FILES AND EXCEPTIONS (DOSYALAR VE HATALAR)

- Variables, lists, tuples, dictionaries, sets, arrays, pandas Series and pandas DataFrames yalnızca geçici veri depolama sunar.
- Yerel değişkenler kapsam dışına çıktığında veya program sona erdiğinde kaybolur.
- Bilgisayarlar, dosyaları katı hal sürücüleri, sabit diskler ve da fazlasını ikincil depolama aygıtlarında depolar.
- Metin dosyalarını birkaç popüler formatta ele alıyoruz:
    - plaintext (düz metin)
    - JSON (JavaScript Object Notation) (JavaScript Nesnesi Gösterimi)
    - CSV (comma-separated values) (Virgülle Ayrılmış Değerler)
- İkincil depolamaya kaydetmek ve bunları İnternet üzerinden iletmek üzere nesneleri seri hale getirmek ve seri durumundan çıkarmak için JSON kullanılır.
- CSV verilerinin yüklenmesi ve işlenmesi için kullanılanlar:
    - Python Standart kütüphanesinin csv modülü
    - pandas kütüphanesi
- Python Standart kütüphanesinin pickle modülü ile verileri seri hale getirme ve serisini kaldırma
- Exception Handling (Hata İşleme)
    - Hata, yürütme zamanı sorununu (execution time problem) gösterir.
    - ZeroDivisionError, NameError, ValueError, StatisticsError, TypeError, IndexError, KeyError and RuntimeError.
    - Bu hatalara yönelik try except blokları tanımlanabilir.
    - Kendi hatalarımızı da tanımlayabiliyoruz.
- Bilgisayarda depolanan bir dosyaya erişmeye çalıştığımız anda onu ele geçirmiş oluyoruz. Bunlar aynı anda yalnızca bir program tarafından kullanılabilir. Biz diğer kaynaklar kullansın diye serbest bırakana kadar başka bir kullanıcı tarafından kullanılamıyor.
- Bu problemi çözmek için with statement geliştirilmiştir.

Bazı Hata Çeşitleri

- **ZeroDivisionError:**
    - Sayının sıfıra bölünmesi durumunda ortaya çıkar.
- **NameError**
    - Python'da daha önce tanımlamadığımız değişkenleri kullanmaya çalışırsak bu error ile karşılaşırız.

- **ValueError:**
  - Eğer kullanıcı sayı değerli veri yerine harf değerli bir veri girerse oluşur.
- **StatisticsError:**
  - İstatistik Hatası
- **TypeError:**
  - Birbirinden farklı veri tipleri ile aritmetik işlemler yapmak istediğimiz durumlarda karşılaşılır.
- **IndexError:**
  - Dizin Hatası
- **SyntaxError:**
  - Syntax kurallarına uymayan durumlarda bu hata ile karşılaşılır:
    - Yanlış veya eksik noktalama işaretleri ya da parantez kullanımı
    - Geçersiz değişken veya fonksiyon isimleri
- **KeyError:**
  - Anahtar Hatası
- **RuntimeError:**
  - Çalışma Hatası

FILES (DOSYALAR)

- Karakter dizisi olarak bir metin dosyası olabilir.
- Binary dosya (resim, video ve daha fazlası) bir bayt dizisi olabilir.
- Bir metin dosyasındaki ilk karakter ve ikili dosyadaki bayt 0 konumunda bulunur.
  - n karakter veya baytlık bir dosyada en yüksek konum numarası n – 1'dir.
- Açılan her dosya için Python, dosyayla etkileşim kurmak için kullanılan bir dosya nesnesi oluşturur.

End of File (Dosya Sonu İşaretçisi)

- Her işletim sistemi, bir dosyanın sonunu belirtmek için bir mekanizma sağlar.
- Bazıları dosya sonu işaretçisi kullanır.
- Diğerleri ise dosyadaki toplam karakterlerin veya baytların sayısını tutar.

Standard File Objects (Standart Dosya Nesneleri)

- Python bir programı yürütmeye başladığında, üç standart dosya nesnesi oluşturur:
  - sys.stdin (the standard input file object)(standart girdi dosyası nesnesi)
  - sys.stdout (the standard output file object) (standart çıktı dosyası nesnesi)
  - sys.stderr (the standard error file object) (standart hata dosyası nesnesi)
- Bunlar dosya nesneleri olarak kabul edilse de, varsayılan olarak dosyalardan okumaz veya dosyalara yazmazlar.
- Girdi işlevi, klavyeden kullanıcı girdisi almak için dolaylı olarak sys.stdin'i kullanır.
- print işlevi, dolaylı olarak komut satırında görünen sys.stdout'a çıktı verir.
- Python dolaylı olarak program hatalarını ve geri izlemeleri komut satırında da görünen sys.stderr dosyasına verir.
- Bu nesnelere erişmek istendiğinde import sys ile sys modülünü içe aktarmak gerekir.

