PYTHON 101: PROGRAMLAMAYA GİRİŞ

YAZGİT
(ANKARA ÜNİVERSİTESİ YAPAY ZEKA VE GÖRÜNTÜ İŞLEME TOPLULUĞU)

YAZGİT 01.10.2024
Web Sitesi: www.yazgit.com.tr
Instagram: au_yazgit
YAZAR: Ömer Doğan
EDİTÖR: Elifnur Boyraz

İÇİNDEKİLER

PYTHON101: PROGRAMLAMAYA GİRİŞ

Python 101: Programlamaya Giriş

Giriş

Programlamanın dünyasına hoş geldiniz! Bu kitap, sizi Python programlama dili ile tanıştırarak yazılım geliştirme yolculuğunuza başlamak için bir rehber olacaktır. Python, hem öğrenmesi kolay hem de güçlü bir dil olmasıyla bilinir ve web geliştirme, veri bilimi, yapay zeka gibi pek çok alanda yaygın olarak kullanılır. Bu kitap boyunca, sıfırdan başlayarak Python ile program yazmayı öğreneceksiniz.

Bölüm 1: Giriş ve Python'a Başlangıç

Bu bölümde öğreneceksiniz:

- Yazılım nedir?
- Python nedir ve nerelerde kullanılır?
- Python'un temel yapı taşları: Değişkenler, veri türleri, matematiksel işlemler

1.1 Yazılım Nedir?

Yazılım, bilgisayarların belli bir görevi gerçekleştirebilmesi için verilen talimatlar bütünüdür. Programlama ise bu talimatların yazıldığı sürece verilen isimdir. Yazılım dünyası, işletim sistemlerinden mobil uygulamalara, oyunlardan veri analizine kadar çok geniş bir yelpazeyi kapsar. İşte bu nedenle yazılım, modern dünyada birçok sektörün temelini oluşturur.

1.2 Python Nedir?

Python, 1991 yılında Guido van Rossum tarafından geliştirilen, genel amaçlı ve yüksek seviyeli bir programlama dilidir. Python, özellikle basit ve okunabilir bir sözdizimine sahip olması nedeniyle öğrenmesi kolay bir dildir. Kullanım alanlarına bakıldığında, web geliştirme, bilimsel hesaplamalar, yapay zeka, veri analizi ve daha birçok alanda karşımıza çıkar. Bu çeşitlilik, Python'un popülerliğini artıran önemli faktörlerden biridir.

1.3 Python'un Kurulumu ve İlk Adımlar

Python programlamaya başlamak için bilgisayarınıza Python dilini kurmanız gerekir. Python'un en güncel sürümünü python.org adresinden indirip kurulum yapabilirsiniz.

Kurulum tamamlandığında, Python'un çalışıp çalışmadığını test etmek için komut satırına şu komutu yazabilirsiniz:

```
python --version
```

Bu komut size yüklü olan Python sürümünü gösterecektir. Kurulumun doğru yapıldığını gördüğünüzde, artık ilk Python programınızı yazmaya hazırsınız!

1.4 İlk Program: "Hello, World!"

Bir programlama dilinde geleneksel olarak yazılan ilk program, "Hello, World!" mesajını ekrana yazdırmaktır. Python'da bu program şu şekilde yazılır:

```
print("Hello, World!")
```

Bu kod satırı, Python'un en temel işlevlerinden biri olan print() fonksiyonunu kullanarak ekrana bir metin yazdırır.

1.5 Python'un Temel Yapı Taşları

1.5.1 Değişkenler ve Veri Türleri

Python'da bir program yazarken, verileri saklamak ve işlemek için değişkenler kullanırız. Değişkenler, bilgiyi geçici olarak hafızada tutan kutucuklar gibidir. Python'da birkaç temel veri türü bulunur:

- **String (Metinler):** Yazı tipi veriler. Örneğin: "Merhaba"
- **Integer (Tamsayılar):** Tam sayı değerleri. Örneğin: 42
- **Float (Ondalıklı sayılar):** Ondalık sayılar. Örneğin: 3.14
- **Boolean (Mantıksal değerler):** Doğru ya da yanlış anlamına gelen iki değer: True veya False.

Değişkenlerin nasıl tanımlandığını görelim:

```
isim = "Ömer"
yas = 22
pi = 3.14
ogrenci_mi = True
```

Bu örnekte, isim bir string, yas bir integer, pi bir float ve ogrenci_mi bir boolean değeri tutuyor.

