

# Python

- Bir başka faydalı kaynak olarak [buradaki repo](#)'ya bakabilirsin.
- Python örnekleri için [buraya](#) bakabilirsin.
- Mobile dev için Kiv çok iyidir, çoklu dokunma desteği sağlar.

Ek kaynak için [harici kaynaklara](#) bakmayı unutma

## İçerik

[HOME](#) tuşu ile yukarı yönlenebilrsiniz.

- [Kurulum ve Kullanım](#)
  - [VsCode Üzerinde Python](#)
    - [Python Kodlarını Formatlama](#)
    - [VsCode Debug Yapılandırması](#)
    - [VsCode Jupyter Desteği](#)
    - [VsCode Python Ortamı Ayarlama](#)
    - [VsCode Ek Python Ayarları](#)
    - [VsCode Python Eklentileri](#)
    - [Anaconda üzerindeki Python'ı Desteklemeyen Eklentiler](#)
    - [VsCode Python Kısayolları](#)
    - [VsCode Python Ortam Değişkenleri](#)
  - [Faydalı Soru & Cevaplar](#)
- [Python ile Programlamaya Hazırlanma](#)
  - [Yazım Kuralları](#)
  - [Dökümantasyon PyDoc](#)
- [Temel Python](#)
  - [Anahtar Kelimeler \(Keywords\)](#)
    - [Fonksiyon Oluşturma Anahtar Kelimeleri](#)
    - [Fonksiyon Anahtar Kelimeleri](#)
  - [Değişkenler](#)
    - [Ana Değişkenler](#)
    - [Ek Değişkenler](#)
    - [Değersiz Değişken Tanımlama](#)
    - [Sabit Değerler \(Constants\)](#)
    - [Değişkenler Arası Takılıma \(Casting\)](#)
    - [Değişken Tipleri için Ek Kaynak](#)
    - [Değişken ve Sabitlerde Gizlilik](#)
  - [Operatörler](#)
    - [Aritmatik Operatörler](#)
      - [Ek Aritmatik Operatörler](#)
    - [Karşılaştırma Operatörleri](#)
    - [Mantıksal Operatörler](#)
    - [Bit Düzeyinde Operatörler](#)
    - [Kimlik Belirleme Operatörleri](#)

- Kimlik Belirleme Operatörleri Örneği
- Üyelik Operatörleri
  - Üyelik Operatörleri Örneği
- If / Else Koşul (Constraints) Yapısı
  - Tek satır (üçlü) If / Else Yapısı
- Döngüler (Loop)
  - For Döngüsü
    - Değişken içinde For Döngüsü
    - İki Liste Üzerinde Paralel For Döngüsü
  - While Döngüsü
  - Range Fonksiyonu
- Break / Continue
- Fonksiyonlar
  - Dahili Fonksiyon Kullanımları
    - Ekrana Yazma / Print İşlemleri
    - String İşlemleri
  - Harici Fonksiyon Kullanımları
    - Harici String İşlemleri
    - Dizin ve Yol İşlemleri
  - Fonksiyon Oluşturma
    - Fonksiyon İskeleti
    - Fonksiyon Örneği
    - Fonksiyon Dökümantasyonu
    - Fonksiyon Varsayılan Parametreler
    - Fonksiyonlarda Keyfi Parametreler
    - Özyineleyen Fonksiyonlar
      - Özyineleyen Fonksiyonların Avantajları
      - Özyineleyen Fonksiyonların Zararları
  - Lambda Fonksiyonlar
    - Filter ile Lambda Kullanımı
    - Map ile Lambda Kullanımı
- Global, Local ve Nonlocal Kavramları
  - Global, Local ve Nonlocal Kavramlarına Örnek
  - Global Kullanımına Örnek
- Modüller
  - Modül Kullanım Örnekleri
  - Python Modül Dosyaları
    - Sistemin Python Modüllerine Bakma
    - Modül İçinde Tanımlanan İsimleri Alma
- Paketler (Package)
  - Paketten ve Modül Örnekleri
  - Sık Kullanılan Paketler
    - Windows Paketleri
    - Görüntü İşleme Paketleri
    - Giriş Çıkış (I/O) Kontrol Paketleri
  - Paketler için Harici Bağlantıları

- Sayılar, Sayılar Arası Dönüşüm ve Matematik
  - Tabanlı Sayılar
  - Ondalıklı Sayılar (Decimals / Floats)
    - Decimal Float Kullanımları ve Farkı
  - Kesirli Sayılar (Fractions)
    - Kesirli Sayılarla İşlemler
  - Matematik İşlemleri
    - Matematikte Rastgelelik
- Class
  - Class Anahtar Kelimeleri
  - Basit Class Örneği
  - Metodlu Class Örneği
    - Obje Özelliği Silme
    - Class Silme
  - Scopes and Namespaces
  - Enumeration
    - Basit Kullanım
    - Enum Özellikleri
      - Benzersin Enum Tanımlaması
- Komut İsteminden Python (CLI)
  - Argparse Modülü Detayları
  - Argüman Ekleme
  - Argüman Action Özelliği
  - Örnek CLI Kodu
- Python Görsel Programlama (GUI)
  - PyQt5 Kurulumu
  - Basit GUI Yapımı
  - PyQt Widgets
- PyInstaller ile Executable Dosya Oluşturma
- İleri Seviye Python
  - Assertion (Kural Koyma)
    - Assertion Örnekleri
  - Try / Except Yapısı
  - Dosya İşlemleri
    - Dosya Okuma
  - Thread
    - Basit Thread Yapısı
    - Zamanlayıcı Yapısı (Timer)
    - Bir Plana göre Fonksiyon Çalıştırma
  - Paralel İşlemler (Multiprocessing)
    - Multiprocessing Örneği
  - Kod Parçaları (Code Snippet)
    - Ekran Görünüsünü Alma ve Kaydetme
    - Kisayol ile Ekran Alanı Seçme
    - Url Encode İşlemi
- Google Colabrotory Üzerinden Python

- IPython Operatorleri
- Python Değişkenlerinin Bash Üzerinde Kullanımı
- Ortam Değişkenleri
  - PyCharm Uygulamasında Ortam Değişkeni Tanımlama
- Yapıacaklar
- Harici Kaynaklar

## Kurulum ve Kullanım

- Temel python kurulumunu, resmi sitesinden [buraya](#) tıklayarak tamamlayabilirsin.
- pip paket yöneticisini kullanır
- pip install komutu ile modül yüklemesi yapılır
- Makine öğrenimi ve veri bilimi gibi işlemlerle uğışacak isen, bu iş için geliştirilmiş olan [Anaconda](#) veya [MiniConda](#) yazılımı tavsiye edilir
  - Resmi paket yönetici conda olmasına rağmen pip üzerinden de kurulumu izin verir
  - conda install komutu ile modül yüklemesi yapılır
  - Temel yükleme yapısı 'Conda ile yüklenemezse pip kullan' idir
  - Anaconda, MiniConda ve VirtualEnv farkı kullanımı için [buraya](#) bakabilirsın

Anaconda ile alakalı bilgiler [burada](#) derlenmektedir.

## VsCode Üzerinde Python

Başlangıç dökümanı için [buraya](#) bakabilirsın.

