Python ile Programlamaya Giriş

Mert Can Demir – A. Oğulcan Çankaya

Programlama Dili Nedir?

- **Programlama dili**, bir bilgisayara veya programlanabilir bir elektronik cihaza, hangi veri üzerinde işlem yapılacağı, verinin nasıl depolanıp iletileceği, hangi koşullarda hangi işlemlerin hangi sıra ile yapılacağı gibi konularda talimat vermeyi sağlayan ve belirli bir yazım kuralı olan dildir.

Sistem Programlama: C ve C++

- 70'li yıllarda geliştirilen C ve 80'li yıllarda geliştirilen nesneye yönelik özellikler eklenmiş hali olan C++ günümüzde halen en çok kullanılan **sistem programlama** dilleridir.
- İşletim sistemleri, veri tabanı yönetim sistemleri, gömülü sistemler, aygıt sürücüleri, oyun motorları gibi yazılımlar geliştirilirken donanıma direkt erişim gerekebileceği için kimi zaman alt seviye bir dil olan Assembler tercih edilse de, çoğu zaman programlamanın daha kolay olması için orta seviye diller olan C ve C++ kullanılır.
- C++ 90'lı yıllara kadar uygulama programlama alanında da en çok kullanılan dil olmuştur.

Java

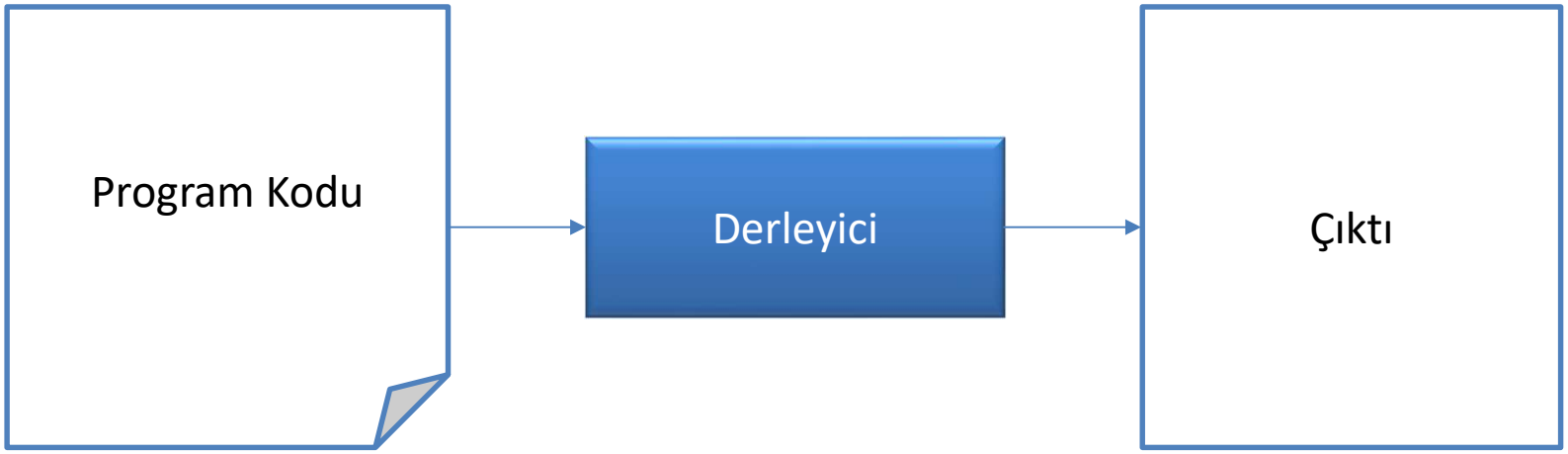
- Platform bağımsız, üst seviye bir dildir.
- Platform bağımsızlığı sayesinde özellikle akıllı telefonlar öncesi kullanılan cep telefonları için uygulamalar ve oyunlar geliştirilirken Java yaygın biçimde kullanılmıştı.

Python

- Guido van Rossum tarafından 1990 yılında geliştirilmeye başlanan Python, nesneye yönelik, yorumlamalı, açık kaynak kodlu, platform bağımsız ve yüksek seviyeli bir programlama dilidir.
- Bu özellikleri ile Java'ya benzemesine rağmen girintilere dayalı basit sözdizimi sayesinde programlamaya ilk başlayanlar için öğrenilmesi daha kolaydır.

Derleyici (Compiler) Nedir?

- Bir programlama dili ile yazılmış program kodunubilgisayarın anlayabileceği makine diline çeviren programa derleyici denir.



Yorumlayıcı (Interpreter) Nedir?

