

MUDANYA
UNIVERSITY



BMB 151 Bilgi Teknolojileri ve Programlamaya Giriş

18.12.2023

Dr. Öğr. Üyesi Aslı SEBATLI-SAĞLAM



MUDANYA
UNIVERSITY



Python ile Programlamaya Giriş

Veri Yapıları



- IDE (integrated development environment – entegre geliştirme ortamı)
 - Bir editör, derleyici ve hata ayıklayıcı içeren bir programlama ortamı
 - PyCharm, MS Visual Studio Code, IDLE, Spyder, Anaconda ...
- Metin düzenleyici
 - Bir programın veri dosyasının içeriği gibi bir metin dosyasının içeriğini girmek için kullanılan bir yazılım uygulaması
- Terminal ekranı
 - Metinsel komutlar aracılığıyla bir işletim sistemi ile etkileşimde bulunan pencere

- Python yorumlayıcısı (interpreter)
 - Python kaynak kodunu bayt (0-1) koduna çeviren ve Python sanal makinesinde çalıştıran bir program
- Shell ekranı
 - Python yorumlayıcısı ile etkileşimde bulunmak için kullanılabilecek bir kullanıcı arayüz ekranı
- Büyük/Küçük harfe duyarlılık
 - Büyük/küçük harf özelliklerini ayırt edici özellik

- Nesne-tabanlı programlama
- Web-tabanlı programlama
- Çoklu miras yapısını destekleme
- Açık-kaynak kodlu
- Kolay kurulum
- Üçüncü parti yardımcı kütüphaneler
- Veri madenciliği, istatistiksel analizler, yapay zeka uygulamaları vb için uygun altyapı
- Günlük dile yakın sözdizim kuralları
- Girintiye duyarlılık !!!

- Derleme-aşaması (compile-time) hatası
 - Bir program derlendiğinde tespit edilen bir hata
- Sözdizimi (syntax) hatası
 - Programlama dili kurallarına uymayan ve derleyici tarafından reddedilen bir komut (bir tür derleme zamanı hatası)
- İstisna (exception)
 - Programın normal şekilde devam etmesini engelleyen bir durumu işaret eden bir sınıf
 - Böyle bir durum oluştuğunda, ilgili hata sınıfından bir nesne atılır
- Çalışma-zamanı (run-time) hatası
 - Sözdizimsel olarak doğru bir programda, programın belirttiğinden farklı davranmasına neden olan bir hata
- Mantık (logic) hatası
 - Sözdizimsel olarak doğru bir programda, programın belirttiğinden farklı davranmasına neden olan bir hata (bir tür çalışma zamanı hatası)

- Veri
 - İşlenen tüm bilgiler veri olarak adlandırılır
- Değişken
 - Farklı değerleri tutmak üzere hafızada ayrılan bir depolama konumunu tanımlayan sembol
- Atama
 - Bir değişkene yeni bir değer verme
 - `sayac = sayac + 1`
- Sabit
 - Bir program tarafından değiştirilemeyen bir değer
 - Python'da sabitler genellikle büyük harflerden oluşan isimlere sahiptir
- Yorum
 - Kullanıcının ilgili bölümü anlamasına yardımcı olmaya yönelik yapılan açıklama
 - `#` sembolü ile ilgili satıra yorum yazılır
 - `"""` sembolü arasına çok satırlı yorum yazılır `"""`

- String

- Metin içeren değişkenlerdir
- Tırnak içinde temsil edilen bitişik bir karakter seti olarak tanımlanır
- Tek veya çift tırnak kullanılabilir

```
x = "Hello World!"
```

```
print(type(x))
```

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

- len(): dizinin karakter sayısını verir

```
x = "Hello World!"
```

```
print(len(x))
```

```
>> 12
```

- index(): parametre olarak verilen karakteri string üzerinde arayarak ilgili indisin değerini döndürür

```
print(x.index("W"))
```

```
>> 6
```


- String
 - split(): belirtilen karaktere göre stringi böler

```
y=x.split(" ")
print(y)
>> ['Hello', 'World!']
```
 - strip(): stringin başındaki ve sonundaki boşluk karakterlerini siler

```
x = "   Hello World!   "
print(x.strip())
>> Hello World!
```
 - replace(): belirtilen karakteri belirtilen başka bir karakter ile değiştirir

```
x = "Hello World!"
y = x.replace("World", "Mudanya")
print(y)
Hello Mudanya!
```