1.5.2 Basit Matematiksel İşlemler

Python'da matematiksel işlemler oldukça basittir. Aşağıdaki gibi temel işlemleri yapabilirsiniz:

```
x = 10
y = 5
toplam = x + y # 15
fark = x - y # 5
carpim = x * y # 50
bolum = x / y # 2.0
```

1.5.3 Uygulama: Basit Bir Hesap Makinesi

Şimdi öğrendiklerimizi bir araya getirerek basit bir hesap makinesi yazalım. Bu program, kullanıcıdan iki sayı alıp toplama işlemi yapacak:

```
# Kullanıcıdan iki sayı alma
sayi1 = float(input("Birinci sayıyı girin: "))
sayi2 = float(input("İkinci sayıyı girin: "))
# Toplama işlemi
toplam = sayi1 + sayi2
# Sonucu ekrana yazdırma
print("Sonuç:", toplam)
```

ile kullanıcıdan veri alıyor, float() ile aldığımız veriyi sayıya dönüştürüyor ve sonucu ekrana yazdırıyoruz.

Bölüm 2: Karar Yapıları ve Döngüler

Bu bölümde programların akışını yönlendiren karar yapıları ve tekrar eden işlemleri gerçekleştiren döngüler hakkında bilgi edineceksiniz. Karar yapıları, bir koşula göre farklı kodların çalıştırılmasını sağlar. Döngüler ise belirli koşullar sağlandığında işlemleri tekrar tekrar yapar.

2.1 Koşullu İfadeler (if, elif, else)

Bir programın, belirli bir koşul doğru olduğunda farklı, yanlış olduğunda farklı işlemler yapmasını istiyorsanız koşullu ifadeleri kullanabilirsiniz. Python'da koşullu ifadeler if, elif ve else anahtar kelimeleri ile yazılır.

2.1.1 Temel if yapısı:

```
yas = 22
if yas >= 18:
 print("Reşitsiniz.")
else:
 print("Reşit değilsiniz.")
```

veya daha büyükse, "Reşitsiniz" mesajı ekrana yazdırılacak. Aksi takdirde "Reşit değilsiniz" mesajı yazdırılır.

2.1.2 elif kullanımı:

elif, "else if" anlamına gelir ve birden fazla koşulun kontrol edilmesini sağlar.

```
sıcaklık = 25
if sıcaklık > 30:
 print("Çok sıcak.")
elif sıcaklık > 20:
 print("Ilık.")
else:
 print("Soğuk.")
```

Bu programda, sıcaklık 30'dan büyükse "Çok sıcak", 20'den büyükse "Ilık", aksi halde "Soğuk" mesajı yazdırılır.

2.1.3 Kullanıcıdan Veri Alma (input)

Bir programı daha etkileşimli hale getirmek için kullanıcıdan veri almak gerekir. Bunu input() fonksiyonu ile yapabilirsiniz. Kullanıcının verdiği bilgiyi işlemeden önce string olarak alır, ancak gerekli olduğunda sayıya dönüştürülmesi gerekir.

```
isim = input("Adınızı girin: ")
yas = int(input("Yaşınızı girin: "))
if yas >= 18:
 print(f"Merhaba {isim}, reşitsiniz.")
else:
 print(f"Merhaba {isim}, reşit değilsiniz.")
```

Bu kodda, kullanıcıdan isim ve yaş alınıyor. Yaş bilgisi sayıya dönüştürülüp bir karar yapısı ile değerlendiriliyor.

2.2 Döngüler

Döngüler, belirli bir koşul sağlandığı sürece tekrar eden işlemler için kullanılır. Python'da iki temel döngü tipi vardır: for ve while döngüleri.

2.2.1 For Döngüsü

for döngüsü, belirli bir sayı veya liste üzerinden tekrar eder.

```
for i in range(5):
 print(i)
```

Bu döngü, 0'dan başlayarak 4'e kadar olan sayıları sırayla ekrana yazdırır. range(5) fonksiyonu, 0'dan başlayarak 5'e kadar (5 dahil değil) bir sayı dizisi oluşturur.

2.2.2 While Döngüsü

while döngüsü, belirli bir koşul doğru olduğu sürece çalışır.

```
sayi = 1
while sayi <= 5:
 print(sayi)
 sayi += 1
```

Bu kod, sayi değişkeni 5'e ulaşana kadar döngüyü tekrar eder ve her tekrarda sayi değişkeni bir artırılır.

