

Python Programlama Diline Giriş

Python Hollandalı bir **bilgisayar bilimcisi ve matematikçi** olan **Guido Van Rossum**'un 1989 yılında çalıştığı bir projeyi **dünyaya hediye etmeye karar vermesiyle** ortaya çıkmıştır. İlk olarak 20 Şubat 1991'de piyasaya sürüldü. Python bir **piton yılan olarak tanınmasına** rağmen gerçekte bu programlama dilinin adı eski bir İngiliz komedi dizisindeki **Monty Python**'dan gelmektedir.

Python'ın önemli özelliklerinden biri de aslında **tek bir kişinin eseri** olmasıdır. Elbette Guido van Rossum tüm Python bileşenlerini **kendi başına oluşturup geliştirmedir**. Python'ın dünya çapında yayılması, binlerce (çoğu zaman anonim) **programcı, test uzmanı, kullanıcı** (çoğu Bilişim-Teknoloji uzmanı değil) ve meraklıların sürekli çalışmasının bir sonucudur. Python 2001 yılından bu yana **Python Yazılım Vakfı (Python Software Foundation)** tarafından desteklenmektedir. Bu vakıf, Python **dilini ve ortamını geliştirmeye**, iyileştirmeye, genişletmeye ve popülerleştirmeye kendilerini adanmış bir topluluk ve **kâr amacı gütmeyen** üyelik bazlı bir kuruluştur.

Python açık kaynak kodlu ve ücretsiz bir yazılımdır. Nesne (obje) tabanlı, modüler, **etkileşimli (interactive)** ve **yorumlayıcı (interpreter)** bir programlama dilidir. Diğer programlama dillerine kıyasla öğrenilmesi ve kullanılması kolay olan pythonun **ingilizce komutları yalın** ve kendine özgü bir sözdizimine (*syntax*) sahiptir. *Unix, Lunix, Mac, Windows, Amiga ve Symbian* gibi **farklı birçok işletim sistemi üzerinde** çalışabilmektedir. Özellikle bazı Unix, Lunix ve Mac sürümlerinde sistem ile birlikte gelmektedir.



Python'un mucidi Guido van Rossum ve Python'un günümüzdeki logosu

Python Programlama Diline Giriş

Çok geniş bir yelpazede kullanım imkanına sahip olan Python;

- Sistem programlama,
- Ağ programlama,
- Veritabanı yönetimi,
- Uygulama geliştirme,
- Görsel arayüz (GUI) hazırlama,
- Web programlama

gibi birçok bilgisayar programcılığı alanında kullanılmaktadır. Ayrıca python yorumlayıcı bir dil olması nedeniyle PHP, VBasic gibi script (komut dizisi) dillerinin yaptığı birçok işeri kolaylıkla yerine getirebilir.

Günümüzde gelişmiş kütüphaneleri ve kolay sözdizimi imkanlarıyla Python büyük ve kompleks programlarda artık C/C++ yerine tercih edilebilir hale gelmiştir. Modüler yapısı sebebiyle Python her programcıya kendine özgü veri tipleri ve fonksiyonlarını oluşturma imkânı vermektedir. Böylece programcılar belirli amaçlar için ayarlanmış programları kolaylıkla geliştirip diğer kullanıcıların kullanımı için Python kütüphanelerine ekleyebilmektedirler.

Python Sürümleri Hakkında

İlk çıkan Python sürümü Python 2 olarak geliştirilmiş ve en son kararlı sürümü 2.7 olarak çıkmıştır. Artık Python 2'nin geliştirilmesi bırakılmıştır. Python 3 ilk sürüm olarak 2008 Aralık ayında yapılmış olup günümüzde geliştirilmesine halen devam edilmektedir. En son kararlı sürümü 3.12 olarak yayınlanmıştır. Burada dikkat edilmesi gereken bir nokta vardır: **Python2 ile yazılmış bir program Python3'te çalışmaz. Aynı şekilde Python3 ile yazdığınız bir program da Python2'de çalışmaz.** Bu bakımdan Python'ı öğrenmeye yeni başlayanların Python 3'ü kullanmaları biraz zorunlu gibidir.

Python Kurulumu

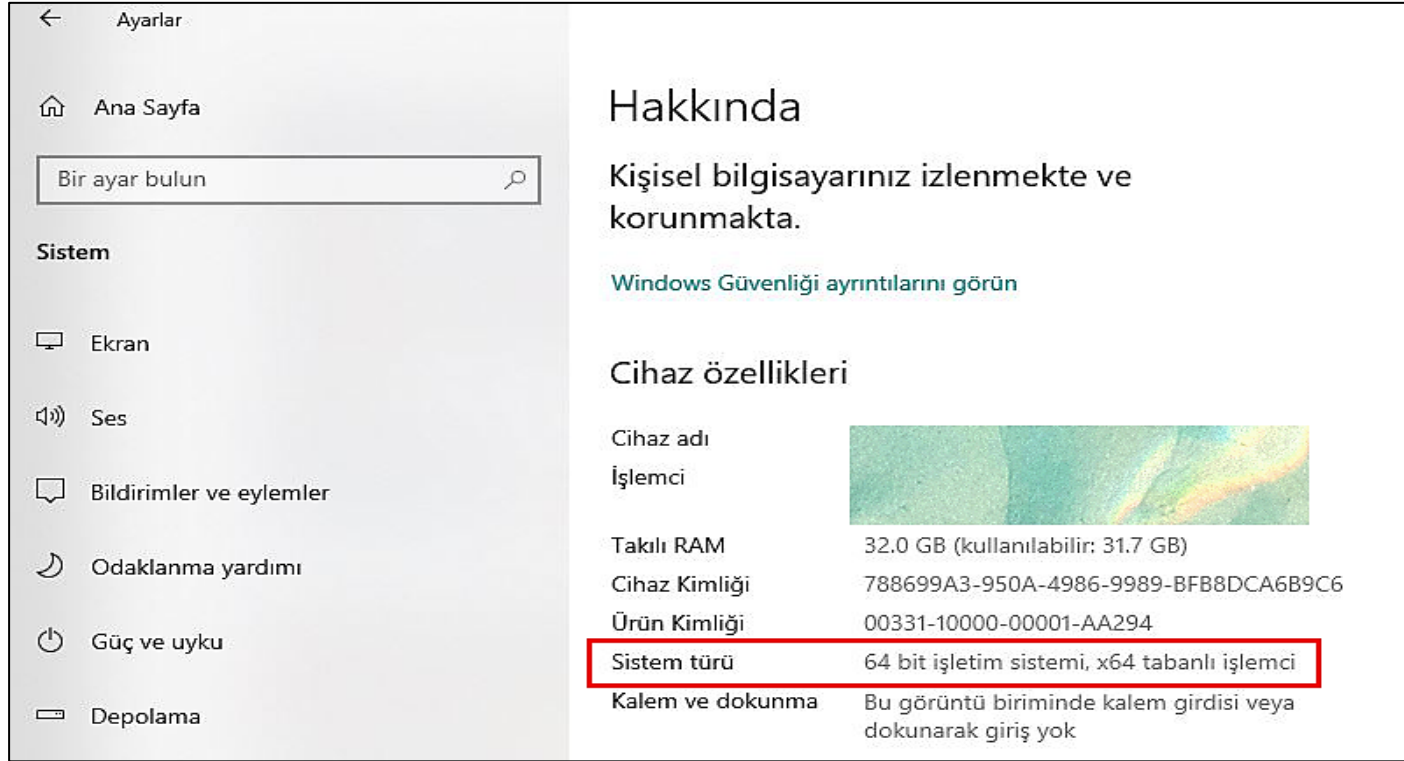
Programın kurulması sahip olunulan bilgisayarın işletim sistemine uygun çalıştırabilir dosyaların internetten indirilmesiyle başlar. Bunun için <https://www.python.org/downloads/> web sitesine gidilerek işletim sistemine uygun dosyaların bilgisayara indirilmesi gerekir. Kullanılan işletim sistemine göre kurulum aşamaları birbirinden farklıdır. Bu bakımdan kurulum aşamaları **benim de aşına olduğum** ve en yaygın işletim sistem olan Windows'a göre anlatılacaktır.



