PYTHON GİRİŞ

Python Nedir?

Python popüler bir programlama dilidir. Guido van Rossum tarafından yaratılmış ve 1991 yılında piyasaya sürülmüştür.

Ne İçin Kullanılır:

- Web geliştirme (sunucu tarafı),
- Yazılım geliştirme
- Matematik
- Sistem komut dosyası.

Python ile Neler Yapılabilir?

- Web uygulamaları oluşturmak için bir sunucuda kullanılabilir.
- İş akışları oluşturmak için yazılımla birlikte kullanılabilir.
- Veritabanı sistemlerine bağlanabilir. Ayrıca dosyaları okuyabilir ve değiştirebilir.
- Büyük verileri işlemek ve karmaşık matematik işlemlerini gerçekleştirmek için kullanılabilir.
- Hızlı prototip oluşturma veya üretime hazır yazılım geliştirme için kullanılabilir.

Neden Python

- Farklı platformlarda çalışır (Windows, Mac, Linux, Raspberry Pi, vb.).
- İngilizce diline benzer basit bir sözdizimine sahiptir.
- Geliştiricilerin diğer bazı programlama dillerinden daha az satırlı programlar yazmasına izin veren sözdizimine sahiptir.
- Bir yorumlayıcı sistemde çalışır, yani kod yazılır yazılmaz çalıştırılabilir. Bu, prototiplemenin çok hızlı olabileceği anlamına gelir.
- Prosedürel bir şekilde, nesne yönelimli bir şekilde veya işlevsel bir şekilde ele alınabilir.

Bunu bilmeniz iyi olur

- Python'un en yeni ana sürümü, ders kapsamında kullanacağımız Python 3'tür. Ancak Python 2, güvenlik güncellemeleri dışında hiçbir şeyle güncellenmese de hala oldukça popüler.
- Derste Python bir metin düzenleyicide veya PyCharm editörü kullanılarak yazılacaktır. Python dosyalarının büyük koleksiyonlarını yönetirken özellikle yararlı olan Thonny, Pycharm, Netbeans veya Eclipse gibi Entegre Geliştirme Ortamında Python yazmak mümkündür.

Diğer programlama dillerine kıyasla Python sözdizimi

- Python okunabilirlik için tasarlanmıştır ve matematikten etkilenen İngilizce ile bazı benzerliklere sahiptir.
- Python, genellikle noktalı virgül veya parantez kullanan diğer programlama dillerinin aksine, bir komutu tamamlamak için yeni satırlar kullanır.

- Python, kapsamı tanımlamak için boşluk kullanarak girintiye güvenir; döngüler, işlevler ve sınıfların kapsamı gibi. Diğer programlama dilleri genellikle bu amaçla küme parantezleri kullanır.

Örnek

Python'da konsola istenilen ifadeyi yazdırma

```
print("Merhaba Dünya!")
```

PYTHONA'A BAŞLARKEN

Python Kurulumu

Birçok PC ve Mac'te Python zaten kurulu olacaktır.

Windows PC'de Python'un kurulu olup olmadığını kontrol etmek için, başlangıç çubuğunda Python için arama yapın veya Komut Satırında (cmd.exe) aşağıdakini çalıştırın:

```
C:\Users\Your Name>python –version
```

Python'un Linux veya Mac'te kurulu olup olmadığını kontrol etmek için linux'ta komut satırını açın veya Mac'te Terminal'i açın ve şunu yazın:

```
python --version
```

Bilgisayarınızda Python yüklü olmadığını fark ederseniz, aşağıdaki web sitesinden ücretsiz olarak indirebilirsiniz: https://www.python.org/

Python hızlı başlangıç

Python yorumlanmış bir programlama dilidir, bu, geliştirici olarak Python (.py) dosyalarını bir metin düzenleyiciye yazmanız ve ardından bu dosyaları yürütülmek üzere python yorumlayıcısına koymanız anlamına gelir.

Bir python dosyasını çalıştırmanın yolu komut satırında şöyledir:

```
C:\Users\Your Name>python merhabadunya.py
```

"merhabadunya.py", python dosyanızın adıdır.

Herhangi bir metin düzenleyicide yapılabilen merhabadunya.py adlı ilk Python dosyamızı yazalım.

merhabadunya.py

```
print("Hello, World!")
```

Bu kadar basit. Dosyanızı kaydedin. Komut satırınızı açın, dosyanızı kaydettiğiniz dizine gidin ve çalıştırın:

```
C:\Users\Your Name>python merhabadunya.py
```

Çıktı şöyle olmalıdır:

```
Merhaba Dünya!
```

Tebrikler, ilk Python programınızı yazıp çalıştırdınız.