## Python Kodlarını Formatlama

- CTRL + SHIFT + P yapın
- Çıkan alana Python: Select Linter yazın
- pylint düzenleyicisini seçin
  - pylint aynı dizindeki modulleri bulamamakta, bu hatananın çözümü için .pylintrc dosyasını düzenlemek gerekmekte
  - 
  - pylint --generate-rcfile .pylintrc komutunu çalışma dizininde yazdıktan sonra, içini açıp #init-hook satırını init-hook='import sys; system.path.append("\${workspaceFolder}")' ile değiştirin. (Yorum satırı olmaktan kaldırın)
  - Eğer girintiyi TAB ile yapıyorsanız pylint'de bug'a sebebiyet vermekte, SPACE kullanın
- Python derleyicinize autopep8 paketini aşağıdaki komutlarla veya vscode arayüzü ile yükleyin
  - pip install autopep8
  - conda install autopep8
- Artık SHIFT + ALT + F ile kodları düzenleyebilirsınız.
- Dosyaya sağ tıklayarak derleyebilirsiniz.

## VsCode Debug Yapılandırması

Detaylar için [buraya](#) bakabilirsın.

- **CTRL + SHIFT + D** ile debug ekranını açın
- Sol üstte açılan ekrandan **ayarlar butonuna** tıklayın
- **Python** kısmını seçin

## VsCode Jupyter Desteği

Detaylar için [buraya](#) bakabilirsin.

- Kod alanının üstüne **#%%** yazarak oluşturabilirsiniz.

## VsCode Python Derleyicisi Ayarlama

Aktif olan derleyici ortamı, en alta bulunan durum çubuğunun solunda gösterilmektedir. Değiştirmek için:

- **CTRL + SHIFT + P** tuş kombinasyonuna basın
- Çıkan alana **Python: Select Interpreter** yazınız
- Çıkan ekrandan istediğiniz derleyiciyi seçiniz

## VsCode PYTHONPATH Oluşturma

- Çalışma dizininde **.env** dosyası oluşturun
- **.env** dosyasının içerişine **PYTHONPATH=** satırını ekleyin
  - Örnek için bir alttaki başlığa bakınız
- Vscode ayarlarına **"python.envFile": "\${workspaceFolder}/.env"** satırını ekleyin
- Vscode'u yeniden başlatın

Kaynak için [buraya](#) bakabilirsin. Ek olarak [buraya](#) bakmada da fayda var.

## Vscode PYTHONPATH Örneği

Resmi döküman için [buraya](#) bakabilirsin.

- VsCode birden fazla satır sahip değişken değerlerini kabul etmez
- Ortam değişiklenleri oluşturmak için proje ayarlarından **env file** seçmemiz gerekmekte
- Ardından içine değişkenlerimizi tanımlamamız gerekmekte

```
"python.envFile": "${workspaceFolder}/prod.env"
```

```
# prod.env
# Python kaynak dizinleri
RESEARCH_FOLDER=C:/Users/YEmre/Documents/Tensorflow/models/research
OBJECT_FOLDER=C:/Users/YEmre/Documents/Tensorflow/models/research/object_detection
SLIM_FOLDER=C:/Users/YEmre/Documents/Tensorflow/models/research/slim
SCRIPT_FOLDER=C:/Users/YEmre/Documents/Tensorflow/scripts

# Python modül yolu
```

```
PYTHONPATH=${RESEARCH_FOLDER}: ${OBJECT_FOLDER}: ${SLIM_FOLDER}: ${SCRIPT_FOLD  
ER}
```

```
PYTHONPATH=./src:${PYTHONPATH}
```

Kaynak için [buraya](#) bakabilirsin.

## VsCode Ek Python Ayarları

Ek python ayarları için [buradaki](#) resmi dökümana bakabilirsin.

## VsCode Python Eklentileri

Eklenti	Açıklama
<a href="#">Python</a>	Dil desteği
<a href="#">Visual Studio IntelliCode - Preview</a>	Sık kullanılan kod önerileri ( <b>eksik öneriler olabilir</b> )
<a href="#">autoDocstring</a>	Dökümantasyon parçaları sağlayan eklenti
<a href="#">Better Comment</a>	Yorum satırı renklediricisi

## Anaconda üzerindeki Python'ı Desteklemeyen Eklentiler

Eklenti	Açıklama
<a href="#">AREPL for python</a>	Anlık çıktıları gösterme
<a href="#">Code Runner</a>	Kod koşturucusu

## VsCode Python Kısayolları

Alttağı kısayollar `keybindings.json` dosyası içerisinde bulunmalıdır.

```
// Place your key bindings in this file to override the defaultsauto[]  
[  
 {  
   // Kod parçasını metoda çevirme  
   "key": "ctrl+shift+m ctrl+shift+m",  
   "command": "python.refactorExtractMethod"  
 },  
 {  
   // Kod parçasını metoda taşıma  
   "key": "ctrl+shift+v ctrl+shift+v",  
   "command": "python.refactorExtractVariable"  
 },  
 {  
   // İmport'ları sıralama  
 }
```

```
        "key": "ctrl+shift+s ctrl+shift+s",
        "command": "python.sortImports"
    },
    {
        "key": "shift+f10",
        "command": "python.execInTerminal"
    }
]
```

## Faydalı Soru & Cevaplar

- [What's the difference between a pip install and conda install?](#)
- [Module Package Library Meaning](#)

## Python ile Programlamaya Hazırlanma

### Yazım Kuralları

Orjinal dökümantasyon için [buraya](#) bakabilirsin.

- Her python dosyasına **modül** denir
  - `import` ile dahil edilirler
  - `.` ile içlerine erişilir
- Class isimleri için **camel case** yazım kuralı geçerlidir
  - Boşluk karakteri **harfi büyüterek** temsil edilir
  - `camelCase`
- Geri kalanlar için **snake case** yazım kuralı geçerlidir
  - Boşluk karakteri `_` ile temsil edilir
  - `snake_case`
- Girintiler (`\t` karakteri) `{}` işlevi görür
- `:` karakteri ile yeni bir scope (alt alan) açılır
  - `for`, `def` gibi döngü veya metod işlemlerinde kullanılır
- Metotlar arasında 2 satır bırakılır
- Metodların en son satırları boş olmalıdır (return için)
- Kodun en son satırı boş olmalıdır (End of File)

Daha fazla bilgi için harici linklerdeki [Should I use underscores or camel case for Python?](#) bağlantısına tıklayabilirsin.

### Dökümantasyon PyDoc

- `'''` ile fonksiyonların üstüne dökümantasyon (açıklama) eklenir
- `#` ile koda yorum eklenir

```
def func(a):
    """ 1 Değeri döndürür """
    return 1 # Döndürme keywordu
```

## PyDoc videosu

# Temel Python

## Anahtar Kelimeler (Keywords)

Harici link için [buraya](#) tıklayabilirsin.

Anahtar	Anlamı
<code>pass</code>	Tanımsız (null)
<code>is</code>	Eşitlik (==)
<code>in</code>	İçerisindeki elemanlar
<code>with</code>	Açık olduğu sürece anlamı taşıır

Döngü veya metodların *içeri* *doldurulana* kadar yer kaplayıcı olarak `pass` kullanılır.

## Fonksiyon Oluşturma Anahtar Kelimeleri

Anahtar	Oluşturma	Erişim
<code>Lambda</code>	<code>m_lambda = lambda x: x*2</code>	<code>m_lambda(2)</code>
<code>Function</code>	<code>def m_func(param):</code>	<code>m_func(5)</code>

## Fonksiyon Anahtar Kelimeleri

Anahtar	Anlamı
<code>return</code>	Veri döndürme
<code>yield</code>	Her çağrılmada tek bir veri döndürme (generator)

## Değişkenler

### Ana Değişkenler

Tip	Açıklama	Örnek
<code>bool</code>	2'li değer, bit	<code>True</code>
<code>int</code>	Sayı	<code>1</code>
<code>float</code>	Virgülü sayı	<code>1.2</code>
<code>complex</code>	Karmaşık sayılar	<code>2+3j</code>
<code>str</code>	String, metin	<code>"Hello" / 'Hello'</code>

### Ek Değişkenler

Tip	Oluşturma	Erişim
List	liste = [1, 2]	liste[index]
Set	kume = {1.0, "Hello", (1, 2, 3)}	kume.add(1)
Dictionary	site = {"adi": "yemreak"}	site['adi']
Tuple	konum = (1, 2)	x, y = konum

- [ for  in <dizi veya liste> if <koşul>] istenen koşullardaki elemanların listesini verir
 
  - Örn: [x for x in a if x != 20]

## Değersiz Değişken Tanımlama

```
degersiz = None
```

## Sabit Değerler (Constants)

Python'da *constant*'lar yoktur. Sabit değerler büyük harfler ile belirtilir.

