برنامه نویسی کامپیوتر ـ زمستان ۹۷
تدریس توسط: حسین جوهری
دانشگاه خواجه نصیرالدین طوسی

خلاصه هفته دهم

مفاهیم: **خواندن و نوشتن داده در فایل**

تا به حال با ورود و خروج اطلاعات تنها از طریق کیبورد و صفحه نمایش آشنا شده‌ایم. یک برنامه علاوه بر کیبورد باید بتواند داده‌های مورد نیاز را از فایلهای ذخیره شده روی دیسک نیز دریافت بکند و نتایج را در فایلهای مربوطه بنویسد. در این هفته با ورودی ـ خروجی از طریق فایل آشنا می‌شویم. به اصطلاح یاد می‌گیریم که چگونه داده‌ها را از فایل بخوانیم و در فایل بنویسیم. برای سادگی، در این کلاس تنها با فایل متنی کار می‌کنیم.

۱ کار با فایل در پایتون

مهمترین دستور در زمینه کار با فایل در پایتون دستور open است. این دستور دو پارامتر اصلی دارد. پارامتر اول نام فایلی است که قرار است برای خواندن یا نوشتن باز شود. در پارامتر دوم شیوه دسترسی به فایل تعیین می‌شود. به مثالهای زیر توجه کنید.

۱.۱ شیوه دسترسی به فایل

```
file = open ("data.txt","r")
```

در مثال بالا فایل data.txt صرفا به منظور خواندن اطلاعات آن باز شده است. در اینجا مقدار برگشتی در متغیر file (متغیری که از طریق آن اطلاعات فایل گرفته می‌شود) قرار داده شده است. اسم file صرفا اختیاری است و هر شناسه معتبر دیگری می‌تواند باشد.

حال می‌توان از طریق متغیر file داده درون فایل data.txt را خواند. لازم به ذکر است چون تنها اسم فایل ذکر شده است (و آدرس کامل فایل ذکر نشده است) فایل باید در همان پوشه باشد که برنامه پایتون قرار گرفته است. اگر فایلی به اسم data.txt در آنجا نباشد، مفسر پیام خطا تولید می‌کند.

```
file = open ("data.txt","w")
```

در مثال بالا فایل data.txt صرفا به منظور نوشتن (درج اطلاعات در فایل) باز می‌شود. اگر فایلی به این اسم در پوشه برنامه نباشد، فایلی خالی به نام data.txt ایجاد می‌شود.

```
file = open ("data.txt","a")
```

در مثال بالا فایل data.txt صرفا به منظور نوشتن و درج در انتهای فایل (حرف a از کلمه append گرفته شده است) باز می‌شود. اگر فایلی به این اسم در پوشه برنامه نباشد، فایلی خالی به نام data.txt ایجاد می‌شود.

۲.۱ خواندن داده از فایل

در صورتی که دستور open با موفقیت اجرا شود، داده‌های فایل از طریق متغیر file قابل دسترسی هستند.
فرض کنید محتوای فایل data.txt بصورت زیر باشد.

```
Ask, and it will be given you.
Seek, and you will find.
Knock, and it will be opened to you.
For every one who asks receives, and
he who seeks finds, and
to him who knocks it will be opened.
```

قطعه کد زیر فایل data.txt را در صورت وجود باز می‌کند و متن داخل آن را می‌خواند. متن داخل فایل بصورت تمام و کامل با استفاده از فراخوانی متد read در رشته s قرار می‌گیرد. با دستور print محتوای رشته s در خروجی چاپ می‌شود.

```
file = open ("data.txt","r")
s = file.read()
print(s)  # prints the content of data.txt
```

اگر بخواهیم تنها به تعدادی مشخص کاراکتر از فایل بخوانیم می‌توانیم در متد read تعداد را مشخص کنیم.

```
file = open ("data.txt","r")
s = file.read(8)
print(s)  # prints the content of data.txt
```

خروجی کد بالا بصورت زیر است.

```
Ask, and
```

فایل را می‌توان خط به خط نیز خواند. این کار با استفاده از متد readline قابل انجام است. در مثال زیر متغیر s حاوی خط اول فایل است (تا جایی که اولین کاراکتر سر خط وارد شده است.)

```
file = open ("data.txt","r")
s = file.readline()
print(s)  # prints the first line
```

