Global ve Yerel değişkenler

Bu konuda **global** ve **yerel (local)** değişkenleri öğrenmeye çalışacağız.

Pythonda her bir değişkenin, fonksiyonun ve ileride göreceğimiz sınıfların(class) aslında bir **kapsamı(scope)** bulunur ve Python herbir değişkeni bir **isim alanında (namespace)** tanımlar. Değişkenlerin **İsim alanı** ise, bu değişkenlerin nerelerde var olduğunu ve nerelerde kullanılabileceğini gösterir.

Pythonda fonksiyonlarda tanımlanan değişkenler Python tarafından **Yerel (Local) değişkenler** olarak tanımlanırlar. Yani bir **fonksiyon bloğunda** oluşturulan değişkenler fonksiyona özgüdür ve **fonksiyon çalışmasını** bitirdikten sonra bu değişkenler bellekten silinir ve yok olur. Böylelikle , fonksiyon içinde tanımlanmış bir değişkene başka bir yerden **erişilemez**.

Pythonda en genel kapsama sahip değişkenler ise **Global değişkenler** olarak tanımlanırlar ve global değişkenlere **tanımlandığı andan itibaren** programın her yerinden ulaşabiliriz.

Yerel değişkenleri anlamak için bir tane fonksiyon tanımlayalım.

In [3]:
```
def fonksiyon():
    a = 10 # Yerel isim alanında bir değişken
    print(a)


fonksiyon()
print(a)  # a değişkeni yok oldu.
```

```
10

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-58e149712583> in <module>()
      5
      6 fonksiyon()
----> 7 print(a)

NameError: name 'a' is not defined
```

Burada fonksiyon içinde tanımlanan **a** değişkeni **fonksiyon çağrıldığında** bellekte oluşur ve fonksiyon bloğunu çalıştırdıktan sonra **yok olur**. Yani, **a** değişkeni burada bir ***yerel değişkendir***.

Global Değişkenleri anlamak içinse şöyle bir örnek yapalım.

In [4]:
```
a = 5 # Global isim alanında bir değişken .

def fonksiyon():
    print(a) # a değişkeni globalde tanımlandığı için burada tanımlı.

fonksiyon()
```

```
5
```

Peki şöyle bir kodda nasıl bir sıkıntı çıkıyor ?

In [7]:
```
def fonksiyon():
    print(s)

fonksiyon() # s global değişkeni henüz tanımlanmadığı için Python hata veriyor.
s = "Python"
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-7-1065c9737baa> in <module>()
      2     print(s)
      3
----> 4 fonksiyon() # s global değişkeni henüz tanımlanmadığı için Python hata veriy
or.
      5 s = "Python"

<ipython-input-7-1065c9737baa> in fonksiyon()
      1 def fonksiyon():
----> 2     print(s)
      3
      4 fonksiyon() # s global değişkeni henüz tanımlanmadığı için Python hata veriy
or.
      5 s = "Python"

NameError: name 's' is not defined
```

Peki şöyle bir durumda kodumuz nasıl bir çıktı verecek ?

In [12]:
```
c = 10 # Globalde tanımlanmış bir değişken
def fonksiyon():
    c = 2 # Yerelde tanımlanmış bir değişken
    print(c)  # Yerel değişken kullanılıyor.

fonksiyon()
print(c)
```

```
2
10
```

Kodumuz çalıştığında ilk olarak **c** isimli global değişken oluşuyor. **fonksiyon çağrıldığında ise c** isimli başka bir yerel değişken oluşuyor gibi düşünebilirsiniz. Böyle bir durumda elimizden iki tane **c** değişkeni var. Python bu durumda **global c** değişkeni yerine **kendi yerel c** değişkenini kullanıyor.

Global Deyimi

Peki bir fonksiyonda globalde tanımlanmış bir değişkeni nasıl kullanacağız ? Bunun için Pythonda **global** ifadesi bulunmaktadır. Şimdi aşağıdaki kodu beraber inceleyelim.

In [13]:
```
d = 10

def fonksiyon():
    global d

    d = 4
    print(d)
fonksiyon()
print(d)
```

```
4
4
```

Bu durumda kodumuz ne yapıyor ? İlk olarak program başladığı zaman, bir tane **global d** değişkeni oluşuyor ve fonksiyonumuz çağrıldığında **global d** ifadesiyle globaldeki **d** değişkenini kullanmak istediğimizi söylüyoruz. Böyle **d = 4** ifadesiyle bir tane daha **d** değişkeni **oluşturmuyoruz**. Böylelikle **d =4** ifadesiyle **globaldeki değişkeninin değerini** değiştirmiş oluyoruz.

İşte **Global** ve **Yerel** değişkenler bu şekilde düşünülebilir. Burada gördüğümüz gibi, **Yerel değişkenler** bir fonksiyon bloğunda içinde tanımlanır. Peki bir **if veya while** bloğunda **yerel bir değişken** tanımlanır mı hemen bakalım.

In [14]:
```
if True:
    t = 10
    print(t)

print(t)
```

```
10
10
```

In [15]:
```
while True:
    deger =  10
    print(deger)
    break

print(deger)
```

```
10
10
```

Burada gördüğümüz gibi, **if ve while** bloklarında tanımlanan değişkenler **yerel bir değişken yerine global bir değişken** olmaktadır.

Bir sonraki konuda **lambda ifadelerini** öğrenmeye çalışacağız.

In [ ]:
```

```