**GENEL KOD YAZIM KURALLARI**

* Satırlardaki karakter sayısı azami 79 karakter olmalıdır.
* Devam edecek uzun satılarda satır uzunluğu 72 karakterle sınırlandırılmalıdır.
* Boşluklar tercih edilen girinti yöntemidir. Sekmeler, yalnızca zaten sekmelerle girintili olan kodla tutarlı kalmak için kullanılmalıdır. Python 3, girinti için sekme ve boşlukların karıştırılmasına izin vermez.
* Aşağıdaki durumlarda gereksiz boşluk bırakılmamalı
* Yuvarlak, süslü ve kapalı parantezler içerisinde boşluk bırakılmamalıdır
* Sondaki virgül ile onu takip eden yakın parantez arasında
* Virgül, noktalı virgül veya iki nokta üst üste işaretinden hemen önce:
* Bir fonksiyonun adını takip eden parantez öncesinde “print ()” olmamalı
* Bir atama (veya başka) operatörünü bir başkasıyla hizalamak için birden fazla boşluk
* Bu ikili operatörleri her iki tarafta da tek bir boşlukla çevreleyin: atama (=), artırılmış atama (+ =, - =, vb.), Karşılaştırmalar (==, <,>,! =, <>, <=,> =, içinde, içinde değil, eşittir, değil), Booleanlar (ve, veya değil)

**COMMENT VE DOCSTRİNG KULLANIMI**

* Kod yazarken kendimize veya başkalarına yönelik olarak açıklayıcı notlar yazmak istediğimizde, programımıza “comment” veya 'docstring' olarak daha detaylı olarak ekleyebiliriz.
* Comment kısa olmalıdır. Uzun comment’ler bölünmelidir. Üç farklı comment tipi vardır:
* # This is a single line comment
* print('the cosmos has no superiority to chaos') # This is an inline comment
* print(3 + 4)

# This is the multi-line comment, line-1

# This is the multi-line comment, line-2

# This is the multi-line comment, line-3

* Docstring’ler - normal commnetler’den farklı olarak – fonksiyon veya modülün bir niteliği olarak saklanır, bu da onlara programlı olarak erişebileceği anlamına gelir.
* Docstring’ler “”” “”” 3’er adet tırnak içerisinde ifade edilir.

**DEĞİŞKENLERİN ADLANDIRILMASI**

* PEP 8’e göre değişkenlerin adlandırılmasında aşağıdaki kurallara dikkat edilmeli
* Küçük harf kullanılmalı, birden fazla kelime alt tire ile ayrılmalı
* Küçük “le” harfi, büyük “O” harfi ve büyük “I” harfi tek karakterli değişken adı olarak kullanılmamalı
* Anlamlı olmalı, mümkünse 3 kelimeyi geçmeyen isimler tercih edilmeli.

**BASİT VERİ TİPLERİ**

String (str) a, b, ., ?

Numbers Signed Integer (int) 7, -6, 0

Floating Numbers (float) 35.90, 7.0

Complex Numbers (complex) 5+5j

Boolean (bool) True, False (T ve F harfleri büyük olmalıdır.)

**VERİ DÖNÜŞÜMLERİ**

str() String tipe çevirir.

float() Float tipe çevirir.

int() Integer tipe çevirir.

**BASİT İŞLEMLER**

* print(4+11) 15 # integer toplamı integer verir.
* print(4+11.5) 15.5 # Integer ile float toplamı float verir.
* a = b = 5 # Bir değeri birden fazla değişkene atayabiliriz
* a, b = 5, 20 # Birden fazla değer aynı anda virgül kullanarak birden fazla değişkene atanabilir.
* no\_1 = 46 print(no\_1/2) 23.0 # Bölüm işleminin sonucu her zaman float olur
* print((3 \* 4)/2) 6.0 # Parantezler matematikte olduğu gibi kullanılır
* print(7 // 2) 3 # Bölümün integer kısmını verir.
* print(9 % 2) 1 # Kalanı verir.
* print(3\*\*2) 9 # Üssü almayı sağlar.
* print(64\*\*0.5) 8.0 # Karekök almayı sağlar. Float varsa sonuç float
* İşlemler esnasında öncelikler şu şekildedir:
* Parantez ( )
* Üssü \*\*
* Unary minus -3
* Çarpma/bölme \*, /
* Toplama/çıkarma +, -
* print () fonksiyonu ile birden fazla ifadeyi virgül ile ayırarak yazabiliriz.
* number = 2020

text = "children deserve respect as much as adults in"

print(text, number)