Aynı dosya içerisinde büyük harflerle yazılsa bile değiştirilebilir.

### sabitler.py dosyası

```
PI = 3.14
YER_CEKIMI = 9.8
```

### main.py dosyası

```
import sabitler

print(sabitler.PI) # 3.14
print(sabitler.GRAVITY) # 9.8
```

## Değişkenler Arası Takılıma (Casting)

```
ondalikli = 5.8
tam = int(5.8) # 5 atanır
sonuc = int(7/3.5) # 2 atanır
sonuc = int(7/3) # 2 atanır
sonuc = float(7 / 3.5) # 2.0 atanır
sonuc = 7 / 3 # 2.33 atanır
```

## Değişken Tipleri için Ek Kaynak

- [Basic Data Types in Python](#)

## Değişken ve Sabitlerde Gizlilik

- `_` ile gizli anlamında gelmektedir.
  - Dışarıdan sadece `_<class>.__<değişken>` şeklinde erişilebilir

Detaylar için [buraya](#) bakabilirsin.

## Değişkenin Tanımlı Olduğunu Kontrol Etme

```
if 'myVar' in locals():
    # myVar exists.
if 'myVar' in globals():
    # myVar exists.
if hasattr(obj, 'attr_name'):
    # obj.attr_name exists.
```

Kaynak için [buraya](#) bakabilirsin.

## Operatörler

### Operatör      Açıklama

`\` Satır atlatmayı geçersiz kılma

## Aritmatik Operatörler

Operatör	Açıklama
<code>+, -, /, *</code>	4 işlem
<code>=</code>	Atama işlemi
<code>a, b = c, d</code>	Tek satırda çoklu atama
<code>+=, -=, /=, *=</code>	Kendisiyle işleme sokup kendisine atama
<code>&lt;operatör&gt;=</code>	Kendisiyle işleme sokup kendisine atama
<code>( )</code>	Parantez ile öncelik belirleme

`<operatör>` herhangi bir operatörü temsil eder.

## Ek Aritmatik Operatörler

### Operatör      Açıklama      Örnek      Çıktı

Operatör	Açıklama	Örnek	Çıktı
%	Mod alma işlemi	6 % 2	0
**	Kuvvet alma	6 ** 2	36
//	Kalansız bölümü alma	13 // 2	6

## Karşılaştırma Operatörleri

Operatör	Açıklama	Örnek	Çıktı
>	Büyük	3 > 2	True
<	Küçük	3 < 2	False
==	Eşit	3 == 3	True
!=	Eşit değil	2 != 2	False
>=	Büyük eşit	2 >= 5	False
<=	Küçük eşit	2 <= 2	True

## Mantıksal Operatörler

Operatör	Açıklama	Örnek	Çıktı
and	Ve işlemi	True and False	False
or	Veya işlemi	False or True	True
not	Değili	not False	True

## Bit Düzeyinde Operatörler

Operatör	Açıklama	Örnek
&	Ve	x & y = 0 (0000 0000)
	Veya	x   y = 14 (0000 1110)
~	Değili	~ x = -11 (1111 0101)
^	XOR	x ^ y = 14 (0000 1110)
>>	Sağda kaydırma	x >> 2 = 2 (0000 0010)
<<	Sola kaydırma	x << 2 = 40 (0010 1000)

## Kimlik Belirleme Operatörleri

Operatör	Açıklama	Örnek	Çıktı
----------	----------	-------	-------

Operatör	Açıklama	Örnek	Çıktı
is	Aynı objeye işaret etme	[1, 2, 3] and [1, 2, 3]	False
is not	Farklı objeye işaret etme	1 is not 1	False

Ek değişkenlerde objelerin adresleri farklı olduğunda ilk çıktı False olur.

### Kimlik Belirleme Operatörleri Örneği

```
x1 = 5
y1 = 5
x2 = 'Hello'
y2 = 'Hello'
x3 = [1,2,3]
y3 = [1,2,3]

# Output: False
print(x1 is not y1)

# Output: True
print(x2 is y2)

# Output: False
print(x3 is y3)
```

### Üyelik Operatörleri

Operatör	Açıklama	Örnek	Çıktı
in	Anahtar var	5 in x	False
not in	Anahtar yok	1 not in x	False

x = [1, 2, 3, 4]

### Üyelik Operatörleri Örneği

```
x = 'Hello world'
y = {1:'a',2:'b'}

print('H' in x) # True
print('hello' not in x) # True (h'si büyük değil)
print(1 in y) # True
print('a' in y) # False ('a' bir değerdir anahtar değildir)
```

### If / Else Koşul (Constraints) Yapısı

- `:` ile if / else satırı sonlandırılır
- Tab kadar boşluk atılırsa if scope\*'u içerisinde olur

```
num = float(input("Sayı giriniz: "))
if num >= 0:
    if num == 0:
        print("Sıfır")
    elif num == 1:
        print("Bir")
    else:
        print("Pozitif sayı")
else:
    print("Negatif sayı")
```

### Tek satır (üçlü) If / Else Yapısı

```
fruit = 'Apple'
isApple = True if fruit == 'Apple' else False
```

## Döngüler (Loop)

### For Döngüsü

```
sayilar = [6, 5, 3, 8, 4, 2, 5, 4, 11]
toplam = 0 # Toplam değeri tutacak değişken

for sayı in sayilar: # Liste üzerinde döngü ile ilerleme
    toplam = toplam + sayı

print("Toplam değer:", sum) # Toplam Değer: 48
```

### Değişken içinde For Döngüsü

```
values = [item.value for item in Fruit] # [4, 5, 6]
values = set(item.value for item in Fruit) # {4, 5, 6}
```

### İki Liste Üzerinde Paralel For Döngüsü

```
for num, cheese, color in zip([1,2,3], ['manchego', 'stilton', 'brie'],
                               ['red', 'blue', 'green']):
    print('{} {} {}'.format(num, color, cheese))
```

```
1 red manchego  
2 blue stilton  
3 green brie
```

## While Döngüsü

```
sayac = 0  
  
while sayac < 3:  
    print("Döngü içinde")  
    sayac = sayac + 1  
else:  
    print("Döngü dışında")
```

Döngü içinde  
Döngü içinde  
Döngü içinde  
Döngü dışında

## Range Fonksiyonu

```
print(range(10)) # range(0, 10)  
print(list(range(10))) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  
print(list(range(2, 8))) # [2, 3, 4, 5, 6, 7]  
print(list(range(2, 20, 3))) # [2, 5, 8, 11, 14, 17]
```

## Break / Continue

```
for deger in "string":  
    if deger == "i":  
        break # Döngüyü sonlandırır  
    if deger == "t":  
        continue # Döngüdeki adımı sonlandırır  
    print(deger)  
  
print("Son")
```

s  
r

Son

## Fonksiyonlar

### Dahili Fonksiyon Kullanımları

#### Ekrana Yazma / Print İşlemleri

Fonksiyon	Açıklama	Örnek	Cıktı
<code>print(&lt;string&gt;)</code>	Ekrana yazma	<code>print(f"X: {a}, Y: {2}")</code>	X: 1, Y: 2
<code>print(f'...{&lt;python_kodu&gt;}')</code>	Ekrana formatlı yazma	<code>print(f"X: {a}, Y: {2}")</code>	X: 1, Y: 2

#### String İşlemleri

Cok önemli ve ileride kullanılacak bir konudur. 