MacOS'da python kurulum için <https://www.dataquest.io/blog/installing-python-on-mac/> linkindeki açıklamalara bakılabilir. Ubuntu'da kurulum için ise <https://phoenixnap.com/kb/how-to-install-python-3-ubuntu> linkindeki açıklamalar takip edilebilir.

Windows 10 – Python Kurulumu

Kullanılan bilgisayar için doğru Python seçimini yapmak için iki hususa dikkat etmek gerekir. Birinci husus, kullanılan Windows işletim sisteminin Windows 7 mi, Windows 8 mi yoksa Windows 10 mu olduğudur. İkinci husus ise sistemin 32-bit mi yoksa 64-bit mi olduğudur. Bu iki hususa dair bilgi edinmek için bilgisayarınızda sırasıyla “**Denetim Masası > Sistem ve Güvenlik > Sistem**” sayfası açılmalı ve “**Sistem türü**” bilgisi kontrol edilmelidir. Ayrıca sistem bilgi sayfasına masaüstünde “**Bu bilgisayar**” simgesine sağ tuşla tıklanıp menüden “**Özellikler**” seçilerek de erişilebilir.



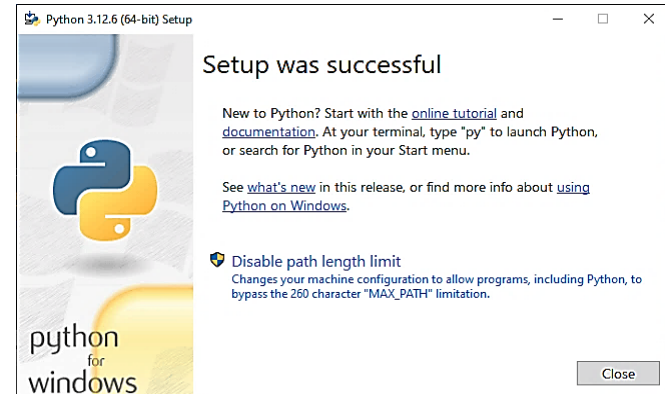
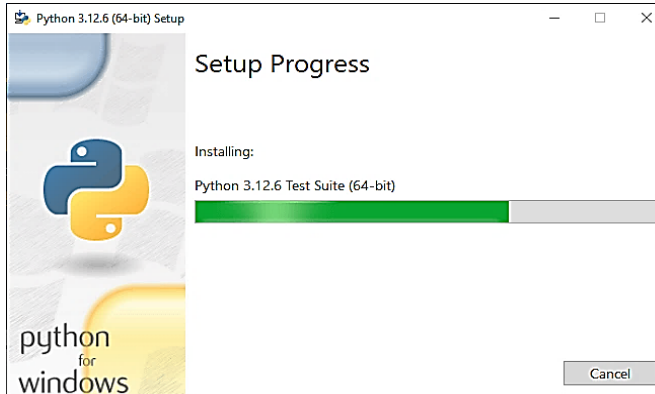
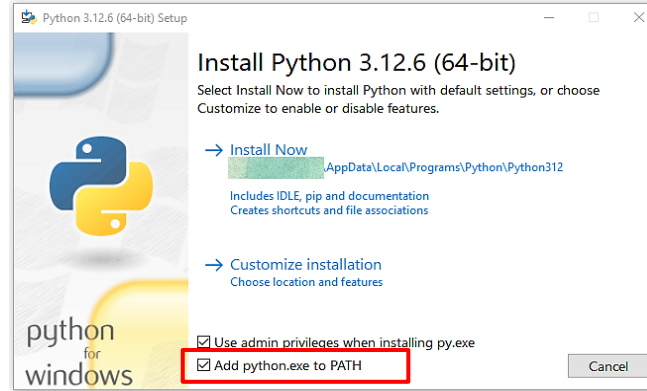
Windows 10'da sistem türünün kontrol edildiği pencere

Python Kurulumu – ilk aşama

Python web sitesinden doğru kurulum paketi indirildikten sonra pakete çift tıklanarak kurulumla geçilir. Kurulum 1. aşamasında ön ayarla (default) mı yoksa özel ayarla (custom) mı kurulacağına karar verilmelidir. Ayrıca kurulum penceresinde “**Add Python.exe to PATH**” seçeneğinin seçili olduğuna dikkat edilmelidir. Böylece Python.exe dosyası herhangi bir komut satırından çalıştırılabilir olacaktır. Eğer ön ayarla kurulum yapılmak istenirse penceredeki “**Install Now**” yazısına tıklanır ve kurulumun 2. aşamasının bitmesi beklenir. Sonra “**Close**” tuşuna basılarak son pencere kapatılır.



