur()
b=matris_olustur()
c=matris_olustur()

print a
print b
c=topla(a,b)
print c
```

Liste üreteci ile 1/4, 1/16, 1/36 .. 1/(2n)^2 serisini üretmek

```
#Kısa yol
x=[1./i**2 for i in range(1,100) if i % 2==0]
print x


#Uzun yol
"""
x=list()
for i in range(1,100):
    if i%2==0:
        x.append(1./i**2)

print x
"""
```

Dosya İşlemleri

- Kabukta dizini görmek

  pwd

  «Bulunulan dizini gösterir»

- Dosya Okuma/Yazma işlemleri istisna işleme mekanizması ile ele alınmalıdır
- Temel işlemler için dosyanın bulunulan dizinde (pwd çıktısı) olduğundan emin olalım

Dosya İşlemleri

Dosya Açma formatı

f = open(dosya_adı, modu)

Modlar

w => yazma

r => okuma

a => ekleme

a+ => ekleme ve okuma

Dosya İşlemleri

f.write(*string_ifade*)

- Dosyaya yazar

f.read()

- Tüm dosyayı okur

f.readlines()

- Tüm dosyayı bir diziye atar

Dosya İşlemi yazma modu ile dosya açıp içine 1-100 sayılarını alt alta yazıyoruz programdan sonra deneme.txt kontrol ediniz

```
f=open("deneme.txt","w") #Yazma modu ile dosyayı aç

for i in range(100):
    f.write(str(i)+"\n")



f.close()  #Dosyayı kapat
```

```
dosya = open("şiir.txt", "w")

dosya.write("Bütün güneşler batmadan,\nBi türkü daha söyleyeyim bu yerde\n\t\t\t\t --Orhan Veli--")

dosya.close()


with open("şiir.txt", "r") as dosya:

 a=dosya.read()

 print a
```

```
#dosyayı basa sarma
dosya.seek(0)
b=dosya.read()
print b

#dosyanın basına verı ekleme
with open("şiir.txt", "r+") as f:
    veri = f.read()
    f.seek(0)
    f.write("Selin Özden\t: 0212 222 22 22\n"+veri)
```

```
#ortaya veya herhangı bır yere ekleme
with open("şiir.txt", "r+") as f:
    veri = f.readlines()
    veri.insert(2, "Sedat Köz\t: 0322 234 45 45\n")
    f.seek(0)
    f.writelines(veri)
```

```
a = open("cift.txt","w")
b = open("tek.txt","w")
for i in range(1,100):
    if i % 2 == 0 :
        a.write(str(i))
        a.write(",")

    else :
        b.write(str(i))
        b.write(",")

a.close()
b.close()
```

ikidosyaoku_dosyaya_yazma

```
dosya = open("C:\\Users\\FeyzaÖzbay\\Documents\\Python\\sayi1.txt","r")
sayilar = [int(satir) for satir in dosya.readlines()]
dosya.close() # artik dosyayla isimiz bittigine gore kapatabiliriz.
print (sayilar)
ortalama = sum(sayilar) / len(sayilar) # ortalama almak icin kolay bir yontem!
print (ortalama)
dosya2 = open("C:\\Users\\FeyzaÖzbay\\Documents\\Python\\sayi2.txt","r")
sayilar2 = [int(satir2) for satir2 in dosya2.readlines()]
dosya2.close() # artik dosyayla isimiz bittigine gore kapatabiliriz.
print (sayilar2)
ortalama2 = sum(sayilar2) / len(sayilar2) # ortalama almak icin kolay bir yontem!
print (ortalama2)
sayilar3 = list()
for i in range(10):
   sayilar3.append (sayilar[i]+sayilar2[i])
print  (sayilar3)
dosya3=open("C:\\Users\\FeyzaÖzbay\\Documents\\Python\\sayi3.txt","w") #Yazma modu ile dosyayı aç
dosya3.write(str(sayilar3)+"\n")
dosya3.close()
```

sesli_bul

```
alfabe = ['a', 'b', 'c', 'd', 'e', 'f', 'i', 'j', 'o']

# function that filters vowels
def filtersesli(alfabe):
    sesliler = ['a', 'e', 'i', 'o', 'u']

    if(alfabe in sesliler):
        return True
    else:
        return False
filtersesli = filter(filtersesli, alfabe)
print('Filtrelenen sesli harfler:')
for ses in filtersesli:
    print(ses)
```

Recursive Fonksiyon nedir?

- Recursive fonksiyon, içerisinde kendini çağıran fonksiyondur.