Link	Metot	Açıklama	Örnek	Cıktı
	<code>len</code>	Uzunluk	<code>len("yemreak")</code>	7
	<code>format</code>	Formatlama	<code>"X: {}, Y: {}".format(1, 2)</code>	'X: 1, Y: 2'
	<code>%</code>	Operatör ile formatlama	<code>'new(%s %d)' % ('help', 5)</code>	'new(help 5)'
	<code>f</code>	Format string ön eki	<code>f'X: {a}'</code>	'X: 2'
	<code>r</code>	Raw String ön eki	<code>r"C:\Users"</code>	C:\\Users
	<code>"""</code>	Çok satırlı string		
	<code>split</code>	Parçalama	<code>"ye mre ak".split(" ")</code>	['ye', 'mre', 'ak']
Slice	<code>[&lt;başlangıç&gt;:&lt;bitiş&gt;]</code>	Kesme	<code>"yemreak".[2:5], "yemreak".[-3:-1]</code>	"mre", "ea"
	<code>join</code>	Birleştirme	<code>', '.join(['do', 're', 'mi'])</code>	'do,re,mi'
	<code>split &amp; join</code>	Yeniden formatlama	<code>arr.split("\t").join(" ")</code>	'İsim Soyisim Numara

Link	Metot	Açıklama	Örnek	Cıktı
	replace	Metin değiştirme	"yemreak".replace("ak", "")	'yemre'
	strip	Metin düzeltme	' abc '.strip()	'abc'
	lstrip	Metnin solunu düzeltme	' abc '.lstrip()	'abc '
	rstrip	Metnin sağını düzeltme	' abc '.rstrip()	' abc'
	sort	Metni sıralama	['n', 'a', 'i']	['a', 'i', 'n']

Daha fazla bilgi için [buraya](#) ve [buraya](#) bakabilirsin.

## Harici Fonksiyon Kullanımları

- Fonksiyonları kullanmadan önce `import <paket>` ile paketi dahil etmeniz lazım
- Fonksiyonların kullanımı `<paket>.<fonksiyon>` şeklindedir

### Harici String İşlemleri

Paket	Fonksiyon	Açıklama
re	<code>split(&lt;ayırıcı_karakterler&gt;, &lt;string&gt;)</code>	Birden fazla karaktere göre parçalama
<ul style="list-style-type: none"> <li><code>&lt;ayırıcı_karakterler&gt;</code> Metni hangi karakterlere göre böleceğimizi ifade eder           <ul style="list-style-type: none"> <li>Birden fazla olacaksa   ile birbirinden ayrılır</li> <li>Ayrılma sırasında <code>bosluk karakteri</code>nin kullanılması sorun oluşturur</li> <li>Örn: '\n \t \'*</li> </ul> </li> <li><code>&lt;string&gt;</code> Ayırtılacak metin           <ul style="list-style-type: none"> <li>Örn: 'yemreak.com'</li> </ul> </li> </ul>		

### Dizin ve Yol İşlemleri

Paket	Fonksiyon	Açıklama
os	<code>listdir(&lt;yol&gt;)</code>	Yolu verilen dizinin içindekileri döndürür
os	<code>rename(&lt;eski_ad&gt;, &lt;yeni_ad&gt;)</code>	Dosya veya dizin adlandırma
os.path	<code>isfile(&lt;yol&gt;)</code>	Dosya mı kontrolü

Paket	Fonksiyon	Açıklama
os.path	join(<yol>, <dosya_adı>)	Dizinleri birleştirme
os.path	basename(<yol>)	Yolu verilen dosyanın salt adını ve uzantısını bulma
os.path	splittext(<dosya_adı>)	Dosyanın başlığını ve uzantısını döndürür (title, ext)
glob	glob(<yol_şablonu>)	Verilen sorguya veya yola uygun dosya ve dizinleri döndürür
glob	iglob(<yol_şablonu>)	Verilen sorguya veya yola uygun dosya ve dizinleri generator yapısı ile döndürür

- <yol> Path, dosya yolu
  - Örn: C:\Users\Username\help.txt
- <dosya\_adı> Dosyanın uzantısıyla birlikteki adı
  - Örn: help.txt
- <yol\_şablonu> Özel dizin sorguları
  - Örn: \*.txt, ../help

## Fonksiyon Oluşturma

### Fonksiyon İskeleti

```
def function_name(parameters):
    """docstring"""
    statement(s)
```

### Fonksiyon Örneği

```
def greet(name):
    """This function greets to
    the person passed in as
    parameter"""
    print("Hello, " + name + ". Good morning!")
```

### Fonksiyon Dökümantasyonu

```
>>> print(greet.__doc__)
This function greets to
the person passed into the
name paramete
```

### Fonksiyon Varsayılan Parametreler

```

def greet(name, msg = "Good morning!"):
    """
    This function greets to
    the person with the
    provided message.

    If message is not provided,
    it defaults to "Good
    morning!"
    """

    print("Hello", name + ', ' + msg)

greet("Kate") # Varsayılan parametreyi kullanma
greet("Bruce", "How do you do?") # Sıralı parametre verme
greet("Bruce", msg="Naber") # İşaretleyerek parametre verme

```

## Fonksiyonlarda Keyfi Parametreler

```

def greet(*names):
    """
    This function greets all
    the person in the names tuple.

    # names is a tuple with arguments
    for name in names:
        print("Hello", name)

greet("Monica", "Luke", "Steve", "John")

```

\* öne eki ile ile kaç tane isim gelirse o kadar kullanıyoruz.

## Özyineleyen Fonksiyonlar

```

def calc_factorial(x):
    """
    This is a recursive function
    to find the factorial of an integer"""

    if x == 1:
        return 1
    else:
        return (x * calc_factorial(x-1))

num = 4
print("The factorial of", num, "is", calc_factorial(num))

```

```
calc_factorial(4)          # 1st call with 4
4 * calc_factorial(3)      # 2nd call with 3
4 * 3 * calc_factorial(2)  # 3rd call with 2
4 * 3 * 2 * calc_factorial(1) # 4th call with 1
4 * 3 * 2 * 1             # return from 4th call as number=1
4 * 3 * 2                 # return from 3rd call
4 * 6                      # return from 2nd call
24                         # return from 1st call
```

#### Özyineleyen Fonksiyonların Avantajları

- Özyineleyen fonksiyonlar kodun daha temiz ve zarif gözükmesini sağlar
- Karmaşık bir görev alt görevlere ayrılarak rahat çözülebilir
- İç içe döngülere göre daha iyidir

#### Özyineleyen Fonksiyonların Zararları

- Bazı durumlarda anlaşılması zordur
- Uzun tekrarlarda çok fazla vakit ve zaman harcarlar
- Hata ayıklama oldukça zordur

## Lambda Fonksiyonlar

```
double = lambda x: x * 2 # lambda fonksiyon

def double(x): # Fonksiyon
    return x * 2
```

#### Filter ile Lambda Kullanımı

Sadece koşulu sağlayan değerleri döndürür.

```
listem = [1, 5, 4, 6, 8, 11, 3, 12]

cift_listem = list(filter(lambda x: (x%2 == 0) , listem))
print(cift_listem) # [4, 6, 8, 12]
```

#### Map ile Lambda Kullanımı

Her eleman için işlem yapar.

```
listem = [1, 5, 4, 6, 8, 11, 3, 12]

katlanmis_listem = list(map(lambda x: x * 2 , listem))
print(katlanmis_listem) # Output: [2, 10, 8, 12, 16, 22, 6, 24]
```

## Global, Local ve Nonlocal Kavramları

Kavram	Açıklama
global	Tüm modülde geçerli değişkenler
local	Fonksiyonların içerisindeki yerel değişkenler
nonlocal	Modül ile fonksiyon arasında kalan, genellikle iç içe fonksiyonlarda kullanılan değişkenler

## Global, Local ve Nonlocal Kavramlarına Örnek

```
x = 5 # Global

def fonksiyonum():
    x = 3 # Nonlocal

    def degisitirici():
        x = 1 # Local
```

## Global Kullanımına Örnek

```
x = 5
# Yerel x değişkenine 3 değeri atanır, evrensel x değişmez.
def xDegistir():
    x = 3

# Evrensel x değişir
def globalXDegistir():
    global x
    x = 4
```

## Modüller

Her python dosyasına modül denir.