2.3 Uygulama: Basit Bir Sayı Tahmin Oyunu

Şimdi öğrendiklerimizi bir oyun yaparak uygulayalım. Bu oyunda bilgisayar rastgele bir sayı seçecek ve kullanıcı bu sayıyı tahmin etmeye çalışacak.

```
import random
# Rastgele bir sayı seç
rastgele_sayi = random.randint(1, 100)
# Kullanıcıdan tahmin al
tahmin = None
while tahmin != rastgele_sayi:
 tahmin = int(input("1 ile 100 arasında bir sayı tahmin edin: "))
 if tahmin < rastgele_sayi:
     print("Daha büyük bir sayı deneyin.")
```

```
elif tahmin > rastgele_sayi:
    print("Daha küçük bir sayı deneyin.")
else:
    print("Tebrikler, doğru tahmin!")
```

Bu oyunda:

- Bilgisayar 1 ile 100 arasında rastgele bir sayı seçiyor.
- Kullanıcı sayı tahmini yapıyor ve bu tahmin rastgele sayı ile karşılaştırılıyor.
- Tahmin doğru değilse, kullanıcıya daha büyük veya daha küçük bir sayı denemesi söyleniyor.
- Doğru tahmin edildiğinde oyun bitiyor.

2.4 Döngüler ve Koşulların Pekiştirilmesi

Bu bölümde öğrendiğiniz bilgileri pekiştirmek için aşağıdaki alıştırmaları yapabilirsiniz:

1. 1'den 10'a kadar olan sayıları toplayan bir Python programı yazın.
2. Kullanıcıdan bir kelime alın ve bu kelimenin her harfini ayrı ayrı yazdıran bir döngü oluşturun.
3. Kullanıcının girdiği bir sayının tek mi çift mi olduğunu bulan bir program yazın.

Bu bölümle birlikte karar yapıları ve döngüler konusunda temel beceriler kazandınız. Bir sonraki bölümde fonksiyonlar ve modüler programlama ile programlarınızı daha verimli hale getirmeyi öğreneceğiz.

Bölüm 3: Fonksiyonlar ve Modüler Programlama

Fonksiyonlar, bir programda tekrar kullanılabilir kod bloklarıdır. Programlarımızı daha okunabilir, düzenli ve yeniden kullanılabilir hale getirmek için fonksiyonlar önemli bir araçtır. Bu bölümde fonksiyonlar hakkında detaylı bilgi edinecek ve Python'un modüler yapısına dair temel bilgileri öğreneceksiniz.

3.1 Fonksiyon Nedir?

Bir fonksiyon, belirli bir görevi yerine getiren ve gerektiğinde çağrılabilen kod parçacığıdır. Fonksiyonlar, aynı işlemi tekrar tekrar yazmak yerine tek bir kez tanımlanıp, birçok kez kullanılabilir.

Python'da bir fonksiyon şu şekilde tanımlanır:

```
def merhaba():
 print("Merhaba, dünya!")
```

Bu örnekte, merhaba() adında bir fonksiyon tanımladık ve bu fonksiyon çağrıldığında ekrana "Merhaba, dünya!" yazdırılacak.

3.1.1 Fonksiyonun Çağrılması

Bir fonksiyonu tanımladıktan sonra, onu çağırarak çalıştırabilirsiniz:

```
merhaba()
```

Bu çağrı, fonksiyonun içindeki kodu çalıştıracak ve ekrana "Merhaba, dünya!" yazdıracaktır.

3.2 Parametreler ve Geri Dönüş Değerleri

Fonksiyonlar, bazen dışarıdan veri alarak bu verilerle işlem yapar ve bir sonuç üretir. Bu tür fonksiyonlar, parametreler ve geri dönüş değerleri ile çalışır.

3.2.1 Parametreli Fonksiyonlar

Bir fonksiyona parametre eklemek, fonksiyonun daha dinamik olmasını sağlar. İşte basit bir örnek:

```
def toplama(a, b):
 sonuc = a + b
 print(f"Toplam: {sonuc}")
```

Bu fonksiyon, iki parametre alır (a ve b) ve bu iki sayıyı toplar. Fonksiyonu şu şekilde çağırabilirsiniz:

```
toplama(5, 10) # Çıktı: Toplam: 15
```

3.2.2 Geri Dönüş Değerleri

Bir fonksiyon bazen bir değer hesaplayıp bu değeri geri döndürmek isteyebilir. Bunun için return ifadesi kullanılır.

```
def carpma(a, b):
 return a * b
```

Bu fonksiyon çağrıldığında, iki sayıyı çarpacak ve sonucu geri döndürecek:

```
sonuc = carpma(4, 5)
print(sonuc) # Çıktı: 20
```

return ifadesi, fonksiyonun sonucu dışarıya iletmesini sağlar, böylece bu değeri başka işlemler için kullanabilirsiniz.