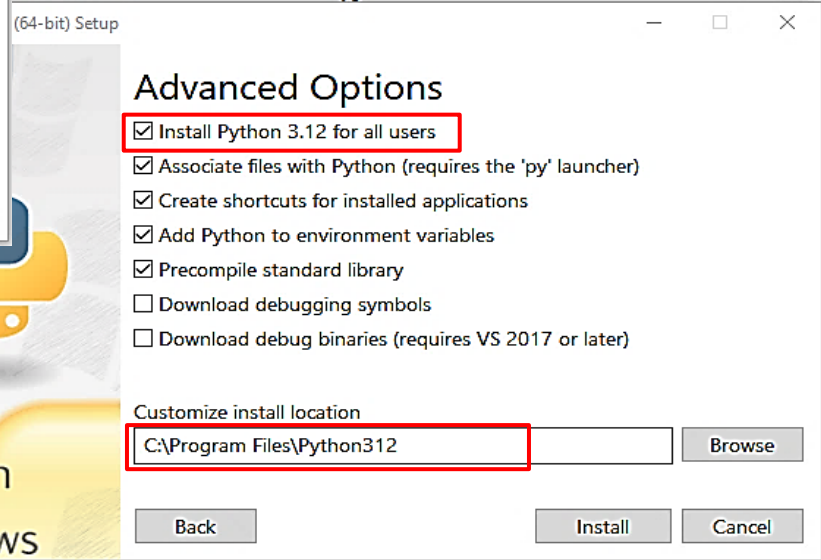
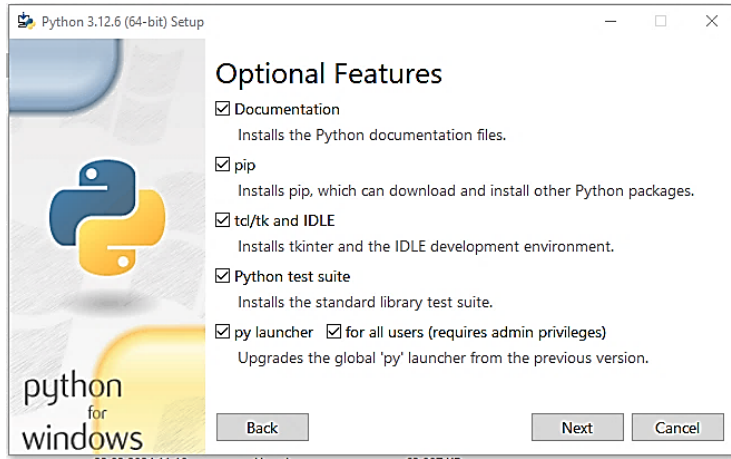
Kurulum paketi



Ön ayarla kurulumun tamamlanması, 2.aşama

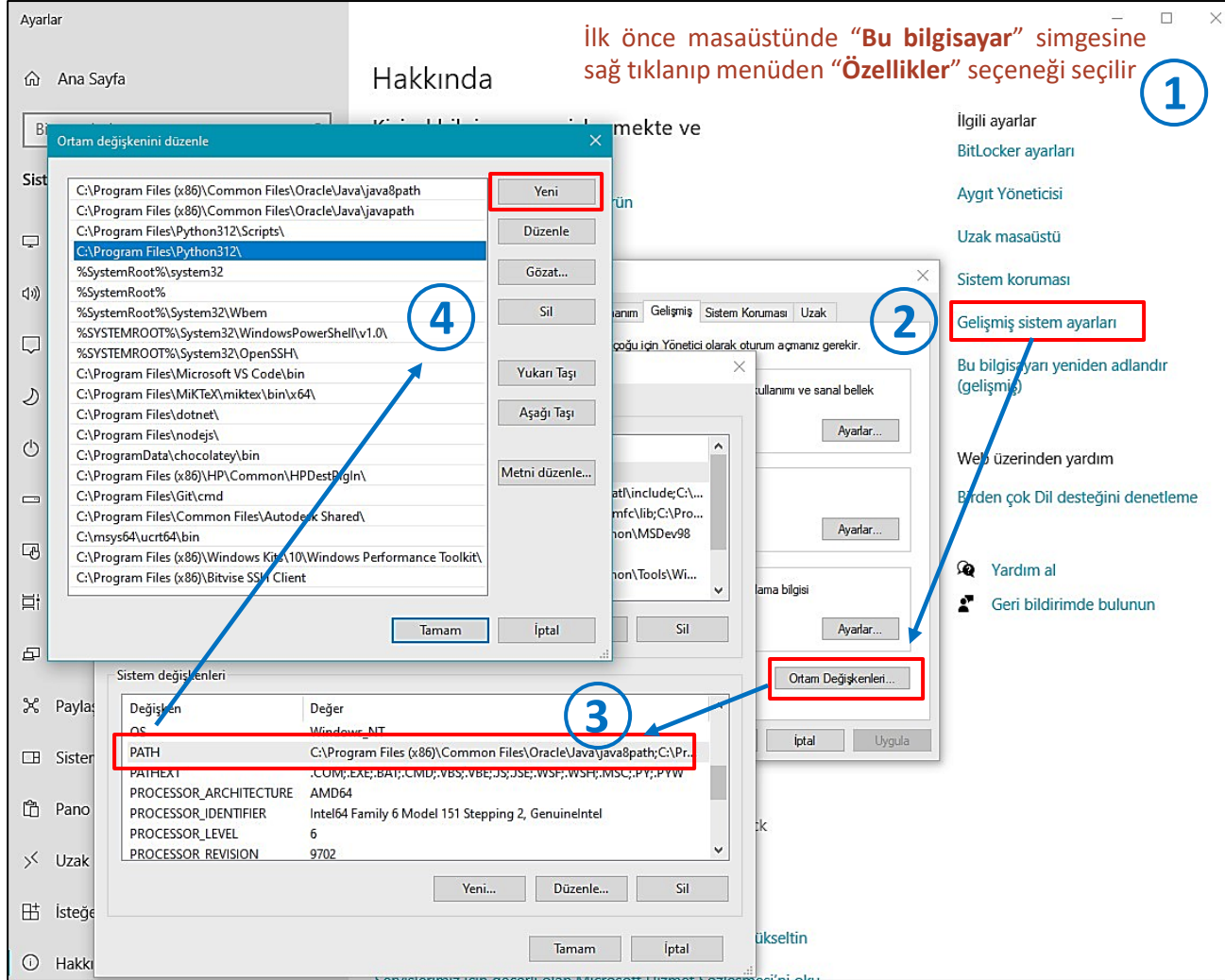
Windows

Eğer özel ayarla kurulum yapmak istenirse penceredeki “**Customize Installation**” yazısına tıklanır. 3. aşamada pencere değişecek ve “**Optional Features**” (Tercihli Özellikler) başlıklı özellikler görünecektir. Genel olarak tüm özellikler seçili olarak görünür ve bir şey değiştirmeden “Next” tuşuna basılarak diğer pencereye geçilir. Burada “**Advanced Options**” başlığı altında bazı seçenekler bulunur birkaçı işaretlidir. Burada “**Install Python 3.12 for all users**” seçeneğinin seçili olduğuna ve kurulum klasörünün doğru olduğuna dikkat edilmelidir. Ardından “**Install**” tuşuna basılır ve 2. aşamadaki adımların bitmesi beklenir.