Text-File Processing (Metin Dosyası İşleme)

- Birçok kaydı alıp depolamak
- Mevcut kayıtları alıp onlarla işlemler yapmak

Python File Open (Python Dosya Açma)

- **"r" -- Read-Default value(Varsayılan Okuma değeri)** Okumak için bir dosya açar, dosyalar yoksa hata verir.
- **"a" -- Append(Ekleme)** Eklemek için bir dosya açar, yoksa dosyayı oluşturur.
- **"w" -- Write(Yazma)** Bir dosyayı yazmak için açar, yoksa dosyayı oluşturur.
- **"x" -- Create(Oluşturma)** Belirtilen dosyayı oluşturur, dosya varsa bir hata döndürür.
- **"t" -- Text-Default value(Metin-Varsayılan değer)** Metin modu
- **"b" -- Binary-Binary mode** (resim vb.)

text dosyası "r" ile açılımı

```
1 f = open("accounts.txt", "r")
2 print(f.read()) # text dosyasının içini yazdırır
```

```
100 Jones 24.98
200 Doe 345.67
300 White 0.00
400 Stone -42.16
500 Rich 224.62
```

```
1 f = open("accounts.txt", "r")
2 print(f.read(5)) # text dosyasının ilk 5 karakterini yazdırır
```

```
100 J
```

```
1 f = open("accounts.txt", "r")
2 print(f.readline()) # text dosyasının bir satırını yazdırır
```

```
100 Jones 24.98
```

```
1 f = open("accounts.txt", "r")
2 print(f.readline())
3 print(f.readline()) # text dosyasının ilk iki satırını yazdırır.
```

```
100 Jones 24.98

200 Doe 345.67
```

text dosyası uzantı şeklinde açılımı

```
1 f = open("D:\\myfiles\demo.txt","r")
2 print(f.read())
```

```
---------------------------------------------------------------------------
-
FileNotFoundError                         Traceback (most recent call
last)
~\AppData\Local\Temp\ipykernel_20240\2877203242.py in <module>
----> 1 f = open("D:\\myfiles\demo.txt","r")
      2 print(f.read())

FileNotFoundError: [Errno 2] No such file or directory:
'D:\\myfiles\\demo.txt'
```

ERROR SEBEBİ: Böyle bir dosya olmadığı için uzantıdan dosyaya erişim sağlanamadı.

açılan text dosyasını kapatma

```
1 f = open("accounts.txt", "r")
2 for x in f:
3     print(x)
4 f.close()
```

```
100 Jones 24.98

200 Doe 345.67

300 White 0.00

400 Stone -42.16

500 Rich 224.62
```

Eğer kapatılmazsa kaynağa erişim devam eder ve risk oluşturur.

Python File Write (Python Dosya Yazma)

```
f = open("accounts.txt", "a") #önceden var olan dosyaya ekleme
f.write("\nNow the file has more content!") #eklenecek yazı
f.close()
#ekleme işleminden sonra dosyayı açme ve okuma
f = open("accounts.txt", "r")
print(f.read())
```

```
100 Jones 24.98
200 Doe 345.67
300 White 0.00
400 Stone -42.16
500 Rich 224.62

Now the file has more content!
```

```
f = open("accounts.txt", "w") #var olan veriyi silip
f.write("\nNow the file has more content!") #yazıyı yazar
f.close()

f = open("accounts.txt", "r")
print(f.read())
```

```
Now the file has more content!
```

```
f = open("accounts2.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()

f = open("accounts2.txt", "r")
print(f.read())
```

```
Woops! I have deleted the content!
```

Python Delete File (Python Dosya Silme)

os modülü

- Dosyalar hakkında bilgi almamızı sağlar.
- **os.remove()** işlemi ile dosyayı siler.
- **PermissionError** Eğer dosya kapanmadan başka bir işlem yapılırsa hata gerçekleşir. Örneğin; accounts2.txt dosyasını silmeye kalkarsak bu error ile karşılaşırız. Çünkü **.read** işleminden sonra **.close** işlemi ile dosyayı kapatmak gerekmektedir.

```
1 import os
2 os.remove("a.txt")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_20240\563518979.py in <module>
      1 import os
----> 2 os.remove("a.txt")

FileNotFoundError: [WinError 2] Sistem belirtilen dosyayı bulamıyor: 'a.txt'
```

```
1 import os
2 if os.path.exists("demofile.txt"):
3     os.remove("demofile.txt")
4 else:
5     print("The file does not exist")
6 #Delete Folder
7 import os
8 os.rmdir("myfolder")
```

```
The file does not exist
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_20240\3461017587.py in <module>
      6 #Delete Folder
      7 import os
----> 8 os.rmdir("myfolder")

FileNotFoundError: [WinError 2] Sistem belirtilen dosyayı bulamıyor: 'myfolder'
```

Writing to a Text File: Introduing the with Statement (Bir Metin Dosyasına Yazma: with İfadesine Giriş)

- Birçok uygulama kaynak alır.
  - dosyalar, ağ bağlantıları, veritabanı bağlantıları ve daha fazlası

- Artık ihtiyaç duyulmayan kaynakları serbest bırakmalıdır.
- Diğer uygulamaların kaynakları kullanabilmesini sağlar.
- with ifadesi
    - Bir kaynak alır ve ona karşılık gelen nesneyi bir değişkene atar.
    - Uygulamanın, kaynağı bu değişken aracılığıyla kullanmasına izin verir.
    - Kaynağı serbest bırakmak için kaynak nesnesinin kapatma yöntemini çağırır.

```
with open('accounts.txt', mode='w') as accounts:
    accounts.write('100 Jones 24.98\n')
    accounts.write('200 Doe 345.67\n')
    accounts.write('300 White 0.00\n')
    accounts.write('400 Stone -42.16\n')
    accounts.write('500 Rich 224.62\n')
```