* children deserve respect as much as adults in 2020
* print () fonksiyonunun default değer olarak sep =’ ‘ ve end =’\n’ olarak tanımlıdır. Yani otamatik olarak sonraki satıra geçer.
* Yerinde işlem yapmak için = karakterinden önce herhangi bir matematik operatörü kullanılabilir:

-=  decrements the variable in place,

+=  increment the variable in place,

\*=  multiply the variable in place,

/=  divide the variable in place,

//=  ﬂoor divide the variable in place,

%=  returns the modulus of the variable in place

\*\*=  raise to power in place.

* \  escape sequence olarak adlandırılan özel bir işarettir. Yanındaki karakterin özelliğini kaybetmesini sağlar.
* \n : means new line,
* \t : means tab mark,
* \b : means backspace. It moves the cursor one character to the left.

**BOOLEN LOGİC İŞLEMLERİ**

Python üç farklı built in-yapıya sahiptir.: and, or, not. Bunlardan “not” haricindekiler binary operatörler olup, iki argümana ihtiyaç duymaktadır.

**AND** operatörü, tüm ifadeleri değerlendirir ve tüm ifadeler **True** olarak değerlendirilirse, son True ifadeyi döndürür. Aksi takdirde, ilk False değeri döndürür.

**Koşullu İfadeler (Conditional Statements)**

### **If Statement Yapıları**

* In some cases, your program needs to execute some part of the code only if a specific condition is true. The simple structure of an if statement is :

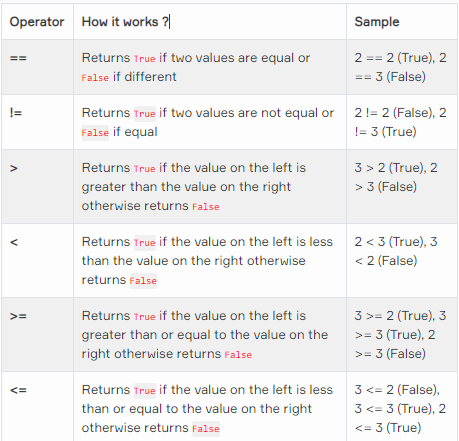


* The **if** statements check the **condition**. The **condition** is always a **Boolean**  expression, that is, its value equals either True or False. Remmber truthy or falsy (None, 0 and [], (), {})
* If it evaluates to True, it executes the **body** of the **if** statement. If it evaluates to False, it skips the **body**. This logic works like the English language.

### **Comparison Operators**

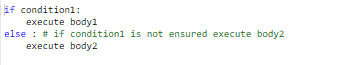
Boolean values basically make it clear if a piece of code needs to be executed. Because comparisons result in bool, it's always best to use them as a condition.

Therefore, it is time to examine **comparison operators** :



### **'if-else' Statements**

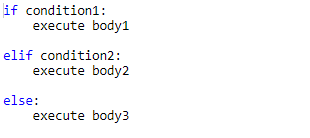
* An **if-else** statement is another kind of conditional statements in Python. It is used with an additional keyword: else.
* else works like an if statement. If none of the conditions in if are ensured, "else" will be used to specify all remaining conditions. The simple structure of an if-else looks like