Özel ayarla kurulum adımları, 3. aşama

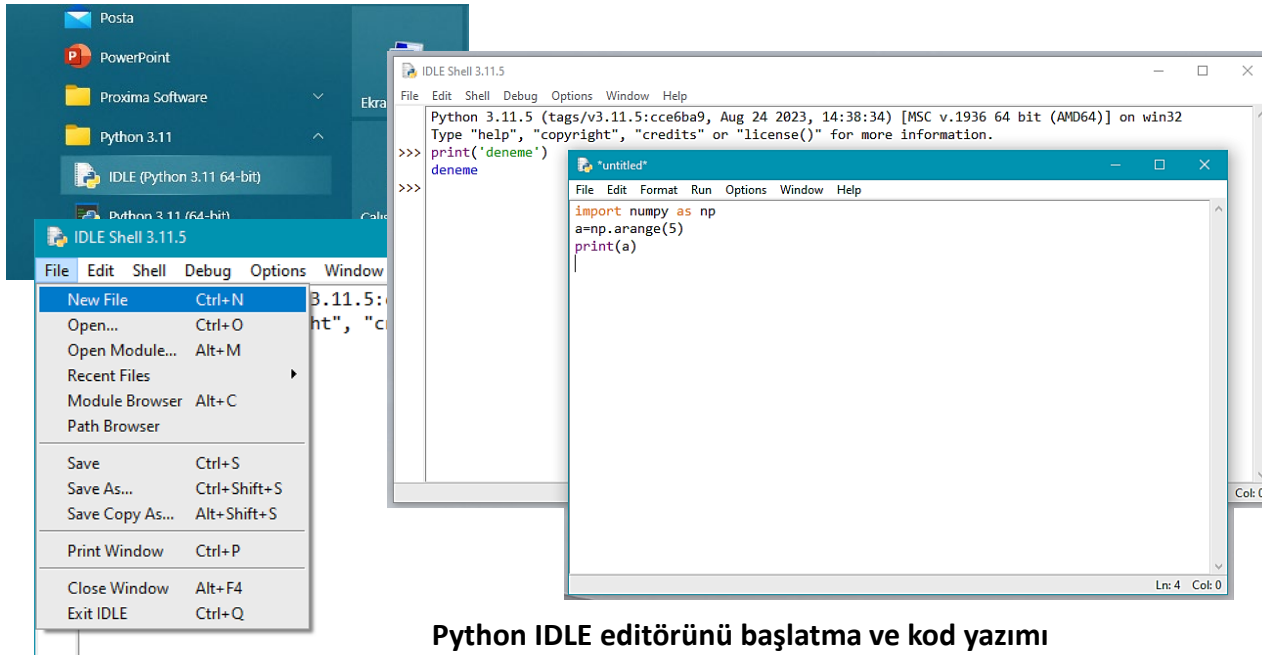
Python Programlama Diline Giriş



Ardından bütün pencereler tek tek “**Tamam**” tuşuna basılarak kapatılır ve yapılan düzenlemelerin geçerli olması için ya kullanıcı oturum kapatılıp yeniden başlatılır ya da bilgisayar yeniden başlatılır.

Python Program Editörü

Python kurulumundan sonraki diğer önemli bir adım program komutlarının yazılıp çalıştırılması için bir editör yani metin yazım programının kurulmasıdır. Normal olarak python kurulumu ile gelen **IDLE (Integrated Development and Learning Environment)** isimli bir editör bulunmaktadır. Bu editörü iki farklı şekilde kullanmak mümkündür; *komut satırı* ve *kod editörü* olarak. IDLE çalıştırıldığında ilk etapta komut satırı şeklinde **"IDLE Shell"** penceresi açılmaktadır. Burada satırların başında pencerenin en solunda **">>>"** şeklinde simge görünmektedir. Bu python komutlarının direkt çalıştırabileceğini göstermektedir. Ancak burada komutlar tek tek çalıştırılmaktadır, yani her yazdığınız satırı **"Enter"** tuşuna basarak çalıştırmamız gereklidir. Sonradan satırı düzeltmeniz mümkün değildir. Bu sadece kısa programları yazmak ve belirli kodları test etmek için kullanılabilir, uzun ve ön ayarlamalar gereken program yazımları için uygun değildir. Normal editörde kod yazmak için shell penceresindeki menüden **"File > New File"** seçilerek kod yazım penceresinin açılması gereklidir. Böylece bütün kodlar alt alta yazılıp sonradan hepsinin çalıştırılması sağlanabilir.

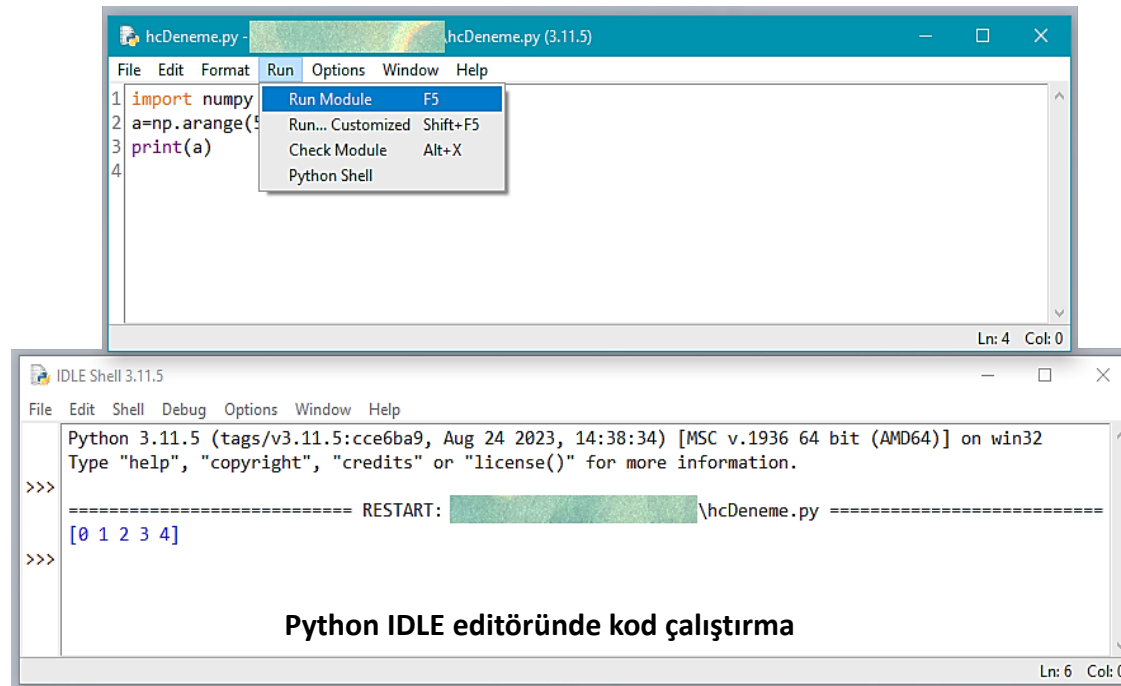


Python IDLE editörünü başlatma ve kod yazımı

Python IDLE Editörü

Gerekli kodlar yazıldıktan sonra bunları çalıştırmak için editör menüsünden **“Run > Run Module”** seçilmeli yada **“F5”** tuşuna basılmalıdır. Yazılan kod metni diske kaydedilmemiş ise derleme ve çalıştırmaya geçmeden önce metni diske kaydetme işleminin yapılması gereklidir. Bunun için açılan pencereden metin dosyası için bir konum seçilmesi ve dosya için bir isim girilmesi gerekir. Dosya kaydedildiğinde editör penceresinin başlığında daha önce **“untitled”** olan dosya ismi değişir ve verilen dosya ismi ve dosya yolu görünür. Kaydetme işleminin tamamlanmasının ardından yazılan kod bilgisayar tarafından çalıştırılır. Kod çalıştıktan sonra bütün çıktılar shell penceresinde görünecektir.

