Değişkenler, Veri Tipleri ve Operatörler

Table of contents

2.1 Değişkenler: Bilgileri Saklama Kutuları

Programlamada, verileri saklamak ve işlemek için **değişkenleri** kullanırız. Değişkenler, tıpkı bir kutu gibi, içinde farklı türde bilgiler tutabilir. Python'da bir değişken tanımlamak için değişken adını yazıp, ardından eşittir işareti (=) kullanarak değeri atarız. Örneğin:

```
isim = "Ahmet"
yas = 30
boy = 1.85
ogrenci_mi = True
```

Bu örnekte, `isim`, `yas`, `boy` ve `ogrenci_mi` değişkenleridir. Her bir değişken farklı bir **veri tipini** tutar:

- `isim`: String (Metin)
- `yas`: Integer (Tam sayı)
- `boy`: Float (Ondalıklı sayı)
- `ogrenci_mi`: Boolean (Doğru/Yanlış)

2.2.1 Değişken Yazım Kuralları

Python'da değişken isimleri belirlerken bazı kurallara uymamız gerekir:

- Değişken isimleri harf, rakam ve alt çizgi (_) karakterlerinden oluşabilir.
- Değişken isimleri bir rakamla başlayamaz.
- Değişken isimleri büyük/küçük harf duyarlıdır (`isim` ile `Isim` farklı değişkenlerdir).
- Python'da rezerve edilmiş kelimeler (keywords) değişken ismi olarak kullanılamaz (örneğin, `if`, `else`, `for`, `while`, `print`, `True`, `False`).

> **Anlamlı Değişken İsimleri Seç**
>
> Değişken isimleri seçerken, anlamlı ve okunabilir isimler kullanmaya özen gösterin. Örneğin, bir kişinin yaşını tutacak bir değişken için `yas` ismi, `x` isminden daha anlaşılırdır.

2.3 Veri Tipleri

Python, farklı türdeki verileri depolamak ve işlemek için çeşitli **veri tipleri** sunar. Her değişkenin bir veri tipi vardır ve bu veri tipi, değişkenin hangi tür verileri saklayabileceğini belirler.

İşte Python'daki temel veri tipleri:

- **`int` (Tamsayı - Integer):** Tam sayıları temsil eder (örneğin, 10, -5, 0, 1000).
- **`float` (Ondalıklı Sayı - Floating-point):** Ondalıklı sayıları temsil eder (örneğin, 3.14, -2.5, 0.0, 2.71828).
- **`str` (Metin - String):** Metinleri temsil eder. Tek tırnak (`'`) veya çift tırnak (`"`) içinde yazılır (örneğin, "Merhaba", 'Python').
- **`bool` (Mantıksal Değer - Boolean):** `True` veya `False` değerlerini alır. Koşullu ifadelerde kullanılır.

İlerleyen bölümlerde, `list` (liste), `tuple` (demet), `dict` (sözlük) ve `set` (küme) gibi daha karmaşık veri yapılarına da değineceğiz.

Örnek:

```
isim = "Ali"       # String
yas = 30           # Integer
fiyat = 99.99      # Float
aktif_mi = True    # Boolean
```

2.3.1 `type()` Fonksiyonu ile Veri Tipini Öğrenme

Bir değişkenin veri tipini öğrenmek için `type()` fonksiyonunu kullanabiliriz. `type()` fonksiyonu, argüman olarak verilen değişkenin veri tipini döndürür.

Örnek:

```
isim = "Ahmet"
yas = 30
boy = 1.85
ogrenci_mi = True

print(type(isim))      # <class 'str'>
print(type(yas))       # <class 'int'>
print(type(boy))       # <class 'float'>
print(type(ogrenci_mi))  # <class 'bool'>
```

Bu örnekte, `type()` fonksiyonu her değişkenin veri tipini yazdırır. Örneğin, `isim` değişkeni `str` (string) tipinde olduğu için, `type(isim)` ifadesi `<class 'str'>` sonucunu verir.

2.3.2 Type Casting (Tip Dönüşümü)

Bazen bir veri tipini başka bir veri tipine dönüştürmeniz gerekebilir. Örneğin, bir kullanıcıdan alınan string (metin) bir değeri integer (tam sayı) olarak kullanmak isteyebilirsiniz. Python'da bu işlem **type casting** olarak adlandırılır ve `int()`, `float()`, `str()` gibi fonksiyonlar kullanılarak yapılır.

