Programlamaya Giriş kursuna hoş geldiniz! Daha önce tek bir satır kod yazmadıysanız ve veri bilimi ve makine öğrenimi öğrenmekle ilgileniyorsanız, bu kurs tam size göre. (Programlama deneyiminiz varsa ve Python diline yeni başlıyorsanız, Python kursu başlamak için daha uygun bir seçenektir.)

Bu kursta, bir bilgisayarın sizin için belirli görevleri gerçekleştirmesini sağlamak için kodu nasıl kullanacağınızı öğreneceksiniz. Python, veri bilimi için en popüler programlama dillerinden biridir ve bu kursta öğreneceğiniz dildir. Bu kursu tamamladığınızda, Python kursuna ve ardından Makine Öğrenimine Giriş kursuna geçmeye tamamen hazır olacaksınız.

Bu eğitimde, Python koduna dair birkaç örnek göreceksiniz. Alıştırmada kendi kodunuzu yazma fırsatı bulacaksınız. (Alıştırmayı önizlemek isterseniz, hemen göz atabilirsiniz. Ayrıca, bu eğitimin sonunda alıştırmaya bir bağlantı da sunuyoruz.)

Printing

Bir bilgisayardan isteyebileceğiniz en basit (ve en önemli!) görevlerden biri, bir mesajı yazdırmak.

Python'da, print() yazarak ve mesajı parantez içine alıp tırnak işaretleri arasına alarak bilgisayardan bizim için bir mesaj yazdırmasını istiyoruz. Aşağıda, bilgisayardan "Merhaba dünya!" mesajını yazdırmasını istiyoruz.

print("Hello, world!")

Hello, world!

Kod, kutunun (kod hücresi olarak bilinir) içindedir ve bilgisayarın yanıtı (kodun çıktısı olarak adlandırılır) kutunun altında gösterilir. Gördüğünüz gibi, bilgisayar istediğimiz mesajı yazdırdı.

# Arithmetic

Ayrıca bazı aritmetik işlemlerin (toplama, çıkarma, çarpma veya bölme gibi) değerini de yazdırabiliriz.

Örneğin, bir sonraki kod hücresinde bilgisayar 2'yi 1'e ekler ve ardından sonucu, yani 3'ü yazdırır. Sadece metin yazdırırken kullandığımızın aksine, tırnak işareti kullanmadığımızı unutmayın.

print(1 + 2)

3

Python'da çıkarma işlemi de yapabiliriz. Bir sonraki kod hücresi 9'dan 5'i çıkarır ve sonucu yazdırır: 4.

print(9 - 5)

4

Aslında Python ile pek çok hesaplama yapabilirsiniz! Bazı örnekler için aşağıdaki tabloya bakın.

|  |  |  |
| --- | --- | --- |
| **Operation** | **Symbol** | **Example** |
| Addition | + | 1 + 2 = 3 |
| Subtraction | - | 5 - 4 = 1 |
| Multiplication | \* | 2 \* 4 = 8 |
| Division | / | 6 / 3 = 2 |
| Exponent | \*\* | 3 \*\* 2 = 9 |

Uzun hesaplamalarda işlem sırasını parantez kullanarak kontrol edebilirsiniz.

print(((1 + 3) \* (9 - 2) / 2) \*\* 2)

196.0

Genel olarak Python, işlem sırasını belirlerken PEMDAS kuralını takip eder.

# Comments

Kodun ne yaptığını açıklamak için yorumlar kullanırız. Başkalarının kodunuzu anlamasına yardımcı olurlar ve ayrıca kendi kodunuza bir süredir bakmadıysanız da faydalı olabilirler. Şimdiye kadar yazdığımız kod oldukça kısa, ancak çok fazla kod yazdığınızda açıklamalar daha da önemli hale gelir.

Örneğin, bir sonraki kod hücresinde 3'ü 2 ile çarpıyoruz. Ayrıca, kodun ne yaptığını açıklamak için kodun üzerine bir yorum (# 3'ü 2 ile çarp) ekliyoruz.