Kod editöründe yazım işi tamamlandıktan sonra istenirse yazılan kodlar **“File > Save As”** ile bir dosyaya çalıştırılmadan önce de kayıt edilebilir. Ayrıca kaydedilmiş bir kod yazımı ise shell penceresinden **“File > Open”** ile tekrar editöre yüklenerek kod yazma yada düzenleme işlemine devam edilebilir.



The screenshot displays the Python IDLE 3.11.5 interface. The top window, titled 'hcDeneme.py (3.11.5)', shows a Python script with the following code:

```
1 import numpy
2 a=np.arange(5)
3 print(a)
4
```

The 'Run' menu is open, showing options: 'Run Module' (F5), 'Run... Customized' (Shift+F5), 'Check Module' (Alt+X), and 'Python Shell'. The bottom window, titled 'IDLE Shell 3.11.5', shows the output of the script:

```
Python 3.11.5 (tags/v3.11.5:cce6ba9, Aug 24 2023, 14:38:34) [MSC v.1936 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: \hcDeneme.py =====
[0 1 2 3 4]
>>>
```

The text 'Python IDLE editöründe kod çalıştırma' is centered at the bottom of the shell window.

Python IDLE Editörü

Bir kod editörünün satır numaralarını göstermesi her zaman tercih edilir. Çünkü Shell'den gelen hata bildirimlerinde satır bilgisi verilir. Yani kod içindeki hatanın ilk görüldüğü satır hangisi ise o satır numarası Shell'de gösterilir. IDLE'in ön tanımlı (default) ayarlarında satır numaralarının gösterimi kapalıdır. Bunları göstermek için menüden “**Options > Show Line Numbers**” seçeneği seçilmelidir. Bu durumdaki editörde her satırın başında bir satır numarası görünecektir.

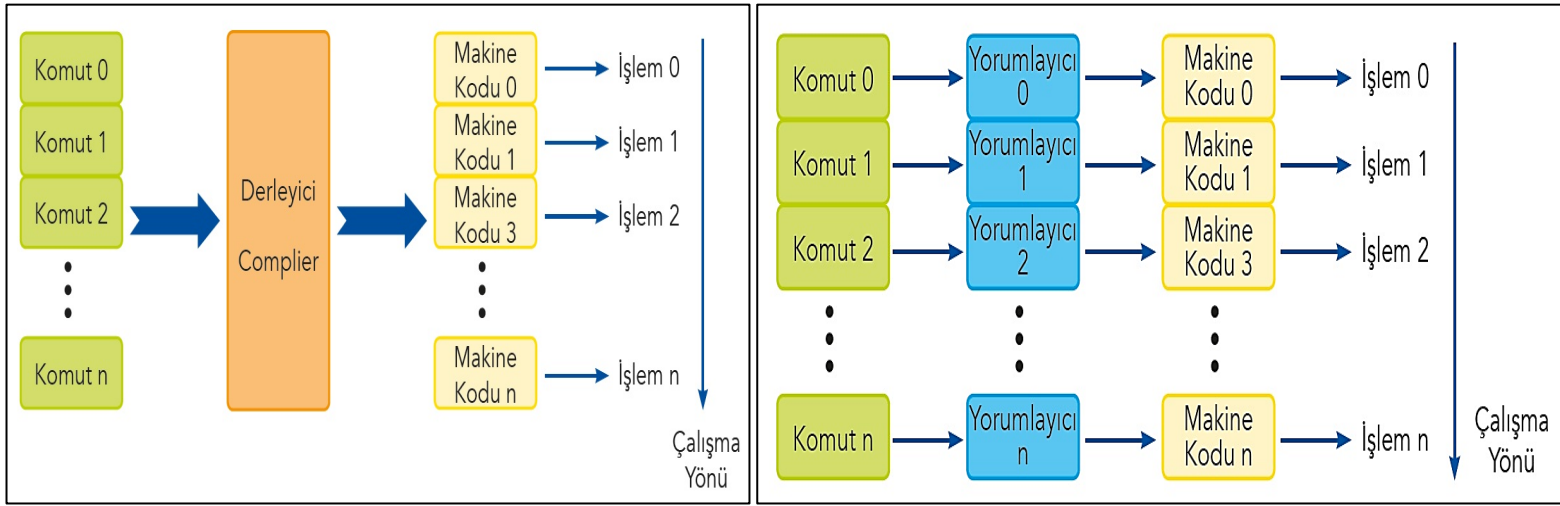
Python IDLE editörü temel seviyede oldukça basit bir editördür. Uzun ve karmaşık programları yazmak biraz zor olabilir. Çünkü satırların çoğaltılması, taşınması, komut bloklarının ayarlanması, düzenlenmesi, birden fazla pencerenin kullanılması gibi durumlarda editörün pek fazla özelliği yoktur. Python kod yazımı için birçok editör mevcuttur ve her bir editörün kendine özgü özellikleri de olabilmektedir. Bu editörlerden bazıları şunlardır:

- Visual Studio Code (VS Code) <https://code.visualstudio.com/Download>
- Pycharm <https://www.jetbrains.com/pycharm/>
- Wing IDE <https://wingware.com/>
- Sublime Text <https://www.sublimetext.com/>
- Spyder <https://www.spyder-ide.org/>
- Jupyter Notebook <https://jupyter.org/>
- Eclipse <https://eclipseide.org/>

Bu editörlerden her hangi biri seçilip kullanılabilir.

Python'a Giriş

Python, nesne tabanlı (**object-oriented**) ve yorumlamalı (**interpreted**) bir programlama dilidir. **Nesne tabanlı yapısı**, yazılımın daha **düzenli ve yeniden kullanılabilir** olmasını sağlayan bir model sunar. Bu modelde, veriler ve bu verilere uygulanacak **işlemler "nesne" adı verilen yapılar içinde** toplanır. Böylece, daha karmaşık projeler bile daha modüler ve yönetilebilir hale gelir. **Yorumlamalı** bir dil olması ise, Python'un satır satır çalıştırıldığı anlamına gelir; bu da geliştiricilerin **kodu anlık olarak çalıştırıp** hataları hızlı bir şekilde tespit etmelerini ve düzeltmelerini kolaylaştırır.