- `import` ile dahil edilirler
- `.` ile içlerindekilere erişilir

## Modül Kullanım Örnekleri

- Python aynı modülü birden fazla kez `import` etmez
  - Kullanıcı birden fazla `import` işlemi yaparsa tepki vermez
- Baştan `import` edilmek istenirse `imp.reload(modül)` şeklinde kullanılır

```
import math # Doğrudan ödülü alma
print("Pi: ", math.pi) # Pi: 3.141592653589793
```

```
import math as m # Modülü özel isimlendirme
print("Pi: ", m.pi) # Pi: 3.141592653589793
```

```
from math import pi # Modül içinden özel değeri alma
print("Pi: ", pi) # Pi: 3.141592653589793
```

```
from math import * # Modül içindeki her şeyi alma
print("Pi: ", pi) # Pi: 3.141592653589793
```

## Python Modül Dosyaları

Modül dosyalarının aranma yerleri:

- Çalışılan dizin
- Ortam değişkenlerindeki `PYTHONPATH` değişkeni değeri
- Kuruluma bağlı varsayılan dizin

## Sistemin Python Modüllerine Bakma

```
>>> import sys
>>> sys.path
['',
'C:\\\\Python33\\\\Lib\\\\idlelib',
'C:\\\\Windows\\\\system32\\\\python33.zip',
'C:\\\\Python33\\\\DLLs',
'C:\\\\Python33\\\\lib',
'C:\\\\Python33',
'C:\\\\Python33\\\\lib\\\\site-packages']
```

## Modül İçinde Tanımlanan İsimleri Alma

```
>>> dir(example)
['__builtins__',
```

```
'__cached__',
 '__doc__',
 '__file__',
 '__initializing__',
 '__loader__',
 '__name__',
 '__package__',
 'add']
```

```
>>> import example
>>> example.__name__
'example'
```

```
>>> a = 1 # Modül değişkenlerine ekleniyor
>>> b = "hello" # Modül değişkenlerine ekleniyor
>>> import math # Modül değişkenlerine ekleniyor
>>> dir()
['__builtins__', '__doc__', '__name__', 'a', 'b', 'math', 'pyscripter']
```

## Paketler (Package)

- Birden fazla modülü içinde barındırır
- . ile modüllere erişilir
  - Tekrar . atılırsa modülün içindekilere erişilir

## Paketten ve Modül Örnekleri

```
import Game.Level.start
```

```
from Game.Level import start
```

```
from Game.Level.start import select_difficulty
```

## Sık Kullanılan Paketler

Modül	Odaklılığı İşlemler
os	İşletim sistemi
time	Zaman

## Modül Odaklı İşlemler

<a href="#">datetime</a>	Tarih
<a href="#">numpy</a>	Matematiksel
<a href="#">openCV</a>	Görüntü
<a href="#">pillow</a>	Resim
Tensorflow	Makine öğrenimi

## Windows Paketleri

### Modül Odaklandığı İşlemler

Modül	Odaklandığı İşlemler	Dökümanlar
<a href="#">pywinauto</a> ☆	Önplanda olmasalar dahi windows uygulamaları (pywin32'i barındırır)	
<a href="#">pygetwindow</a>	Windows pencereleri (basit)	
<a href="#">pywin32</a>	Resmi windows API (pencere dahil)	
<a href="#">pyautogui</a>	Arayüz, fare, klavye ...	

## Görüntü İşleme Paketleri

Modül	Açılıkama	Dökümanlar
<a href="#">pillow</a>	Python resim kütüphanesi	
<a href="#">opencv</a>	Görüntü işleme	
<a href="#">pytesseract</a>	Görüntüdeki yazıyı bulma	

## Giriş Çıkış (I/O) Kontrol Paketleri

### Paket Odaklı İşlemler Dökümanlar

<a href="#">pynput</a>	Fare, klavye vs...	
------------------------	--------------------	---

## Paketler için Harici Bağlantıları

- [Python Kütüphaneleri](#)
- [Argparse Tutorial](#)
- [PyAutoGUI vs Pywinauto](#)

## Sayılar, Sayılar Arası Dönüşüm ve Matematik

### Tabanlı Sayılar

Taban	Ön ek	Örnek	Çıktı
2'lük	<code>0b</code> ya da <code>0B</code>	<code>print(0b1101011)</code>	107

Taban	Ön ek	Örnek	Çıktı
8'lük	0o ya da 00	print(0xFB + 0b10)	253 (251 + 2)
16'lık	0x ya da 0X	print(0o15)	13

## Ondalıklı Sayılar (Decimals / Floats)

```
>>> (1.1 + 2.2) == 3.3
False
>>> 1.1 + 2.2
3.3000000000000003
```

```
import decimal

print(0.1) # 0.1
print(decimal.Decimal(0.1)) #
Decimal('0.100000000000000055511151231257827021181583404541015625')
```

```
from decimal import Decimal as D

print(D('1.1') + D('2.2')) # Decimal('3.3')
print(D('1.2') * D('2.50')) # Decimal('3.000')
```

## Decimal Float Kullanımı ve Farkı

- Decimal daha fazla bellek kaplar
- Finansal işlemlerde decimal tercih edilir

## Kesirli Sayılar (Fractions)

```
import fractions

print(fractions.Fraction(1.5)) # 3/2
print(fractions.Fraction(5)) # 5
print(fractions.Fraction(1,3)) # 1/3
```

```
import fractions

# Floatlar virgülünden sonra da sayı barındırdığından dolayı farklı sonuç
verir
```

```
print(fractions.Fraction(1.1)) # 2476979795053773/2251799813685248
print(fractions.Fraction('1.1')) # 11/10
```

## Kesirli Sayılarla İşlemler

```
from fractions import Fraction as F

print(F(1,3) + F(1,3)) # 2/3
print(1 / F(5,6)) # 6/5
print(F(-3,10) > 0) # False
print(F(-3,10) < 0) # True
```

## Matematik İşlemleri

```
import math

print(math.pi) # 3.141592653589793
print(math.cos(math.pi)) # -1.0
print(math.exp(10)) # 22026.465794806718
print(math.log10(1000)) # .0
print(math.sinh(1)) # 1.1752011936438014
print(math.factorial(6)) # 720
```

## Matematikte Rastgelelik

```
import random

x = ['a', 'b', 'c', 'd', 'e']

print(random.randrange(10,20)) # Rastgele 10, 20 arasında sayı yazdırma
print(random.choice(x)) # Rastgele seçim yapma
random.shuffle(x) # Karıştırma
print(x) # Karışım sonucunu yazma
print(random.random()) # Rastgele eleman yazma
```