Dosya içeriği görüntüleme

- macOS/Linux Kullanıcıları:

!cat accounts.txt

- Windows Kullanıcıları:

!more accounts.txt

```
!more accounts.txt
```

```
100 Jones 24.98
200 Doe 345.67
300 White 0.00
400 Stone -42.16
500 Rich 224.62
```

Ayrıca şu şekilde de dosyaya yazdırma

print('100 Jones 24.98', file = accounts)

Built-In Function open

- **accounts.txt** dosyasını açar ve onu bir dosya nesnesiyle ilişkilendirir.
- **mode** bağımsız değişkeni, dosya açma modunu belirtir:
  - Bir dosyanın dosyadan okumak için mi, dosyaya yazmak için mi yoksa her ikisi için mi açılacağı
  - **'w'** modu dosyayı yazmak için açar, yoksa dosyayı oluşturur
  - Dosyanın yolunu belirtmezseniz Python dosyayı geçerli klasörde oluşturur.
  - **ÖNEMLİ!** Bir dosyayı yazmak için açmak, dosyadaki tüm mevcut verileri siler.
  - Geleneksel olarak **.txt dosya uzantısı** düz metin dosyasını belirtir.

Writing to File (Dosyaya Yazma)

- **with** ifadesi yan tümcesindeki değişken hesapları açarak döndürülen nesneyi atar.
- **with** deyim paketi, dosyayla etkileşime geçmek için hesapları kullanır.
  - dosya nesnesinin yazma yöntemi, dosyaya her seferinde bir kayıt yazar.
- **with** deyimi, dosyayı kapatmak için dolaylı olarak dosya nesnesinin **close** yöntemini çağırır.

Reading Data from a Text File (Metin Dosyasından Veri Okuma)

```
1 with open('accounts.txt', mode='r') as accounts:
2     print(f'{"Account":<10}{"Name":<10}{"Balance":>10}')
3     for record in accounts:
4         account, name, balance = record.split()
5         print(f'{account:<10}{name:<10}{balance:>10}')
6 f.close()
```

```
Account   Name         Balance
100       Jones          24.98
200       Doe           345.67
300       White           0.00
400       Stone         -42.16
500       Rich          224.62
```

- Bir dosyanın içeriğinin değiştirilmemesi gerekiyorsa, dosya sadece okumak için açılmalıdır.
  - Programın yanlışlıkla dosyayı değiştirmesini engeller.
- Önceki for ifadesinde gösterildiği gibi, bir dosya nesnesi aracılığıyla yineleme, dosyadan her seferinde bir satır okur ve onu bir dize olarak döndürür.
- Dosyadaki her kayıt (yani satır) için string yöntemi split, satırdaki belirteçleri bir liste olarak döndürür.

File Method readlines (Dosya Yöntemi Satır satır okuma)

- Bir metin dosyasının tamamını okumak için de kullanılabilir.
- Listedeki her satırı bir dizi olarak döndürür.
- Küçük dosyalar için bu iyi çalışır fakat yukarıda gösterildiği gibi bir dosya nesnesindeki satırları yinelemek daha verimli olabilir.
    - Tüm dosyayı yüklemeyi beklemek yerine, programınızın her metin satırını okunduğu gibi işlemesini sağlar.

Seeking to a Specific File Position (Belirli Bir Dosya Pozisyonunu Aramak)

- Bir dosyayı okurken sistem, okunacak bir sonraki karakterin konumunu temsil eden bir dosya konumu işaretçisi tutar.
- Bir programın yürütülmesi sırasında bir dosyayı baştan birkaç kez sırayla işlemek için, dosya konumu işaretçisini dosyanın başına yeniden konumlandırmanız gerekir:
    - Dosyayı kapatıp yeniden açarak yapabilir.
    - Dosya nesnesinin arama yöntemini aşağıdaki gibi çağırarak yapılabilir.

      file_object.seek(0)

! İkinci yaklaşım daha hızlıdır.

Updating Text Files (Metin Dosyalarını Güncelleme)

- Bir metin dosyasına yazılan biçimlendirilmiş veriler, diğer verileri bozma riski olmadan değiştirilemez.
- accounts.txt dosyasında "White" adının "Williams" olarak değiştirilmesi gerekiyorsa, eski adın üzerine yazılamaz.

300 White 0.00

300 Williams00

- Williams'taki ikinci "i"nin ötesindeki karakterler, satırdaki diğer karakterlerin üzerine yazılır.
- Sorun, biçimlendirilmiş girdi-çıktı modelinde, kayıtların ve alanlarının boyutlarının değişebilmesidir.
- Önceki adı değiştirmek için şunları yapabiliriz:
    - 300 White 0.00 'dan önceki kayıtları geçici bir dosyaya kopyalayın.
    - 300 için güncellenmiş ve doğru biçimlendirilmiş kaydı bu dosyaya yazın.
    - 300 White 0.00 'dan sonraki kayıtları geçici dosyaya kopyalayın.
    - Eski dosyayı silin.
    - Orijinal dosyanın adını kullanmak için geçici dosyayı yeniden adlandırın.

