File Handling(Dosya Yönetimi)

Verilerimizi genellikle farklı dosya formatlarında saklarız. Dosyaları işlemeye ek olarak, bu bölümde farklı dosya formatlarını da (.txt, .json, .xml, .csv, .tsv, .excel) göreceğiz. İlk olarak, ortak dosya formatına (.txt) sahip dosyaları işlemeye alışalım.

Dosya işleme, dosyaları oluşturmamıza, okumamıza, güncellememize ve silmemize izin veren programlamanın içe aktarılan bir parçasıdır. Python'da verileri işlemek için open() yerleşik işlevini kullanırız.

Opening Files for Reading(Dosyaları Okumak İçin Açma)

Varsayılan açma modu okumadır, bu nedenle 'r' veya 'rt' belirtmemiz gerekmez. Dosyalar dizininde read\_file.txt adlı bir dosya oluşturup kaydettim

f  =  open( 'read\_file.txt','r')

print ( f )

#<\_io.TextIOWrapper name='read\_file.txt' mode='r' encoding='cp1254'>

* Yukarıdaki örnekte gördüğümüz gibi 'r' değeri görmekteyiz. Okumak için bir dosya açar, dosya yoksa hata verir anlamında yazılır. read() , readline , readlines . Açılan bir dosya close() metodu ile kapatılmalıdır .
* read()

Örnek:

f = open('read\_file.txt','r')

txt = f.read()

print(type(txt))

print(txt)

f.close()

<class 'str'>

Tüm metni yazdırmak yerine, metin dosyasının ilk 10 karakterini yazdıralım.

f = open('read\_file.txt','r')

txt = f.read(10)

print(type(txt))

print(txt)

f.close()

readline() : sadece ilk satırı okur. Aşağıda bir dosyanın nasıl açılıp okunacağını gösteren bir örneği inceleyelim.

f = open('read\_file.txt','r')

line = f.readline()

print(type(line))

print(line)

f.close()

* readlines() : tüm metni satır satır okur ve bir satır listesi döndürür şimdi bir dosyanın açılıp okunduğu örneği inceleyelim.

f = open('read\_file.txt','r')

line = f.readlines()

print(type(line))

print(line)

f.close()

Aynı işlemi “f.close()” kullanmak zorunda olmadan “with” ile de halledebiliriz. Aşağıdaki örnek kodu inceleyelim.

with open('read\_file.txt','r') as f:

    lines = f.read().splitlines()

    print(type(lines))

    print(lines)

## Opening Files for Writing and Updating(Dosyaları Yazma ve Güncelleme için Açma)

Az önce 'r' den bahsetmiştik şimdi ise 'a' ve 'w' ele alalım.

"a" (ekle ): dosya yeni bir dosya oluşturmazsa, dosyanın sonuna eklenir.

"w" (yazma): oluşturduğu dosya mevcut değilse, mevcut içeriğin üzerine yazar. Okumakta olduğumuz dosyaya biraz metin ekleyelim:

Örnek:

with open('read\_file.txt','a') as f:

    f.write('ali ')

Örnek:

with open('read\_file.txt','w') as f:

    f.write('ali ')

## Deleting Files(Dosyaları Silme)

Dosyayı kaldırmak için aşağıdaki kodu kullanabilriz.

import os

os.remove('read\_file.txt')

# File Types(Dosya Türleri)

## File with txt Extension(txt uzantılı dosya)

txt uzantılı dosya çok yaygın bir veri biçimidir ve önceki bölümde ele aldık. JSON dosyasına geçelim

## File with json Extension(json uzantılı dosya)

JSON, JavaScript Nesne Gösterimi anlamına gelir. Aslında, dizilenmiş bir JavaScript nesnesi veya Python sözlüğüdür.

person\_json  =  '''{

    "name":"Büşra",

    "country":"Türkiye",

    "city":"İstanbul",

    "skils":[ "c++", "R","Python"]

}'''

#JSON sölükten bir dize

## Changing JSON to Dictionary(json'u sözlüğe değiştrime)

Bir JSON'u sözlüğe dönüştürmek için önce json modülünü içe aktarıyoruz ve ardından load() yöntemini kullanıyoruz.

Örnek:

import json

person\_json  =  '''{

    "name":"Büşra",

    "country":"Türkiye",

    "city":"İstanbul",

    "skils":[ "c++", "R","Python"]

}'''

# json sözlükte değiştirelim

person\_dct = json.loads(person\_json)

print(type(person\_dct))

print(person\_dct)

print(person\_dct['name'])

Çıktı:

{'name': 'Büşra', 'country': 'Türkiye', 'city': 'İstanbul', 'skils': ['c++', 'R', 'Python']}

Büşra

## Changing Dictionary to JSON(Sözlüğü Json'a değiştirme)

Bir sözlüğü JSON olarak değiştirmek için json modülünden dumps yöntemini kullanırız.

Örnek:

import json

person\_json  =  '''{

    "name":"Büşra",

    "country":"Türkiye",

    "city":"İstanbul",

    "skils":[ "c++", "R","Python"]

}'''

person= json.dumps(person\_json, indent=4)

print(type(person\_json))

print(person\_json)

Çıktı:

<class 'str'>

{

    "name":"Büşra",

    "country":"Türkiye",

    "city":"İstanbul",

    "skils":[ "c++", "R","Python"]

}

## Saving as JSON File(Json dosyası olarak kaydetme)

Verilerimizi bir json dosyası olarak da kaydedebiliriz. Aşağıdaki adımları kullanarak json dosyası olarak kaydedelim. Bir json dosyası yazmak için json.dump() yöntemini kullanıyoruz.

Örnek:

import json

person\_json  =  {

    "name":"Büşra",

    "country":"Türkiye",

    "city":"İstanbul",

    "skils":[ "c++", "R","Python"]

}

with open('json\_example.json', 'w', encoding='utf-8') as f:

    json.dump(person, f, ensure\_ascii=False, indent=4)

Yukarıda görüldüğü gibi yeni bir dosya olarak kayıt yapmıştır.

## File with csv Extension(csv uzantılı dosya)

CSV, virgülle ayrılmış değerler anlamına gelir. CSV, elektronik tablo veya veritabanı gibi tablo verilerini depolamak için kullanılan basit bir dosya biçimidir. CSV, veri biliminde çok yaygın bir veri formatıdır.

## File with xlsx Extension(xlsx uzantılı dosya)

Excel dosyalarını okumak için xlrd paketini kurmamız gerekiyor . Bunu pip kullanarak paket kurulumunu yaptıktan sonra ele alacağız.

import xlrd

excel\_book = xlrd.open\_workbook('read\_file.xls)

print(excel\_book.nsheets)

print(excel\_book.sheet\_names)

## File with xml Extension(xml uzantılı dosyalar)

XML, HTML'ye benzeyen başka bir yapılandırılmış veri biçimidir. XML'de etiketler önceden tanımlanmamıştır. İlk satır bir XML bildirimidir. Kişi etiketi, XML'in köküdür. Kişinin bir cinsiyet özelliği vardır Örneği inceleyelim:

<?xml version="1.0"?>

<person gender="female">

  <name>Asabeneh</name>

  <country>Finland</country>

  <city>Helsinki</city>

  <skills>

    <skill>JavaScrip</skill>

    <skill>React</skill>

    <skill>Python</skill>

  </skills>

</person>

Örnek:

import xml.etree.ElementTree as ET

tree = ET.parse('read\_file.xml')

root = tree.getroot()

print('Root tag:', root.tag)

print('Attribute:', root.attrib)

for child in root:

    print('field: ', child.tag)