Python'da Dosya İşlemleri

Python programlama dili, dosya işlemleri konusunda güçlü ve kullanışlı araçlar sunar. Dosya okuma, yazma, silme gibi işlemleri bilmek, Python geliştiricilerinin çoğu zaman karşılaştığı temel gereksinimlerdendir. Bu makalede, Python'da dosya işlemleri ile ilgili bilmeniz gereken her şeyi adım adım öğreneceksiniz.

Lütfen başlamadan önce sözlükçe'yi okuyun, bilmediğiniz terimler burada yazar.

Dosya açma

Python'da bir dosya açmak için `open(dosya_yolu, mod, encoding?)` fonksiyonu kullanılır. Bu fonksiyon, iki ana ve 1 opsiyonel (`?` ile belirttik) parametre alır:

- **dosya_yolu**: Açılacak olan dosyanın yolu, bulunan dosyaya olan konumu (örn. ./xyz.txt, ./ içinde bulunduğumuz dosyanın dizininis ifade eder) ya da tam konumu (örn. C:/Users/Py/xyz.txt gibi tam bir yol) kullanılır.
- **mod**: Dosyanın nasıl açılacağını ifade eder, modları buradan okuyabilirsiniz.

Dosya Modları:

Temel Modlar

- `r` : **(Read - Oku)** Default mod. Dosyayı sadece okumak için açar. Dosya yoksa hata verir.
- `w` : **(Write - Yaz)** Dosyayı sadece yazmak için açar. Dosya varsa içeriğini siler. Yoksa yeni oluşturur.
- `a` : **(Append - Ekleyerek Yaz)** Dosyanın sonuna yazmak için açar. Dosya yoksa oluşturur. İçeriği silmez.
- `x` : **(Exclusive Create - Yeni Dosya Oluştur)** Yeni bir dosya oluşturur. Eğer dosya varsa hata verir.

Okuma ve Yazma İçin Kullanılan Modlar

- `r+` : **(Read & Write - Oku & Yaz)** Mevcut dosyayı okumak ve yazmak için açar. İçeriği silmez. Dosya yoksa hata verir.
- `w+` : **(Write & Read - Yaz & Oku)** Mevcut dosyayı okumak ve yazmak için açar. İçeriği siler. Dosya yoksa oluşturur.
- `a+` : **(Append & Read - Ekleyerek Yaz & Oku)** Mevcut dosyanın sonuna ekleme yapmak için açar. Okuma da yapılabilir. Dosya yoksa oluşturur.
- `x+` : **(Exclusive Create & Read - Yeni Dosya & Oku)** Yeni bir dosya oluşturur ve okuma/yazma yapabilir. Dosya varsa hata verir.

İkili (Binary) Modlar

Bunlar metin yerine ikili (binary) veri işlemek için kullanılır. Özellikle resim, ses, video ve diğer binary dosyalarla çalışırken gerekir.

- `rb` : **(Read Binary - İkili Oku)** Dosyayı binary (ikili) modda okumak için açar. Dosya yoksa hata verir.
- `wb` : **(Write Binary - İkili Yaz)** Dosyayı binary modda yazmak için açar. Dosya varsa içeriğini siler. Dosya yoksa oluşturur.
- `ab` : **(Append Binary - İkili Ekleyerek Yaz)** Dosyanın sonuna binary modda veri eklemek için açar. Dosya yoksa oluşturur.
- `xb` : **(Exclusive Create Binary - Yeni İkili Dosya Oluştur)** Yeni bir binary dosya oluşturur. Dosya varsa hata verir.

Dosya kapama

Basit ama önemli bir ayrıntıdır, dosyalar kapatılmazsa RAM'i tüketmeye devam eder. Bunun 2 yolu var:

- `file.close()` : ile açıldıktan ve iş bittikten sonra manuel olarak kapatılabilir.
- **Önerilen!** Dosyayı açarken `with` anahtar kelimesi ile otomatik bir kapanma sağlar. Örneğin:

```
with open('dizin', 'mod') as file: # İstenen bütün işlemler otomatik oluşturulan 'file' değişkeniyle yapılabilir ve iş bittiğinde otomatik olarak kapanır.
```

Python'da bir değeri varsayılandan farklı bir isim ile referans almak için `as` anahtar kelimesi kullanılır.