```
def faktoriyel(x):
    if x == 0:
        return 1
    else:
        return x * faktoriyel(x-1)
```

Aldığımız sayıdan 0'a kadar olan bütün sayıların toplamını hesaplayan bir recursive fonksiyon yazalım

```
def toplam(sayi):
    # Base Case(Temel Durum)
    if sayi == 1:
        # Bu fonksiyon icin sinirimiz recursive kisimda
        # azaltilan sayinin 1'e esit olma durumu
        return 1
    # Recursive Case(Yinelenen Durum)
    else:
        return sayi + toplam(sayi - 1)
```

Çift bir sayıyı parametre olarak alan ve aldığı sayıdan 0'a kadar olan çift sayıların toplamını yazdıran bir recursive fonksiyonu yazalım.

```
def cift_sayi_toplam(sayi):
    #base case
    if sayi == 0:
        return 0
    #recursive case
    else:
        return sayi + cift_sayi_toplam(sayi - 2)
```

Aldığı sayının kaç basamaklı olduğunu yazdıran bir recursive fonksiyonu yazalım.

```
def basamak_sayisi(sayi):
    if sayi%10 < 1:
        return sayi
    else:
        return 1 + basamak_sayisi(sayi/10)
```

Fibonacci dizisini aldığı sayıya kadar devam ettiren bir recursive fonksiyon yazalım.

```
def fibonacci(sayi):
    # Base Case
    if sayi == 0:
        return 0
    elif sayi == 1:
        return 1
    # Recursive Case
    else:
        return fibonacci(sayi-1) + fibonacci(sayi-2)
```

Bir listenin en büyük ve en küçük elemanını

```
def min_r(liste):
    if len(liste) == 1:
        return liste[0]
    else:
        if liste[0] < liste[1]:
            return min_r([liste[0]] + liste[2:])
        else:
            return min_r([liste[1]] + liste[2:])


def max_r(liste):
    if len(liste) == 1:
        return liste[0]
    else:
        if liste[0] > liste[1]:
            return max_r([liste[0]] + liste[2:])
        else:
            return max_r([liste[1]] + liste[2:])
```

Python Programlama Dili

Gömülü Fonksiyonlar

Gömülü Fonksiyonlar

- Fonksiyon
  - Kullanıcı tarafından geliştirilir
- Builtin fonksiyon
  - Geliştirici tarafından geliştirilip dile entegre edilmiştir

Gömülü fonksiyonlar genel olarak, problemin türünden bağımsız sık gereksinim duyulan fonksiyonlardır.

print, len gibi

Gömülü Fonksiyonlar

- Matematiksel Fonksiyonlar
  - abs() => Mutlak değerini al
  - round() => Yuvarla
  - bin() => İkili sayıya dönüştür
  - pow() => Kuvvetini al
- all() => Hepsi True mı?
- any() => En az biri True mı ?

Gömülü Fonksiyonlar

- Tür dönüşümleri
  - chr()
  - str()
  - int()
  - float()

Gömülü Fonksiyonlar

- Liste, Demet, Küme, Sözlük Oluşturanlar
  - list()
  - tuple()
  - set()
  - dict()

  - enumerate()

Gömülü Fonksiyonlar

- Yardım Alma
  - dir()
  - help()
  - type()

Gömülü Fonksiyonlar

- Kullanıcıdan Giriş Alma
  - input()
  - raw_input()
- Liste fonksiyonları
  - len()
  - range()
  - sum(liste)
  - min()
  - max()
  - sort()
  - reverse()

Gömülü Fonksiyonlar

- İki veriyi karşılıklı birleştirme

zip()

```
a=("ali","veli")
b=(1,2)
z=zip(a,b)

z=[('ali', 1), ('veli', 2)]
```

Gömülü Fonksiyonlar

filter()

```
def suz(x):
    return x>=70


print filter(suz, [78,45,67,97])
```

map()

```
def karesinial(x):
    return x**2


print map(karesinial,[2,3,5])
```

Rekürsif Fonksiyonlar

- Rekursiflik (Kendi kendini çağıran fonksiyonlar)

```
def faktoriyel(n):
    if n==1:
        return 1
    else:
        return n*faktoriyel(n-1)

print faktoriyel(5)
```

Rekürsif Fonksiyonlar

- Faktoriyel örneğinin daha kısa bir hali