*# Multiply 3 by 2*

print(3 \* 2)

6

Python'a bir satırın yorum olduğunu (ve Python kodu olmadığını) belirtmek için, ilk karakter olarak bir diyez işareti (#) yazmanız gerekir.

Python, diyez işaretini görüp satırın bir yorum olduğunu anladığında, bilgisayar tarafından tamamen yok sayılır. Bu önemlidir, çünkü tıpkı İngilizce veya Hintçe (veya başka herhangi bir dil!) gibi, Python da uyulması gereken çok katı kurallara sahip bir dildir. Ancak Python, bir insan dinleyiciden daha katıdır ve kodu anlayamadığında hata verir.

Bunun bir örneğini aşağıdaki kod hücresinde görebiliriz. Diyez işaretini kaldırırsak Python hata verir, çünkü yorumdaki metin geçerli bir Python kodu değildir ve bu nedenle doğru şekilde yorumlanamaz.

Multiply 3 by 2

File "/tmp/ipykernel\_18/3750420471.py", line 1

Multiply 3 by 2

^

SyntaxError: invalid syntax

# Variables

Şimdiye kadar bir hesaplama yapmak ve sonucu yazdırmak için kod kullandınız ve sonuç hiçbir yere kaydedilmedi. Ancak, daha sonra üzerinde çalışmak için sonucu kaydetmek isteyebileceğinizi tahmin edebilirsiniz. Bunun için değişkenler kullanmanız gerekecek.

## **Creating variables**

Bir sonraki kod hücresi test\_var adlı bir değişken oluşturur ve bu değişkene 5'i 4'e eklediğimizde elde ettiğimiz değeri atar.

Ardından değişkene atanan değeri, yani 9'u yazdırırız.

*# Create a variable called test\_var and give it a value of 4+5*

test\_var = 4 + 5

*# Print the value of test\_var*

print(test\_var)

9

Genel olarak, bir değişkenle çalışmak için, kullanmak istediğiniz adı seçerek başlamanız gerekir. Değişken adları ideal olarak kısa ve açıklayıcıdır. Ayrıca birkaç gereksinimi karşılamaları gerekir:

Boşluk içeremezler (örneğin, test değişkenine izin verilmez)

Yalnızca harf, rakam ve alt çizgi içerebilirler (örneğin, test\_var! değişkenine izin verilmez)

Bir harf veya alt çizgiyle başlamalıdırlar (örneğin, 1\_var değişkenine izin verilmez)

Ardından, değişkeni oluşturmak için, istediğiniz değeri atamak üzere = işaretini kullanmanız gerekir.

print() kullanarak ve değişkenin adını parantez içine alarak değişkene atanan değere her zaman göz atabilirsiniz.

Zamanla, Python değişkenleri için iyi adlar seçmeyi öğreneceksiniz. Şimdilik rahatsız edici hissetmeniz tamamen normal ve öğrenmenin en iyi yolu bolca Python kodu incelemektir!

## **Manipulating variables**

Bir değişkene atanan değeri, önceki değeri geçersiz kılarak istediğiniz zaman değiştirebilirsiniz.

Aşağıdaki kod hücresinde, my\_var değişkeninin değerini 3'ten 100'e değiştiriyoruz.

*# Set the value of a new variable to 3*

my\_var = 3

*# Print the value assigned to my\_var*

print(my\_var)

*# Change the value of the variable to 100*

my\_var = 100

*# Print the new value assigned to my\_var*

print(my\_var)

3

100

Genel olarak, bir kod hücresinde bir değişken tanımladığınızda, onu takip eden tüm kod hücrelerinin de değişkene erişebileceğini unutmayın. Örneğin, bir sonraki kod hücresini, yukarıdaki kod hücresinden my\_var ve bu eğitimin önceki bölümlerinden test\_var değerlerine erişmek için kullanırız.

print(my\_var)

print(test\_var)

100

9

Bir sonraki kod hücresi, Python'a my\_var değişkeninin mevcut değerini 3 artırmasını söyler.

