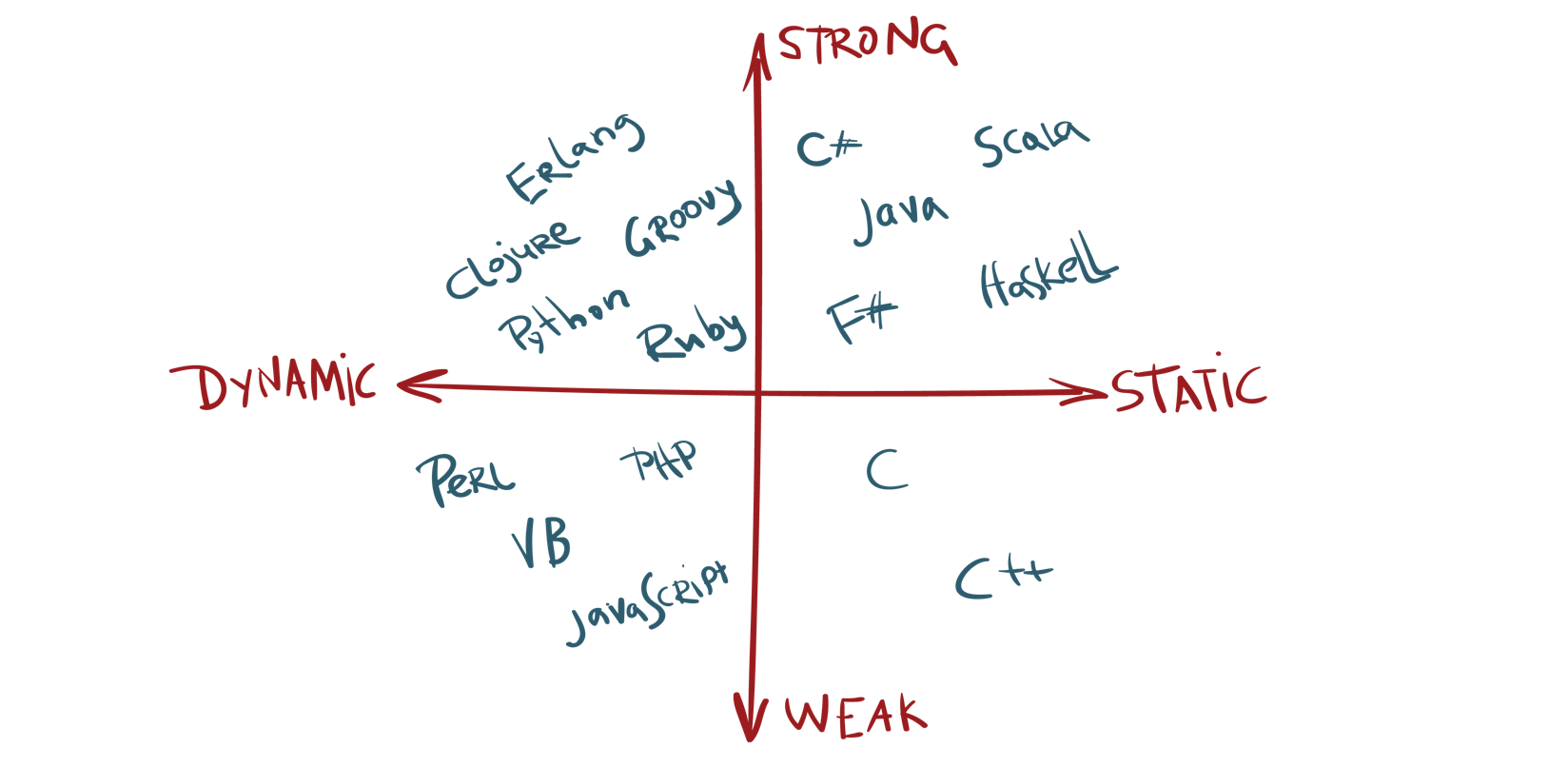
**Python’ın C# İle Karşılaştırılması**



Yeni bir programlama dili öğrenmenin en etkili yollarından biri daha önceden bildiğimiz farklı bir dil ile kıyaslamaktır. Böylece öğrendiğimiz yeni dilin eksiklerini, artılarını, neleri farklı sunduğunu daha rahat gözlemleyebiliriz. Bu durum sadece yeni bir programlama dili öğrenirken değil, pek çok farklı alan için de geçerlidir.

