

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 دوره بعدی درباره دانلود کردن ماژول‌ها صحبت خواهیم کرد).

پایتون یک تابع بازکننده داخلی دارد که به ما اجازه می‌دهد با انواع پایۀ 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 ویژه نوت‌بوک‌های Jupyter است! به طور جایگزین، می‌توانید به سرعت یک فایل ساده‌ی test.txt را با ویرایشگر sublime ایجاد کنید.

In [1]:

```
1 %%writefile test.txt
2 Hello, this is a quick test file.
```

Overwriting test.txt

باز کردن یک فایل در پایتون

بیاید 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 [2]:

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

```
-----  
-  
FileNotFoundError                                Traceback (most recent call last)  
Cell In[2], line 1  
----> 1 myfile = open('whoops.txt')  
  
File C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:284, in _modified_open(file, *args, **kwargs)  
    277 if file in {0, 1, 2}:  
    278     raise ValueError(  
    279         f"IPython won't let you open fd={file} by default "  
    280         "as it is likely to crash IPython. If you know what you are  
e doing, "  
    281         "you can use builtins' open."  
    282     )  
--> 284 return io_open(file, *args, **kwargs)  
  
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 [3]:

```
1 pwd
```

Out[3]:

```
'C:\\Users\\babak\\Python For Everyone\\00-Python Object and Data Structures 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\\پوشه\\خانه\\نامکاربری\\myfile.txt")
```

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

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

In [5]:

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

In [6]:

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

Out[6]:

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

In [7]:

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

Out[7]:

```
''
```

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 [8]:

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

Out[8]:

```
0
```

In [9]:

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

Out[9]:

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

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 دوره بعدی خواهیم آموخت که چگونه بر روی فایل‌های بزرگ حلقه زنی کنیم.

In [10]:

```
1 # Readlines returns a list of the lines in the file
2 my_file.seek(0)
3
4 my_file.readlines()
```

Out[10]:

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

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

In [11]:

```
1 my_file.close()
```

نوشتن در یک فایل

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

In [12]:

```
1 # Add a second argument to the function, 'w' which stands for write.
2 # Passing 'w+' Lets us read and write to the file
3 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 می شود، به این معنی که هر چیزی که در فایل اصلی بود پاک می شود!

In [13]:

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

Out[13]:

18

In [14]:

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

Out[14]:

'This is a new line'

In [15]:

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

افزودن به یک فایل

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

In [16]:

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

Out[16]:

23

In [17]:

```
1 my_file.seek(0)
2 print(my_file.read())
```

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

In [18]:

```
1 my_file.close()
```

افزودن با استفاده از %%writefile

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

In [19]:

```
1 %%writefile -a test.txt
2
3 This is text being appended to test.txt
4 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

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

In [20]:

```
1 %%writefile test.txt
2 First Line
3 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 [21]:

```
1 for line in open('test.txt'):
2     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 [22]:

```
1 # Pertaining to the first point above
2 for i in open('test.txt'):
3     print(i)
```

First Line

Second Line