Derleyici bir dilin çalışma şekli

Yorumlamalı bir dilin çalışma şekli

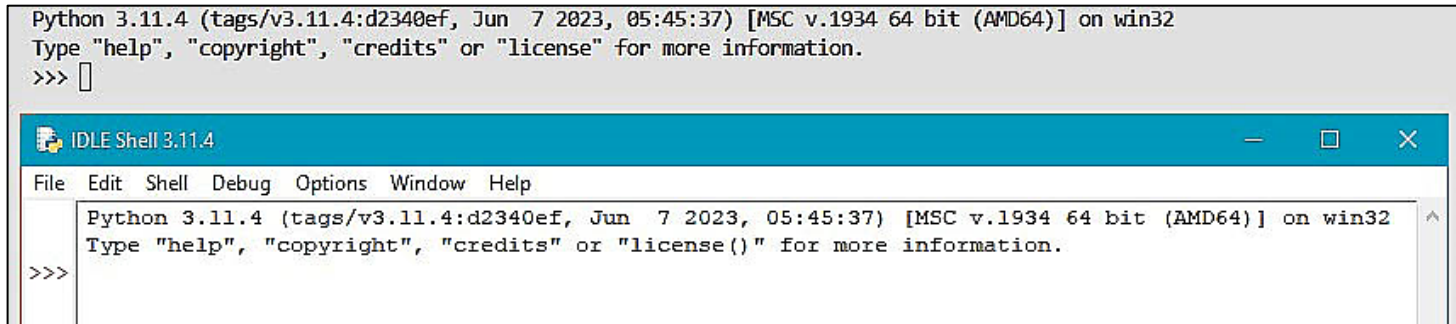
Python, yorumlamalı bir dil olduğu için, **kodlar satır satır çalıştırılır** ve **anında sonuç alınabilir**. Bu özellik, Python'un interaktif olarak kullanılmasına imkan verir. Bu da özellikle deneme-yanılma yaparken, **kodu test ederken veya hataları hızlıca bulup** düzeltirken oldukça kullanışlıdır. **İnteraktif kullanım sayesinde**, uzun programları baştan çalıştırmak yerine küçük kod parçalarını adım adım denenebilir ve üzerinde anında işlem yapılabilir.

<https://python-istihza.yazbel.com/index.html>

Python hakkında Türkçe hazırlanmış geniş kaynak pdf olarak indirilebilir.

Python'ın Çalıştırılması

Python yorumlamalı bir dil olması sebebiyle kodları satır satır işleyebilmektedir. Bu da hem **etkileşimli terminalde** hem de **metin tabanlı dosya** kullanılmasına imkan verir. Etkileşimli terminal kullanımı iki şekilde yapılabilir. İlki Python kurulumu ile gelen **IDLE editörünü** kullanmak, ikinci ise komut satırından "**py**" veya "**python**" yazıp "**enter**"e basarak etkileşimli moda geçmektir.



Etkileşimli terminal kullanımı

Metin tabanlı dosya kullanımı ise ilk önce her hangi bir metin editöründen gerekli kodlar yazılır ve bir dosya ismi verilerek (deneme.py) kaydedilir. Daha sonra terminal açılarak komut satırından "**py <dosya_ismi>.py**" komutu yazılır ve "**enter**"e basılarak dosya python derleyicisiyle çalıştırılır. Çıkan sonuçlar yine terminale yazılır.

```
print("deneme")  
a = 12 + 5  
print(a)
```

deneme.py

```
PS D:\VeriBilUyg> py deneme.py  
deneme  
17  
PS D:\VeriBilUyg> █
```

Terminalden dosya çalıştırma

Python kod yazım kuralları

Python kodu yazarken, daha temiz, anlaşılır ve sürdürülebilir kod yazmak için bazı kurallar ve kod yazım düzeni (kodlama standartları) takip edilmelidir. Python'un temel kod yazım standartları, **PEP 8** adı verilen bir belge ile belirlenmiştir. İşte Python'da uyulması gereken başlıca kurallar ve kod yazım düzeni hakkında bilgiler:

1. Girintileme (Indentation)

Python'da bloklar (if, for, while, fonksiyonlar gibi) süslü parantez yerine girintileme ile belirlenir. Girinti için genellikle **4 boşluk** kullanılır. Tab veya karışık boşluk kullanılması önerilmez.

Örnek

```
if True:
    ....print("Bu blok 4 boşluk ile girintilenmiştir.")
```

2. Boşluk Kullanımı

Operatörler arasında ve liste ya da sözlük tanımlarında uygun yerlerde boşluk bırakılmalıdır.

Yanlış

```
x=5+3
list=[1,2,3]
```

Doğru

```
x = 5 + 3
list = [1, 2, 3]
```

Python kod yazım kuralları

3. Satır Uzunluğu

Bir satırın uzunluğu 79 karakteri geçmemelidir. Daha uzun satırlar kodun okunabilirliğini azaltabilir. Eğer bir satırda kod çok uzarsa, satırın bölünmesi tavsiye edilir.

Örnek

```
total = first_variable + second_variable + third_variable + fourth_variable
```

Satırın çok uzun olması durumunda () parantezler kullanılarak şöyle bölünebilir

Örnek

```
total = (first_variable + second_variable +  
        third_variable + fourth_variable)
```

4. Fonksiyon ve Değişken İsimlendirme

Fonksiyon ve değişken isimleri küçük harflerle yazılır ve kelimeler arasında alt çizgi (`_`) kullanılır. **CamelCase** (**DeveHâli**) yerine **snake_case** (**yılan_hâli**) önerilir.

Yanlış

```
myVariable = 10  
def MyFunction():  
    pass
```

Doğru

```
my_variable = 10  
def my_function():  
    pass
```


Python kod yazım kuralları

5. Yorum Satırları (Comments)

Kodun anlaşılabilirliğini artırmak için yorumlar kullanılmalıdır. Yorumlar `#` işareti ile başlar ve açıklayıcı olmalıdır. Ancak fazla yorumdan kaçınılmalıdır. Yorumlar, kodun neden yapıldığını açıklamalı, nasıl yapıldığını değil.

Örnek

```
# Kullanıcının girdiği ismi yazdırır  
print(name)
```

6. Fonksiyonlar ve Sınıflar Arasında Boş Satır

Fonksiyonlar ve sınıflar arasında iki boş satır bırakılmalıdır. Fonksiyon içindeki bloklar ise bir boş satır ile ayrılabilir.

Örnek

```
def my_function():  
    pass  
  
def another_function():  
    pass
```

7. Sabit Değerler

Sabit değişkenler (değişmeyecek veriler) genellikle büyük harflerle tanımlanır.

Örnek

```
PI = 3.14159  
MAX_SPEED = 120
```

Python kod yazım kuralları

8. Modül ve Kütüphane Aktarımı (import)

Gerekli olan modül ve kütüphaneler, dosyanın başında bulunmalı ve her satırda sadece bir modül aktarımı yapılmalıdır. Ayrıca tek satırda çoklu aktarım yapmak da mümkündür.