Bu yazıda Python’ı karşılaştırmak için C#’ı kullanacağız. C#’ı seçmemdeki sebep iki dilin de nesne yönelimli olması, yüksek seviyeli diller olmaları ve C#’ın Python’dan farklı olarak statik tipli (statically typed or strongly typed) bir dil olması. Python, C#’ın aksine dinamik tipli (dynamically typed) bir dildir. Bu sayede bazı yönleriyle benzeşen bazı yönleriyle ayrışan iki dili karşılaştırarak, çeşitli senaryolarda hangi dili kullanmanın daha işlevli olacağına karar verebilir hale geleceğiz. Aynı zamanda da Python’ın özelliklerini öğrenmiş olacağız.

**Statically Typed - Dynamically Typed**



**Statik Tipli (Statically Typed)**

Statically typed en basit tanımıyla, dil içerisindeki tüm değişkenlerin tip zorunluluğunu (int, float, string, int array, list vb.) ifade eder. Yani değişkenler tanımlanırken tipleri belirtilmek zorundadır. Aksi takdirde compile-time (derleme anı) hatalarıyla karşılaşırız.

**Dinamik Tipli (Dynamically Typed)**

Dynamically typed kavramı ise en basit tanımıyla değişkenlerin tip zorunluluğunun olmadığını ifade eder. Yani Python’da bir değişken tanımlanırken bu değişkenin tipinin özel olarak *“int a”* gibi belirtilmesine gerek yoktur.

**Aradaki Fark Tam Olarak Nedir?**

Dinamik tipli diller tip kontrolünü yürütme esnasında (run time) gerçekleştirirler. Öte yandan statik tipli diller tip kontrolünü derleme (compile) esnasında yaparlar. Dinamik tipli dillerin bu özelliği içerisinde problem olan bir script’in ne olursa olsun compile edileceği anlamına gelir. Örnekle somutlaştırmak gerekirse, elimizde bozuk bir araba olmasına rağmen bu arabanın bozukluğunu ancak çalıştırdığımızda anlayabiliriz yani çalıştırmadan bozukluğu fark edemeyiz.

C#’ta değişken tanımlamaya bir örnek:

int a = 5;

Python’da değişken tanımlamaya bir örnek:

a = 5

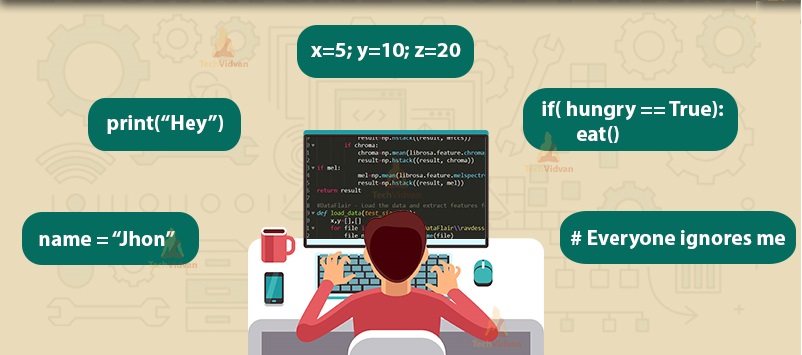
Özet olarak Python bir değişkenin tipini içerisindeki değeri kontrol ederek kendi belirler. C# ise bizden değişkenin tipini girmemizi beklemektedir. Bunun olası sonuçlarını daha iyi anlayabilmemiz için şöyle bir örnek kullanabiliriz:

variable = 5

varible = (variable + 25) / 2

Yukarıdaki örnekte ikinci satırda “variable” yazmak yerine yanlışlıkla “varible” yazdık. Bu konuda hiçbir uyarı almayacağız çünkü Python bizden değişkenler için tip tanımlaması (declare) beklemiyor. Sonuç olarak dinamik tipli, geliştiriciye hız ve esneklik sağlasa da güvenli kod yazımı açısından statik tiplinin gerisinde kalmaktadır.

**Syntax (Sözdizim)**



Python’ın, C# ve benzeri yüksek seviyeli dillerle ayrıştığı bir diğer nokta “indentation” (girinti) kullanımıdır. Bir kod bloğunu C#’ta süslü parantez içerisinde tanımlarken, Python’da bu işlem için girinti kullanılır. Örnek olarak:

*C#’ta bir if bloğu:*

if (a<5) {

a = a \* 5;

Console.WriteLine(“curly braces”);

}

*Python’da bir if bloğu:*

if a < 5:

a = a \* 5

print(“indentation”)

Yukarıdaki örnekte Python’da if bloğunun girinti ile nasıl oluşturulduğunu görüyoruz. Bir girinti tanımlamak için minimum bir space alanı bırakmanız gerekmektedir. Üst sınırı yoktur. Fakat programın okunurluğu ve tutarlılığı açısından genellikle bir tab space’i kadar boşluk bırakmak tercih edilir.