3.3 Python'un Temel Kütüphaneleri

Python, birçok kullanışlı fonksiyon ve veri yapısının yanı sıra geniş bir standart kütüphane sunar. Bu kütüphaneler, belirli görevler için önceden yazılmış fonksiyonlar içerir. Örneğin, matematiksel işlemler için math kütüphanesini, rastgele sayı üretmek için random kütüphanesini kullanabilirsiniz.

3.3.1 math Kütüphanesi

Matematiksel işlemler için Python'un standart math kütüphanesini kullanabilirsiniz. İşte birkaç örnek:

```
import math
print(math.sqrt(16)) # 16'nın karekökünü alır, çıktı: 4.0
print(math.pi) # Pi sayısını yazdırır, çıktı: 3.141592653589793
```

3.3.2 random Kütüphanesi

random kütüphanesi, rastgele sayı üretmek veya rastgele seçimler yapmak için kullanılır:

```
import random
rastgele_sayi = random.randint(1, 100) # 1 ile 100 arasında rastgele
bir sayı üretir
print(rastgele_sayi)
```

3.3.3 datetime Kütüphanesi

Bu kütüphane, tarih ve saat ile ilgili işlemleri yapmanızı sağlar:

```
import datetime
simdi = datetime.datetime.now() # Şu anki tarih ve saati alır
print(simdi)
```

3.4 Uygulama: Rastgele Sayı Tahmin Oyunu (Gelişmiş Versiyon)

Bu bölüme kadar öğrendiklerinizi pekiştirmek için, önceki bölümdeki sayı tahmin oyununu biraz daha geliştirip, fonksiyonlar kullanarak daha modüler hale getirelim.

```
import random
def rastgele_sayi_uret(baslangic, bitis):
 return random.randint(baslangic, bitis)
def tahmin_yap():
 sayi = rastgele_sayi_uret(1, 100)
 tahmin = None
 hak = 0

 while tahmin != sayi:
 tahmin = int(input("1 ile 100 arasında bir sayı tahmin edin: "))
 hak += 1

 if tahmin < sayi:
 print("Daha büyük bir sayı deneyin.")
 elif tahmin > sayi:
 print("Daha küçük bir sayı deneyin.")

 print(f"Tebrikler! Doğru tahmin ettiniz. {hak} denemede buldunuz.")
# Oyunu başlat
tahmin_yap()
```

Bu sürümde, rastgele sayı üretme ve tahmin yapma işlemleri ayrı fonksiyonlar olarak düzenlenmiştir. Böylece kodunuzu daha modüler ve okunabilir hale getirmiş olduk.

3.5 Modüler Programlama

Modüler programlama, bir programı daha küçük, bağımsız modüller halinde organize etme yöntemidir. Her modül, belirli bir işlemi yerine getiren fonksiyonlar içerir. Bu yaklaşım, kodun daha anlaşılır olmasını sağlar ve yeniden kullanılabilirliği artırır.

3.5.1 Modül ve Paket Nedir?

Bir Python dosyası (örneğin, fonksiyonlar.py), bir modül olarak kullanılabilir. Farklı Python dosyalarından fonksiyonları ve değişkenleri çağırabilirsiniz.

Örneğin, fonksiyonlar.py dosyasında şu kodları yazdığınızı düşünelim:

```
def selam_ver():
 print("Merhaba!")
```

Başka bir Python dosyasında bu fonksiyonu şu şekilde kullanabilirsiniz:

```
import fonksiyonlar
fonksiyonlar.selam_ver()
```

Bu bölümde, fonksiyonları nasıl tanımlayıp kullandığınızı, parametreleri nasıl ilettiğinizi ve fonksiyonlardan değerleri nasıl geri döndürebileceğinizi öğrendiniz. Ayrıca Python'un güçlü kütüphanelerini kullanarak programlarınızı nasıl genişletebileceğinizi gördünüz.

Bir sonraki bölümde Python'da listeler, diziler ve daha karmaşık veri yapıları ile nasıl çalışılacağını keşfedeceğiz.

Bölüm 4: Listeler ve Diziler

Bu bölümde Python'da verileri nasıl depolayıp işleyebileceğinizi öğreneceksiniz. Listeler, birden fazla değeri tek bir değişkende saklamanızı sağlayan veri yapılarıdır ve Python'da çok sık kullanılır.