```
def faktoriyel(n):
    return 1 if n==1 else  n*faktoriyel(n-1)


print faktoriyel(5)
```

Kısa if

```
if şart:
    doğru_ise
else:
    yanlış_ise
```

- **Tek satırda yazma**

```
doğru_ise if şart else yanlış_ise
```

Fonksiyonel programlama

- Map
- Filter
- Reduce
- Lamda
- Liste işleçleri

Fonksiyonel programlama araçları programcıya esneklik ve zaman kazandıran programlama yaklaşımlarıdır.

Fonksiyonel programlama avantaj/dezavantajları

- Kod yapısını kısaltır, kod geliştirme süresini uzatır.
- Test dostu yazılım geliştirmeyi sağlar.
- Performans: Bu tür fonksiyonlar kullanıldıktan sonra Garbage Collector tarafından silinirler.

Nedir ?

- Problem: Bir liste'de ki küçük harfle başlayan kelimeleri bulmak istiyoruz (Filter problemi)
- Yaklaşım: liste üzerinde bir döngü kurmak.
- Fonksiyonel programlama yaklaşımı: Tek parametre alan bir metot tanımlayıp, tüm listeyi filter aracılığı ile metota göndermek.

Filter

- **Filter** ile tanımlanacak metot bir bool ifade ile gelen parametreyi seçmelidir.

```
a=range(11)
def suz(x):
    return x%2==0

print filter(suz,a)
```

Örnekte **filter** suz metoduna parametreleri tek tek göndermekte, metot ise bool sonucuna göre parametreyi geri döndürmekte yada döndürmemektedir.

Map

- **Map** ile tanımlanacak metot parametreyi güncelleyerek dönderir.

```
a=range(11)
def ekle(x):
    return x*x

print map(ekle,a)
```

Örnekte **map** ekle metoduna parametreleri tek tek göndermekte, metot ise geriye yine bir parametre göndermektedir. (Gelen giden parametre sayısı eşit)

Reduce

- Map ve Filter amaçları farklı olsa da şekil olarak birbirlerine çok benzerler. Reduce ise liste elemanlarını ardışıl şekilde parametre olarak alır ve en sonunda bir parametre dönderir.

```
a=range(11)
def topla(x,y): return x+y
print reduce(topla,a)
```

Lambda

Geçici ve tek satırdan yazılabilecek basit fonksiyonlar yazılmasını sağlar.

```
def carp(x,y):
    return x*y
```

```
carp=lambda x,y:x*y
```

Yapı

Fonksiyon_adı=**lambda** parametreler : geri_dönecek_değer

Fonksiyonlarda Kısaltma

- **Lambda**

  carp=lambda x,y:x*y

- **Kısaltma**

  def carp(x,y): return x*y

  Her iki yapı birbirine eşittir.

Aynı örnekleri liste işleçleri ile yapmak:Filter

- **Filter**

```
a=range(11)
def suz(x):
    return x%2==0

print filter(suz,a)
```

- **Liste**

```
[i for i in a if i%2==0]
```

Aynı örnekleri liste işleçleri ile yapmak:Map

- **Map**

```
a=range(11)
def ekle(x):
    return x*x

print map(ekle,a)
```

- **Liste**

```
[i**2 for i in a]
```

Aynı örnekleri liste işleçleri ile yapmak:Reduce ?

- **Reduce**

```
a=range(11)
def topla(x,y): return x+y
print reduce(topla,a)
```

- **Liste**

```
sum([i for i in a])
```

Alıştırmalar

1) Reduce kullanarak döngüsüz faktöriyel hesaplayın ?

liste=["Python", "Ruby", "PHP", "jAVA", "scala","go"]

2) Listedeki tüm sözcükler büyük harfle mi başlıyor ?

3) Listede en az bir tane tüm harfleri büyük kelime varmı ?

Python Programlama Dili

Hesap Makinesi

```
giriş = """
(1) topla
(2) çıkar
(3) çarp
(4) böl
(5) karesini hesapla
(6) karekök hesapla
"""

print(giriş)

soru = input("Yapmak istediğiniz işlemin numarasını girin: ")

if soru == "1":
    sayı1 = int(input("Toplama işlemi için ilk sayıyı girin: "))
    sayı2 = int(input("Toplama işlemi için ikinci sayıyı girin: "))
    print(sayı1, "+", sayı2, "=", sayı1 + sayı2)

elif soru == "2":
    sayı3 = int(input("Çıkarma işlemi için ilk sayıyı girin: "))
    sayı4 = int(input("Çıkarma işlemi için ikinci sayıyı girin: "))
    print(sayı3, "-", sayı4, "=", sayı3 - sayı4)
```

