HAFTA-1-2-3:

Genel Özellikler:

```
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 ?
```

Temel tipler (integer, string, boolean,double)

Diğer veri tipleri

- Liste (List)
- Demet (Tuple)
- Küme (Set)
- Sözlük (Dictionary)

VERİ TİPLERİ:

1-Listeler: [] Dinamiktirler

```
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
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
help(a.append) # a.append nasıl çalışır
dir(a)          # a ile başka ne yapabilirim
```

Soru: Döngülerde indislere de erişmek istersek ?

Cevap: enumerate

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

Listeleri birleştirmek istersek:

```
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 diziler:

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

2-Demetler: 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!

3-Kümeler: a={1,2,"a",5}

#Bu methodları desteklerler

add

revome

pop

#Ancak aşağıdaki methodlar çalışmaz, kümeler sıralıdır ve çift (ikili)değer içermezler

a.index

a[1]

a.count(5) ?

4-Sözlükler: {key:value}

```
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
```

For Döngüsü:

```
for i in range(a,b,c):      #a'dan b'ye (b dahil değil) c
    print i                 arttırımlı olarak i'yi yazdırır.
```

break:

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

Continue:

Döngüyü pas geçer (bir sonraki adımdan devam eder)

Ör:

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 yöntemi 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"
```

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
```

Ör:

```
def hesapla(a,b):
    x=a*b
    return x
print hesapla(3,4)
```

Ör:

```
def hesapla(a=2,b=3):
    x=a*b
    return x
print hesapla()      # Sonuç 6 olacak
print hesapla(5)     #  Sonuç ?(a=5 alır b yine 3'tür)
# 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 rasyonl 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
```

NOT: join ve split birbirlerinin tam tersi iş yaparlar.

ÖDEV , ÖRNEKLERE BAK

Ör: Liste Üretme ile ilgili örnek

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

veya

```
x=range(100)
print x
```

Ör: Liste üreteçleri ilgili örnek

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

Ör: Satır içi Fonksiyonlar

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

Biraz Daha İleri Seviye

```
str="Hayat Kısa Python Oğrenin"
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

Ör:

```
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.

ÖDEV –ÖRNEKLERE BAK

HAFTA-4:

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 Açma formatı:

- f = open(dosya_adı, modu)
- **Modlar**

  w => yazma

  r => okuma

  a => ekleme

  a+ => ekleme ve okuma

f.write(*string_ifade*) Dosyaya yazar

f.read() Tüm dosyayı okur

f.readlines() Tüm dosyayı bir diziye atar.

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

- 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 ?
- Tür dönüşümleri
  - chr() = KARAKTER
  - str() = STRİNG
  - int() = TAM SAYI
  - float() = RASYONEL SAYI
- Liste, Demet, Küme, Sözlük Oluşturanlar:
  - list()
  - tuple()
  - set()
  - dict()
  - enumerate()
- Yardım Alma
  - dir()
  - help()
  - type()
- Kullanıcıdan Giriş Alma
  - input()
  - raw_input()
  - Liste fonksiyonları
  - len()
  - range()
  - sum(liste)
  - min()
  - max()
  - sort()
  - reverse()

zip(): İki kümeyi karşılıklı olarak birleştirir

Ör:

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

filter():

```
def suz(x):
    return x>=70
print filter(suz, [78,45,67,97])
#70'ten büyük sayıları filtreler
```

map():

```
def karesinial(x):
    return x**2
print map(karesinial,[2,3,5])
```

Rekürsif Fonksiyonlar:

Kendi kendini çağıran fonksiyonlar

Ör:

```
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

ALIŞTIRMA – ÖDEV VE ÖRNEKLERE BAK

HAFTA-5:

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 bağlayan 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

```
a=range(11)

def suz(x):
```

```
    return x%2==0

print filter(suz,a)
```

Liste Şekli:

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

Aynı örnekleri liste işleçleri ile yapmak: Map

```
a=range(11)

def ekle(x):
    return x*x

print map(ekle,a)
```

Liste Şekli:

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

Aynı örnekleri liste işleçleri ile yapmak: Reduce

```
a=range(11)

def topla(x,y): return x+y

print reduce(topla,a)
```

Liste Şekli:

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

<u>String İşlemleri:</u>

Stringler karakterlerden oluşan liste yapısı olarak ele alınabilir.

```
s="Python"

print len(s)

print s[0:3]

print s[-1]

print s.count("a")

s[0:6:2]

s[::-1]
```

String Üzerinde Döngü Kurmak:

```
for i in s:
    print i

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

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

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()

.upper()

.capitalize()

.title()

.swapcase()
```

Sorgulama:

```
.isalpha()

.isupper()

.islower()

.isdigit()
```

Sorgulamak:

```
s="bilimsel programlama olarak python"

s.endswith("on")      => True döner

s.startswith("bi")    => False döner
```

Temizlemek:

Verilen parametreye göre başındaki/sonundaki/her iki karakteri temizlemek

```
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()

Ödev örneklere bak

HAFTA-6:

Modül Ekleme:

1. import random

   random.randint(1,10)

2. import random as rnd

   rnd.randint(1,10)

3. from random import randint as r

   r(1,10)

4. 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.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
```

ÖDEV – ÖRNEKLERE BAK

HAFTA-7:

Arama Algoritmaları:

- Listeler Üzerinde Arama
  - **Doğrusal Arama**
  - **İkili Arama**
  - İnterpolasyon Araması
- Şekiller Üzerinde Arama
  - Kruskal, Dijkstra, Binary Tree, B-Tre
  - ...
  - Metin Arama Algoritmaları
- Diğer (A*)

Yapısal Arama Algoritmaları:

Yapısal olarak arama algoritmalarını iki gruba ayırabiliriz

- Bilerek Arama
  - *Listenin sıralı olduğunu biliyoruz*
- Bilmeden Arama
  - *Listenin sıralı olduğunu bilmiyoruz*

Doğrusal Arama:

- Sırasız bir liste üzerinde arama yapıyorsak
- Aranan eleman bulunana kadar listeyi tara

İkili Arama:

- Sıralı bir liste üzerinde arama yapıyorsak tüm elemanları tek tek kontrol etmemize gerek yok

- Algoritma (Rekürsif)
  1. Aranan ortadaki eleman ise döndür ve bitir
  2. Aranan ortadaki elemandan büyükse listenin sağı için aramayı tekrarla
  3. Aranan ortadaki elemandan küçükse listenin solu için aramayı tekrarla

Sıralama Algoritmaları:

- Listeler Üzerinde (veri tabanı)
- Şekil üzerinde arama (haritada iki nokta arasındaki en kısa yolu bulma)
- Metin arama (metin indexleme, arama motoru)
- Diğer (oyunlar vb)

Seçmeli Sıralama:

En basit arama algoritması

- En küçük elemanı en başa al!
- Arama süresi N(N-1)/2

Eklemeli Sıralama:

- Eleman daha küçükse kalanları ötele!
- 2(1+2+...+(N-2)+(N-1))=N(N-1)

Kabarcık Sıralama:

- Eklemeli sıralamaya benzer, öteleme yerine yer değiştirir. Arama süresi (N-1)(N-1)

Quick Sort: Hızlı

- Pivot seç

- Büyükleri pivotun sağına
- Küçükleri soluna al
- Rekursif olarak devam et
- Boyut<=1 ise dur

Arama süresi n(log2n)