* else doesn't require any condition and the body2 requires 4-space indentation

### **if-elif-else' Statements**

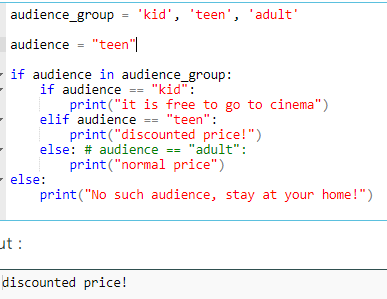
* The elif statement is used when it requires to specify several conditions in our program.
* In Python you can deﬁne a series of conditionals using :
* if for the **ﬁrst** one,
* elif for the **rest**, up until the ﬁnal (optional),
* else for **anything not caught by the other conditionals**.
* The basic structure of these statements looks like :



* We can use as many elif statements as we need, so your conditions can be varied.

### **Nested 'if-elif-else' Statements**

* Both if-else and if-elif statements can be nested. Let's see the nested structure on the same movie ticket example.



* Nested if-elif-else structures may seem a bit complicated to you, the best way to overcome this is to examine and practice plenty of samples.

**Not:** If yapılarında kullanıcıdan Input fonksiyonu ile “string” bir veri istendiğinde, case sensitive bir dil olması sebebiyle **değişken\_adı.lower () == “XXXX”** formatı kullanılabilir.

**ülke = USA**

**if ülke.**lower() == “usa”

**print(“”)**

## Loops

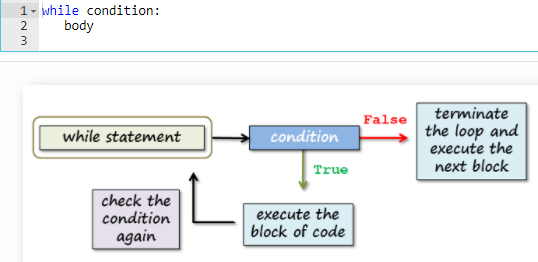
* When writing programs in Python, in some cases it is not enough to execute our block of code only once. **The loops are used to repeat (iterate) the execution of a block of code.**
* As one of the most main functions in programming, loops are an important part of almost every programming language. Loops enable programmers to set certain sections of their code to repeat through a number of loops which are referred to as iterations.

This topic covers using multiple types of loops and applications of loops in Python. You will learn two types of loops which are :

* **while Loop**,
* **for Loop**.

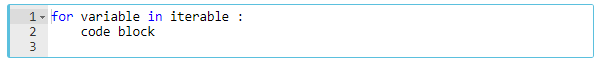
### **'while' Loop**

* while loops have a boolean logic, similar to if statements.
* As long as the result of the condition returns True, the code block under while loop runs. When the condition returns to False, the loop execution is terminated and the program control moves further to the next operation.
* Here is the simple structure of a while loop :

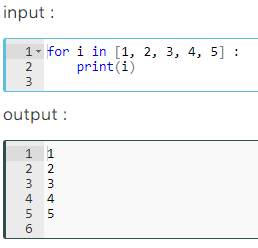


### **'for' Loop**

* When you want to iterate a block of code you will use for loop. To create a for loop, you need a **variable** and an **iterable object**. Here is the simple structure of a for loop :



* We can use list as an iterable



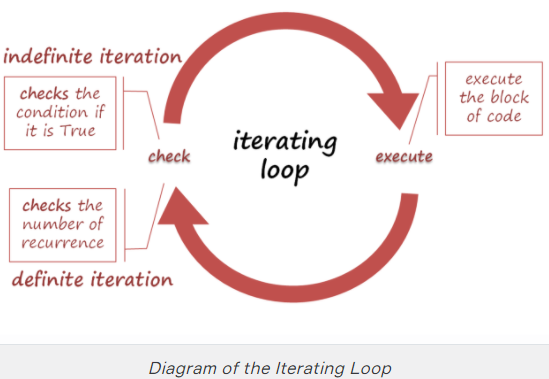
### **Working with the Iterators**

Let us explain the term **iteration** a little more.

**Iterable** object can be anything for which items are received one by one, forward only. In Python, the process of recurrent execution of a block of code is called an **iteration**.