Encoding nedir?

Açılan dosyanın şifrelenme biçimidir. Eğer dosyanızda Türkçe karakter varsa soruna yol açar çünkü default encoding İngilizceyi destekler. Bunu söyle değiştirebiliriz:

```
# ...
open(yol, mod, encoding="utf-8")
# ...
```

Temel İşlemler: Dosya okuma, yazma, oluşturma

Dikkat! Dosyayı açıp kullandıktan sonra onu kapatmak önemlidir. Eğer kapatmazsanız kullandıktan sonra RAM'i tüketmeye devam eder ve birden fazla dosya kapatılmadan kullanıldığında büyük performans kaybına yol açar. Nasıl yapıldığını burada detaylıca anlattık. Olası hatalar için de sonuna kadar okuyunuz.

Dosya okuma:

Dosyamızı şu şekilde açalım:

```
file = open('dosya.txt', 'r') # İçinde bulunduğumuz dizindeki dosya.txt'yi (r)eading modunda açtık.
```

Şimdi dosyamızın içeriğini `str` veri türünde bir değişkene atayalım:

```
content = file.read()
```

Biraz daha derine inersek 2 metodumuz daha var:

- `.readline()` : Her kullanıldığında sonraki satırı okur.
- `.readlines()` : Bütün satırları bir liste (`list`) olarak döndürür.

Karşılaştırma yapalım! Dosya içeriği şu olsun:

```
Merhaba!
Na'ber?
Nasılsın?
```

ilk `readline()` yaptığımızda `Merhaba!` yazısını okuruz, ikincisinde `Na'ber?` ve böyle sonsuza kadar gider. **Eğer dosyanın sonuna gelirseniz `boş string` ("") döner.**

Ama `readlines()` dersek şöyle bir veri döner:

```
["Merhaba!\n", "Na'ber?\n", "Nasılsın?\n"]
```

`read()` ise bütün metni `str` olarak döndürür.

Dosya yazma:

Dosyamızı yazma modunda açalım ama bu sefer iki seçeneğimiz var:

- `w` ile içeriği silip baştan yazma.
- `a` ile dosyanın sonuna ekleme yapma.

İlk olarak `w` modu ile deneyelim:

```
file = open('dosya.txt', 'w') # dosya.txt'yi (w)rite modunda açtık. Dosya varsa içeriği tamamen silinir. Eğer dosya bulunamaz ise oluşturulacak.

file.write("Hello, World!\n") # İçerik baştan yazıldı
```

Şimdi de `a` modu ile deneyelim:

```
file = open('dosya.txt', 'a') # dosya.txt'yi (a)ppend modunda açtık.

file.write("Hello, World!\n") # Dosyanın sonuna eklendi.
```

Dosya oluşturma:

Şimdi de dosya oluşturuyoruz:

```
file = open('dosya.txt', 'x') # İçinde bulunduğumuz dizinde dosya.txt'yi oluşturduk.
```

Diğer bütün modlar için kullanımlar benzerdir.

Olası hatalar:

- **FileNotFoundError**: Dosyanın belirtilen dizinde olmadığını söyler.
- **PermissionError**: Dosyayı açmak için Python betiğinin yeterli yetkiye sahip olmadığını söyler.
- **FileExistsError**: Dosya oluşturulurken bu dosyanın zaten mevcut olduğunu söyler.
- **UnicodeDecodeError**: Dosya ile işlem yaparken Türkçe veya desteklenmeyen bir dil kullanıldığını söyler. Encoding'e göz atın.

Sözlükçe

- **default**: Varsayılan demektir, bir şeyi eklemezseniz bile böyle kabul edilir.
- **UTF-8**: Unicode karakterlerini temsil etmek için kullanılan ve geriye dönük uyumluluğa sahip, değişken uzunluklu bir karakter kodlama standardıdır.
- **parametre**: Bir fonksiyonda ya da metod içinde, her kullanımda değişen ve kullanıcının seçtiği değerlerdir.
- **Binary**: 0 ve 1'den oluşan bilgisayar kodlama standardı, Python'da da desteklenir.

> *Mehmet Kaan Aksoy, 1.03.2025, repo: @mkaksoy/python-dosya-islemleri*