```
elif soru == "3":
    sayı5 = int(input("Çarpma işlemi için ilk sayıyı girin: "))
    sayı6 = int(input("Çarpma işlemi için ikinci sayıyı girin: "))
    print(sayı5, "x", sayı6, "=", sayı5 * sayı6)

elif soru == "4":
    sayı7 = int(input("Bölme işlemi için ilk sayıyı girin: "))
    sayı8 = int(input("Bölme işlemi için ikinci sayıyı girin: "))
    print(sayı7, "/", sayı8, "=", sayı7 / sayı8)

elif soru == "5":
    sayı9 = int(input("Karesini hesaplamak istediğiniz sayıyı girin: "))
    print(sayı9, "sayısının karesi =", sayı9 ** 2)

elif soru == "6":
    sayı10 = int(input("Karekökünü hesaplamak istediğiniz sayıyı girin: "))
    print(sayı10, "sayısının karekökü = ", sayı10 ** 0.5)

else:
    print("Yanlış giriş.")
    print("Aşağıdaki seçeneklerden birini giriniz:", giriş)
```

```
d1 = open("isimler1.txt") # dosyayı açıyoruz
d1_satırlar = d1.readlines() # satırları okuyoruz

d2 = open("isimler2.txt")
d2_satırlar = d2.readlines()

for i in d2_satırlar:
    if not i in d1_satırlar:
        print(i)

d1.close()
d2.close()
```

Karakter Dizisindeki Karakterleri Sayma

```
metin = """Bu programlama dili Guido Van Rossum adlı Hollandalı bir programcı
tarafından 90'lı yılların başında geliştirilmeye başlanmıştır. Çoğu insan,
isminin Python olmasına aldanarak, bu programlama dilinin, adını piton
yılanından aldığını düşünür. Ancak zannedildiğinin aksine bu programlama dilinin
adı piton yılanından gelmez."""

harf = input("Sorgulamak istediğiniz harf: ")

sayı = []
for s in metin:
    if harf == s:
        sayı += harf    # kullanıcıdan gelen bu harfi sayı değişkenine yolla

print(len(sayı))
```

İç İçe (Nested) Fonksiyonlar

İsminden anlayabileceğimiz gibi içe içe olan birden fazla fonksiyonumuz olunca bunlara nested, yani iç içe fonksiyonlar diyoruz.

```
def fonk1():
    def fonk2():
        ...
```

```
def yazıcı(mesaj):
    def yaz():
        nonlocal mesaj
        mesaj += " Dünya"
        print(mesaj)
    return yaz


>>> y = yazıcı("Merhaba")
>>> y()
Merhaba Dünya
```

```
liste=["Python", "Ruby", "pHp", "jAVA", "scala","go"]

def suz(x):
    return x.istitle()  #Büyük harfle başlıyorsa dönder



def cevir(x):
    return x.capitalize() #İlk harfi büyük harfe çevirip dönder



print filter(suz,liste)
print map(cevir,liste)
```

Rekürsif float bölme

```
def bol(x,y,bolum=0,kalan=0,ondalik=0,s="",b=""):
    if x<y:
        x=x*10
        b=""
        for i in range(5):
                while (x-y>=0):
                    x=x-y
                    ondalik+=1
                b=b+str(ondalik)
        return (x,y,bolum,kalan,ondalik,b)

    else:
        bolum+=1
        return bol(x-y,y,bolum,kalan,ondalik,s="",b="")



sonuc= bol(10,5)
s=str(sonuc[2])+"."+str(sonuc[5])
print "sonuc=",s
```

String İşlemleri

- Stringler karakterlerden oluşan liste yapısı olarak ele alınabilir.