- Yalnızca bir kaydı güncellemeniz gerekse bile dosyadaki her kaydın işlenmesini gerektirir.
- Bir uygulamanın dosyanın tek geçişinde birçok kaydı güncellemesi gerektiğinde daha verimlidir.

Updating accounts.txt (accounts.txt Güncelleme)

```
1 accounts = open('accounts.txt', 'r')
```

```
1 temp_file = open('temp_file.txt', 'w')
```

accounts.txt dosyasındaki değeri değiştirmeden temp değişken dosyasına atanıp asıl dosyada değişiklik olmaması için:

```
1 with accounts, temp_file:
2     for record in accounts:
3         account, name, balance = record.split()
4         if account ≠ '300':
5             temp_file.write(record)
6         else:
7             new_record = ' '.join([account, 'Williams', balance])
8             temp_file.write(new_record + '\n')
```

- Bu with ifadesi, with'den sonra virgülle ayrılmış bir listede belirtilen iki kaynak nesnesini yönetir.
  - Hesap '300' değilse, temp_file'a (yeni satır içeren) kayıt yazarız.
  - Aksi takdirde, 'White' yerine 'Williams' içeren yeni kaydı birleştirir ve dosyaya yazarız.

```
1 !more temp_file.txt
```

```
100 Jones 24.98
200 Doe 345.67
300 Williams 0.00
400 Stone -42.16
500 Rich 224.62
```

os Module File-Processing Functions (os Modülü-Dosya İşleme İşlevleri)

- Güncellemeyi tamamlamak için eski accounts.txt dosyasını silin, ardından temp_file.txt dosyasını accounts.txt olarak yeniden adlandırın.

```
1 import os
```

```
os.remove('accounts.txt')
```

- Geçici dosyayı 'accounts.txt' olarak yeniden adlandırmak için yeniden adlandırma işlevini kullanın.

```
os.rename('temp_file.txt', 'accounts.txt')
```

```
!more accounts.txt
```

```
100 Jones 24.98
200 Doe 345.67
300 Williams 0.00
400 Stone -42.16
500 Rich 224.62
```

Serialization with JSON

- JSON (JavaScript Object Notation), nesneleri (sözlükler, listeler ve daha fazlası gibi) ad-değer çiftleri koleksiyonları olarak temsil etmek için kullanılan, metin tabanlı, insan ve bilgisayar tarafından okunabilen, veri alışverişi biçimidir.
- Nesneleri platformlar arasında iletmek için tercih edilen veri formatıdır.

JSON Data Format (JSON Veri Formatı)

- Benzer Python sözlükleri
- Her JSON nesnesi, virgülle ayrılmış bir özellik adları listesi ve süslü ayraçlar içindeki değerler içerir.
  - {"account": 100, "name": "Jones", "balance": 24.98}
- Python listeleri gibi JSON dizileri, köşeli parantez içindeki virgülle ayrılmış değerlerdir.
  - [100,200,300]
- JSON nesneleri ve dizilerindeki değerler şunlar olabilir:
  - çift tırnak içindeki dizeler
  - JSON Boole değerleri true veya false
  - null (Python'daki None gibi)
  - diziler
  - diğer JSON nesneleri

Python Standart Library Module json (Python Standart Kitaplık Modülü json)

- **json modülü**, nesneleri JSON (JavaScript Object Notation) metin biçimine dönüştürmenizi sağlar.

- Verileri **serializing** (serileştirme) olarak bilinir.
- Aşağıdaki sözlük, ilişkili anahtar 'accounts' değerinin bir anahtar-değer çiftini içerir, iki hesabı temsil eden sözlüklerin listesidir.

```
accounts_dict = {'accounts': [
    {'account': 100, 'name': 'Jones', 'balance': 24.98},
    {'account': 200, 'name': 'Doe', 'balance': 345.67}]}
```

Serializing an Object to JSON (Bir Nesneyi JSON'a Serileştirme)

- Bir dosyaya JSON yaz
- json modülünün **dump fonksiyonu** accounts_dict sözlüğünü dosyaya seri hale getirir.

```
import json
```

```
with open('accounts.json', 'w') as accounts:
    json.dump(accounts_dict, accounts)
```

- Ortaya çıkan dosya, okunabilirlik için yeniden biçimlendirilmiş aşağıdaki metni içerir:

```
    {"accounts":
    [{"accounts": 100, "name": "Jones", "balance": 24.98},
     {"accounts": 200, "name": "Doe", "balance": 345.67}]}
```

```
{'accounts': [{'accounts': 100, 'name': 'Jones', 'balance': 24.98},
  {'accounts': 200, 'name': 'Doe', 'balance': 345.67}]}
```

- JSON, dizeleri çift tırnaklı karakterlerle sınırlandırır.