Python Komut Satırı

Python'da kısa miktarda kodu test etmek için bazen kodu bir dosyaya yazmamak en hızlı ve en kolay yoldur. Bu, Python'un kendisi bir komut satırı olarak çalıştırılabildiği için mümkün olmuştur.

Windows, Mac veya Linux komut satırına aşağıdakini yazın:

```
C:\Users\Your Name>python
```

Veya "python" komutu çalışmadıysa "py" komutunu deneyebilirsiniz:

```
C:\Users\Your Name>py
```

Buradan, önceki bölümde söz edilen merhaba dünya örneğimiz de dahil olmak üzere herhangi bir python kodu yazabilirsiniz:

```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Merhaba Dünya!")
```

Komut satırında "Merhaba, Dünya!" yazar:

```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello
Hello, World!
```

Python komut satırında işiniz bittiğinde, python komut satırı arayüzünden çıkmak için aşağıdakini yazmanız yeterlidir:

```
exit()
```

PYTHON SYNTAX

Python Sözdizimini Yürütmek

Önceki sayfada öğrendiğimiz gibi, Python sözdizimi doğrudan Komut Satırına yazılarak yürütülebilir:

```
>>> print("Hello, World!")
Hello, World!
```

Veya sunucuda bir python dosyası oluşturarak, .py dosya uzantısını kullanarak ve bunu Komut Satırında çalıştırarak:

```
C:\Users\Your Name>python dosyam.py
```

Python'da Girinti

Girinti, bir kod satırının başındaki boşlukları ifade eder.

Diğer programlama dillerinde koddaki girinti yalnızca okunabilirlik içindir, Python'daki girinti çok önemlidir.

Python, bir kod bloğunu belirtmek için girinti kullanır.

Örnek

```
if 5 > 2:
  print("Beş ikiden büyüktür!")
```

Girintiyi atlarsanız Python size bir hata verecektir:

```
if 5 > 2:
print("Beş ikiden büyüktür!") -> Hata verecektir!!!
```

Bir programcı olarak boşluk sayısı size kalmış, en yaygın kullanım dört ama en az bir olması gerekiyor.

Örnek

```
if 5 > 2:
  print("Beş ikiden büyüktür!")
if 5 > 2:
     print("Beş ikiden büyüktür!")
```

Aynı kod bloğunda aynı sayıda boşluk kullanmanız gerekir, aksi takdirde Python size bir hata verecektir:

Örnek

Sözdizimi Hataları:

```
if 5 > 2:
  print("Beş ikiden büyüktür!")
     print("Beş ikiden büyüktür!")
```

Python Değişkenleri

Python'da, ona bir değer atadığınızda değişkenler oluşturulur:

Örnek

Python'daki Değişkenler:

```
x = 5
y = "Merhaba Dünya!"
```

Python'un değişken bildirmek için bir komutu yoktur.

Python Değişkenleri bölümünde değişkenler hakkında daha fazla bilgi edineceksiniz.

Yorumlar

Python, kod içi dokümantasyon amacıyla yorum yapma yeteneğine sahiptir.

Yorumlar # ile başlar ve Python satırın geri kalanını yorum olarak oluşturur:

Örnek

Python'daki yorumlar:

```
#Bu bir yorumdur.
print("Merhaba Dünya!")
```

PYTHON YORUMLAR

Python kodunu açıklamak için yorumlar kullanılabilir.

Yorumlar, kodu daha okunabilir hale getirmek için kullanılabilir.

Yorumlar, kodu test ederken yürütmeyi önlemek için kullanılabilir.

Yorum Oluşturma

Yorumlar # ile başlar ve Python onları görmezden gelir:

Örnek

```
#Bu bir yorumdur.
print("Merhaba Dünya!")
```

Yorumlar bir satırın sonuna yerleştirilebilir ve Python satırın geri kalanını yok sayar:

Örnek

```
print("Merhaba Dünya!") # Bu bir yorumdur.
```

Yorumun kodu açıklayan bir metin olması gerekmez, ayrıca Python'un kod yürütmesini önlemek için de kullanılabilir:

Örnek

```
#print("Merhaba Dünya!")
print("Selamlar!")
```

Çok Satırlı Yorumlar

Python'un gerçekten çok satırlı yorumlar için bir sözdizimi yoktur.