**Önemli**: Python’da değişken, fonksiyon, sınıf gibi değerlere isim verirken diğer dillerde kullanılan camelCase tarzı (ilk kelime hariç değişken ismindeki tüm kelimelerin büyük harfle başlaması ve bitişik yazılmaları barışCanKurt gibi) kullanılmaz. Python’da bunun yerine okunabilirliği arttırmak için alt tire “\_” notasyonu kullanılır, “barış\_can\_kurt” gibi.

**Comment (Yorum)**

C#’ta tek satırlı yorum satırı yapmak için “//” ifadesini kullanılır. Çok satırlı yorum için yorumun başlangıcını “/\*” sembolüyle bitişini de “\*/” sembolüyle ifade ederiz. Bu ikisinin arasında kalan alan compile edilmez. Örnek:

//bu tek satırlı bir yorumdur

/\* bu çok

satırlı

bir yorumdur\*/

Python’da tek satırlı yorum için “#” sembolü kullanılır. Çok satırlı yorum içinse başlangıç ve bitişte “”” (üç çift tırnak) kullanılır. Örnek:

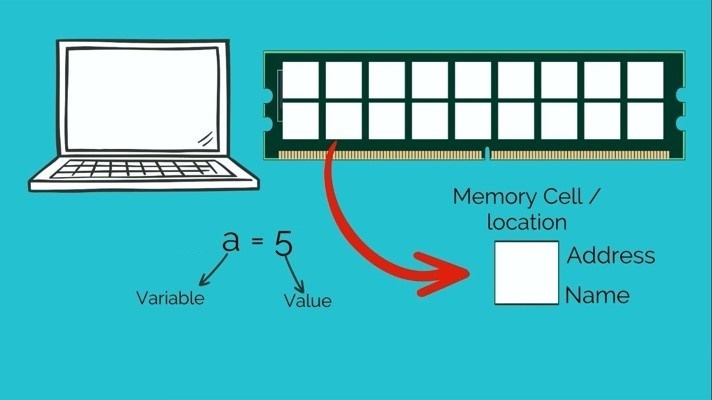
# bu tek satırlı bir yorumdur

“”” bu çok satırlı

Bir yorumdur

python için “””

**Variables (Değişkenler)**



Python’da variables (değişkenler) içerisine değer atandığında oluşturulurlar. Yani aşağıdaki örnekte x değişkenine 2 farklı değer atayarak 2 kere yaratmış oluyoruz. Her x atama esnasında tekrardan yaratıldığı için farklı adreslere sahip.

x = 3

x = 5

Bunu C#’ta aşağıdaki örnekteki gibi yapmış olsaydık değişkeni bir kere yaratıp 2 kere değer atamış olurduk. C#’ta değer ataması yeni bir değişken yaratmaz.

int x = 5;

x = 3;

**Multiple Value Assigning (Çoklu Değişken Atama)**

Python’da çoklu değişkenlere farklı veri atamak için şu yazım kullanılır:

a, b, c = “baris”, 2, “can”

Eğer çoklu değişkenlere aynı veriyi atamak istersek şu yazımı kullanırız:

a = b = c = “baris”

**Output Variables (Değişkenleri Yazdırma)**

Python’ın print kullanımı diğer yüksek seviyeli dillerle hemen hemen aynı syntax’a sahiptir.

x = “awesome”

print(“python is ” + x)

# burada çıktı olarak “python is awesome” cümlesini alırız

**Global Variables (Global Değişkenler)**

Fonksiyon dışarısında oluşturulan değişkenler global değişken olarak tanımlanır. Bunları hem fonksiyon dışında hem fonksiyon içinde kullanabiliriz. Örnek:

y = 15

def my\_func():

print(“he is ” + y)

Fonksiyon içerisinde global değişken tanımlamak istersek değişken isminin başına “global” keyword’u (anahtar sözcüğü) getirilir. Örnek:

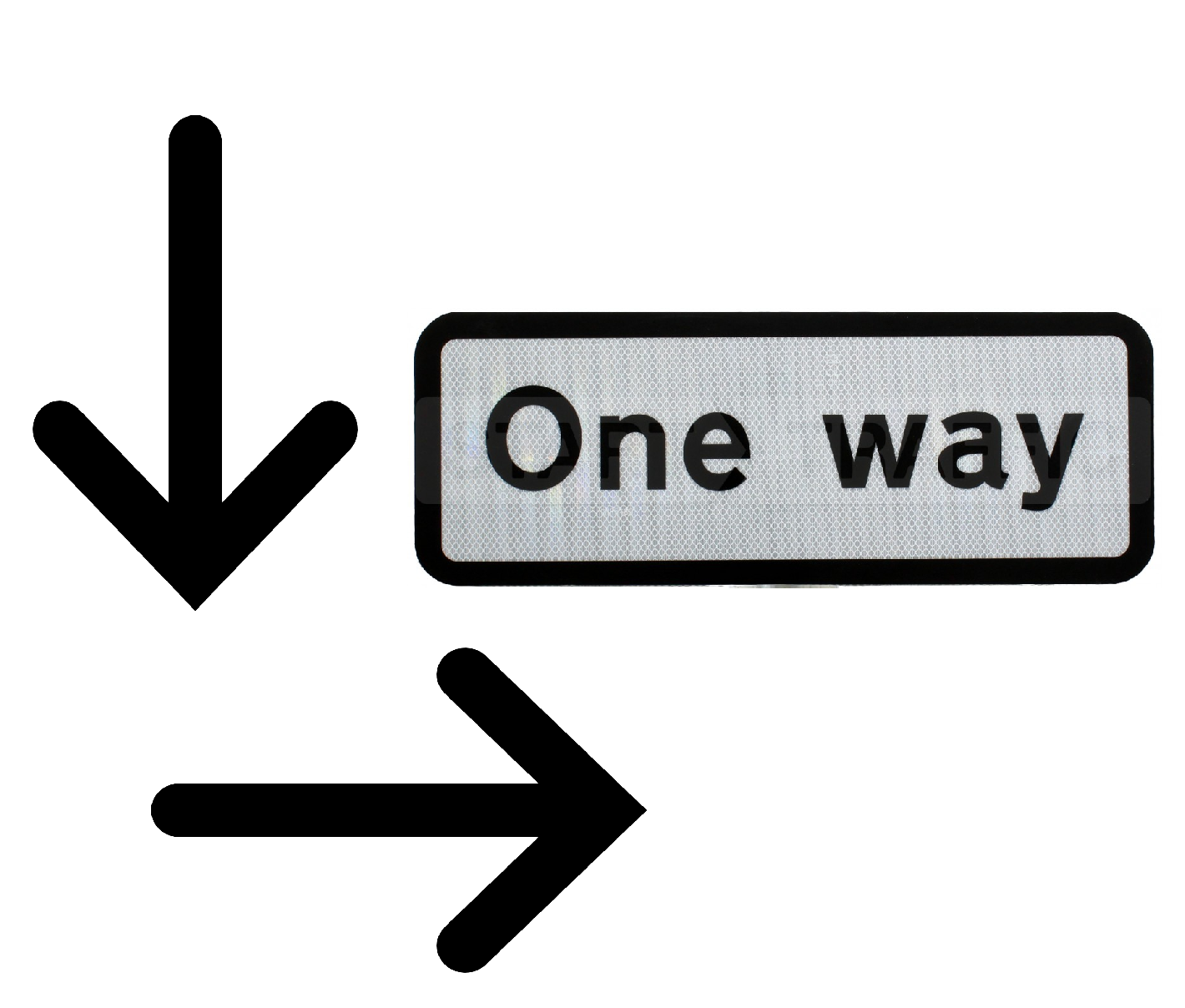
y = 15

def my\_func():

global z = 15

print(z)

“”” burada alacağımız çıktı 15 olacaktır. Gördüğünüz gibi z değişkenimiz fonksiyon içerisinde olmasına rağmen global keyword’u sayesinde fonksiyon dışında da tanımlı hale geldi “””



**Önemli**: Yukarıdaki görselde görüldüğü gibi Python’da kod yukarıdan aşağıya ve soldan sağa interpreter tarafından okunur. C#’ta da aynı yön geçerlidir fakat bazı kod parçalarına okuma önceliği tanınır. Bu sayede C#’ta metotları yazıldıkları yerin yukarısında çağırabiliriz.