Deserializing the JSON Text (JSON Metninin Seri Halini Kaldırma)

- json modülünün **load fonksiyonu**, dosya nesnesi bağımsız değişkeninin tüm JSON içeriğini okur ve JSON'u bir Python nesnesine dönüştürür.
- Verilerin serisini kaldırma olarak bilinir.

```
with open('accounts.json', 'r') as accounts:
    accounts_json = json.load(accounts)
```

```
accounts_json
```

```
{'accounts': [{'account': 100, 'name': 'Jones', 'balance': 24.98},
  {'account': 200, 'name': 'Doe', 'balance': 345.67}]}
```

```
1 type(accounts_json)
```

```
dict
```

- Sözlük için kullanılan her fonksiyon kullanılabilmektedir.

```
1 accounts_json['accounts']
```

```
[{'account': 100, 'name': 'Jones', 'balance': 24.98},
 {'account': 200, 'name': 'Doe', 'balance': 345.67}]
```

```
1 accounts_json['accounts'][0]
```

```
{'account': 100, 'name': 'Jones', 'balance': 24.98}
```

```
1 accounts_json['accounts'][1]
```

```
{'account': 200, 'name': 'Doe', 'balance': 345.67}
```

Displaying the JSON Text (JSON Metnini Görüntüleme)

- json modülünün **dumps fonksiyonu** (dumps, "dump string"in kısaltmasıdır), JSON biçiminde bir nesnenin Python dize temsilini döndürür.
- Daha okunabilir, hoş bir şekilde yazdırmak için kullanılır.

```
1 with open('accounts.json', 'r') as accounts:
2     print(json.dumps(json.load(accounts), indent=4))
```

```
{
    "accounts": [
        {
            "account": 100,
            "name": "Jones",
            "balance": 24.98
        },
        {
            "account": 200,
            "name": "Doe",
            "balance": 345.67
        }
    ]
}
```

Focus on Security: pickle Serialization and Deserialization (Güvenliğe Odaklanma: pickle Serileştirme ve Seri Kaldırma)

- Python Standart Kütüphanesinin **pickle modülü**, nesneleri Python'a özgü bir veri biçiminde seri hale getirebilir.

- **Dikkat:** Python belgeleri, pickle hakkında aşağıdaki uyarıları sağlar:
  - “pickle dosyaları hacklenebilir. Ağ üzerinden bir işlenmemiş pickle dosyası alırsanız, ona güvenmeyin! İçinde, ayıklamaya çalıştığınızda rastgele Python çalıştıran kötü amaçlı kod olabilir. Ancak, kendi pickle yazma ve okuma işleminizi yapıyorsanız, güvendesiniz. (Tabii ki pickle dosyasına başka kimsenin erişimi olmaması koşuluyla).”
  - “Pickle, keyfi olarak karmaşık Python nesnelerinin serileştirilmesine izin veren bir protokoldür. Bu nedenle Python'a özgüdür ve diğer dillerde yazılmış uygulamalarla iletişim kurmak için kullanılamaz. Ayrıca varsayılan olarak güvensizdir: Güvenilmeyen bir kaynaktan gelen pickle verilerinin seri durumundan çıkarılması, eğer veriler yetenekli bir saldırgan tarafından hazırlanmışsa rastgele kod çalıştırabilir.”

Additional Notes Regarding Files (Dosyalarla İlgili Ek Notlar)

- Metin dosyaları için dosya açma modları
  - Dosya mevcut değilse, okuma modları bir FileNotFoundError hatası verir.
  - Her metin dosyası modunun, 'rb' veya 'wb+'da olduğu gibi b ile belirtilen karşılık gelen bir ikili dosya modu vardır.
- **'r+'** Okuma ve yazma bir metin dosyası açın.
- **'w+'** Okuma ve yazma bir metin dosyası açın. Mevcut dosya içerikleri silinir.
- **'a+'** Sonunda okuma ve ekleme yapan bir metin dosyası açın. Yeni veriler dosyanın sonuna yazılır. Dosya yoksa, oluşturulur.

Other File Object Methods (Diğer Dosya Nesnesi Yöntemleri)

- **read**
  - Bir metin dosyası için, yöntemin tamsayı bağımsız değişkeni tarafından belirtilen karakter sayısını içeren bir dizi döndürür.
  - Bir ikili dosya için, belirtilen bayt sayısını döndürür.
  - Herhangi bir bağımsız değişken belirtilmezse yöntem dosyanın tüm içeriğini döndürür.
- **readline**
  - Varsa yeni satır karakteri de dahil olmak üzere metnin bir satırını dizi olarak döndürür.
  - Dosyanın sonuyla karşılaştığında boş bir dizi döndürür.
- **writelines**
  - Dizilerin bir listesini alır ve içeriğini bir dosyaya yazar.
- Python'un dosya nesneleri oluşturmak için kullandığı sınıflar, Python Standart kütüphanesinin io modülünde tanımlanır.

