فایل‌ها

پایتون از اشیای فایل برای تعامل با فایل‌های خارجی در رایانه‌تان استفاده می‌کند. این اشیای فایل می‌توانند هر نوع فایلی که در رایانه‌تان دارید باشند، مانند فایل صوتی، فایل متنی، ایمیل‌ها، اسناد اکسل و غیره. توجه: شاید نیاز داشته باشید کتابخانه‌ها یا ماژول‌های خاصی را نصب کنید تا با این انواع مختلف فایل تعامل کنید، اما آن‌ها به راحتی در دسترس هستند. (در دورهٔ بعدی درباره دانلود کردن ماژول‌ها صحبت خواهیم کرد).

پایتون یک تابع باز کنندهٔ داخلی دارد که به ما اجازه می‌دهد با انواع پایهٔ فایل بازی کنیم. اما ابتدا به یک فایل نیاز داریم. ما قصد داریم از یکی از قدرت‌های IPython برای ایجاد یک فایل متنی استفاده کنیم!

نوشتن یک فایل در IPython

این تابع ویژهٔ نوت‌بوک‌های Jupyter است! به طور جایگزین، می‌توانید به سرعت یک فایل سادهٔ txt. را با ویرایشگر sublime ایجاد کنید.

In [1]:

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

```
Overwriting test.txt
```

باز کردن یک فایل در پایتون

بیایید با شروع، فایل test.txt را که در همان دایرکتوری این نوت‌بوک قرار دارد، باز کنیم. در حال حاضر، ما با فایل‌هایی که در همان دایرکتوری نوت‌بوک یا اسکریپت py. قرار دارند کار خواهیم کرد.

در این مرحله بسیار آسان است که خطایی دریافت کنید:

In [1]:

```
myfile = open('whoops.txt')
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call las
t)
<ipython-input-1-dafe28ee473f> in <module>()
----> 1 myfile = open('whoops.txt')

FileNotFoundError: [Errno 2] No such file or directory: 'whoops.txt'
```

برای جلوگیری از این خطا، مطمئن شوید که فایل txt. شما در همان محلی که نوت‌بوک شما قرار دارد، ذخیره شده باشد. برای بررسی مکان نوت‌بوک خود، از **pwd** استفاده کنید:

In [1]:

```
pwd
```

Out[1]:

```
'C:\\Users\\babak\\Complete-Python-3-Bootcamp-master\\00-Python Object and
Data Structure Basics'
```

به عنوان یک روش جایگزین، برای دریافت فایل‌ها از هر محلی در رایانه‌تان، به سادگی مسیر کامل فایل را به عنوان ورودی ارسال کنید.

برای ویندوز، باید دو بار علامت \ استفاده کنید تا پایتون علامت \ دوم را به عنوان یک کاراکتر فرار متصور نکند، مسیر فایل به صورت زیر است:

```
myfile = open("C:\\Users\\نام‌کاربری\\خانه\\پوشه\\myfile.txt")
```

برای مک و لینوکس، از خط کج به سمت معکوس استفاده می‌کنید:

```
myfile = open("/Users/نام‌کاربری/پوشه/myfile.txt")
```

In [2]:

```
# Open the text.txt we made earlier
my_file = open('test.txt')
```

In [3]:

```
# We can now read the file
my_file.read()
```

Out[3]:

```
'Hello, this is a quick test file.'
```

In [4]:

```
# But what happens if we try to read it again?
my_file.read()
```

Out[4]:

```
''
```

این اتفاق می‌افتد زیرا می‌توانید تصور کنید که "مکان نمایشگر" خواندنی در انتهای فایل قرار دارد بعد از آنکه فایل را خوانده‌اید. بنابراین، چیزی برای خواندن باقی نمی‌ماند. می‌توانیم "مکان نمایشگر" را مجدداً تنظیم کنیم به این شکل:

In [5]:

```
# Seek to the start of file (index 0)
my_file.seek(0)
```

Out[5]:

```
0
```

In [6]:

```
# Now read again
my_file.read()
```

Out[6]:

```
'Hello, this is a quick test file.'
```

شما می‌توانید با استفاده از متد readlines یک فایل را به صورت خط به خط بخوانید. با فایل‌های بزرگ، با احتیاط عمل کنید زیرا همه چیز در حافظه نگهداری خواهد شد. در دورهٔ بعدی خواهیم آموخت که چگونه بر روی فایل‌های بزرگ حلقه زنی کنیم.