**Data Types (Veri Tipleri)**

Python’ın varsayılan olarak sahip olduğu veri tipleri şunlardır:

*Text Type:* str

*Numeric Types:*  int, float, complex

*Sequence Types:* list, tuple, range

*Mapping Type:* dict

*Set Types:* set, frozenset

*Boolean Type:* bool

*Binary Types:* bytes, bytearray, memoryview

**C# içerisindeki veri tiplerini gösteren bir görsel:**



**Get Type (Tip Bulma)**

Python’da bir verinin tipini bulabilmek için type() fonksiyonu kullanılır.

C#’ta bunun karşılığı GetType() fonksiyonudur.

**Casting (Veri Tipi Belirleme)**

Python’da bir değişkene spesifik veri tipi ataması yapmak istersek şu tarz bir tanımlama yapmamız gerekir:

x = str(“Hello World”)

x = int(20)

Yukarıdaki tanımlama bize değişken tipleri üzerinde az da olsa bir kontrol verir.

**Numbers (Sayılar)**

Python içerisinde kullanılan sayı tipleri:

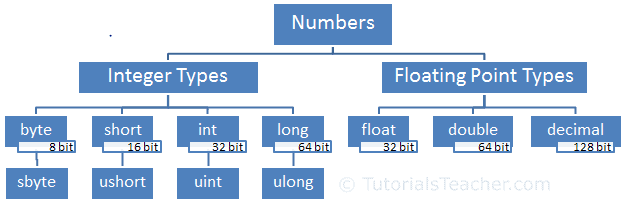
* int
* float
* complex

Birer örnek vermek gerekirse:

sayi1 = 5 #int tipinde bir örnek

sayi2 = 5.2 #float tipinde bir örnek

sayi3 = 3+5j #complex tipinde bir örnek



*C# içerisindeki numbers kategorisini yukarıdaki görselde görebilirsiniz.*

**Strings (Diziler)**

Python’da strings tıpkı diğer çoğu dilde olduğu gibi çift tırnak veya tek tırnak işaretleri arasında yazılır ve Unicode karakterlerden oluşan bir dizi grubudur. Örnek:

isim = “Barış”

Aynı tanımın C# üzerindeki karşılığı:

string isim = “Barış”;

Python’da stringler üzerinde kullanılmak için pek çok farklı fonksiyon tanımlanmıştır. Bunlardan bazıları:

isim = “Barış”.upper() #stringi küçük harfli yapar

isim.lower() #stringi büyük harfli yapar

isim.replace(“r”,”k”) #stringteki r harflerini k harfi ile değiştirir

len(isim) #stringin uzunluğunu almamızı sağlar

Python’da çok satırlı string (multiline string) tanımlamak için “”” (üç çift tırnak) kullanılır. Bu semboller arasında yazılan stringlerin her satırının sonuna otomatik olarak \n eklenir. Örnek:

cumle = “””Bugün hava çok sıcak

fakat yağmur yağma ihtimaline karşılık

yine de yanıma şemsiye alacağım.”””

Pek çok programlama dilinde olduğu gibi Python’da da stringler, Unicode karakterlerini ifade eden bit dizileridir. Bu da index kullanarak stringin içeriğine erişebiliriz anlamına gelir. Örnek olarak isim stringinin 4. harfine erişmek istersek:

isim = “Hello World!”

dorduncu\_harf = isim[3]

**Booleans**

Bir programlama dilinde çoğu zaman bir ifadenin doğru (true) mu yanlış (false) mı olduğunu bilmeye ihtiyaç duyarız. Bu alışkanlık, programlama dillerinin başlangıcından çok daha geriye gider. Boolean ifadeleri daha iyi anlamak için matematikte mantık konusuna bakabilirsiniz. Python’da da boolean ifadeler sık sık kullanılır. Boolean ifadeler C# üzerinde de aynı mantıkla çalışırlar. Bu yüzden kullanım açısından aralarında pek bir fark yoktur. Örnek birkaç boolean kullanımı:

print(3>5) # 3, 5’ten büyük olmadığı için bize false değerini yazacaktır

print(“ahmet” == “ahmet”) # iki string aynı olduğu için bize true değerini yazacaktır

bool\_ifade = True # bool\_ifade değişkenine doğrudan True değerini de atayabiliriz

**Operators (İşlemler)**



Operators (operatörler) değişkenler ve değerler üzerinde işlemler yapabilmemizi sağlarlar. Python ve C#’taki operatör çeşitleri şunlardır:

* Arithmetic Operators (Aritmetik Operatörleri)
* Assignment Operators (Atama Operatörleri)
* Comparison Operators (Karşılaştırma Operatörleri)
* Logical Operators (Mantıksal Operatörler)

**Arithmetic Operators**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| + | Toplama | İki değeri toplar | x + y |  |
| - | Çıkarma | Bir değeri diğerinden çıkarır | x - y |  |
| \* | Çarpma | İki değeri çarpar | x \* y |  |
| / | Bölme | Bir değeri diğerine böler | x / y |  |
| % | Mod Alma | Bölümden kalan sonucu verir | x % y |  |
| \*\* | Üs Alma | Değişkenin üssünü alır | x\*\*y |  |
| // | Yuvarlamalı Bölme | Bölüm sonucunu yakın değere yuvarlar | x//y |  |

*Yukarıdaki tabloda Python’daki aritmetik operatörler görülüyor.*

C#’ta farklı olarak değişkeni bir arttırma “++” ve değişkeni bir azaltma “--” operatörleri vardır. Üs alma ve yuvarlamalı bölme işlemleri için farklı yöntemler kullanılır.

|  |  |  |  |
| --- | --- | --- | --- |
| = | x = 5 | x = 5 |  |
| += | x += 3 | x = x + 3 |  |
| -= | x -= 3 | x = x - 3 |  |
| \*= | x \*= 3 | x = x \* 3 |  |
| /= | x /= 3 | x = x / 3 |  |
| %= | x %= 3 | x = x % 3 |  |
| //= | x //= 3 | x = x // 3 |  |
| \*\*= | x \*\*= 3 | x = x \*\* 3 |  |
| &= | x &= 3 | x = x & 3 |  |
| |= | x |= 3 | x = x | 3 |  |
| ^= | x ^= 3 | x = x ^ 3 |  |
| >>= | x >>= 3 | x = x >> 3 |  |
| <<= | x <<= 3 | x = x << 3 |  |

*Yukarıdaki tabloda Python’daki atama operatörleri bulunmaktadır.*

|  |  |  |  |
| --- | --- | --- | --- |
| == | Equal | x == y |  |
| != | Not equal | x != y |  |
| > | Greater than | x > y |  |
| < | Less than | x < y |  |
| >= | Greater than or equal to | x >= y |  |
| <= | Less than or equal to | x <= y |  |

*Yukarıdaki tabloda Python’daki karşılaştırma operatörleri bulunmaktadır.*

|  |  |  |  |
| --- | --- | --- | --- |
| and | Returns True if both statements are true | x < 5 and  x < 10 |  |
| or | Returns True if one of the statements is true | x < 5 or x < 4 |  |
| not | Reverse the result, returns False if the result is true | not(x < 5 and x < 10) |  |

*Yukarıdaki tabloda Python’daki mantıksal operatörler bulunmaktadır.*

Ufak farklılıklar dışında C# ve Python’da operatörler aynı mantıkla çalışır ve benzer kullanımlara sahiptirler.

**Lists (Listeler)**

Python’da listeler birden fazla değeri aynı değişken içinde tutmak için kullanılırlar. Farklı tipte veriler barındırabilir. C#’taki gibi initialize edilmesine gerek yoktur ve içerik tipi tanımlaması yapılmaz. Python’da veri koleksiyonu tutmak için kullanılan 4 veri tipinden biridir. Sıralıdır (ordered), içindeki nesnelere indeks ile erişilir. sort, remove, append vb. fonksiyonlar ile üzerinde işlemler yapılabilir. Liste içerisinde bir nesne tekrar edebilir (duplication).

Python: thislist = [“apple”, ”apple”, 1, False]

C#: List<String> thislist = new List<String>(){“apple”,”banana”,”orange”};

**Tuples**

Listelerden farklı olarak içerisindeki nesneler değiştirilemezdir. Python’da veri koleksiyonu tutmak için kullanılan dört veri tipinden ikincisidir.

thislist = (“apple”, “apple”, 1, False)

**Sets**

Lists ve tuplesdan farkı hem sırasız (unordered) hem de unindexed olmasıdır. Tuples da olduğu gibi içerisindeki nesneler değiştirilemezdir. Python’da veri koleksiyonu tutmak için kullanılan dört veri tipinden üçüncüsüdür. İçerisindeki nesnelere erişmek için in anahtar sözcüğü kullanılır.