Handling Exceptions (Hataları Ele Alma)

- Dosyalarla çalışırken çeşitli türde hatalar oluşabilir.
- **FileNotFoundError**
  - Var olmayan bir dosyayı 'r' veya 'r+' modlarıyla okumak için açmaya çalışılması.
- **PermissionError**
  - İzniniz olmayan bir işlemi denemek
  - Hesabınızın erişmesine izin verilmeyen bir dosyayı açmanın denenmesi
  - Hesabınızın yazma izninin olmadığı bir klasörde bir dosya oluşturun
- **ValueError**
  - Zaten kapatılmış bir dosyaya yazmaya çalışıldı.

```
1 import sys
```

```
1 def fonk():
2 
3     if (len(sys.argv)≠2):
4         return
5     else:
6         dosya = sys.argv[1]
7         with open(dosya, 'r+') as d:
8             d.write("deneme")
9             #print("deneme", dosya)
```

Division by Zero and Invalid Input (Sıfıra Bölme ve Geçersiz Giriş)

Division By Zero (Sıfıra Bölme)

0'a bölmeye çalışmanın bir ZeroDivisionError ile sonuçlandığını hatırlamalıyız.

```
1 10 / 0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_20240\1725448531.py in <module>
----> 1 10 / 0

ZeroDivisionError: division by zero
```

- interpreter'ın ZeroDivisionError türünde bir hata oluşturduğu söyleniyor.
- Python'da hata:

- snippet'i sonlandırır.
- **NOT :** Sözlük anlamı makasla kesilmiş parça anlamına gelen, kendi başına çalışmayan ancak kod içerisinde kısayollar aracılığı ile kullanılan kaynak kodu parçacıklarına **snippet** denir.
- Hatanın geri izlemesini görüntüler.
- Ardından, sonraki parçacığı girebilmeniz için sonraki In[] istemini gösterir.

- Bir scriptte hata oluşursa IPython scripti sonlandırır ve hatanın geri izlemesini görüntüler.

Invalid Input (Geçersiz Giriş)

- int, a tamsayısına dönüştürmeye çalışırsanız bir ValueError üretir.

```
1 value = int(input('Enter an integer: '))
```

```
Enter an integer: hello
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call
last)
~\AppData\Local\Temp\ipykernel_20240\1466288023.py in <module>
----> 1 value = int(input('Enter an integer: '))

ValueError: invalid literal for int() with base 10: 'hello'
```

try Statements (try İfadeleri)

- İşlemeye devam etmek için kodu etkinleştirebilir.
- Aşağıdaki kod, ortaya çıkan herhangi bir ZeroDivisionError ve ValueError'ı yakalamak ve işlemek için hata işlemeyi kullanır. Bu durumda, kullanıcının girişi yeniden girmesine izin verir.

```
 1 while True:
 2     try:
 3         number1 = int(input('Enter numerator: '))
 4         number2 = int(input('Enter denominator: '))
 5         result = number1 / number2
 6     except ValueError:
 7         print('You must enter two integers\n')
 8     except ZeroDivisionError:
 9         print('Attempted to divide by zero\n')
10     else:
11         print(f'{number1:.3f} / {number2:.3f} = {result:.3f}')
12         break
```

```
Enter numerator: hello
You must enter two integers
```

```
Enter numerator: 10
Enter denominator: 0
Attempted to divide by zero

Enter numerator: 5
Enter denominator: 2
5.000 / 2.000 = 2.500
```

try Clause

- try ifadeleri hata işlemeyi etkinleştirir.
- Ardından hatalara yol açabilecek bir dize ifade

except Clause

- try caluse'ın suite'in ardından bir veya daha fazla hata cümlesi gelebilir.
- Hata işleyicileri olarak bilinir.
- Her biri, işlediği hata türünü belirtir.

else Clause

- Yalnızca try kısmındaki kod hata oluşturmadığında çalıştırılması gereken kodu belirtir.

Flow of Control for a ZeroDivisionError (ZeroDivisionError için Kontrol Akışı)

- Programda bir hatanın meydana geldiği noktaya genellikle **raise point** (yükselme noktası) denir.
- Bir **try** kısmında bir hata meydana geldiğinde hemen sonlandırılır.
- try kısmını takip eden herhangi bir hata işleyici varsa program kontrolü ilkine aktarılır.
- Hata işleyicileri yoksa yığın(stack) çözme adı verilen bir işlem gerçekleşir.
- Bir hata cümlesi hatayı başarılı bir şekilde ele aldığında, program yürütmesi, **finally** clause (eğer varsa), ardından **try** ifadesinden sonraki sonraki ifadeyle devam eder.

Catching Multiple Exceptions in One except Clause (Birden Fazla Hatayı except Yakalamak)

- except'i aşağıdaki gibi tanımlarsak tek bir değişkende tanımlanmış olur:

```
1 except(type1,type2,...) as variable_name:
```

finally Clause

try ifadesinin finally ifadesi

- **try** deyimi, herhangi bir hata cümlesinden veya başka bir cümleden sonra son cümle olarak bir finally cümlesine sahip olabilir.
- **finally** kısmının yürütülmesi kesinleşecek.
    - finally diğer dillerde, finally kısmını kaynak ayırma kodunu yerleştirmek için ideal bir konum haline getirir.
    - Python'da bu amaçla with deyimini tercih ederiz.

```
try:
    print('try suite with no exceptions raised')
except:
    print('this will not execute')
else:
    print('else executes because no exceptions in the try suite')
finally:
    print('finally always executes')
```

```
try suite with no exceptions raised
else executes because no exceptions in the try suite
finally always executes
```

```
try:
    print('try suite that raises an exception')
    int('hello')
    print('this will not execute')
except ValueError:
    print('a ValueError occured')
else:
    print('else will not execute because an exception occured')
finally:
    print('finally always executes')
```

```
try suite that raises an exception
a ValueError occured
finally always executes
```