- Yorumlayıcı (interpreter), yüksek seviyeli programlama dili ile yazılmış bir programı adım adım makine diline çeviren ve makine dilindeki talimatları çalıştıran programdır.
- Derleyici'den (compiler) farklı olarak kaynak kod her ne zaman çalışırsa her adımı için makine dilindeki karşılığı tekar oluşturulur ve çalıştırılır.

Değişken (Variable)

- Programlarda kullanılan verileri saklamak ve gerektiğinde o veriler üzerinde işlem yapmak için değişkenler kullanılır.
- Birçok programlama dilinde bir değişken tanımı yapıldığında, o değişkenin türüne göre istenilen büyüklükte bir hafıza bölgesi işletim sisteminden istenir. Tanımlanan değişkene bir değer atandığında, değişken için ayrılan hafıza bölgesine o değer yazılır.
- Python gibi dillerde değişken tanımı yapılmadan da o değişken kullanılabilir. Python'da bir değişkene ilk defa değer atandığında o değerin türüne göre değişkenin türü belirlenir.

Değişken Türleri ve Değişkene Değer Atama

- **Örnekler:**

```
a = 7
```
a değişkeni tamsayı (integer) türünde yaratılır

```
b = 7.0
```
b değişkeni ondalıklı (float) sayı türünde yaratılır

```
a = 'ali'
```
a değişkeni artık karakter dizisi (string) türünde oldu

Değer Atamada Kurallar

- Değişken ismi solda, atanacak değer sağda olmalıdır (7 = a doğru değildir):

a = b	a değişkenine b değişkenindeki değeri atar
b = a	b değişkenine a değişkenindeki değeri atar

- Aynı anda iki yada daha fazla değişkene değer atanabilir.

a, b = 4, 5	a değişkenine 4, b değişkenine 5 atar
a, b = b, a	a'ya b'nin değerini, b'ye a'nın değerini atar

Değişkenlerde Kısıtlamalar

- Değişken isminde : ; , . / ' # [ ] ! " $ % ^ & * ( ) { } karakterleri ve boşluk karakteri kullanılamaz. Ancak _ kullanılabilir.
- Değişken isminin ilk karakteri harf olmalıdır. Yani değişkenler rakamla başlamaz (sayi2 olur 2sayi olmaz).
- Bir fonksiyon veya prosedürde aynı değişken ismi birçok defa tanımlanamaz (farklı fonksiyonlarda tanımlanabilir).
- Programlama diline ait bir komut ismi, değişken ismi olarak tanımlanamaz.
- Python dilinde büyük/küçük harf ayrımı vardır (case sensitive).

Operatörler

- Aritmetik işlem operatörleri:

  `+` `-` `*` `/` `%` `//` `**`

  `%` → mod alma

  `//` → tam sayı bölme

  `**` → üs alma

- Mantıksal operatörler:

  `and` `or` `not`

- Karşılaştırma operatörleri:

  `<` `>` `<=` `>=` `==` `!=`

  `==` → eşit

  `!=` → eşit değil

- Atama operatörleri:

  `=` `+=` `-=` `*=` `/=` `%=` `//=` `**=`

Değişkenin değerini; belli bir sayı kadar arttırma (+=), belli bir sayı kadar azaltma (-=), belli bir sayı kadar çarpma (*=), belli bir sayı kadar bölme (/=), belli bir sayı kadar modunu alma (%=), belli bir sayı kadar tam sayı bölme (=//) ve belli bir sayı kadar üssünü alma (**=) için kullanılırlar.

Veri Türleri

- Python'da temel olarak 3 veri türü vardır:

  int: tamsayıları saklar (integer)

  float: ondalıklı sayıların saklanması içindir

  str: Bir veya daha çok karakterin saklanmasını sağlar. Tek tırnak veya çift tırnak içinde yazılan her şey rakam bile olsa str (string) olarak saklanır.
- Bir değişkenin veya veri yapısının türünü öğrenmek için type() fonksiyonu kullanılabilir.
  - type(a) yazdığınızda <class 'int'> gibi bir çıktı alırsınız

Birçok dilde tamsayılar ve ondalıklı sayılar için birden çok veri türü kullanılır. Örneğin C dilinde tamsayılar için short, int ve long; ondalıklı sayılar için ise float ve double gibi hafızada farklı büyüklüklerde yer kaplayan farklı veri türleri vardır.

Fonksiyon, Parametre ve Argüman

- Programlama dillerindeki fonksiyonlar matematikteki fonksiyonlara benzer. Yani bir yada daha çok değer alıp, bir değer veya veri yapısı (ileride göreceğiz) döndürürler.
  - Örneğin 2 veya daha çok sayı alıp, bunların en büyük ortak bölenini bulup döndüren fonksiyon yazabiliriz (ileride göreceğiz)
- Fonksiyonlar isimlerinden sonra gelen parantez içinde farklı değerler verilerek çağrılabilir. Virgül ile birbirinden ayrılan bu değerlere parametre denir.
- Fonksiyonları çağırırken, parametrelerine gönderdiğimiz değerlere ise argüman denir. Argüman olarak sabit bir değer, bir değişken, aritmetiksel bir işlem veya bir fonksiyon kullanılabilir.