## Class

### Class Anahtar Kelimeleri

Anhatar	Açıklama	Örnek
<code>self</code>	Diğer dillerdeki <code>this</code> anlamına gelir	<a href="#">Basit Class Örneği</a>
<code>__init__</code>	Constructer fonksiyonudur	<a href="#">Basit Class Örneği</a>

Anhatar	Açıklama	Örnek
<code>def function(param):</code>	Fonksiyon tanımlama	Metodlu Class Örneği

## Basit Class Örneği

```
class Person:  
    def __init__(self, name, age):  
        self.name = name  
        self.age = age  
  
p1 = Person("John", 36)  
  
print(p1.name)  
print(p1.age)
```

John  
36

## Metodlu Class Örneği

```
class Person:  
    def __init__(self, name, age):  
        self.name = name  
        self.age = age  
  
    def myfunc(self):  
        print("Hello my name is " + self.name)  
  
p1 = Person("John", 36)  
p1.myfunc()
```

Hello my name is John

## Objenin Özellikini Silme

```
del p1.age
```

## Class Silme

```
del p1
```

## Scopes and Namespaces

```
def scope_test():
    def do_local():
        spam = "local spam"

    def do_nonlocal():
        nonlocal spam
        spam = "nonlocal spam"

    def do_global():
        global spam
        spam = "global spam"

    spam = "test spam"
    do_local()
    print("After local assignment:", spam)
    do_nonlocal()
    print("After nonlocal assignment:", spam)
    do_global()
    print("After global assignment:", spam)

scope_test()
print("In global scope:", spam)
```

```
After local assignment: test spam
After nonlocal assignment: nonlocal spam
After global assignment: nonlocal spam
In global scope: global spa
```

## Enumeration

Resmi dökümantasyon için [buraya](#) bakabilirsın.

- Sıralı ve sabit veriler oluşturmak için kullanılır
- `from enum import Enum` ile projeye dahil edilir

## Basit Kullanım

```
from enum import Enum

class Color(Enum):
    RED = 1
```

```

GREEN = 2
BLUE = 3

# Erişim şekli
Color.RED # 1
Color.RED.name # RED
type(Color.RED) # <enum 'Color'>
Color(1) # <Color.RED: 1>
Color(3) # <Color.BLUE: 3>
isinstance(Color.GREEN, Color) # True

# Obje olarka kullanımı
color = Color.RED
color.value # 1
color.name # RED

```

## Enum Özellikleri

Aynı özelliklere sahip objeler oluşturulamaz

```

# Oluşturulmaz!
class Shape(Enum):
    SQUARE = 2
    SQUARE = 3

# Oluşturabilir
class Shape(Enum):
    SQUARE = 2
    DIAMOND = 1
    CIRCLE = 3
    ALIAS_FOR_SQUARE = 2

Shape.SQUARE # <Shape.SQUARE: 2>
Shape.ALIAS_FOR_SQUARE # <Shape.SQUARE: 2>
Shape(2) # <Shape.SQUARE: 2>

```

## Benzersin Enum Tanımlaması

@unique etiketi ile tanımlama yapılır

```

from enum import Enum, unique
@unique
class Mistake(Enum):
    ONE = 1
    TWO = 2
    THREE = 3
    FOUR = 3

```

```
# Traceback (most recent call last):
# ValueError: duplicate values found in <enum 'Mistake'>: FOUR -> THREE
```

## Komut İsteminden Python (CLI)

- Komut isteminden gelen argümanları **argparse** adlı modül ile yönetmekteyiz
- Otomatik kod tamamlaması için **buraya** bakmada fayda var.
- Kullanıcı cmd üzerinden **python <dosya\_adi> <argümanlar>** gibi komutlarla programımızı kullanabilir

### Argparse Modülü Detayları

- Argüman ekleme işlemi **parser = argparse.ArgumentParser(...)** ile yapılmaktadır.
- Parametrelerin kullanımı **argparse.ArgumentParser(description='yok')** şeklindedir.

Parametre	Açıklama
<b>description</b>	Uygulama ile alakalı açıklama metnidir
<b>Argüman Ekleme</b>	
• Argüman ekleme işlemi <b>parser.add_argument(...)</b> ile yapılmaktadır.	
Parametre	Açıklama
1. parametre	Kısa kullanım komutunu içerir
2. Parametre	Orjinal kullanım komutunu içerir
<b>help</b>	<b>-h</b> yazıldığında çıkacak olan yardım metni
<b>action</b>	Davranışı belirler
<b>type</b>	Tip bilgisini içerir (int, string ...)
<b>default</b>	Varsayılan değer

### Argüman Action Özelliği

Parametre	Açıklama
<b>'store_true'</b>	Flag* değeri olur ve komutta içerilirse <b>True</b> değeri alır (-h gibi)
<b>count</b>	Kaç kere yazıldığı bilgisini tutar (-vvv için 3)

```
import argparse
parser = argparse.ArgumentParser()
parser.add_argument("--verbose", help="increase output verbosity",
                    action="store_true")
args = parser.parse_args()
```

```
if args.verbose:  
    print("verbosity turned on")
```

### Çıktısı:

```
$ python3 prog.py --verbose  
verbosity turned on  
  
$ python3 prog.py --verbose 1  
usage: prog.py [-h] [--verbose]  
prog.py: error: unrecognized arguments: 1  
  
$ python3 prog.py --help  
usage: prog.py [-h] [--verbose]  
  
optional arguments:  
  -h, --help    show this help message and exit  
  --verbose    increase output verbosity
```

### Örnek CLI Kodu

```
import argparse  
  
def main():  
    # Initiate argument parser  
    parser = argparse.ArgumentParser(  
        description="Sample TensorFlow XML-to-CSV converter")  
    parser.add_argument("-i",  
                      "--inputDir",  
                      help="Path to the folder where the input .xml files  
are stored",  
                      type=str)  
    parser.add_argument("-o",  
                      "--outputFile",  
                      help="Name of output .csv file (including path)",  
                      type=str)  
    args = parser.parse_args()  
  
    if args.inputDir is None:  
        args.inputDir = os.getcwd()  
  
    if args.outputFile is None:  
        args.outputFile = args.inputDir + "/labels.csv"  
  
    assert (os.path.isdir(args.inputDir))  
  
    xml_df = xml_to_csv(args.inputDir)  
    xml_df.to_csv(  
        args.outputFile, index=None)
```

```
print('Successfully converted xml to csv.')

if __name__ == '__main__':
    main()
```

## Python Görsel Programlama (GUI)

Python görsel programlama **PyQt API**'ı ile yapılmaktadır.

- Bu yazıyı oluştururken yararlandığım kaynak için [buraya](#) bakabilirsın.
- Türkçe eğitim serisi için [buraya](#) bakabilirsın.

### PyQt5 Kurulumu

GUI için *cross development* desteği olan **pyqt** kullanılmaktadır.

- `pip install pyqt5`
- `conda install pyqt`

 *Cross development:* Birden çok işletim sisteminde çalışabilen yazılım geliştirmesi: PC, Mac, linux vs..