Örnek

```
import os
import sys
```

9. Fonksiyon ve Sınıf Dokümantasyonu

Fonksiyonlar ve sınıflar için docstring kullanarak ne yaptıklarını açıklayan yorumlar eklenmelidir. Docstringler üç tırnak işareti `'''` kullanılarak yazılır.

Örnek

```
def add_numbers(a, b):
    """
    Bu fonksiyon iki sayıyı toplar.

    Parametreler:
    a (int): Birinci sayı
    b (int): İkinci sayı

    Returns:
    int: Toplam sonucu
    """
    return a + b
```

10. Dosya ve Modül İsimlendirme

Dosya ve modül isimleri küçük harflerden oluşmalı ve kelimeler arasında alt çizgi kullanılmalıdır. Sınıf isimleri ise büyük harfle başlamalı ve **DeveHâli** kullanılarak yazılmalıdır.

Örnek

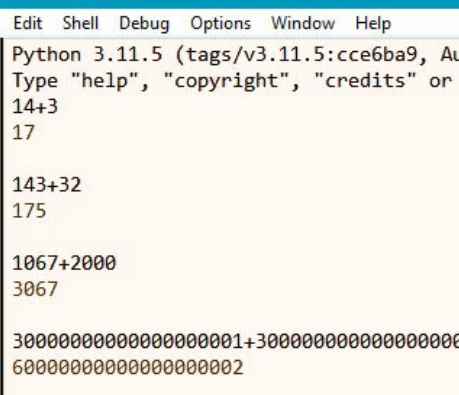
```
# Dosya adı: data_processing.py
class DataProcessor:
    pass
```

Temel Aritmetik İşlemler

Python da tüm programlama dilleri gibi temelde bir hesap makinesidir. Bu amaçla temel matematiksel işlemler için diğer dillerde olduğu matematiksel işaretler (operatörler) kullanılır, hatırlatmak gerekirse;

Operatör	Açıklama	İşlem	Sonuç
+	Toplama	2 + 2	4
-	Çıkarma	3 - 2	1
/	Bölme	4 / 2	2.0
*	Çarpma	3 * 2	6
%	Mod alma (Kalan)	12 % 5	2
**	Üs alma	2 ** 3	8
//	Tamsayı bölme (Bölüm)	14 // 3	4

Birkaç deneme yapalım



Python 3.11.5 (tags/v3.11.5:cce6ba9, Aug 24 2023)

Type "help", "copyright", "credits" or "license()" for more

```
>>> 14+3
17
>>>
>>> 143+32
175
>>>
>>> 1067+2000
3067
>>>
>>> 30000000000000000001+30000000000000000001
60000000000000000002
>>>
>>>
```

toplama

[illegible][illegible]

Python, yazılan bir matematiksel işlemi değerlendirirken matematiksel kurallara uygun olarak

- önce parantez içini,
- sonra üs alma işlemini

ve daha sonra da sırasıyla

- bölme ve çarpma işlemleri ile
- toplama ve çıkarma işlemlerini

dikkate alır. Etkileşimli terminalde bazı örnekler şu şekilde çalışmaktadır.

```
>>> 3**2 * 4
36
```

```
>>> 3**(2 * 4)
6561
```

```
>>> 5 + 4 / 2
7.0
```

```
>>> (5 + 4) / 2
4.5
```

```
>>> a = 12
>>> a
12
```

```
>>> a**2
144
```

```
>>> 10 // 3
3
```

```
>>> 23 // 3
7
```

```
>>> "Bunu yaz"
'Bunu yaz'
```

```
>>> 'Selam'+ ' size'
'Selam size'
```

```
>>> 2+5; 5-4
7
1
```

```
>>> 34*56; 42**3; 45/6
1904
74088
7.57
```

Aynı satırdan birden fazla komut noktalı-virgül (;) ile ayrılarak yazılabilir. Ancak sonuçlar ayrı ayrı yazılır. Sayılar üzerinde kullanılan toplama operatörü metinler üzerinde birleştirme işlemini gerçekleştirir.

Değişkenler

Değişken, en basit tanımıyla bir değeri daha sonra da kullanabilmek için bu değere verilen isimdir, örneğin, **e = 2.71828183** olarak verilirse, bundan sonra her defasında sayıyı uzun uzun yazmak yerine **`e`** yazılması yeterli olacaktır. Burada **`e`** bir değişkendir ve program için **`e`** yazıldığında aslında **`e`**'nin işaret edildiği değer anlaşılmaktadır.

Programların çalışması sırasında bilgisayarların hafızasında farklı türde veriler depolanır ve kullanılır. Verilerin depolandığı yerler **“nesne”** olarak isimlendirilir. Bu nesnelere ulaşmak için de isimlere ihtiyaç duyarız. Bir nesneye verilen isme de ayrıca **“değişken”** adı verilir.

Bir değişken tanımlamak için kullanılacak değişken ismi ve sonrasında **`=`** yazıldıktan sonra değişkene atanacak değer yazdır.

- Atama işleminde mutlaka soldan sağa doğru **değişken-operatör-değer** sıralaması vardır. Örneğin **a = 14;** atamasında **a** değişkeninin karşılığı **14** olur.
- **a = 2+4;** gibi bir atamada, **a** değişkeninin karşılığı olarak sağ taraftaki işlem **(2+4=6)** hesaplanıp, sonuç olarak soldaki **a** değişkenine atanır.

Bir değişkene basit bir değer atanabileceği gibi daha karmaşık veri yapıları da atanabilir.

```
>>> metin = 'Bu denemedir'
>>> pi = 3.141592
>>> x = 18
```

Bazı dillerde bir değişkeni kullanmadan önce mutlaka bu değişkeni ve bu değişkenin verinin tipini tanımlamak gerekir. Python'da ise buna gerek yoktur. Ayrıca, bir değişkenin hangi tipte olduğunu da açıkça belirtmeye de gerek yoktur. Python, değişkenin aldığı değere bakarak hangi tipte olduğunu anlayabilir.

Değişken isimleri

Python programlama dilinde, değişken adı olarak belirleyebileceğimiz kelime sayısı neredeyse sınırsızdır. Yani hemen hemen her kelimeyi değişken adı olarak kullanabiliriz. Ama yine de değişken adı belirlerken dikkat etmemiz gereken bazı kurallar var.

1. Değişken adları bir sayı ile başlayamaz. Yani şu kullanım yanlıştır:

```
>>> 3_kilo_elma = "5 TL"
```

2. Değişken adları aritmetik operatörlerle başlayamaz. Yani şu kullanım yanlıştır:

```
>>> +değer = 4568
```

3. Değişken adları ya bir alfabe harfiyle ya da `_` işaretiyle başlamalıdır:

```
>>> _değer = 4568  
>>> değer = 4568
```

4. Değişken adları içinde Türkçe karakterler kullanabilir. Ancak daha sonra beklenmedik uyum sorunları çıkması ihtimaline karşı değişken adlarında Türkçe karakter kullanmaktan kaçınmak gerekir.