Veri Türü Değiştirme Fonksiyonları

- Veri türleri ile aynı isimde olan ve farklı türdeki verileri o türe dönüştüren fonksiyonlar vardır.
- Örnekler:

```
>>> float(3)
3.0
>>> str(45.32)
'45.32'
>>> int(4.99)
4
>>> round(4.99)
5
```

3 tamsayısını alıp 3.0 ondalıklı sayısını döndürdük

45.32 ondalıklı sayısını alıp str türünde döndürdük

4.99 sayısının tam sayı kısmını (4) döndürdük

4.99 sayısını yuvarladık.

round fonksiyonu ikinci parametre olarak ondalıklı kısmın kaç basamağa indireleceğini de alabilir. Örneğin round(5/3, 4) ifadesinin sonucu 1.6667 olur. Buradaki 5/3 ve 4 değerleri parametrelere gönderilen argümanlardır.

İşlem Örnekleri

```
>>> 5 * 2 + 5 / 2
12.5
>>> 5 % (2 + 5) / 2
2.5
>>> int(9/2) * float('4.5')
18.0
>>> 3**2*5//2
22
>>> 'ali ' * 3
'ali ali ali '
```

Matematikte olduğu gibi parantezin önceliği vardır. % işlemi ise *, / ve // ile aynı önceliklidir. Bunlarda sola yazılan işlem önce yapılır. + ve – en son yapılanlardır

** işleminin diğer aritmetik işlemlere göre önceliği vardır. Yani en sağda bile olsa idi önce yapılırdı.

str üzerinde çarpma işlemi sadece int değer ile kullanılabilir ve birçok programlama dilinde bu özellik yoktur

Klavyeden Veri Alma : input

- ENTER basılana kadar girilen karakter dizisini (string) döndürür. Döndürdüğü değeri bir değişkene atayabilirsiniz:

```
ad = input('adınızı girin: ')
print('merhaba ' + ad)
```

  - Kullanıcının girdiği değerin hepsi rakam olsa bile str türünde ad değişkenine atanır
  - + operatörünün her iki yanında da str türünde veri olduğu için aritmetiksel toplama yapılmaz, 'merhaba' ile ad (kullanıcının girdiği string) ifadeleri birleştirilir

Dört İşlem Programı

```
a = int(input('bir sayı giriniz: '))
b = int(input('bir sayı giriniz: '))
print("Toplam :", a + b)
print("Fark   :", a - b)
print("Çarpım :", a * b)
print("Bölüm  :", a / b)
```

Örnekte görüldüğü gibi int() fonksiyona argüman olarak input() fonksiyonu verilmiştir. input() fonksiyonundan gelen string türü değeri int() ile tamsayıya çevirdik. Bunu yapmasaydık a + b iki string'i birleştirecekti (a = '4' ve b = '5' ise sonuç 9 değil '45' olurdu). '-' işlemi ise str türü veriler üzerinde kullanılamayacağı için program o satırda hata verecekti. Eğer a ve b değişkenlerinde ondalıklı sayılar saklanmak istenirse int() fonksiyonu yerine float() fonksiyonu kullanılmalıdır.

print hakkında ek bilgi (1)

- Önceki örnekte görüldüğü gibi farklı türden verilerin print içinde yan yana gösterilmesi için ',' işareti kullanılır.
- Eğer hepsi str türünde ise ',' yerine '+' kullanılabilir. Fakat '+' aralara boşluk karakteri eklemez.
  - `>>> print('python',3,'öğreniyorum')`
  - `python 3 öğreniyorum`
  - `>>> print('python'+'3'+'öğreniyorum')`
  - `python3öğreniyorum`

Tırnak içinde yazmazsak str olmaz ve hata verir

print hakkında ek bilgi (2)

- Eğer print içinde string ifadenizi çift tırnak içinde yazarsanız içinde tek tırnak geçebilir. Fakat ifade tek tırnak içinde yazılırsa içinde tek tırnak kullanabilmek için önüne \ yazmalısınız:
  - `"Ali'nin kalemi"` `'Ali\'nin kalemi'`
- C dilinde olduğu gibi \ işareti sonrasında n kullanmak alt satıra geçmenizi (new line), t ise TAB tuşu eklemenizi sağlar.
  - `print('Ali\nin kalemi')` → Ali
in kalemi
  - `print('Ali\tnin kalemi')` → Ali nin kalemi