- String
 - find(): belirtilen ifadeyi string içinde arayarak ilk bulduğu indisin değerini döndürür

```
x = "Hello World!"
print(x.find("World"))
>> 6
```
 - join(): stringleri birleştirir
 - Fonksiyonun başına ifade konulursa belirtilen bu ifade ile birleştirir
 - ```
str1 = "asli"
str2 = "saglam"
str3 = " ".join([str1, str2])
print(str3)
>>asli saglam
```

- String

- + işareti ile de stringler birleştirilir

```
str1 = "asli"
```

```
str2 = "saglam"
```

```
print(str1 + " " + str2)
```

```
>> asli saglam
```

- \* işareti ile belirtilen sayıda aynı string defalarca yazılabilir

```
str = "asli"
```

```
print(str*10)
```

```
>> asliasliasliasliasliasliasliasli
```

- input() : kullanıcıdan veri girişi alınır

Dikkat verinin varsayılan tipi string'dir, numerik değerler için dönüşüm yapılmalıdır!

- Integer

- Tamsayılı değer alan numerik değişkenlerdir
- Pozitif, negatif veya sıfır değer alabilirler

```
x = 5
print(x)
print(type(x))
>>5
<class 'int'>
```

- Float

- Ondalıklı sayı

```
x = 3.2
print(x)
print(type(x))
>>3.2
<class 'float'>
```

Dikkat: Ondalık ayracı  
olarak «nokta»  
kullanılmalıdır!

- Boolean
    - True / False tipi değişkenler
- ```
x = True  
print(x)  
print(type(x))  
>>True  
<class 'bool'>
```

- Listeler

- Diğer programlama dillerindeki dizi (array) yapısına benzer tipteki değişkenler
- Sıralı, yinelenen, değiştirilebilir elemanlardan oluşan koleksiyon
- Köşeli parantez kullanarak tanımlanır
- Elemanlar arasında virgül kullanılır
- Diğer programlama dillerinde diziler tek tip değişken içerirken Python'da liste yapıları farklı tip değişkenler içerebilir!

- String, integer, float, boolean tipi değişkenler bir arada tek bir listede yer alabilir

```
liste = ["asli", 5, 4.5]
```

```
print(liste[1])    # ilgili indisteki elemanı verir
```

```
>> 5
```

```
meyveler = ["elma", "portakal", "armut", "kivi", "mandalina"]
```

```
print(meyveler[1:3])    # 1. indisten 3. indise kadar olan elemanları verir, 3. indisi vermez!!!
```

```
>> ['portakal', 'armut']
```

- Listeler

```
meyveler = ["elma", "portakal", "armut", "kivi", "mandalina"]  
meyveler[1] = "ayva"           # 1. indisteki elemanı değiştirdi  
print(meyveler)  
>> ['elma', 'ayva', 'armut', 'kivi', 'mandalina']
```

- len(): eleman sayısını verir

```
meyveler = ["elma", "portakal", "armut", "kivi", "mandalina"]  
print(len(meyveler))  
>> 5
```

- append(): sona eleman ekler

```
meyveler = ["elma", "portakal", "armut", "kivi", "mandalina"]  
meyveler.append("muz")  
print(meyveler)  
>> ['elma', 'portakal', 'armut', 'kivi', 'mandalina', 'muz']
```

- Listeler

- insert(): belirtilen indise göre araya eleman ekler

```
meyveler = ["elma", "portakal", "armut", "kivi", "mandalina"]
```

```
meyveler.insert(2, "cilek")
```

```
print(meyveler)
```

```
>> ['elma', 'portakal', 'cilek', 'armut', 'kivi', 'mandalina']
```

- del : eleman siler

```
meyveler = ["elma", "portakal", "armut", "kivi", "mandalina"]
```

```
del meyveler[1]
```

```
print(meyveler)
```

```
>> ['elma', 'armut', 'kivi', 'mandalina']
```