### Basit GUI Yapımı

GUI oluşturma yardımcı olan **QTDesigner** oldukça faydalı olacaktır. ( Çek-bırak mantığında çalışır.)

```
from PyQt5.QtWidgets import QApplication, QLabel

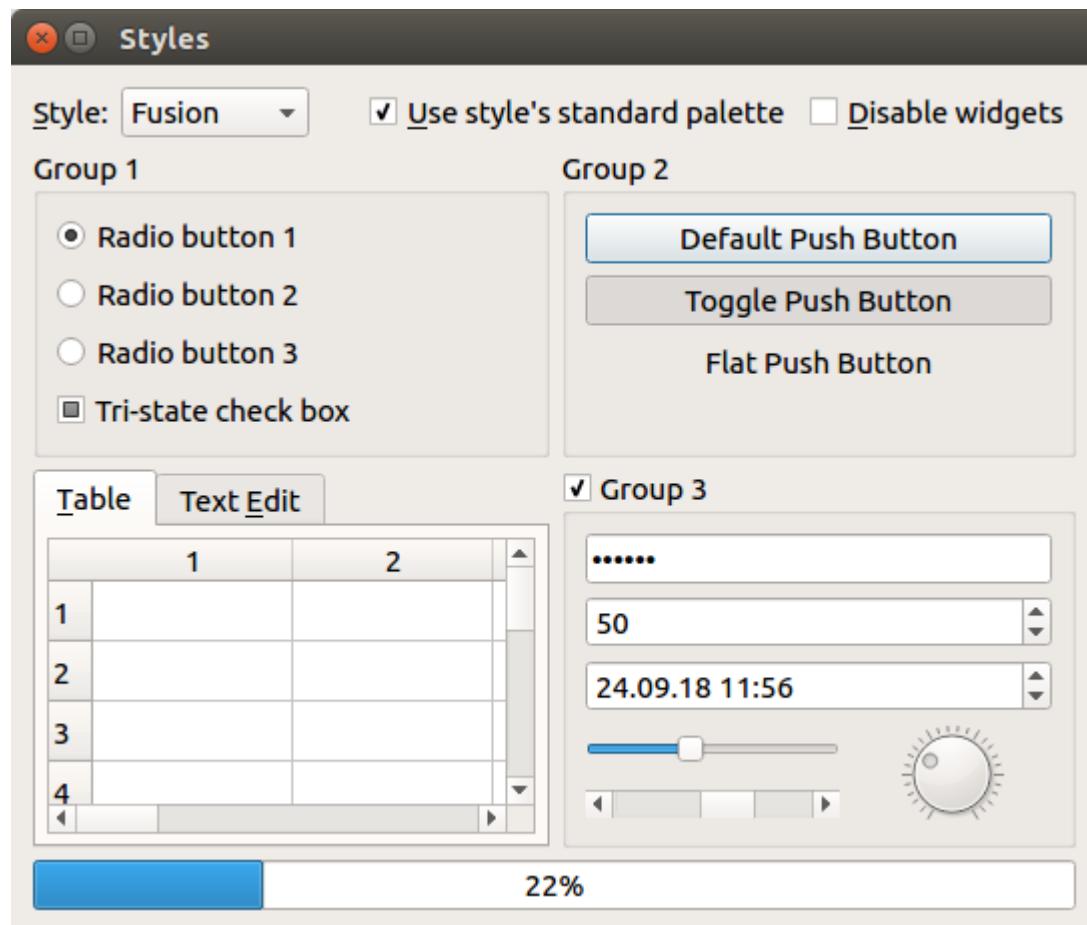
# Uygulamayı tanımlama
# - [] objesi içine aktarılacak argümanları ifade eder
app = QApplication([])

# Pencerenin içine yazı yazma ve görünür kıılma
label = QLabel('~ YEmreAk')
label.show()

# Uygulamayı kullanıcı kapatana kadar çalıştırma (exec olursa arkaplanda da
# çalışır)
app.exec_()
```

## PyQt Widgets

PyQt deki her bir obje widget olarak adlandırılmasa



Yukarıdan-aşağı, soldan-sağ olmak üzere sırayla:

- QLabel
- QComboBox
- QCheckBox
- QRadioButton
- QPushButton
- QTableWidget
- QLineEdit
- QSlider
- QProgressBar

Ekran görüntüsündeki kodu [buraya](#) tıklayarak indirebilirsin.

## PyInstaller ile Executable Dosya Oluşturma

Video açıklaması için [buraya](#) bakabilirsin.

## İleri Seviye Python

### Assertion (Kural Koyma)

Boolean değeri sağlanmazsa hata verir ve programı kapatır.

```
assertion(<bool>, < açıklama>)
```

- <bool> Kontrol değişkeni
  - Örn:  $0 > 5$
- <açıklama> Hatanın neden verildiğine dair metin
  - Örn: *Küçük bir değer girildi*

## Assertion Örnekleri

```
def KelvinToFahrenheit(Temperature):
    assert (Temperature >= 0), "Colder than absolute zero!"
    return ((Temperature-273)*1.8)+32

print (int(KelvinToFahrenheit(505.78)))
print (KelvinToFahrenheit(-5))
```

```
451
Traceback (most recent call last):
  File "test.py", line 9, in <module>
    print KelvinToFahrenheit(-5)
  File "test.py", line 4, in KelvinToFahrenheit
    assert (Temperature >= 0), "Colder than absolute zero!"
AssertionError: Colder than absolute zero!
```

## Try / Except Yapısı

Olası hatalarda programın kapanmasını engelleyerek hata kontrolü sağlar.

```
try:
    a = float("Ben sayı değilim")
except ValueError:
    print("Bu sayı değil")
```

## Dosya İşlemleri

Python üzerinde dosya işlemleri oldukça kolaydır ve **context manager** ile halledilir.

```
with open(<dosya_ismi>, <erişim_modu>, encoding=<kodlama>) as file:
    # İşlemler
    pass
```

- <dosya\_ismi> Dosya yolu veya ism
  - Örn: "text.txt"
- <erişim\_modu> Okuma, yazma veya ekleme
  - Örn: 'a', 'w', 'r', 'r+' ...

- <kodlama> Dosya kodlama formatı
  - Örn: 'utf-8'

## Dosya Okuma

```
file_str = ""
with open("README.md", "r", encoding="utf-8") as file:
    file_str = "\n".join(file.readlines())
```

```
file_str = ""
with open("README.md", "r", encoding="utf-8") as file:
    for line in file:
        file_str += line
```

```
with open(xml_path) as fp:
    for row, line in enumerate(fp):
        pass
```

```
with open("README.md", "r", encoding="utf-8") as file:
    lines = list(file) # Tüm satırları liste olarak döndürür
    line = file.readline() # Tek bir satırı string olarak döndürür
    lines = file.readlines() # Tüm satırları liste olarak döndürür
```

## Thread

Thread modülü ile satır satır ilerleyen kod yerine karma ilerleyen kodlar yazılabilir.

- `threading` modülü kullanılır
- Eş zamanlı işlemler için `multiprocessing` tercih edilir

Class	Açıklama
Thread	Sırasız olarak bir fonksiyonu çalıştırma
Timer	Belirli saniyelerde fonksiyonu çalıştırma
Scheduler	Bir plana göre fonksiyonu çalıştırma

## Basit Thread Yapısı

```
from time import sleep
from threading import Thread

def tekrarla(ne, bekleme):
    while True:
        print ne
        sleep(bekleme)

if __name__ == '__main__':
    dum = Thread(target = tekrarla, args = ("dum",1))
    tis = Thread(target = tekrarla, args = ("tis",0.5))
    ah = Thread(target = tekrarla, args = ("ah",3))

    dum.start()
    tis.start()
    ah.start()
```

```
dum
tis
ah

tis
dumtis

tis
dumtis

tis
ah
tisdum
```

## Zamanlayıcı Yapısı (Timer)

```
import threading

def run_check():
    print("Fonksiyon çalıştı.")
    threading.Timer(5.0, run_check).start()

run_check()
```

## Bir Plana göre Fonksiyon Çalıştırma

```
import sched, time
s = sched.scheduler(time.time, time.sleep)
def do_something(sc):
    print "Doing stuff..."
    # do your stuff
    s.enter(60, 1, do_something, (sc,))

s.enter(60, 1, do_something, (s,))
s.run()
```

## Paralel İşlemler (Multiprocessing)

Python'da eş zamanlı işler `thread` ile yapılamaz

Kaynak için [buraya](#) bakabilirsın.