Çok satırlı bir yorum eklemek için her satıra bir # ekleyebilirsiniz:

Örnek

```
#Çok satırlı yorumlarda
#Yorum başında # işareti koyulur
print("Merhaba Dünya!")
```

Veya tam olarak amaçlandığı gibi değil, çok satırlı bir dize kullanabilirsiniz.

Python, bir değişkene atanmamış dize değişmezlerini yok sayacağından, kodunuza çok satırlı bir dize (üçlü tırnak) ekleyebilir ve yorumunuzu bunun içine yerleştirebilirsiniz:

```
"""
Bir başka çok satırlı
oluşturma biçimi
"""
print("Merhaba Dünya!")
```

Dize bir değişkene atanmadığı sürece, Python kodu okuyacak, ancak daha sonra görmezden gelecek ve çok satırlı bir yorum yaptınız.

PYTHON DEĞİŞKENLERİ

Değişkenler

Değişkenler, veri değerlerini depolamak için kapsayıcılardır.

Değişkenler Oluşturma

Python'un değişken bildirmek için bir komutu yoktur.

Bir değişken, ona bir değer atadığınız anda oluşturulur.

Örnek

```
x = 5
y = "Selim"
print(x)
print(y)
```

Değişkenlerin belirli bir türle bildirilmesi gerekmez ve ayarlandıktan sonra türü bile değiştirebilir.

Örnek

```
x = 3      # x int tipindedir
x = "Tuna" # x şimdi de str tipindedir
print(x)
```

Kalıp

Bir değişkenin veri tipini belirtmek isterseniz bu kalıp yayınlama/döküm ile yapılabilir.

Örnek

```
x = str(3)    # x '3' olacaktır.
y = int(3)    # y 3 olacaktır.
z = float(3)  # z 3.0 olacaktır.
```

Türü Alın

type() işleviyle bir değişkenin veri türünü alabilirsiniz.

Örnek

```
x = 5
y = "Alp"
print(type(x))
print(type(y))
```

Bu ders kapsamında daha sonradan veri türleri ve yayınlama hakkında daha fazla bilgi edineceksiniz.

Tek tırnak mı çift tırnak mı?

Dize değişkenleri, tek veya çift tırnak kullanılarak bildirilebilir:

Örnek

```
x = "Ahmet"
# ikisi de aynı ifadedir
x = 'Ahmet'
```

Büyük/Küçük Harf Duyarlılığı

Değişken isimleri büyük/küçük harf duyarlıdır.

Örnek

Bu iki değişken yaratacaktır:

```
a = 25
A = "Gizem"
#A küçük a üzerine yazılmaz, ayrı bir değişkendir.
```

Değişken İsimleri

Bir değişkenin kısa bir adı (x ve y gibi) veya daha açıklayıcı bir adı (yaş, aracadi, toplam_hacim) olabilir. Python değişkenleri için kurallar:

- Değişken adı bir harf veya alt çizgi karakteri ile başlamalıdır
- Değişken adı bir sayı ile başlayamaz
- Değişken adı yalnızca alfasayısal karakterler ve alt çizgiler içerebilir (A-z, 0-9 ve _)
- Değişken adları büyük/küçük harfe duyarlıdır (yaş, Yaş ve YAŞ üç farklı değişkendir)

Örnek

Geçerli değişken adları:

```
myvar = "Ahmet"
my_var = " Ahmet"
_my_var = " Ahmet"
myVar = " Ahmet"
MYVAR = " Ahmet"
myvar2 = " Ahmet"
```

Örnek

Geçersiz değişken adları:

```
2myvar = "Mehmet"
my-var = " Mehmet"
my var = " Mehmet"
```

NOT: Değişken adlarının büyük/küçük harfe duyarlı olduğunu unutmayın

Çoklu Kelime Değişken İsimleri

Birden fazla kelime içeren değişken isimlerinin okunması zor olabilir.