- Listeler

- extend(): listenin sonuna başka bir listenin elemanlarını ekler

```
meyveler = ["elma", "portakal", "armut", "kivi", "mandalina"]
```

```
sebzeler = ["biber", "kabak", "patlican"]
```

```
meyveler.extend(sebzeler)
```

```
print(meyveler)
```

```
['elma', 'portakal', 'armut', 'kivi', 'mandalina', 'biber', 'kabak', 'patlican']
```

- remove(): belirtilen öğeyi listeden kaldırır

```
meyveler = ["elma", "portakal", "armut", "kivi", "mandalina"]
```

```
meyveler.remove("portakal")
```

```
print(meyveler)
```

```
>> ['elma', 'armut', 'kivi', 'mandalina']
```

- Listeler

- pop(): indis belirtmedikçe sondaki elemanı döndürür ve kaldırır, indis belirtilirse ilgili elemanı döndürür ve kaldırır

```
meyveler = ["elma", "portakal", "armut", "kivi", "mandalina"]
```

```
x = meyveler.pop()
```

```
print(x)
```

```
print(meyveler)
```

```
>> mandalina
```

```
>> ['elma', 'portakal', 'armut', 'kivi']
```

```
meyveler = ["elma", "portakal", "armut", "kivi", "mandalina"]
```

```
x = meyveler.pop(2)
```

```
print(x)
```

```
print(meyveler)
```

```
>> armut
```

```
['elma', 'portakal', 'kivi', 'mandalina']
```

- Listeler

- reverse(): Listeyi tersine çevirir

```
meyveler = ["elma", "portakal", "armut", "kivi", "mandalina"]
```

```
meyveler.reverse()
```

```
print(meyveler)
```

```
>> ['mandalina', 'kivi', 'armut', 'portakal', 'elma']
```

- clear(): Tüm öğeleri temizler

```
meyveler = ["elma", "portakal", "armut", "kivi", "mandalina"]
```

```
meyveler.clear()
```

```
print(meyveler)
```

```
>> [ ]
```

- Listeler

- `copy()`: Listeyi kopyalar

```
meyveler = ["elma", "portakal", "armut", "kivi", "mandalina"]
```

```
x = meyveler.copy()
```

```
print(x)
```

```
>> ['elma', 'portakal', 'armut', 'kivi', 'mandalina']
```

- `sort()`: listeyi sıralar

```
sayilar = [4, 5, 8, 2, 9, 0, 1, 3]
```

```
sayilar.sort()
```

```
print(sayilar)
```

```
>> [0, 1, 2, 3, 4, 5, 8, 9]
```

- Listeler
 - `index()`: belirtilen elemanın kaçınıcı indekste olduğunu verir
`sayilar = [4, 5, 8, 2, 9, 0, 1, 3]`
`ind = sayilar.index(8)`
`print(ind)`
`>> 2`
 - `count()`: belirtilen elemandan liste içinde kaç tane olduğunu sayar
`sayilar = [9, 4, 5, 8, 2, 3, 9, 0, 1, 3, 4, 9]`
`print(sayilar.count(9))`
`>> 3`

- Sözlükler (Dictionary)
 - Diğer dillerdeki map veya hash veri tiplerine benzerler
 - anahtar : değer biçiminde veri içerirler
 - Bir anahtar yalnızca bir kez kullanılır

```
isimler = {1: "asli", 2: 'yesim', 3: 'ufuk', 4: 'deniz'}
```

```
print(type(isimler))
```

```
<class 'dict'>
```

```
isimler = {1: "asli", 2: 'yesim', 3: 'ufuk', 4: 'deniz'}
```

```
print(isimler.keys())
```

```
print(isimler.values())
```

```
>> dict_keys([1, 2, 3, 4])
```

```
>> dict_values(['asli', 'yesim', 'ufuk', 'deniz'])
```

- Sözlükler (Dictionary)

```
isimler = {1: "asli", 2: 'yesim', 3: 'ufuk', 4: 'deniz'}
```

```
print(isimler[1])
```

```
>> asli
```

```
isimler = {1: "asli", 2: 'yesim', 3: 'ufuk', 4: 'deniz'}
```

```
isimler[5] = "bora"
```

```
print(isimler)
```

```
>> {1: 'asli', 2: 'yesim', 3: 'ufuk', 4: 'deniz', 5: 'bora'}
```