thisset = {“apple”, “banana”, “cherry”}

**Dictionaries (Sözlükler)**

Verileri key:value (anahtar:değer) çiftleri şeklinde tutmak için kullanılır. İçerisindeki nesneler sırasız, değişebilir ve tekrarsızdır. Aynı isimde ikinci bir key bulunamaz. JavaScript’teki obje tipine benzerdir. Python’da veri koleksiyonu tutmak için kullanılan dört veri tipinden sonuncusudur.

thisdict = {

“name”: ”baris”,

”lastname”: ”kurt”

}

**If-Else**

Python’da kullandığımız if-else, mantık olarak C# ve diğer tüm yüksek seviyeli dillerdeki ile aynıdır. If kullanımını daha iyi anlayabilmek için koşullu cümle kurduğunuzu düşünebilirsiniz. Örnek:

*“Eğer yağmur yağarsa, dışarı çıkmam.”*

Yukarıdaki cümlede dışarı çıkmama durumu yağmurun yağma koşuluna bağlanmıştır. If-else mantığı da aynı şekilde çalışır.

if rain == True:

going\_out = False

Yukarıdaki if bloğunun Türkçe’si şöyledir: “Eğer rain (yağmur) değişkeni true (doğru) değerine sahipse going\_out (dışarı çıkmak) değişkeni false (yanlış) değerini alır.”

Şimdi ifadeye bir de else ekleyelim. Yukarıda kurduğumuz cümleyi düşünecek olursak şimdi kuracağımız cümle şöyle olmalı:

“Eğer yağmur yağarsa, dışarı çıkmam. Yağmazsa çıkarım.”

Bu cümleyi Python ile if-else bloğu kullanarak yazacak olursak:

if rain == True:

going\_out = False

else:

going\_out = True

Koşuldan sonra bir koşul daha eklemek istersek, Python’da elif sözcüğünü kullanırız. Şimdi cümlemizin sonuna bir koşul daha ekleyelim:

if rain == True:

going\_out = False

elif sun\_is\_up == True:

wearing\_coat = True

Yukarıdaki cümlede “Eğer rain değişkeni true ise going\_out değişkeni false değerini alır. Eğer rain değişkeni false ise ve sun\_is\_up (güneş çıktı) değişkeni true ise wearing\_coat (ceket giymek) değişkeni true olur.”

Yazı içerisinde tablo şeklinde listesini verdiğimiz mantıksal ve karşılaştırma operatörlerinden faydalanarak if-else blokları ile çeşitli uygulamalar yapabiliriz.

**C# tarafında da if-else tamamen aynı mantıkla çalışır ve aralarında sadece syntax farkı bulunur. C# içerisinde if-else bloğuna bir örnek:**

if(rain == True){

goingOut = False;

}

else if(sunIsUp == True){

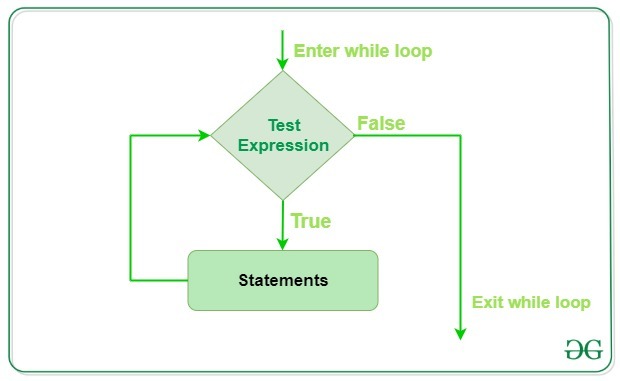
wearingCoat = True;

}

**Loops (Döngüler)**

Programlamanın en önemli parçalarından bir diğeri de döngülerdir (loops). Python’da while döngüleri ve for döngüleri olmak üzere iki çeşit döngü kullanılmaktadır. Şimdi bunları inceleyelim.