- try'de bir hata oluşursa except'e yönlendirir, else çalışmaz.
- try'de bir hata olmazsa except çalışmayacaktır, else çalışacaktır.
- finally ise hata olsa da olmasa da çalışacaktır.

finally'nin özellikle kullanılmasının sebebi: Hata olsun olmasın her koşulu yapması ve bitmesi için kullanılır.

```
open('gradez.txt') #olmayan bir dosya
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call
```

```
try:
    with open('gradez.txt', 'r') as accounts:
        print(f'{"ID":<3}{"Name":<7}{"Grade":}')
        for record in accounts:
            student_id, name, grade = record.split()
            print(f'{student_id:<3}{name:<7}{grade}')
except FileNotFoundError:
    print('The file name you specifiend does not exist')
finally:
    accounts.close()
```

```
The file name you specifiend does not exist
```

```
def fonk():
    for dosya in sys.argv[1:]:
        try:
            with open(dosya) as d:
                toplam = 0
                for i in d:
                    toplam += int(d)
            print(toplam)
        except Exception:
            print(dosya, "problem var")
```

os.path metodu

- **os.path.exist** Bu yöntem belirtilen yolun var olup olmadığını kontrol eder.
- **os.path.isfile** Bu yöntem dosya ile bağlantısı var ise true döndürür.
- **os.path.isdir** Bu yöntem belirtilen yolun mevcut bir dizin olup olmadığını kontrol etmek için kullanılır.

```
with open('accounts.txt', 'r') as accounts:
        print(f'{"ID":<3}{"Name":<7}{"Grade":}')
        for record in accounts:
            student_id, name, grade = record.split()
            print(f'{student_id:<3}{name:<7}{grade}')
```

```
ID Name   Grade
100Jones  24.98
200Doe    345.67
300Williams0.00
400Stone  -42.16
500Rich   224.62
```

```
for record in accounts:
            student_id, name, grade = record.split()
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call
last)
~\AppData\Local\Temp\ipykernel_20240\690359494.py in <module>
----> 1 for record in accounts:
      2             student_id, name, grade = record.split()

ValueError: I/O operation on closed file.
```

ERROR SEBEBİ: with kullanımında hem dosyayı açma hem de kapatma yapılmaktadır. Bu yüzden dosyayı tekrar kullanmak istediğimizde açmadan kullanamayız.

Explicitly Raising an Exception (Açıkça Bir Hata Oşuşturmak)

- raise ile kendi hatamızı tanımlayabiliriz.
- raise ExceptionClassName şeklinde

```
raise ValueError
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call
last)
~\AppData\Local\Temp\ipykernel_20240\4096975414.py in <module>
----> 1 raise ValueError

ValueError:
```

(Optional) Stack Unwinding and Tracebacks ((İsteğe bağlı) Yığın Çözme ve Geri İzlemeler)

```
def function1():
    function2()
```

```
def function2():
    raise Exception('An exception occured')
```

```
function1()
```

```
---------------------------------------------------------------------------
-
Exception                                 Traceback (most recent call
last)
~\AppData\Local\Temp\ipykernel_20240\2505403607.py in <module>
----> 1 function1()

~\AppData\Local\Temp\ipykernel_20240\4005091792.py in function1()
      1 def function1():
----> 2     function2()
```

Hatada stack yapısı:

- İlk olarak (yani en alt kısımda) hatanın olduğu ilk yer gelir.
- Bir üstünde hatanın meydana geldiği yer gelir.
- En sonda hatanın karşılaşıldığı en son yer gelir.

Python Standard Library Module csv (Python Standart Kütüphane csv Modülü)

- Birbirinden virgülle ayrılmış değerleri olan bir csv dosyası oluşturulur.
- writer ve reader nesneleri vardır.

```
import csv
```

```
with open('accounts.csv', mode='w', newline='') as accounts:
    writer = csv.writer(accounts)
    writer.writerow([100, 'Jones', 24.98])
    writer.writerow([200, 'Doe', 345.67])
    writer.writerow([300, 'White', 0.00])
    writer.writerow([400, 'Stone', -42.16])
    writer.writerow([500, 'Rich', 224.62])
```

```
!more accounts.csv
```

```
100,Jones,24.98
200,Doe,345.67
300,White,0.0
400,Stone,-42.16
500,Rich,224.62
```

Reading from a CSV File (Bir CSV Dosyasından Okuma)

```
with open('accounts.csv', 'r', newline='') as accounts:
    print(f'{"Accounts":<10}{"Name":<10}{"Balance":<10}')
    reader = csv.reader(accounts)
    for record in reader:
```

```
5         account, name, balance = record
6         print(f'{account:<10}{name:<10}{balance:>10}')
```

```
Accounts  Name      Balance
100       Jones         24.98
200       Doe          345.67
300       White           0.0
400       Stone        -42.16
500       Rich         224.62
```

Reading CSV Files into Pandas DataFrames (CSV Dosyalarını Pandas DataFrames ile Okuma)

Datasets (Veri Setleri)

- İnternette bir sürü veri seti bulunmaktadır.
- Kendimiz de veri seti oluşturabiliriz.

```
1 !more accounts.csv
```

```
100,Jones,24.98
200,Doe,345.67
300,White,0.0
400,Stone,-42.16
500,Rich,224.62
```

```
1 import pandas as pd
```

```
1 df = pd.read_csv('accounts.csv')
```

```
1 df
```

	100	Jones	24.98
0	200	Doe	345.67
1	300	White	0.00
2	400	Stone	-42.16
3	500	Rich	224.62

Tabloda sütun isimleri yoksa ilk satırda olması gereken kısım yer almıştır.