- Demet (Tuple)
 - Üzerinde değişiklik yapılamayan liste biçimi
 - Değişmeyecek veriler varsa tercih edilebilir
 - Listelere kıyasla çok daha hızlı işlem yapılır
 - Değişmez veriler içerdikleri için `append()`, `remove()`, `pop()` gibi metodlar kullanılamaz
 - Tanımlarken normal parantez kullanılır, listelerdeki gibi köşeli parantez kullanılmaz!
 - Değiştirmek için yeni bir demet yapısı kullanılabilir
 - Değiştirmek için demet yapısı listeye dönüştürebilir, liste üzerinde değişiklik yapılır ve liste tekrar bir demete dönüştürülür
 - `list()`
 - `tuple()` komutları ile bu işlemler yapılır

- Küme (Set)
 - Matematikteki küme veri yapısı gibidir
 - Değişmez nesnelerden oluşur, aynı eleman tekrar eklenmez
 - Süslü parantez ile tanımlanır
- ```
liste = [1, 2, 4, 3, 5, 2, 3]
setim = set(liste)
print(setim)
>> {1, 2, 3, 4, 5}
```
- add(): yeni eleman ekler
  - remove(): ilgili elemanı siler
  - union(): kümeleri birleştirir
  - intersection(): kümelerin kesişimini verir
  - difference(): kümelerin farkını verir

# Python'da Temel Matematiksel İşlemler

26

+ : Toplama

- : Çıkarma

\* : Çarpma

/ : Bölme

\*\* : Üs alma

// : Tam bölme (kalansız bölme)

% : Mod alma

== : Eşittir

!= : Eşit değildir

< : Küçüktür

> : Büyüktür

<= : Küçük eşittir

>= : Büyük eşittir

& : Ve

\ : Veya

- Aşağıda verilen kod kaç satır çıktı üretmektedir?  

```
print("Hello")
print("World!")
```
- Aşağıda verilen komutların çıktılarını gözlemleyiniz.  

```
print("Hi")
print("there")
print(39 + 3)
print("39+3")
print("x =", 7 + 2)
```

- Aşağıda verilen değişken isimlerinden hangileri geçerli bir değişken ismidir?
  - kediler geçerli
  - Sayac1 geçerli
  - 1sayac geçersiz
  - kedi-sayac geçersiz
  - max\_deger geçerli
  - kedi sayisi geçersiz
  - kediSayisi geçerli
  - KediSayisi geçerli
  - C geçerli

- Aşağıdaki kod bölümünü dikkate alarak

num = 12

x = 4

y = 2

verilen ifadelerin sonucunu hesaplayınız.

- $x ** 2 * y$
- $x ** (2 * y)$
- $num + 6 ** 2 + 10$

Aşağıda verilen ifadelerin sonucunu hesaplayınız.

$27 // 4$

$27 \% 4$

- Kullanıcı tarafından bir kenar uzunluğu ve bu kenarın yüksekliği verilen bir üçgenin alanını hesaplayan python programı yazınız

```
taban = float(input("Taban uzunlugunu giriniz: "))
yukseklk = float(input("Yuksekligi giriniz: "))

Alani hesapla
alan = 0.5 * taban * yukseklik

Sonucu yaz
print("Ucgenin alani:", alan)
```

- Kullanıcı tarafından verilen iki sayının toplamını, farkını, çarpımını ve oranını yazan python programı yazınız

```
sayi1 = float(input("İlk sayiyi giriniz: "))
sayi2 = float(input("İkinci sayiyi giriniz: "))

Hesaplamalar
toplam = sayi1 + sayi2
fark = sayi1 - sayi2
carpim = sayi1 * sayi2
oran = sayi1 / sayi2

Sonuclari yazma
print("Toplam: ", toplam)
print("Fark: ", fark)
print("Carpim: ", carpim)
print("Oran: ", oran)
```

- Kullanıcıdan bir sayı alarak bu sayının basamak sayısını yazan programı yazınız

```
sayi = int(input("Bir sayi giriniz: "))

Calculate the number of digits
basamak_sayisi = len(str(abs(sayi)))

Display the result
print(f" {sayi} sayisinin basamak sayisi: {basamak_sayisi}")
```