### Multiprocessing Örneği

```
from multiprocessing import Process

def func1():
    print('func1: starting')
    for i in range(10000000):
        pass
    print('func1: finishing')

def func2():
    print ('func2: starting')
    for i in range(10000000):
        pass
    print ('func2: finishing')

if __name__ == '__main__':
    p1 = Process(target=func1)
    p1.start()
    p2 = Process(target=func2)
    p2.start()
    p1.join() # Threadi çalışma gecikmesini engellemek için
    p2.join()

# func1: starting
# func2: starting
# func2: finishing
# func1: finishing
```

## Kod Parçaları (Code Snippet)

## PYTHONPATH Ayarlama

```
# Tek başına çalışmak isterse
if __name__ == "__main__":
    import os
    import sys
    sys.path.append(os.getcwd())
```

## Ekran Görünüsünü Alma ve Kaydetme

```
from PIL import ImageGrab as ig

import numpy as np
import time
import cv2

# Hata ayıklama ve bilgilendirme notlarını aktif eder
DEBUG = True

# Çıktı kaydını aktif etme
KEEP = False

# Yakalanacak ekranın konum bilgileri (x0, y0, x1, y1)
CAPTURE_AREA = (80, 101, 1111, 923)

# Yakalanan ekranın gösterilme boyutu (Varsayılan için 0 yapın)
WIDTH = 0
HEIGHT = 0

# FPS sayacını tanımlama
if DEBUG:
    frame_count = 0
    last_time = time.time()

out = cv2.VideoWriter(
    'output.avi',
    cv2.VideoWriter_fourcc(*'XVID'),
    5.0,
    (CAPTURE_AREA[2] - CAPTURE_AREA[0], CAPTURE_AREA[3] - CAPTURE_AREA[1]))
) if KEEP else None

while True:
    screen = ig.grab(bbox=CAPTURE_AREA)
    screen_np = np.array(screen)

    # BGR tipindeki görüntüyü RGB yapıyoruz
    screen_np_RGB = cv2.cvtColor(screen_np, cv2.COLOR_BGR2RGB)

    # Gösterilecek ekranın boyutunu ayarlama
    screen_width = WIDTH if WIDTH != 0 else CAPTURE_AREA[2] -
```

```

CAPTURE_AREA[0]
    screen_height = HEIGHT if WIDTH != 0 else CAPTURE_AREA[3] -
CAPTURE_AREA[1]

# Kaydedilen ekranı uygun boyutta görüntüleme
cv2.imshow(
    'Ekran görüntüsü',
    cv2.resize(
        screen_np_RGB,
        (
            screen_width,
            screen_height
        )
    )
)

# Dosyaya yazma
out.write(screen_np_RGB) if KEEP else None

# 'q' tuşuna basıldığında çıkış işlemi
if cv2.waitKey(25) & 0xFF == ord('q'):
    out.release() if KEEP else None
    cv2.destroyAllWindows()
    break

# FPS bilgilerini hesaplama ve ekrana basma
if DEBUG:
    frame_count += 1
    if time.time() - last_time >= 1:
        print('FPS: {}'.format(frame_count))
        frame_count = 0
        last_time = time.time()

```

## Kısayol ile Ekran Alanı Seçme

```

def draw_dimension(hotkey="ctrl_l") -> tuple:
    """Ekrandan seçilen alanın koordinatlarını verir

    Keyword Arguments:
        hotkey {string} -- Klavye kısayolu (default: {None})

    Returns:
        tuple -- Seçilen alanın koordinatları `(x0, y0, x1, y1)`
    """

    # Farenin başlangıç ve bitiş konumları
    start_position, end_position = (0, 0)

    def listen_keyboard():
        with keyboard.Listener(on_press=on_press, on_release=on_release) as

```

```

keyboard_listener:
    keyboard_listener.join()

def on_press(key):
    # Başlangıç koordinatlarını oluşturma
    if key == keyboard.Key[hotkey]:
        nonlocal start_position
        start_position = mouse.Controller().position

def on_release(key):
    # Bitiş koordinatlarını başlangıça ekleme
    if key == keyboard.Key[hotkey]:
        nonlocal end_position
        end_position = mouse.Controller().position

    # Dinleyiciyi kapatma
    return False

print(
    f"Seçmek istediğiniz alanın başlangıç noktasına farenizi getirin ve {hotkey} tuşuna basılı tutarak farenizi alanın bitiş noktasına götürün.")

listen_keyboard()
return start_position + end_position

print(draw_dimension())

```

## Url Encode İşlemi

- TODO

## Google Colabrotory Üzerinden Python

Google Colabrotory IPython modülünü kullanmaktadır.

Detaylı bilgileri içeren google colabrotory notum için [buraya](#) tıklayabilirsin.

## IPython Operatorleri

### Operator      Açıklama

!	Bash komutları ön eki
%	Bash dizini ön eki (?)

## Python Değişkenlerinin Bash Üzerinde Kullanımı

Operatör	Açıklama	Örnek	Çıktı
\$<değişken>	Tek değişkenler için kullanılır	!echo \$filename	test
{<python_kodu>}	Python kodu için kullanılır	{"{} .test".format(1)}	1.test

## Ortam Değişkenleri

- **PYTHONPATH** Python modülleri yollarını barındıran değişkendir.
  - `import` ile verilen yollardaki dizinlerden script dahil edilir

Windows için cmd ortam değişkeni ayarlama yapısı `set name=value;value` şeklindedir.

## PyCharm Uygulamasında Ortam Değişkeni Tanımlama

- Üst sekmeden **Run** kısmına gelin
- **Edit Configuration** yazısına tıklayın
- Yapılandırma ayarınızı seçin
  - Yoksa `+` ile yeni bir tane oluşturun
- **Environment Variables** kısmında en sağdaki dosya simgesine tıklayın
- `+` ile yeni ortam değişkeninizi ekletin

Windows için cmd ortam değişkeni ayarlama yapısı `set name=value;value` şeklindedir.

## Yapılacaklar

- Thread ve Timer eklenecek
  - [Link1](#), [Link2](#), [Link3](#), [Link4](#)
- Alttaiki yapı eklenecek
  - `t2 = Thread(target=time.sleep(3))`
  - `{}` ile fonksiyon
  - return olursa değeri çıkarır

## Harici Kaynaklar

- [String işlemleri](#)
- [Learn Python Programming](#)
- [Python Türkçe Başlangıç](#)
- [Should I use underscores or camel case for Python?](#)
- [Top 10 Python Libs 2017](#)
- [Tensorflow Object Detection API](#)
- [Dosyadak Belli Satırı Değiştirme](#)
- [How do I list all files of a directory](#)
- [Replace single backslash with double backslash](#)
- [What does `if \_\_name\_\_ == '\_\_main\_\_': do?`](#)
- [Gitignore yapılandırması](#)
- [Ekranın Video Görüntüsünü Yakalama](#)
- [Putting a simple if-then-else statement on one line](#)
- [Can python get the screen shot of a specific window?](#)
- [Get window position & size with python](#)
- [Python inactive screen capture](#)
- [Computer Screen Recording using Python & OpenCV](#)
- [How can I code OpenCV to use GPU using Python?](#)
- [Google Keep to Text](#)
- [Python ile QuickDraw Projesi](#)

- [7 Top Python GUI Frameworks](#)
- [Python init.py Dosyası](#)

## Lisans ve Teferruatlar

Bu yazı **MIT** lisanslıdır. Lisanslar hakkında bilgi almak için [buraya](#) bakmando fayda var.

- [Github](#)
- [Website](#)
- [LinkedIn](#)

Yardım veya destek için [iletişime](#) geçebilrsiniz 😊

~ Yunus Emre Ak