Örnekler:

```
yas = "25"  # yas değişkeni string tipinde
yas_int = int(yas)  # yas değişkenini integer tipine dönüştürdük
print(yas_int + 5)  # Çıktı: 30

fiyat = 99.99 # fiyat değişkeni float tipinde
fiyat_str = str(fiyat) # fiyat değişkenini string tipine dönüştürdük
print("Ürün fiyatı: " + fiyat_str) # Çıktı: Ürün fiyatı: 99.99
```

2.4 Operatörler: Veriler Üzerinde İşlemler Yapma

Operatörler, veriler üzerinde çeşitli işlemler yapmak için kullanılır. Python'da birçok farklı operatör bulunur. İşte bazı temel operatörler:

- **Aritmetik Operatörler:** `+`, `-`, `*`, `/`, `%`, `**`, `//` (Toplama, çıkarma, çarpma, bölme, mod alma, üs alma, tam bölme)
- **Karşılaştırma Operatörleri:** `==`, `!=`, `>`, `<`, `>=`, `<=` (Eşittir, eşit değildir, büyüktür, küçüktür, büyük eşittir, küçük eşittir)
- **Mantıksal Operatörler:** `and`, `or`, `not` (Ve, veya, değil)

Bu operatörleri kullanarak veriler üzerinde matematiksel işlemler yapabilir, karşılaştırmalar yapabilir ve mantıksal ifadeler oluşturabilirsiniz.

Örnekler:

```
sayi1 = 10
sayi2 = 5

toplam = sayi1 + sayi2 # 15
fark = sayi1 - sayi2 # 5
carpim = sayi1 * sayi2 # 50
bolum = sayi1 / sayi2 # 2.0
mod = sayi1 % sayi2 # 0
us = sayi1 ** sayi2 # 100000
tam_bolum = sayi1 // sayi2 # 2

esit_mi = sayi1 == sayi2 # False
buyuk_mu = sayi1 > sayi2 # True

dogru = True
yanlis = False

ve = dogru and yanlis # False
veya = dogru or yanlis # True
degil = not dogru # False
```

2.5 Kullanıcıdan Girdi Alma: `input()` Fonksiyonu

Programlarımızda, kullanıcıdan veri almak için `input()` fonksiyonunu kullanabiliriz. `input()` fonksiyonu, isteğe bağlı olarak bir mesaj (prompt) alır ve kullanıcıdan bir metin girdisi bekler. Kullanıcının girdiği metin, bir string olarak döndürülür.

Örnek:

```
isim = input("Adınızı girin: ")
print("Merhaba", isim + "!")
```

Bu örnekte, `input()` fonksiyonu, "Adınızı girin:" mesajını yazdırır ve kullanıcıdan bir metin girdisi bekler. Kullanıcının girdiği metin, `isim` değişkenine atanır. Ardından, `print()` fonksiyonu, "Merhaba" mesajını ve kullanıcının girdiği ismi ekrana yazdırır.

> ⚠ Önemli Not
>
> `input()` fonksiyonu her zaman string bir değer döndürür. Eğer kullanıcıdan sayısal bir girdi almak istiyorsanız, döndürülen string değeri `int()` veya `float()` fonksiyonları ile tip dönüşümü yapmanız gerekir.

Örnek:

```
yas = int(input("Yaşınızı girin: "))
print("Seneye", yas + 1, "yaşında olacaksınız.")
```

Bu örnekte, kullanıcıdan alınan yaş bilgisi string olarak alınır ve `int()` fonksiyonu ile integer'a dönüştürülür. Ardından, kullanıcının yaşı 1 artırılarak ekrana yazdırılır.

2.6 Yorum Satırları (Comments)

Kodunuza açıklama eklemek için **yorum satırlarını** kullanabilirsiniz. Yorum satırları, Python tarafından çalıştırılmaz ve kodun okunabilirliğini artırmak için kullanılır. Python'da yorum satırları `#` işareti ile başlar. Örneğin:

```
# Bu bir yorum satırıdır.
print("Merhaba Dünya!") # Bu da bir yorum satırıdır.
```

Çok Satırlı Yorum (Docstring)

Çok satırlı yorumlar, üç tane tırnak işareti (`"""`) ile başlar ve biter. Bu yorumlar, birden fazla satıra yayılabilir. Genellikle fonksiyonların veya sınıfların ne işe yaradığını açıklamak için kullanılırlar, aynı zamanda kodun belirli bölümlerini geçici olarak devre dışı bırakmak için de kullanılabilirler.

```
"""
Bu bir çok satırlı yorumdur.
Birden fazla satıra yayılabilir.
Kodun belirli bölümlerini geçici olarak devre dışı bırakmak veya
fonksiyonlar ve sınıflar için detaylı açıklamalar yazmak için kullanışlıdır.
"""

print("Merhaba Dünya!") #  Bu kod çalıştırılır.
```

```
"""
Bu kısım geçici olarak devre dışı bırakıldı.
print("Bu satır çalıştırılmaz.")
print("Bu satır da çalıştırılmaz.")
"""

print("Program devam ediyor...") # Bu kod çalıştırılır.
```