5. Aşağıdaki kelimeleri değişken adı olarak kullanamazsınız:

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue',  
'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in',  
'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with',  
'yield']
```

```
>>> import keyword  
>>> keyword.kwlist
```


Değişken atamaları ve tavsiyeler

Diğer programlama dillerinin çoğunda olmayan bir özellik de Python'ın birden fazla değişkeni aynı anda tanımlamaya izin vermesidir. Ayrıca birden fazla değişkeni tanımlarken normal veya köşeli parantez de kullanılabilir.

```
>>> a, b = 3.14, 2.78
>>> a
3.14
>>> b
2.78
```

```
>>> [x, y, z] = [7, 8, 9]
>>> x
7
>>> z
9
```

Mümkün olduğu kadar kısa ancak aynı zamanda anlaşılır değişken isimlerin her zaman tercih edilmesi önerilir. Örneğin, bir programda **müşteri numarası** için değişkene ihtiyaç duyulması halinde **mn**, **m_n**, gibi pratik ancak anlaşılması zor değişken isimleri yerine **müşteri_no** gibi bir değişken ismini tercih edilmelidir.

Öncelikle, değişken isimlerini küçük harflerle tanımlamak, şart olmamakla birlikte tavsiye edilen bir alışkanlıktır. Değişken ismi, birden fazla kelimeden oluşuyorsa kelimeleri alt çizgi ile ayırabilirsiniz (**yılan_hâli**) ya da ilk kelime küçük harf, sonraki her kelimenin ise sadece ilk harfi büyük olacak şekilde bitişik (**DeveHâli**) olarak yazabilirsiniz: **müşteri_no** ya da **musteriNo** gibi.

Diğer dillerde değişkenlerde nokta kullanılması kabul edilse de Python'da değişken isminde **nokta kullanılmaz**. Son olarak değişken isimlerinin büyük ve küçük harfe duyarlı olduğuna dikkat etmek gerekir. Yani, **abc**, **ABC**, **Abc**, **aBc** farklı değişken isimlerini temsil eder.

Python'ın temel bazı komutlar

Python da sıklıkla kullanılan iki komut (fonksiyon) vardır, biri **type()** diğeri ise **print()**'dir. **type()** isminden de anlaşılacağı üzere bir değişkenin veri tipini öğrenmek için kullanılır. **print()** ise genel olarak terminale veya dosyaya verileri yazmak için kullanılır ve bu fonksiyonunun çok çeşitli kullanımları söz konusudur.

```
>>> a = 3.14
>>> type(a)
<class 'float'>
```

```
>>> b = 12
>>> type(b)
<class 'int'>
```

```
>>> c = 'deneme'
>>> type(c)
<class 'str'>
```

```
>>> d = True
>>> type(d)
<class 'bool'>
```

print() fonksiyonunda yazılacak değer veya değişkenler iki yuvarlak parantez arasına konulur ve çoklu değerler aralarına virgül konularak yazılır. Çoklu yazımda çıktıda virgüllerin yerlerine bir boşluk yazılır.

```
>>> print("Aramak istediğiniz kelimeyi yazın: ")
Aramak istediğiniz kelimeyi yazın:

>>> print('Kolay', 'gelsin', 'sizlere')
Kolay gelsin sizlere
```

Ancak bazen değerler arasında boşluk istenmeyebilir. Bu durumda print fonksiyonunun **sep** parametresini kullanmak gereklidir. Separator (ayraç) kelimesinin kısaltmasıdır. Bu parametre ekrana basılacak öğeler arasında hangi karakterin yerleştirileceğini gösterir. Ön tanım olarak **sep=" "** şeklinde bir boşluk karakteri olarak tanımlanmıştır.

```
>>> print("http://", "www.", "sabah.", "com")
http:// www. sabah. com

>>> print("http://", "www.", "sabah.", "com", sep="")
http://www.sabah.com
```

Python'ın temel bazı komutlar

Bazı durumlarda çoklu verilerin ayrı ayrı satırlarda yazılması istenebilir. Bu durumda en kolay metot **sep="\n"** şeklinde bir ayarlama yapmaktır. Burada **"\n"** yeni satır anlamına gelmektedir.

```
>>> print("birinci satır", "ikinci satır", "üçüncü satır", sep="\n")
birinci satır
ikinci satır
üçüncü satır
```

Aynı sonuç aşağıdaki şekilde de elde edilebilir. Hangisinin kullanılacağına amaca göre karar verilmelidir.

```
>>> print("birinci satır\nikinci satır\nüçüncü satır")
birinci satır
ikinci satır
üçüncü satır
```

Print fonksiyonun diğer bir parametresi **end**'dir. Bu parametre yazılan her hangi veri için satır sonu işlemini ayarlamak için kullanılır. Ön tanım olarak **end="\n"** şeklinde tanımlanmıştır, veri yazıldıktan sonra satırın sonlandırılıp yeni bir satıra (new line) geçeceğini belirtir. Ancak bazı durumlarda verilerin yan yana yazılması gerekebilir.

```
>>> print(1, end=""); print(2, end=""); print(3)
123
```

Print fonksiyonun diğer bir parametresi **file**'dir. Bu parametre çıktı işleminin bir dosyaya yapılmasını sağlamak için kullanılır. Verilen kaydedileceği dosya bu parametre ile verilmektedir.

```
>>> dosya = open("deneme.txt", "w")
>>> print("Temel bilgiler", file=dosya)
>>> dosya.close()
```

```
>>> f = open("sonuclar.txt", "w")
>>> print("1. sonuç", file=f)
>>> print("2. sonuç", file=f)
>>> f.close()
```

Python'ın temel bazı komutlar

Python'da **len()** fonksiyonu da sıklıkla kullanılmaktadır. Bu fonksiyon bir metin değişkenin karakter uzunluğunu vermektedir. Sayısal değerler için kullanılmaz. Ayrıca çok elemanlı değişkenlerin de eleman sayısını öğrenmek için kullanılır.

```
>>> bilgi = "Bugün günlerden pazartesi"
>>> len(bilgi)
28

>>> bilgi = ("Bugün", "günlerden", "pazartesi") # bir tuple (demet) verisi
>>> len(bilgi)
3
```

Python'da bir çok fonksiyon bulunmaktadır. Ancak bunların bazıları yeri geldikçe örnek verilerek açıklanacaktır.

```
abs()
round()
all()
any()
ascii()
repr()
bool()
bin()
bytes()
bytearray()
chr()
list()
set()
tuple()
```

```
frozenset()
complex()
float()
int()
str()
dict()
callable()
ord()
oct()
hex()
eval(),exec(),globals(),locals(),compile()
copyright()
credits()
license()
```

```
dir()
divmod()
enumerate()
exit()
help()
id()
input()
format()
filter()
hash()
isinstance()
len()
map()
max()
```

```
min()
open()
pow()
print()
quit()
range()
reversed()
sorted()
slice()
sum()
type()
zip()
vars()
```