```
s="Python"
Print( len(s))
Print( s[0:3])
Print( s[-1])
Print( s.count("a"))
```

String Üzerinde Döngü Kurmak

```
for i in s:
    print i
```

```
for i in range(len(s)):
    print s[i]
```

```
for x,y in enumerate(s):
    print x,y
```

String Manipülasyonları

s="Merhaba"

s.replace("a","A")

Bölmek ve Birleştirmek

x=s.split(" ") => ayırıcı karaktere göre bölüp listeye atar

y=' '.join(x) => birleştirici karaktere göre birleştirip string yapar

splitlines() ise bir paragrafı satır satır böler

Çevirmek

Küçük/Büyük harfe dönüştürme

.lower() #lower fonksiyonu, tüm karakterleri küçük harfe çevirir

adsoyad = «dErYa«

print adsoyad.lower()

.upper() #upper fonksiyonu, tüm karakterleri büyük harfe çevirir

print adsoyad.upper()

.capitalize() #capitalize fonksiyonu, string'deki ilk harfi büyük harfe çevirir, diğerlerini küçük harfe çevirir

print adsoyad. capitalize()

.title() //cümledeki kelimelerin ilk harflerini büyük yapar

.swapcase() //küçük harfleri büyük harfe ve tersini değiştirin

Sorgulama

.isalpha()

Dizedeki tüm karakterler alfabe ise doğrudur (hem küçük hem de büyük harf olabilir).

Yanlış eğer en az bir karakter alfabe değildir.

```
name = "Monica"
print(name.isalpha()) // doğru

 name = "Mo3nicaGell22er"
print(name.isalpha())  // yanlış
```

- **. isalnum ()**

Dize en az 1 karakter içeriyorsa ve tüm karakterler alfanümerik, aksi takdirde false ise true değerini döndürür.

- **.islower()**

Dize en az 1 karakterli karakter içeriyorsa ve tüm karakterli karakterleri küçük harf ve küçük harfle yazıyorsanız true değerini döndürür.

- **.isdigit()**

Dize yalnızca rakam içeriyorsa true, aksi takdirde false değerini döndürür.

Sorgulamak

s="bilimsel programlama olarak python"

.endswith metodu karakter dizileri üzerinde herhangi bir değişiklik yapmamızı sağlamaz. Bu metodun görevi karakter dizisinin durumunu sorgulamaktır.

s.endswith("n") => True döner

s.startswith eğer karakter dizisi gerçekten belirtilen karakterle başlıyorsa Python True çıktısı, yok eğer belirtilen karakterle başlamıyorsa False çıktısı veriyor.

s.startswith("bi") => False döner

Temizlemek

- lstrip öndeki karakterlerini kaldırır
- rstrip sondaki karakterlerini kaldırır
- strip her iki taraftaki karakterlerini kaldırır

```
s="kazak"
s.lstrip("k")
s.rstrip("k")
s.strip("k")
```

Aramak

s="kazak mazak"

s.index("l") => Bulursa indexini, bulamazsa hata döndürür

s.find("l") => Bulursa indexini, bulamazsa -1 döndürür

Aşağıdakiler aynı işlemi sağdan yaparlar

.rindex()

.rfind()

Python Programlama Dili

Modüller

Modül Nedir?

- Modüllerin, bazı işlevleri kolaylıkla yerine getirmemizi sağlayan birtakım fonksiyonları ve nitelikleri içinde barındıran araçlar olduğunu söyleyebiliriz.

Modül Ekleme

- import random

  random.randint(1,10)

- import random as rnd

  rnd.randint(1,10)

- from random import randint as r

  r(1,10)

- from random import *

  randint(1,10)

Modüllerde Yardım

- import random

  dir(random)

OS Modülü

- **import os**
  - os.name => işletim sisteminin adı ney
  - os.sep => işletim sisteminin seperatörü ney
  - os.getcwd. => bulunduğumuz dizin
  - os.chdir('/usr/bin/') => Mevcut klasöre git
  - os.listdir() => Belirtilen dizindeki dosya/klasörleri listeler

    os.listdir(os.getcwd())
  - os.mkdir('yenidizin') => Yeni dizin oluşturur
  - os.rename("untitled1.py","deneme1.py") => adını değiştir
  - os.rmdir('dizin_adı')
  - os.remove('dosya_adı')
  - os.stat('dosya_adı') => Dosya hakkında detaylı bilgi verir

Os Modülü

- os.path.isfile(‘dosya_adi’)
- os.path.isdir()
- os.path.exists()

dizin, dosya = os.path.split('/Desktop/deneme.txt’)

dosya_adi, uzantı = os.path.splitext(‘deneme.txt')

SYS Modülü

- sys.exit()
- sys.argv => Komut satırından girilen parametreler

Random Modülü

- random.randint(a,b)
- random.random()
- random.choice(liste) => Listeden 1 tane seçer
- random.sample(liste,n) => Listeden n tane seçer
- random.shuffle(liste) => Listeyi karıştırır

Datetime Modülü

import datetime

datetime.datetime.now()

month/year/day/hour/second

datetime.datetime.today()

Zaman Formatı

```
import datetime

tarih = datetime.datetime.today()

print datetime.datetime.strftime(tarih,'%d %m %Y')
```

Zaman Aritmetiği

```
import datetime
simdi = datetime.datetime.today()
fark = datetime.timedelta(days=2, seconds=12)
ileri=simdi+fark
geri=simdi-fark
print ileri
print geri
```

Time Modülü

```
import time

time.sleep(1)          => 1 sn bekle
```

Kodun çalışma süresi ?