Bu hatayı düzeltmek için aşağıdaki düzenleme yapılır:

```
1 df = pd.read_csv('accounts.csv', names=['account', 'name', 'balance'])
```

```
1 df
```

	account	name	balance
0	100	Jones	24.98
1	200	Doe	345.67
2	300	White	0.00
3	400	Stone	-42.16
4	500	Rich	224.62

```
1 df.describe()
```

	account	balance
count	5.000000	5.000000
mean	300.000000	110.622000
std	158.113883	166.701248
min	100.000000	-42.160000
25%	200.000000	0.000000
50%	300.000000	24.980000
75%	400.000000	224.620000
max	500.000000	345.670000

```
1 df.to_csv('accounts_from_dataframe.csv', index=False)
```

```
1 !more accounts_from_dataframe.csv
```

```
account,name,balance
100,Jones,24.98
200,Doe,345.67
300,White,0.0
400,Stone,-42.16
500,Rich,224.62
```

VERİ SETLERİNİ ALIP KULLANMAK İLE İLGİLİ ÖRNEK: Reading the Titanic Disaster Dataset (Titanik Kazası Veri Kümesini Okuma)

```
1 import pandas as pd
```

```
1 titanic = pd.read_csv('https://vincentarelbundock.github.io/Rdatasets/csv/carDat
```

```
1 veri = pd.read_csv("TitanicSurvival.csv")
```

```
1 veri
```

	Unnamed: 0	survived	sex	age	passengerClass
0	Allen, Miss. Elisabeth Walton	yes	female	29.0000	1st
1	Allison, Master. Hudson Trevor	yes	male	0.9167	1st
2	Allison, Miss. Helen Loraine	no	female	2.0000	1st
3	Allison, Mr. Hudson Joshua Crei	no	male	30.0000	1st
4	Allison, Mrs. Hudson J C (Bessi	no	female	25.0000	1st
...	...	...	...	...	...
1304	Zabour, Miss. Hileni	no	female	14.5000	3rd
1305	Zabour, Miss. Thamine	no	female	NaN	3rd
1306	Zakarian, Mr. Mapriededer	no	male	26.5000	3rd
1307	Zakarian, Mr. Ortin	no	male	27.0000	3rd
1308	Zimmerman, Mr. Leo	no	male	29.0000	3rd

1309 rows × 5 columns

```
1 titanic.head() # ilk beş eleman yazdırılır.
```

	Unnamed: 0	survived	sex	age	passengerClass
0	Allen, Miss. Elisabeth Walton	yes	female	29.0000	1st
1	Allison, Master. Hudson Trevor	yes	male	0.9167	1st
2	Allison, Miss. Helen Loraine	no	female	2.0000	1st
3	Allison, Mr. Hudson Joshua Crei	no	male	30.0000	1st
4	Allison, Mrs. Hudson J C (Bessi	no	female	25.0000	1st

```
1 titanic.tail() # son beş eleman yazdırılır.
```

	Unnamed: 0	survived	sex	age	passengerClass
1304	Zabour, Miss. Hileni	no	female	14.5	3rd
1305	Zabour, Miss. Thamine	no	female	NaN	3rd
1306	Zakarian, Mr. Mapriededer	no	male	26.5	3rd
1307	Zakarian, Mr. Ortin	no	male	27.0	3rd
1308	Zimmerman, Mr. Leo	no	male	29.0	3rd

```
1 titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Unnamed: 0      1309 non-null   object
 1   survived        1309 non-null   object
 2   sex             1309 non-null   object
 3   age             1046 non-null   float64
 4   passengerClass  1309 non-null   object
dtypes: float64(1), object(4)
memory usage: 51.3+ KB
```

Burada 1309 kişi var ama yaş verisi 1046 yani yaşı belli olmayanlar var demektir. titanic.info() ile bu değerlere kolayca ulaşırız. **Dtype** da önemlidir.

Sütun Adlarını Değiştirme

```
1 titanic.columns = ['name','survived', 'gender', 'age', 'class']
```

```
1 titanic.head()
```

	name	survived	gender	age	class
0	Allen, Miss. Elisabeth Walton	yes	female	29.0000	1st
1	Allison, Master. Hudson Trevor	yes	male	0.9167	1st
2	Allison, Miss. Helen Loraine	no	female	2.0000	1st
3	Allison, Mr. Hudson Joshua Crei	no	male	30.0000	1st
4	Allison, Mrs. Hudson J C (Bessi	no	female	25.0000	1st

```
1 titanic.describe()
```

age

```
1 (titanic.survived == 'yes').describe()
```

```
count      1309
unique        2
top       False
freq        809
Name: survived, dtype: object
```

25% 21.000000

Histogramını Alma

75% 39.000000

```
1 %matplotlib inline
```

```
1 histogram = titanic.hist()
```