Bu kurs, veri biliminde Python kullanmaya başlamanız için ihtiyaç duyacağınız temel Python becerilerini kapsar. Bu kurs, daha önce kodlama deneyimi olan ve Python'ı repertuarına eklemek isteyen kişiler için idealdir. (İlk kez kod yazıyorsanız, Python'a başlamak isteyen yeni başlayanlar için tasarlanmış Programlamaya Giriş kursumuza göz atmanızı öneririz.)

Python sözdizimi, değişken atama ve aritmetik operatörlerine kısa bir genel bakışla başlayacağız.

Hello, Python!

Python, adını İngiliz komedi topluluğu Monty Python'dan alıyor, bu yüzden ilk Python programımızı onların Spam hakkındaki skeçine bir saygı duruşu olarak hazırlayacağız.

Sadece eğlenmek için, aşağıdaki kodu okuyup çalıştırıldığında ne yapacağını tahmin etmeye çalışın. (Hiçbir fikriniz yoksa sorun değil!)

Ardından programımızın sonuçlarını görmek için "çıktı" düğmesine tıklayın.

spam\_amount = 0

print(spam\_amount)

*# Ordering Spam, egg, Spam, Spam, bacon and Spam (4 more servings of Spam)*

spam\_amount = spam\_amount + 4

if spam\_amount > 0:

print("But I don't want ANY spam!")

viking\_song = "Spam " \* spam\_amount

print(viking\_song)

0

But I don't want ANY spam!

Spam Spam Spam Spam

Burada açıklanacak çok şey var! Bu komik program, Python kodunun nasıl göründüğüne ve nasıl çalıştığına dair birçok önemli noktayı gösteriyor. Kodu baştan sona inceleyelim.

spam\_amount = 0

Değişken ataması: Burada spam\_amount adlı bir değişken oluşturuyoruz ve atama operatörü olarak adlandırılan = kullanarak ona 0 değerini atıyoruz.

Not: Eğer Java veya C++ gibi belirli dillerde programlama yaptıysanız, Python'un burada bizden talep etmediği bazı şeyleri fark etmiş olabilirsiniz:

spam\_amount'u atamadan önce "bildirmemize" gerek yok

Python'a spam\_amount'un hangi tür bir değere atıfta bulunacağını söylememize gerek yok. Hatta spam\_amount'u dize veya boolean gibi farklı bir şeye atıfta bulunacak şekilde yeniden atayabiliriz.

print(spam\_amount)

0

Fonksiyon çağrıları:.print, kendisine iletilen değeri ekranda görüntüleyen bir Python fonksiyonudur. Fonksiyonları, adlarının sonuna parantez koyarak ve fonksiyona ait girdileri (veya argümanları) bu parantezlerin içine koyarak çağırırız.

*# Ordering Spam, egg, Spam, Spam, bacon and Spam (4 more servings of Spam)*

spam\_amount = spam\_amount + 4

Yukarıdaki ilk satır bir yorumdur. Python'da yorumlar # sembolüyle başlar.

Sonra bir yeniden atama örneği göreceğiz. Mevcut bir değişkenin değerini yeniden atamak, bir değişken oluşturmakla aynı şekilde çalışır; yine de = atama operatörünü kullanır.

Bu durumda, spam\_amount'a atadığımız değer, önceki değeri üzerinde basit bir aritmetik işlem gerektirir. Python bu satırla karşılaştığında, = ifadesinin sağ tarafındaki ifadeyi (0 + 4 = 4) değerlendirir ve ardından bu değeri sol taraftaki değişkene atar.

if spam\_amount > 0:

print("But I don't want ANY spam!")

viking\_song = "Spam Spam Spam"

print(viking\_song)

But I don't want ANY spam!

Spam Spam Spam

"Koşullar" hakkında daha sonra çok fazla konuşmayacağız, ancak daha önce hiç kod yazmamış olsanız bile, bunun ne işe yaradığını muhtemelen tahmin edebilirsiniz. Python, okunabilirliği ve basitliğiyle değerlidir.

Hangi kodun if'e ait olduğunu nasıl belirttiğimize dikkat edin. "Ama HİÇBİR spam istemiyorum!" ifadesi yalnızca spam\_amount pozitif olduğunda yazdırılmalıdır. Ancak sonraki kod (örneğin print(viking\_song)) ne olursa olsun çalıştırılmalıdır. Bunu nasıl biliyoruz (ve Python)?

If satırının sonundaki iki nokta üst üste (:), yeni bir kod bloğunun başladığını gösterir. Girintili sonraki satırlar, o kod bloğunun bir parçasıdır.

Not: Daha önce kod yazdıysanız, bazı dillerin kod bloklarının başlangıcını ve sonunu işaretlemek için {süslü parantez} kullandığını biliyor olabilirsiniz. Python'ın anlamlı boşluk kullanımı, diğer dillere alışkın programcılar için şaşırtıcı olabilir, ancak pratikte, kod bloklarının girintili olmasını zorunlu kılmayan dillere göre daha tutarlı ve okunabilir kodlar elde etmenizi sağlayabilir.

Viking\_song ile ilgili sonraki satırlar 4 boşlukla girintili olmadığından, if kod bloğunun bir parçası değiller. Girintili kod bloklarına dair daha fazla örneği, fonksiyonları tanımlarken ve döngüleri kullanırken daha sonra göreceğiz.