```
import timeit

bas = timeit.default_timer()
son = timeit.default_timer()

fark=son-bas
print fark

#Saniye cinsinden değer üretir
```

```
import datetime
import locale

locale.setlocale(locale.LC_ALL,'tr_TR') #yada "turkish şeklinde yazılabilir

gun=11
ay=4

yil=2022

#Değişkenlerden tarih oluşturma
tarih=datetime.date(yil,ay,gun) # yil-ay-gün şeklinde olmalı sıra
#tarih=datetime.datetime.today() #Bugün

#tarih.year, tarih.month, tarih.day şeklinde de erişilebilir

print tarih

print datetime.datetime.strftime(tarih,"%d %B %A %Y") #Formatlı yazdırma
```

```
import datetime

bugun=datetime.datetime.today()

fark=datetime.timedelta(weeks=0,days=1,hours=0, minutes=0, seconds=0)

yarin=bugun+fark
dun=bugun-fark
ikigunoncesi=bugun-fark*2



print bugun
print yarin
print dun
print ikigunoncesi

print "dun bugünden küçükmü: ", dun<bugun
```

```
sözlük = {"kitap"      : "book",
          "bilgisayar" : "computer",
          "programlama": "programming"}

def ara(sözcük):
    hata = "{} kelimesi sözlükte yok!"
    return sözlük.get(sözcük, hata.format(sözcük))

def ekle(sözcük, anlam):
    mesaj = "{} kelimesi sözlüğe eklendi!"
    sözlük[sözcük] = anlam
    print(mesaj.format(sözcük))
```

Nesne Yönelimli Programlama

Object Oriented Programming

OOP bir programlama yaklaşımıdır.

1. Sıralı Programlama (Kod ilk satırda doğar, çalışır, son satırda biter)
2. Metotlar (Tekrar kullanılabilirlik, kendini tekrar etmeme!)
3. OOP (Nesnelerin özellikleri ve davranışları vardır)
   - OOP ile gerçek dünyayı daha kolay modelleyebiliriz.

Avantajları

- Modülerlik
- Yazılım bakım kolaylığı
- Genişletilebilirlik
- Değiştirilebilirlik
- Yeniden kullanılabilirlik

Kavramlar

- OOP aşağıdaki 4 temel prensibi desteklemelidir

- Abstraction (Soyutlama)
- Encapsulation (Kapsülleme)
- Inheritance (Miras)
- Polymorphism (Çok biçimlilik)

Sınıf/Nesne Kavramı

- Sınıf soyut bir kavramdır.
  - OOP ile oluşturulan programlama yapısıdır.
- Nesne sınıfın somutlaşan bir cismidir.

Sınıf Oluşturma

```
class ilksinif:                 # Sınıfları class anahtar sözcüğü ile oluşturuyoruz
   i=1                          # Sınıf değişkeni
   def yaz(self):               # Sınıf metotu, self anahtar sözcüğü sınıfa atıf yapar
       print "i=",self.i        # Sınıf değişkenlerine self.değişken şeklinde erişiyoruz


x=ilksinif()                    # x sınıftan oluşturduğumuz nesne
print x.i                       # Sınıf değişkenlerine erişim ve atama x.i=5
x.yaz()                         # Sınıf metodunu çağırma
```

Global Değişkenler

```
class ilksinif:
   global a
   a=2
    i=1
    def yaz(self):
        print "global a=",a

x=ilksinif()
x.yaz()
a=7
x.yaz()
```

- Sınıf değişkenlerine sınıf dışında doğrudan erişemeyiz

- Sınıf içinden
  - ***self.değişken***
- Sınıf dışından
  - ***nesne.değişken***

şeklinde erişebiliriz

Global Değişkenlere

- Her yerden erişebilir, değiştirebiliriz
- Self anahtar sözcüğüne gerek yok

Sınıfları Başlatma

```
class ilksinif:
    def __init__(self):
        print "merhaba"


x=ilksinif()
```

__init__(self):

- Özel bir metottur
- Metot çağrılmaksızın nesne oluşturulduğunda çalışır

- initialize : Başlatmak

Programı Başlatmak

Programı ilk satırdan itibaren biz başlatmış olduk

Programı doğrudan bağımsız olarak çalıştırmak için, bir main metoduna ihtiyacımız var (Java da ki main methodu gibi).

