Python les-materialen Yilmaz Mustafa, Instructeur Java/Python

# Files (Bestanden)

Python gebruikt file-objecten (bestandsobjecten) om te communiceren met externe bestanden op uw computer. Deze bestandsobjecten kunnen elk soort bestand zijn dat u op uw computer heeft, of het nu een audiobestand, een tekstbestand, e-mails, Excel-documenten, enz. is. Opmerking: u zult waarschijnlijk bepaalde bibliotheken of modules moeten installeren om met die verschillende bestandstypen, maar ze zijn gemakkelijk beschikbaar. (Later in de cursus zullen we het downloaden van modules behandelen).

Python heeft een ingebouwde open-functie waarmee we basisbestandstypen kunnen openen en ermee kunnen spelen. Eerst hebben we echter een bestand nodig. We gaan wat IPython-magie gebruiken om een tekstbestand te maken!

## IPython Een bestand schrijven

#### Deze functie is specifiek voor jupyter-notitieboeken! U kunt ook snel een eenvoudig .txt-bestand maken met een sublieme teksteditor.

%%writefile test.txt  
Hello, this is a quick test file.

Overwriting test.txt

## Python Een bestand openen

Laten we dat doen door het bestand test.txt te openen dat zich in dezelfde map als dit notitieblok bevindt. Voor nu zullen we werken met bestanden die zich in dezelfde map bevinden als het notebook- of .py-script dat u gebruikt.

Het is heel gemakkelijk om een foutmelding te krijgen bij deze stap:

myfile = open('whoops.txt')

---------------------------------------------------------------------------  
  
FileNotFoundError Traceback (most recent call last)  
  
<ipython-input-1-dafe28ee473f> in <module>()  
----> 1 myfile = open('whoops.txt')  
  
  
FileNotFoundError: [Errno 2] No such file or directory: 'whoops.txt'

Om deze fout te voorkomen, moet u ervoor zorgen dat uw .txt-bestand op dezelfde locatie als uw notebook is opgeslagen. Gebruik **pwd** om de locatie van uw notebook te controleren:

pwd

'C:\\Users\\Marcial\\Pierian-Data-Courses\\Complete-Python-3-Bootcamp\\00-Python Object and Data Structure Basics'

**Als alternatief, om bestanden van elke locatie op uw computer te pakken, geeft u gewoon het volledige bestand-pad door.**

Voor Windows moet je dubbel  (back slash) gebruiken zodat python de tweede  niet als een escape-teken behandelt, een bestand-pad heeft de vorm:

myfile = open("C:\\Users\\YourUserName\\Home\\Folder\\myfile.txt")

Voor MacOS en Linux gebruik je schuine strepen (forward slash) in de tegenovergestelde richting:

myfile = open("/Users/YouUserName/Folder/myfile.txt")

# Open de text.txt die we eerder hebben gemaakt  
my\_file = open('test.txt')

# We kunnen het bestand nu lezen  
my\_file.read()

'Hello, this is a quick test file.'

# Maar wat gebeurt er als we het opnieuw proberen te lezen?  
my\_file.read()

''

Dit gebeurt hier omdat je kunt voorstellen dat de lezende “cursor” aan het einde van het bestand staat nadat je het hebt gelezen. Er valt dus niets meer te lezen. We kunnen de “cursor” als volgt resetten:

# Zoek naar het begin van het bestand (index 0)  
my\_file.seek(0)

0

# Lees nu nog eens  
my\_file.read()

'Hello, this is a quick test file.'

U kunt een bestand lijn voor lijn lezen met behulp van de readlines-methode. Wees voorzichtig met grote bestanden, aangezien alles in het geheugen wordt bewaard. We zullen later in de cursus leren hoe we grote bestanden kunnen herhalen.

# Readlines retourneert een lijst van de regels in het bestand  
my\_file.seek(0)  
my\_file.readlines()

['Hello, this is a quick test file.']

Wanneer u klaar bent met het gebruik van een bestand, is het altijd een goede praktijk om het te sluiten.

my\_file.close()

## Schrijven naar een bestand

Standaard staat de functie open() ons alleen toe om het bestand te lezen. We moeten het argument ‘’w’’ doorgeven om over het bestand te schrijven. Bijvoorbeeld:

# Voeg een tweede argument toe aan de functie, 'w' wat staat voor schrijven.  
# Door 'w+' door te geven, kunnen we het bestand lezen en erin schrijven  
  
my\_file = open('test.txt','w+')

### Wees voorzichtig!

Het openen van een bestand met 'w' of 'w+' kapt het origineel af, wat inhoudt dat alles wat in het originele bestand stond **wordt verwijderd**!

# Schrijf naar het bestand  
my\_file.write('This is a new line')

18

# Lees het bestand  
my\_file.seek(0)  
my\_file.read()

'This is a new line'

my\_file.close() # doe dit altijd als je klaar bent met een bestand

## Toevoegen aan een bestand

Als u het argument ‘’a’’ doorgeeft, wordt het bestand geopend en wordt de aanwijzer (pointer) aan het einde geplaatst, zodat alles wat is geschreven, wordt toegevoegd. Net zoals ‘’w+’‘, laat’‘a+’’ ons lezen en schrijven naar een bestand. Als het bestand niet bestaat, wordt er een gemaakt.

my\_file = open('test.txt','a+')  
my\_file.write('\nThis is text being appended to test.txt')  
my\_file.write('\nAnd another line here.')

23

my\_file.seek(0)  
print(my\_file.read())

This is a new line  
This is text being appended to test.txt  
And another line here.

my\_file.close()

### Toevoegen (Appending) met %%writefile

We kunnen hetzelfde doen met behulp van IPython-celmagie:

%%writefile -a test.txt  
  
This is text being appended to test.txt  
And another line here.

Appending to test.txt

Voeg een spatie toe als u wilt dat de eerste regel op een eigen regel begint, aangezien Jupyter escape-reeksen zoals \n niet herkent

## Een bestand doorlopen

Laten we een snel voorbeeld van een for-lus krijgen door een tekstbestand te herhalen. Laten we eerst een nieuw tekstbestand maken met wat IPython Magic:

%%writefile test.txt  
First Line  
Second Line

Overwriting test.txt

Nu kunnen we een klein beetje flow (stroom) gebruiken om het programma door elke lijn van het bestand te vertellen en iets te doen:

for line in open('test.txt'):  
 print(line)

First Line  
  
Second Line

Maak je geen zorgen dat je dit nog volledig begrijpt, want er komen binnenkort lussen aan. Maar we zullen afbreken wat we hierboven hebben gedaan. We zeiden dat voor elke regel in dit tekstbestand, ga je gang en druk die regel af. Het is belangrijk om hier een paar dingen op te merken:

1. We hadden het “line” -object alles kunnen noemen (zie voorbeeld hieronder).
2. Door .read() niet aan te roepen op het bestand, werd het hele tekstbestand niet in het geheugen opgeslagen.
3. Let op de inspringing (indent) op de tweede lijn om af te drukken. Deze spatie is vereist in Python.

# Met betrekking tot het eerste punt hierboven  
for asdf in open('test.txt'):  
 print(asdf)

First Line  
  
Second Line

We zullen hier later veel meer over leren, aangezien het meeste applicaties vereisten om bestanden-data te manipuleren.