خروجی کد بالا بصورت زیر است.

```
Ask, and it will be given you.
```

با فراخوانی readline برای بار دوم خط بعدی خوانده می‌شود.

```
file = open ("data.txt","r")
s = file.readline()
print(s)  # prints the first line
s = file.readline()
print(s)  # prints the second line
```

خروجی کد بالا بصورت زیر است.

```
Ask, and it will be given you.
Seek, and you will find.
```

می‌توان همه خطوط فایل را در قالب یک لیست دریافت کرد. این کار را با متد readlines انجام می‌دهیم. در این حالت s یک رشته نیست. بلکه لیستی از رشته هاست. هر عنصر لیست یک خط از فایل است.

```
file = open ("data.txt","r")
s = file.readlines()
print(s)  # prints the list of lines
```

خروجی کد بالا به شکل زیر است.

```
["Ask, and it will be given you.", "Seek, and you will find.",
 "Knock, and it will be opened to you.", "For every one who asks
 receives, and",  "he who seeks finds, and", "to him who knocks
  it will be opened."]
```

می‌توان روی file حلقه for اجرا کرد و خط به خط محتوای فایل را خواند.

```
file = open ("data.txt","r")
for line in file:
     print(line)
```

حال فرض کنید محتوای فایل data.txt بصورت زیر باشد. لیستی از رکوردهای دانشجویان یک کلاس. هر خط اطلاعات مربوط به یک دانشجو را نگه می‌دارد. شماره دانشجویی، نام، نام خانوادگی.

```
96234521,  mohammad, karimi
97232313,  zahra sadat,  hosseini
97239283,  farhad, amiri
96343243,  samira, jalili
```

می‌خواهیم اطلاعات دانشجویان را خوانده و آن را در لیستی ذخیره کنیم. هر عنصر لیست اطلاعات مربوط به یک دانشجو را باید نگه دارد. در واقع هر عنصر لیست خود یک لیست با سه مولفه خواهد بود (برای شماره دانشجویی، نام، نام خانوادگی)

```
students = []
file = open ("data.txt","r")
for line in file:
     record =line.split(",")
     students.append(record)

print(students)
```

در کد بالا از متد split برای جداسازی بخشهای یک رشته استفاده شده است. در اینجا line یک رشته است که هر بار حاوی یک خط از فایل است. متد split در اینجا با آرگومان "،" فراخوانی شده است. این بدین معنی است که از ویرگول برای معیار جداکردن قسمتهای رشته استفاده کن. متد split بدون ذکر آرگومان کاراکتر فاصله space یا کاراکتر tab را به عنوان معیار جداکردن در نظر می‌گیرد.
حاصل split یک لیست از رشته‌هاست. هر بار رکورد مربوط به یک دانشجو به انتهای لیست students اضافه می‌شود.
خروجی کد بالا در زیر نشان داده شده است.

```
[['96234521',  'mohammad', 'karimi'],
['97232313',  'zahra sadat',  'hosseini'],
['97239283',  'farhad', 'amiri'], ['96343243',  'samira', 'jalili']]
```

۳.۱ بستن فایل

هنگامی که کار با فایلی تمام شد (اطلاعات مورد نظر خوانده شد و یا نوشته شد) فایل باز شده باید بسته شود. این کار را با اجرای متد close انجام می‌شود.

```
file = open ("data.txt","r")
for line in file:
     print(line)

file.close()
```

۴.۱ نوشتن در فایل

قطعه کد زیر فایل data.txt را برای صرفا نوشتن باز می‌کند. اگر فایل data.txt از قبل موجود باشد، محتوای قبلی فایل از دست می‌رود و پاک می‌شود. اگر فایل data.txt وجود نداشته باشد، یک فایل به این اسم در پوشه مربوطه ایجاد می‌شود. بعد از باز شدن رشته hello در فایل نوشته می‌شود.

```
file = open ("data.txt","w")
file.write("hello")
file.close()
```

اگر بخواهیم فایل باز شود و محتوای جدید به انتهای فایل اضافه شود (بدون اینکه محتوای قبلی از دست برود) می‌توانیم فایل را در حالت ضمیمه append باز کنیم. در کد زیر کلمه hello در انتهای فایل data.txt درج می‌شود.

```
file = open ("data.txt","a")
file.write("hello")
file.close()
```

۲ کار با ماتریسها: مثالی از خواندن و نوشتن در فایل

با ماتریسها در ریاضیات دبیرستانی آشنا شده‌اید. یک ماتریس $m \times n$ مانند جدولی از m سطر است که هر سطر n خانه دارد. اگر A یک ماتریس $m \times n$ باشد درایه (یا عنصر) واقع در سطر i و ستون j ام A را با $a_{i,j}$ نمایش می‌دهیم. در زیر یک نمونه ماتریس حاوی اعداد نشان داده شده است.