```
if __name__=="__main__":
    x=ilksinif()
```

Sınıflara Başka Programdan Erişmek

```
class ilksinif:
   i=1
   def yaz(self):
       print "i=",self.i
```

Bu programı **dene.py** olarak kaydettiğimizi varsayarsak

```
import dene
x=dene.ilksinif()
print x.goster()
```

test.py programı içinden erişmek için import ediyoruz.

2-100 arasındaki asal sayıları bulur

kontrol değişkenini bayrak olarak kullanıyoruz
döngü başında kontrol=True
içteki döngüde eğer sayı bölünürse kontrol=False yapılarak döngü kırılıyor (break ifadesi) döngü sonunda kontrol halen True ise sayı ekrana yazdırılıyor
if kontrol: kontrol zaten boolean değişken olduğu için == ile karşılaştırmaya gerek yok

```
for x in range(2,100):
    kontrol=True
    for i in range(2,x-1):
        if x%i==0:
            kontrol=False
            break
    if kontrol:
        print x
```

ilk 100 fibonacci sayısını yazdırma

```
"""
a=1
b=2 #sıradaki fibonacci sayısı
for x in range(100):
    c=a+b
    a,b=b,c  #çoklu atama
    print c
```

```
liste=[3,1,"b",5,"a",10,2,3,5]    #Tamamen dinamiktirler
demet=(3,1,5,"a",10,2,3,5)        #Sonradan değiştirilemezler
kume={3,1,5,10,2,3,5}             #Çift eleman içermezler


print "Tipler=>",type(liste),type(demet),type(kume)


liste.pop()     #Son elemanı çek
liste.append(20)  #20 yi ekle
liste.sort()    #Listeyi sırala
```

```
# Liste üzerinde döngü
print "Liste"
for i in liste:
    print i

# demet.append(3) Hata verecektir çünkü demetlerde günceleme
# Demet üzerinde döngü
print "Demet"
for i in demet:
    print i
```

Metot kendisine gelen sayı asal ise True döndürecek parametre verilmez ise default değer 2 alınacak

```
def asalmi(x=2):
    asal=True
    for i in range(2,(x/2) +1):
        if x%i==0:
            asal=False
            break
    return asal

x=107
if asalmi(x):
    print x,"asal sayidir"
else:
    print x,"asal değildir"


print asalmi()     #Parametresiz çağrılınca x=2 default değer geçerli olacak
```

1 ile 1000 arasında rastgele sayıda elemandan oluşan, değerleri 0-1 arasında değişen bir liste oluşturuluyor, ardından eleman sayısı ve toplamı bulunuyor

```
import random

liste=list()

for i in range(random.randint(1,1000) ):
    liste.append(random.random())

x,s=len(liste),sum(liste)
print x,"tane elemandan oluşan listenin toplamı=",s,"ortalaması=",s/x
```

```
#Klasik yöntem - Kontrol değişkeni kullanılıyor
x=23
i=1
asal=True
while i<(x/2)+1:
    i+=1
    if x%i==0:
        asal=False
        break
if asal:
    print x,"asal bir sayidir"
else:
    print x,"asal bir sayi degildir"

"""
x=17
i=1
while i<(x/2)+1:
    i+=1
    if x%i==0:
        print x,"asal bir sayi degildir"
        break

else:
    print x,"asal bir sayidir"
```

Liste üretme örnekleri

1,4,9,16,25 ... n^2 serisinden oluşan bir liste üretmek

#1.yol

```
x=[]
for i in range(100):
    x.append(i**2)
```

#2.yol

```
x=[i**2 for i in range(100)]
print x
```

100 adet random sayıdan birbirlerine en yakın olanı bulma (farkları minumum olanlar)