Bu kod parçası aynı zamanda Python'da bir dizeyi ilk görüşümüz:

"But I don't want ANY spam!"

Dizeler çift veya tek tırnak işaretiyle işaretlenebilir. (Ancak bu özel dize tek tırnak karakteri içerdiğinden, dikkatli olmazsak Python'u tek tırnak işaretiyle çevrelemeye çalışarak karıştırabiliriz.)

viking\_song = "Spam " \* spam\_amount

print(viking\_song)

Spam Spam Spam Spam

\* operatörü iki sayıyı çarpmak için kullanılabilir (3 \* 3, 9'a eşit olur), ancak bir dizeyi bir sayıyla çarparak, o kadar çok tekrarlanmış bir sürüm elde edebiliriz. Python, \* ve + gibi operatörlerin, ne tür bir şeye uygulandığına bağlı olarak farklı anlamlara geldiği, bunun gibi bir dizi küçük ve zaman kazandıran hile sunar. (Bunun teknik terimi operatör aşırı yüklemesidir.)

## **Numbers and arithmetic in Python**

Yukarıda sayı içeren bir değişkenin örneğini gördük:

spam\_amount = 0

"Number" bu tür şeyler için güzel bir gayrı resmi isimdir, ancak daha teknik olmak istersek Python'a spam\_amount'un ne tür bir şey olduğunu nasıl tanımlayacağını sorabiliriz:

type(spam\_amount)

int

Bu bir int'tir - tam sayının kısaltmasıdır. Python'da sıkça karşılaştığımız başka bir sayı türü daha var:

type(19.95)

float

Kayan noktalı sayı, ondalık basamağı olan bir sayıdır; ağırlıklar veya oranlar gibi şeyleri temsil etmek için çok kullanışlıdır.

type(), gördüğümüz ikinci yerleşik fonksiyondur (print()'ten sonra) ve akılda tutulması gereken bir diğer önemli fonksiyondur. Python'a "Bu ne tür bir şey?" diye sorabilmek çok kullanışlıdır.

Sayılarla ilgili yapmak isteyeceğiniz doğal bir şey aritmetiktir. Toplama için + operatörünü ve çarpma için \* operatörünü gördük. Python, hesap makinenizdeki diğer temel düğmeler için de bize yardımcı oluyor:

| Operator | Name | Description |
| --- | --- | --- |
| a + b | Addition | Sum of a and b |
| a - b | Subtraction | Difference of a and b |
| a \* b | Multiplication | Product of a and b |
| a / b | True division | Quotient of a and b |
| a // b | Floor division | Quotient of a and b, removing fractional parts |
| a % b | Modulus | Integer remainder after division of a by b |
| a \*\* b | Exponentiation | a raised to the power of b |
| -a | Negation | The negative of a |

Buradaki ilginç bir gözlem, hesap makinenizde muhtemelen yalnızca bir bölme düğmesi varken, Python'ın iki tür bölme işlemi yapabilmesidir. "Gerçek bölme" temel olarak hesap makinenizin yaptığı şeydir:

print(5 / 2)

print(6 / 2)

2.5

3.0

Bize her zaman bir kayan noktalı sayı verir.

// operatörü bize bir sonraki tam sayıya yuvarlanmış bir sonuç verir.

print(5 // 2)

print(6 // 2)

2

3

Bunun nerede işe yarayacağını düşünebiliyor musunuz? Yakında kodlama yarışmalarında bir örneğini göreceksiniz.

### **Order of operations**

İlkokulda öğrendiğimiz aritmetikte, işlemlerin değerlendirilme sırasına ilişkin kurallar vardır. Bazıları bunları PEMDAS - Parantez, Üs, Çarpma/Bölme, Toplama/Çıkarma gibi bir anımsatıcıyla hatırlar.

Python, hangi hesaplamaların önce yapılacağı konusunda benzer kuralları izler. Bunlar çoğunlukla oldukça sezgiseldir.

8 - 3 + 2

7

-3 + 4 \* 2

5

Bazen varsayılan işlem sırası istediğimiz gibi olmayabilir:

hat\_height\_cm = 25

my\_height\_cm = 190

*# How tall am I, in meters, when wearing my hat?*

total\_height\_meters = hat\_height\_cm + my\_height\_cm / 100

print("Height in meters =", total\_height\_meters, "?")

Height in meters = 26.9 ?

Parantezler burada kullanışlıdır. Python'ın alt ifadeleri istediğiniz sırayla değerlendirmesini sağlamak için parantez ekleyebilirsiniz.

total\_height\_meters = (hat\_height\_cm + my\_height\_cm) / 100

print("Height in meters =", total\_height\_meters)

Height in meters = 2.15

### **Builtin functions for working with numbers**

min ve max, sırasıyla argümanlarının minimum ve maksimum değerlerini döndürür...

print(min(1, 2, 3))

print(max(1, 2, 3))

1

3

abs, bir argümanın mutlak değerini döndürür:

print(abs(32))

print(abs(-32))

32

32

Python'un iki ana sayısal tipi olan int ve float'ın adları olmasının yanı sıra, argümanlarını karşılık gelen tipe dönüştüren fonksiyonlar olarak da çağrılabilirler:

print(float(10))

print(int(3.33))

*# They can even be called on strings!*

print(int('807') + 1)

10.0

3

808