$$A = \begin{bmatrix} 1 & 2 & 2 & 2 \\ 2 & 4 & 6 & 8 \\ 3 & 6 & 8 & 10 \end{bmatrix}$$

فرض کنید می‌خواهیم ماتریس A را در یک ساختارداده پایتون نگه‌داری کنیم. می‌توانیم برای اینکار از لیست استفاده کنیم. برای مثال یک لیست با 12 عنصر عددی. اما بهتر است که از یک لیست دو بعدی استفاده کنیم. لیستی که خودش حاوی سه لیست است. در اینجا هر عنصر لیست محتوای یک سطر را نگه‌داری می‌کند.

```
a = [[1,2,2,2] , [2,4,6,8] , [3,6,8,10]]
```

مزیت این کار این است که می‌توان به درایه واقع در سطر i ام و ستون j ام ماتریس با عبارت a[i][j] دسترسی پیدا کرد. دقت کنید که در پایتون اندیسها از صفر شروع می‌شوند.

```
a = [[1,2,2,2] , [2,4,6,8] , [3,6,8,10]]
prints(a[0][0])  # prints 1
prints(a[1][2])  # prints 6
```

می‌خواهیم برنامه‌ای بنویسیم که ماتریس A را از فایل a.txt و ماتریس B را از فایل b.txt بخواند و حاصلجمع ماتریس $C = A + B$ را محاسبه کرده و در فایل c.txt بنویسد. روشن است برای اینکه عمل جمع ماتریسی انجام شود، A و B باید ابعاد یکسان داشته باشند. فرض کنید ماتریس B بصورت زیر داده شده است.

$$B = \begin{bmatrix} 0 & -1 & 2 & 5 \\ 3 & 7 & -1 & 5 \\ 0 & 0 & 4 & -9 \end{bmatrix}$$

محتوای فایل a.txt بصورت زیر است.

```
1 2 2 2
2 4 6 8
3 6 8 10
```

محتوای فایل b.txt بصورت زیر است.

```
0 -1 2 5
3 7 -1 5
0 0 4 -9
```

اول یک قطعه کد می‌نویسیم که فایل a.txt را بخواند و محتوای آن را در لیست دو بعدی a ذخیره کند.

```
a = []
file = open("a.txt","r")
for line in file:
    row = line.split()
    for i in range(len(row)):
        row[i] = float(row[i])
    a.append(row)
```

یک قطعه کد هم می‌نویسیم که فایل b.txt را بخواند و محتوای آن را در لیست دو بعدی b بریزد.

```
b = []
file = open("b.txt","r")
for line in file:
    row = line.split()
    for i in range(len(row)):
        row[i] = float(row[i])
    b.append(row)
```

حال باید لیست دو بعدی c را محاسبه کنیم که حاصلجمع دو ماتریس A و B را نگه میدارد. اول لیست دوبعدی c را تعریف می‌کنیم. این لیست حاوی سه عنصر است و هر عنصر یک لیست با چهار عدد است. محتوای c کاملا با صفر پر شده است. دقت کنید که $len(a)$ تعداد سطرهای ماتریس A است و $len(a[0])$ تعداد ستونهای ماتریس را بدست می‌دهد. عبارت $[0] * k$ یک لیست با k صفر تولید می‌کند.

```
c = []
for i in range(len(a)):
     c.append([0]*len(a[0]))
```

کد بالا لیست دو بعدی زیر را تولید می‌کند.

```
c = [[0,0,0,0] , [0,0,0,0] , [0,0,0,0]]
```

حال می‌توانیم حاصلجمع b و a را در c بریزیم.

```
for i in range(len(a)):
    for j in range(len(a[0])):
         c[i][j] = a[i][j] + b[i][j]
```

حال که ماتریس C بدست آمده است. می‌توانیم محتوای آن را در فایل c.txt بریزیم. در اینجا متغیر s از نوع رشته است و هر بار یک سطر ماتریس را در خود نگه میدارد. به انتهای s کاراکتر سرخط را چسپانده‌ایم.

```
file = open("c.txt","w")
for i in range(len(a)):
    s = ""
    for j in range(len(a[0])):
        s = s + str(c[i][j]) + " "
    s = s + "\n"
    file.write(s)
```