```
import random

sayilar=list()
minimum=1 #minimum değişkeninin başlangıç değeri
liste=[random.random() for i in range(100)] # Random listeyi oluşturma

for x in liste[0:len(liste)-1]:                    #Her sayıyı listedeki bir sonraki sayı ile karşılaştırıyoruz
    for y in liste[(liste.index(x)+1):len(liste)]:   #Karşılaştırma işlemi listenin sonuna kadar her eleman için yapılacak
        if abs(x-y) <minimum:                        #Farkların mutlak değerini alıyoruz
            sayilar[:2]=[x,y]
            minimum=abs(x-y)



print "minumum fark=",minimum
print "en yakin sayilar=",sayilar
print "tüm liste=",liste
```

nxn Birim matris oluşturma

"""

```
n=5
x=list()
for i in range(n):
    y=list()
    for j in range(n):
        y.append(1) if i==j else y.append(0)  # if-else satır içi kullanımı
    x.append(y)


for i in x:
    print i
```

Bir listenin tüm elemanlarının küplerini alarak yeni bir liste oluşturmak

"""

```
#Kısa yol
x=range(10)
y=[i**3 for i in x]
print x
print y
```

```
#Uzun yol
"""
x=range(10)
y=list()
for i in x:
    y.append(i**3)

print x
print y
```

nxn boyutunda 2 matrisi çarpma, program düzenlenerek axb, ve bxc boyutunda iki matriside çarpabilir matrisler metot ile elemanları random olacak şekilde oluşturulacak

```
n=3
import random

def matris_olustur():
    x=[ [random.randint(1,9) for c in range(n)] for i in range(n)] #Satır içi üreteçle iki boyutlu matris üretme
    return x

def carp(a,b):
    for i in range(n):
        for j in range(n):
            t=0
            for k in range(n):
                t+=a[i][k]*b[k][j]
            c[i][j]=t
    return c

a= matris_olustur()
b=matris_olustur()

print a
print b
c=carp(a,b)
print c
```

Liste üreteci ile 1/4, 1/16, 1/36 .. 1/(2n)^2 serisini üretmek

"""

```
#Kısa yol
x=[1./i**2 for i in range(1,100) if i % 2==0]
print x


#Uzun yol
"""
x=list()
for i in range(1,100):
    if i%2==0:
        x.append(1./i**2)

print x
```

Dosya işlemleri

```
a = open("cift.txt","w")
b = open("tek.txt","w")
for i in range(1,100):
    if i % 2 == 0 :
        a.write(str(i))
        a.write(",")

    else :
        b.write(str(i))
        b.write(",")

a.close()
b.close()
```

```
alfabe = ['a', 'b', 'c', 'd', 'e', 'f', 'i', 'j', 'o']

# function that filters vowels
def filtersesli(alfabe):
    sesliler = ['a', 'e', 'i', 'o', 'u']

    if(alfabe in sesliler):
        return True
    else:
        return False
filtersesli = filter(filtersesli, alfabe)
print('Filtrelenen sesli harfler:')
for ses in filtersesli:
    print(ses)
```

04242370001-1001 şeklinde verilen bir telefon numarası formatını
0424 2370001 1001 şeklinde split etme
"""

```
tel="04242370001-1001"

no,dahili=tel.split("-")

alanadi, numara=no[0:4],no[4:]


print alanadi, numara,dahili


#join örneği
yeni=[alanadi,numara,dahili]

s=':'.join(yeni)
print s
```

```
import datetime
import locale

locale.setlocale(locale.LC_ALL,'tr_TR') #yada "turkish şeklinde yazılabilir

gun=18

ay=3

yil=2018

#Değişkenlerden tarih oluşturma
tarih=datetime.date(yil,ay,gun) # yil-ay-gün şeklinde olmalı sıra
#tarih=datetime.datetime.today() #Bugün

#tarih.year, tarih.month, tarih.day şeklinde de erişilebilir

print tarih

print datetime.datetime.strftime(tarih,"%d %B %A %Y") #Formatlı yazdırma
```

Arama Algoritmaları lineer, ikili ,lineer search => sırasız liste üzerinde binary search => sıralı olduğunu bildiğimiz liste üzerinde çalışacak

"""

```
def lineer_search(liste,aranan):
    boyut=len(liste)
    for i in range(boyut):
        if aranan==liste[i]:
            return i
    else:
        return 0

def binary_search(liste,aranan):
    indis=len(liste)/2
    if not indis:
        return "Hata"
    elif aranan==liste[indis]:
        return indis
    elif aranan>liste[indis]:
        return indis+binary_search(liste[indis:],aranan)
    elif aranan<liste[indis]:
        return binary_search(liste[:indis],aranan)
    else:
        return 0

liste=range(50)
print binary_search(liste, 45)
```