**While Loop**



Yukarıdaki akış diyagramında bir while döngüsünün nasıl çalıştığı görülüyor. Kısaca açıklamak gerekirse, Python’da while sözcüğünün hemen yanında bulunan mantıksal ifade “True” döndürdüğü sürece döngü devam eder. Bu sebeple döngüyü kırmak için gerektiği zamanda mantıksal ifadeyi “False” yapacak bir adım yazmamız gerekir. Buna alternatif olarak “break” sözcüğünü kullanarak döngüyü kırabiliriz ama ilk tercihimiz bu olmamalıdır. Örnek:

a = “barış can kurt”

counter = 0

while a != “barış can kurt2”:

counter += 1

if counter == 50:

a = “barış can kurt2”

Yukarıdaki örnekte while döngüsünün koşulu “a değişkeni eşit değildir ‘barış can kurt2’ stringine” şeklinde tanımlanmıştır. Bu da döngünün bu eşitsizlik devam ettiği sürece dönmesini sağlar. Döngü içerisinde tanımladığımız counter değişkeni yardımıyla 50 tur sonucunda if bloğunun içine girip a değişkenini “barış can kurt2” stringine eşitleriz. Bu sayede döngüden çıkılır. While döngülerinin genel çalışma mantığı bu şekildedir.

**Önemli: Çoğu dilde döngüler, koşullar, mantıksal karşılaştırmalar vb. aynı çalışma mantığına sahiptir. Bu sayede yüksek seviyeli bir programlama dilini bildiğinizde aynı seviyede olan bir başka programlama dilini kolaylıkla öğrenebilirsiniz.**

C# içerisinde bulunan while döngüsüne bir örnek:

string a = “barış can kurt”;

int counter = 0;

while (a != “barış can kurt2”){

counter++;

if(counter==50)

a = “barış can kurt2”;

}

Python’da herhangi bir döngü içerisindeyken “break” sözcüğü kullanılırsa döngü kırılır. Örnek:

while True:

print(“İlk turda döngü kırıldı.”)

break

**For Loop**

Python’ın diğer programlama dillerinden ayrıştığı bir diğer nokta da for döngüleridir. Python’da for keyword’u C#’taki foreach’in karşılığı olarak düşünülebilir. Iterator bir metot olarak çalışır. Örnek:

a = “abcdefg”

for harf in a:

print(harf)

Yukarıdaki örnekte stringin iterable olduğunu görüyoruz. Stringteki harflere indis ile erişebiliriz. O yüzden C#’taki foreach mantığına benzer şekilde a stringi içerisindeki harfleri print ile ekrana yazdırdık. Aynı işlemin C# karşılığı:

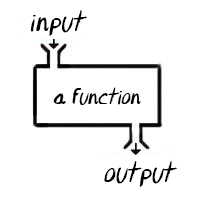
string a = “abcdefg”;

for(int i=0; i<a.Length; i++){

Console.WriteLine(a[i]);

}

**Functions**



Python’da fonksiyonlar, diğer yüksek seviyeli programlama dillerinde olduğu gibi sadece çağırıldığında çalışan paketlenmiş bir kod bloğudur. Fonksiyonların içine işlenmesi için veri (parametre) gönderebilir ve fonksiyonların dışına sonuç olarak veri (return value) döndürebiliriz. Python’da fonksiyon tanımlamak için def keywordu kullanılır. Örnek:

def test\_fonksiyonu():

print(“Bu bir test fonksiyonudur.”)

Fonksiyon sadece bu haliyle bir işlem yapmaz, çünkü henüz çağırmadık. Fonksiyonu çağırmak için kodun herhangi bir yerinde test\_fonksiyonu() yazmamız yeterlidir.

Python’da fonksiyonlara argüman olarak veri gönderebiliriz. Daha sonra bu veriyi fonksiyonda kullanırız. Örnek:

def toplam(sayi1,sayi2):

return sayi1 + sayi2

Yukarıdaki kodu kullanarak toplam(1,2) yazarsak fonksiyon bize 3 değerini döndürecektir.

***Önemli: Python’da değişkenler gibi fonksiyonlarda dinamiktir. Bu yüzden C#’taki gibi döndürecekleri değeri belirtmemiz gerekmez.***

C#’ta toplam metodu (evet, fonksiyonlara C#’ta metot denir) için bir örnek:

float toplam(int sayi1, int sayi2){

return sayi1+sayi2;

}

Yukarıdaki kodu kullanarak toplam(1,2) yazarsak metot bize 3 değerini döndürecektir.

Yazının devamında gelişmiş fonksiyon kullanımlarından ve nesne yönelimli programlama (object-oriented-programming) konularından bahsedilecektir.

**Yazan: BARIŞ CAN KURT**

Kaynakça:

<https://www.w3schools.com/python/>

<https://www.w3schools.com/cs/>

<https://www.geeksforgeeks.org/csharp-programming-language/>

<https://docs.oracle.com/en/>