We can basically classify iterations as two headings :

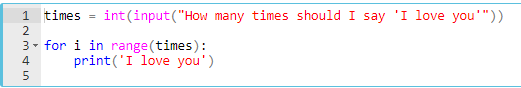
* If the number of repetitions is predetermined, it is called **definite** iteration.
* The repetition structure that makes the code block run as long as the predetermined condition generates True is called **indefinite** iteration.



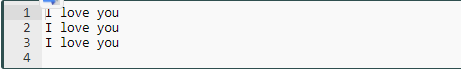
### **Operations with the 'for' Loop**

In this topic, you will learn about how we use the for loop using several functions and methods.

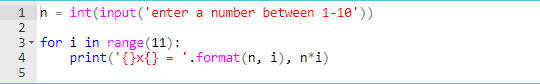
* In the example below, you'll get a number from the user and print a sentence the number of times we receive from the user.



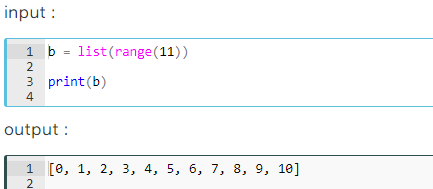
* As we stated before, input() function can get the value of different data types and assign a variable you chose. In the example above, it gets a number and assigns it to times. If the user enters 3 then the output will be:

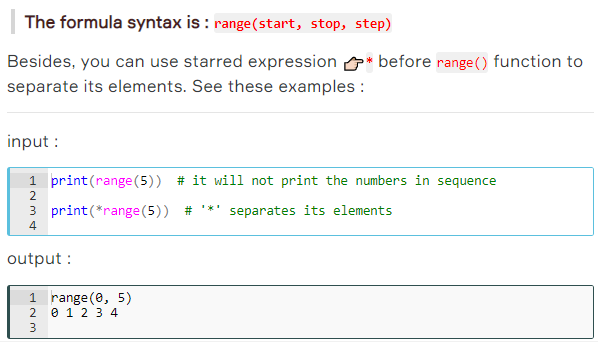


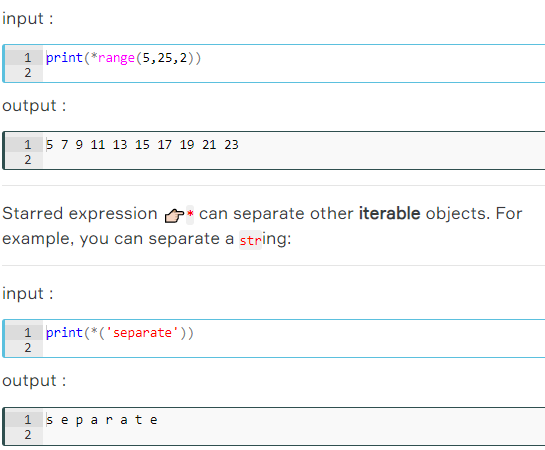
* Now, let's write a code that asks the user a number between 1 and 10 and puts that number into the **multiplication table**.

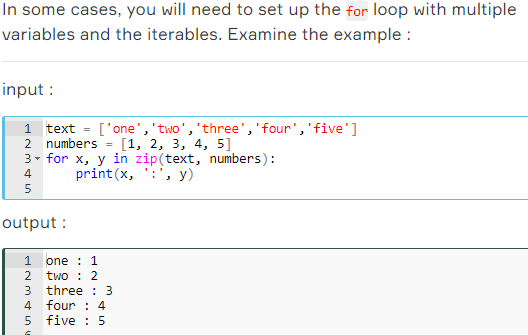


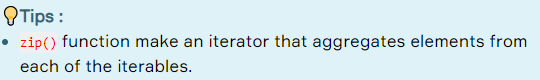
* If you want the user to input numbers, use the input() function together with the int() function. Otherwise, the value entered by the user will be in the **string** data type.
* The range() function creates an iterable sequence of numbers. And it can be simply converted into an iterable object: list, set, and tuple. For example :











### **Nested 'for' Loop**