4.1 Listeler

Liste, sıralı ve değiştirilebilir veri yapılarıdır. Python'da listeler köşeli parantezler [] kullanılarak tanımlanır.

```
meyveler = ["elma", "muz", "kiraz"]
print(meyveler) # Çıktı: ['elma', 'muz', 'kiraz']
```

4.1.1 Listeye Eleman Ekleme ve Silme

- Listeye eleman eklemek için append() fonksiyonu kullanılır.

```
meyveler.append("portakal")
print(meyveler) # Çıktı: ['elma', 'muz', 'kiraz', 'portakal']
```

- Listedeki bir elemanı silmek için remove() veya pop() fonksiyonlarını kullanabilirsiniz.

```
meyveler.remove("elma")
print(meyveler) # Çıktı: ['muz', 'kiraz', 'portakal']
meyveler.pop() # Son elemanı siler
print(meyveler) # Çıktı: ['muz', 'kiraz']
```

4.1.2 Listelerle Döngüler

Bir liste üzerinde döngü oluşturmak için for döngüsü kullanabilirsiniz.

```
for meyve in meyveler:
 print(meyve)
```

4.2 Liste Metodları

Python listeleri, çeşitli kullanışlı metodlar sunar. İşte birkaç örnek:

- len(): Listenin uzunluğunu verir.

  ```
  print(len(meyveler)) # Çıktı: 2
  ```

- sort(): Listeyi sıralar.

  ```
  sayilar = [3, 1, 4, 2]
  sayilar.sort()
  print(sayilar) # Çıktı: [1, 2, 3, 4]
  ```

- reverse(): Listeyi tersine çevirir.

  ```
  sayilar.reverse()
  print(sayilar) # Çıktı: [4, 3, 2, 1]
  ```

4.3 Uygulama: Basit Not Defteri

Bu uygulamada kullanıcıdan alınan notlar bir listeye kaydedilecek ve ardından tüm notlar ekrana yazdırılacaktır.

```
notlar = []
while True:
 not_ekle = input("Bir not ekleyin (çıkmak için 'q'): ")

 if not_ekle == 'q':
 break

 notlar.append(not_ekle)
print("Notlarınız:")
for notu in notlar:
 print(notu)
```

Bölüm 5: Dosya İşlemleri

Programlarımızda verilerin kalıcı olmasını istiyorsak, dosya okuma ve yazma işlemlerini kullanabiliriz. Bu bölümde Python ile nasıl dosya okuyup yazabileceğinizi göreceksiniz.

5.1 Dosya Yazma

Bir dosyaya veri yazmak için open() fonksiyonunu write modunda kullanabilirsiniz.

```
with open("notlar.txt", "w") as dosya:
 dosya.write("Bu bir deneme notudur.\n")
```

Bu kod, notlar.txt isimli bir dosya oluşturur ve içine belirtilen metni yazar.

5.2 Dosya Okuma

Bir dosyadan veri okumak için open() fonksiyonu ile dosyayı read modunda açabilirsiniz.

```
with open("notlar.txt", "r") as dosya:
 icerik = dosya.read()
 print(icerik)
```

5.3 Kapsamlı Uygulama: Öğrenci Notları Sistemi

Bu projede öğrenci notlarını dosyaya kaydedeceğiz ve daha sonra dosyadan bu notları okuyacağız.

```
def notlari_yaz():
 with open("ogrenci_notlari.txt", "w") as dosya:
 while True:
 isim = input("Öğrenci ismi (çıkmak için 'q'): ")
 if isim == 'q':
 break
 notu = input(f"{isim} adlı öğrencinin notu: ")
 dosya.write(f"{isim}: {notu}\n")
def notlari_oku():
 with open("ogrenci_notlari.txt", "r") as dosya:
 icerik = dosya.read()
 print("Öğrenci Notları:")
 print(icerik)
# Notları dosyaya yaz
notlari_yaz()
# Notları dosyadan oku
notlari_oku()
```

Sonuç

Bu kitap, Python ile programlamaya giriş yapmanız için gerekli temel bilgileri sundu. Temel veri türlerinden, fonksiyonlar ve döngülere; listelerden dosya işlemlerine kadar pek çok konuya değindik. Öğrendiklerinizle kendi projelerinizi geliştirerek daha ileri seviyede Python becerileri kazanabilirsiniz.

Unutmayın, en iyi öğrenme yöntemi bol bol pratik yapmaktır. Her öğrendiğiniz konuyu uygulamaya dökmek, programlama becerilerinizi hızla ilerletecektir.