Bunu yapmak için, daha önce olduğu gibi my\_var = kullanmamız gerekir. Ayrıca, tıpkı daha önce olduğu gibi, değişkene atamak istediğimiz yeni değer = işaretinin sağındadır.

*# Increase the value by 3*

my\_var = my\_var + 3

*# Print the value assigned to my\_var*

print(my\_var)

103

## **Using multiple variables**

Kodun birden fazla değişken kullanması yaygındır. Bu, özellikle birden fazla girdiyle uzun bir hesaplama yapmamız gerektiğinde faydalıdır.

Bir sonraki kod hücresinde, dört yıldaki saniye sayısını hesaplıyoruz. Bu hesaplama beş girdi kullanır.

*# Create variables*

num\_years = 4

days\_per\_year = 365

hours\_per\_day = 24

mins\_per\_hour = 60

secs\_per\_min = 60

*# Calculate number of seconds in four years*

total\_secs = secs\_per\_min \* mins\_per\_hour \* hours\_per\_day \* days\_per\_year \* num\_years

print(total\_secs)

126144000

Yukarıda hesaplandığı gibi, dört yılda 126144000 saniye vardır.

Bu hesaplamayı değişkenler olmadan sadece 60 \* 60 \* 24 \* 365 \* 4 olarak yapmak mümkündür, ancak değişkenler olmadan hesaplamanın herhangi bir hata içermediğinden emin olmak çok daha zordur, çünkü okunabilirliği düşüktür. Değişkenler (örneğin num\_years, days\_per\_year vb.) kullandığımızda, hesaplamanın her bir bölümünü daha iyi takip edebilir ve hataları daha kolay kontrol edip düzeltebiliriz.

Girdi değerlerinin değişebileceği durumlarda değişken kullanmanın özellikle faydalı olduğunu unutmayın. Örneğin, artık yılları hesaba katmak için bir yıldaki gün sayısının değerini 365'ten 365,25'e güncelleyerek tahminimizi biraz iyileştirmek istediğimizi varsayalım. Ardından, diğer değişkenleri değiştirmeden days\_per\_year'a atanan değeri değiştirebilir ve hesaplamayı yeniden yapabiliriz.

*# Update to include leap years*

days\_per\_year = 365.25

*# Calculate number of seconds in four years*

total\_secs = secs\_per\_min \* mins\_per\_hour \* hours\_per\_day \* days\_per\_year \* num\_years

print(total\_secs)

126230400.0

Not: Sayının sonuna eklenen .0'ı fark etmiş olabilirsiniz; bu gereksiz görünebilir. Bu durum, ikinci hesaplamada kesirli kısmı olan bir sayı (365,25) kullanmamızdan, ilk hesaplamada ise kesirli kısmı olmayan sayıların çarpılmasından kaynaklanmaktadır. Veri türlerini ele aldığımız 3. Ders'te bu konu hakkında daha fazla bilgi edineceksiniz.

## **Debugging**

Değişkenlerle çalışırken sık karşılaşılan hatalardan biri, yanlışlıkla yazım hataları yapmaktır. Örneğin, hours\_per\_day ifadesini hours\_per\_dy olarak yazarsak, Python "NameError: name 'hours\_per\_dy' is not defined" hatasıyla karşılaşır.

print(hours\_per\_dy)

---------------------------------------------------------------------------

NameError Traceback (most recent call last)

/tmp/ipykernel\_18/142450907.py in <module>

----> 1 print(hours\_per\_dy)

NameError: name 'hours\_per\_dy' is not defined

NameError'ı bu şekilde gördüğünüzde, "tanımlanmamış" olarak referans verdiği değişkenin yazımını kontrol etmeniz gerektiği anlamına gelir. Hatayı düzeltmek için sadece yazımı düzeltmeniz yeterlidir.

print(hours\_per\_day)

24