Bunları daha okunaklı hale getirmek için kullanabileceğiniz birkaç teknik vardır:

Deve Durumu

İlk kelime hariç her kelime büyük harfle başlar:

```
myVariableName = "Ayşe"
```

Pascal Durumu

Her kelime büyük harfle başlar:

```
MyVariableName = "Ayşe"
```

Yılan Durumu

Her kelime bir alt çizgi karakteri ile ayrılır:

```
my_variable_name = "Ayşe"
```

Birden Çok Değer Atama

Birden Çok Değişkene Birçok Değer

Python, bir satırda birden çok değişkene değer atamanıza izin verir:

Örnek

```
x, y, z = "Portakal", "Muz", "Kiraz"
print(x)
print(y)
print(z)
```

NOT: Değişken sayısının değer sayısıyla eşleştiğinden emin olun, aksi takdirde bir hata alırsınız.

Birden Çok Değişkene Bir Değer

Ve aynı değeri bir satırda birden çok değişkene atayabilirsiniz:

Örnek

```
x = y = z = "Portakal"
print(x)
print(y)
print(z)
```

Koleksiyonu Açma

Bir listede değerler koleksiyonunuz varsa, tuple vb. Python, değerleri değişkenlere çıkarmanıza izin verir. Buna ambalaj açma denir.

Örnek

Bir listeyi açın:

```
meyveler = ["elma", "muz", "kiraz"]
x, y, z = meyveler
print(x)
print(y)
print(z)
```

Çıkış Değişkenleri

Python'da print() işlevi genellikle değişkenleri çıktılamak için kullanılır.

Örnek

```
x = "Python nesne yönelimli bir dildir"
print(x)
```

print() işlevinde, virgülle ayrılmış birden çok değişken çıktısı alırsınız:

Örnek

```
x = "Python"
y = "nesne yönelimli"
z = "bir dildir"
print(x, y, z)
```

Birden çok değişkenin çıktısını almak için + operatörünü de kullanabilirsiniz:

Örnek

```
x = "Python"
y = "nesne yönelimli"
z = "bir dildir"
print(x + y + z)
```

"Python" ve "yönelimli" den sonraki boşluk karakterine dikkat ediniz, aksi halde onlar olmadan sonuç "Pythonnesne yönelimlibir dildir" şeklinde çıktı oluşur.

Sayılar için + karakteri matematiksel bir operatör olarak çalışır:

Örnek

```
x = 5
y = 10
print(x + y)
```

print() işlevinde, + operatörüyle bir dize ve sayıyı birleştirmeye çalıştığınızda Python size bir hata verecektir:

Örnek

```
x = 5
y = "Ahmet"
print(x + y)
```

print() işlevinde birden çok değişkeni çıktı almanın en iyi yolu, onları farklı veri türlerini bile destekleyen virgüllerle ayırmaktır:

Örnek

```
x = 5
y = "Ahmet"
print(x, y)
```

Genel (Global) Değişkenler

Bir fonksiyonun dışında oluşturulan değişkenler (yukarıdaki tüm örneklerde olduğu gibi) global değişkenler olarak bilinir.

Global değişkenler hem fonksiyonların içinde hem de dışında herkes tarafından kullanılabilir.

Örnek

Bir fonksiyonun dışında bir değişken oluşturun ve onu fonksiyonun içinde kullanın

```
x = "bir dildir"

def fonk():
  print("Python " + x)

fonk()
```

Bir fonksiyon içinde aynı isimde bir değişken oluşturursanız, bu değişken yerel olur ve sadece fonksiyon içinde kullanılabilir. Aynı ada sahip global değişken, olduğu gibi, global ve orijinal değerde kalacaktır.

Örnek

Global değişkenle aynı ada sahip bir fonksiyon içinde bir değişken oluşturun

```
x = "muhteşem"

def fonk():
  x = "harika"
  print("Hava " + x)

fonk ()

print("Hava " + x)
```

‘global’ Anahtar Kelimesi

Normalde, bir işlev içinde bir değişken oluşturduğunuzda, bu değişken yereldir ve yalnızca o işlevin içinde kullanılabilir.

Bir fonksiyon içinde global bir değişken oluşturmak için global anahtar sözcüğünü kullanabilirsiniz.

Örnek

Global anahtar sözcüğünü kullanırsanız, değişken global kapsama aittir:

```
def fonk():
  global x
  x = "harika"

fonk()

print("Hava " + x)
```

Ayrıca, bir fonksiyon içindeki global bir değişkeni değiştirmek istiyorsanız global anahtar sözcüğünü kullanın.

Örnek

Bir fonksiyon içindeki global değişkenin değerini değiştirmek için global anahtar kelimeyi kullanarak değişkene bakın:

```
x = "muhteşem"

def fonk():
  global x
  x = "harika"

fonk()

print("Hava " + x)
```