In [7]:

```
# Readlines returns a list of the lines in the file
my_file.seek(0)
my_file.readlines()
```

Out[7]:

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

وقتی که از یک فایل استفاده کردید، همیشه عمل خوبی است که آن را ببندید.

In [8]:

```
my_file.close()
```

نوشتن در یک فایل

به طور پیش‌فرض، تابع `open()` فقط به ما اجازه می‌دهد فایل را بخوانیم. برای اینکه بتوانیم در فایل بنویسیم، باید آرگومان `'w'` را به آن بدهیم. به عنوان مثال:

In [9]:

```
# Add a second argument to the function, 'w' which stands for write.
# Passing 'w+' lets us read and write to the file

my_file = open('test.txt','w+')
```

با احتیاط عمل کنید!

با باز کردن یک فایل با استفاده از `'w'` یا `'w+'` ، محتوای اصلی فایل بریده می‌شود، به این معنی که هر چیزی که در فایل اصلی بود **پاک می‌شود**!

In [10]:

```
# Write to the file
my_file.write('This is a new line')
```

Out[10]:

```
18
```

In [11]:

```
# Read the file
my_file.seek(0)
my_file.read()
```

Out[11]:

```
'This is a new line'
```

In [12]:

```
my_file.close()  # always do this when you're done with a file
```

افزودن به یک فایل

ارسال آرگومان 'a' فایل را باز می‌کند و نشانگر را در انتها قرار می‌دهد، بنابراین هر چیزی که نوشته می‌شود به فایل اضافه می‌شود. مانند 'w+' و 'a+' به ما اجازه می‌دهد برای یک فایل بخوانیم و بنویسیم. اگر فایل وجود نداشته باشد، یک فایل ساخته می‌شود.

In [13]:

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

Out[13]:

```
23
```

In [14]:

```
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.
```

In [15]:

```
my_file.close()
```

افزودن با استفاده از %%writefile

می‌توانیم همین کار را با استفاده از قدرت‌های خاص IPython انجام دهیم:

In [16]:

```
%%writefile -a test.txt

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

```
Appending to test.txt
```

اگر می‌خواهید خط اول بر روی خط خودش آغاز شود، یک فضای خالی اضافه کنید، زیرا Jupyter ترتیب‌های فرار مانند \n را تشخیص نمی‌دهد.

حلقه ها در یک فایل

بیایید با استفاده از یک حلقه for بر روی یک فایل متنی، یک نمایش سریع از عملکرد آن را ببینیم. ابتدا بیایید یک فایل متنی جدید با استفاده از برخی از قدرت‌های IPython ایجاد کنیم:

In [17]:

```
%%writefile test.txt
First Line
Second Line
```

```
Overwriting test.txt
```

حالا می‌توانیم از یک کمی روند استفاده کنیم تا به برنامه بگوییم که در هر خط از فایل حلقه بزند و عملی انجام دهد:

In [18]:

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

```
First Line

Second Line
```

فعلاً نگران جزئیات کامل نباشید، حلقه‌های for در ادامه قرار دارند. اما ما در بالا آنچه انجام دادیم را تجزیه کردیم. ما گفتیم که برای هر خط در این فایل متنی، ادامه دهید و آن خط را چاپ کنید. در اینجا چند نکته مهم وجود دارد:

1. ما می‌توانستیم هرچه که بخواهیم به "اشیای خط" نامیده شود (نمونه زیر را ببینید).
2. با فراخوانی .read() بر روی فایل، کل محتوای فایل متنی در حافظه ذخیره نشده است.
3. توجه کنید که برای دستور چاپ، فاصله درونی در خط دوم لازم است. این فاصله سفید در پایتون الزامی است.

In [19]:

```
# Pertaining to the first point above
for asdf in open('test.txt'):
    print(asdf)
```

```
First Line

Second Line
```