دوره جامع پایتون:

بخش تسلط بر کدنویسی به زبان پایتون

جلسه هشتم

دکتر ذبیح اله ذبیحی

ماژول عبارات با قاعده Regular Expression

```
import  re
```

توابع

- تابع findall() نتیجه دستور داخل آرگومان را بر می گرداند
- تابع search() جستجو می کند آیا عبارت داخل ارگومان برقرار است یا نه؟(تطبیق الگو)
- تابع split() هر جایی از نتیجه داخل آرگومان قطع شده باشد یک لیست برمی گرداند.
- تابع sub() یک یا چند نتیجه را با یک رشته replace (عوض کردن) می کند.

کاراکترها

- [] به معنای نماینده ی مجموعه ای از کاراکترها است. به طور مثال [a-k] یعنی تمام حروف الفبای انگلیسی بین حرف a و k البته از آنجا که آن ها را با حرف کوچک نوشتیم، فقط به دنبال حروف کوچک خواهد گشت.
- [axz] یعنی یکی از کاراکترهای a یا r یا z یا همه آنها در رشته حضور داشته باشند.
- [a-x] یعنی هر کاراکتری بین a و x (بر اساس الفبای انگلیسی) باشد؛ البته فقط حروف کوچک.
- [^axz] یعنی هر کاراکتری به جز a و x و z
- [567] یعنی یکی از این اعداد در رشته باشد
- [3-8] یعنی یکی از اعداد بین ۳ تا ۸ در رشته باشد.

- [0-9][0-5] یعنی هر عدد دو رقمی بین ۰۰ و ۵۹
- [a-zA-Z] یعنی هر حرفی بین aتا z، چه با حروف بزرگ و چه با حروف کوچک
- [+] تمام کاراکترهای خاص (مانند + و * و . و | و () و $ و {} و ...) هیچ معنای خاصی در setها ندارند.

مثال

```
import re
pas="zabihi435678"
x=re.search("[xyz]",pas)
if  x:
    print("ok")
else:
    print("try again")
```

مثال

```
import re
pas="zabihi435678"
x=re.search("[90]",pas)
if  x:
    print("ok")
else:
    print("try again")
```

مثال

```
import re
pas="zabihi435678"
x=re.search("[^90]",pas)
if  x:
    print("ok")
else:
    print("try again")
```

مثال

```
import re
pas="zabihi435678"
x=re.search("[+]",pas)
if  x:
    print("ok")
else:
    print("try again")
```

- . نشان دهنده ی هر کاراکتری که باشد (به غیر از کاراکتر new line – برای رفتن به خط بعد). به طور مثال he..o یعنی رشته ای که با heشروع شود و سپس ۲ کاراکتر داشته باشد (دو نقطه) و در آخر با oتمام شود. علامت نقطه اصلا برایش مهم نیست چه نوع کاراکتری است، فقط باید وجود داشته باشد (عدد، حرف و ...)

مثال

```
import re
pas="zabihi435678"
x=re.search("za..h",pas)
if  x:
    print("ok")
else:
    print("try again")
```

مثال

```
import re
pas="zabihi435678"
x=re.search("za..i",pas)
if  x:
    print("ok")
else:
    print("try again")
```

مثال

```
import re
pas="zabihi435678"
x=re.search(“zabihi",pas)
if  x:
    print("ok")
else:
    print("try again")
```

مثال

```
import re
pas="zabihi435678"
x=re.search(“43",pas)
if  x:
    print("ok")
else:
    print("try again")
```

- ^ نشان دهنده ی شروع شدن با چیزی. مثلا "^hi" یعنی رشته ی ما باید با عبارت hi شروع شود ولی بقیه ی آن مهم نیست.
- $ نشان دهنده ی پایان یافتن با چیزی. به طور مثال "world$" یعنی رشته ی ما باید با worldتمام شود ولی قبل از آن مهم نیست.
- * نشان دهنده ی این است که فلان قسمت هیچ بار (صفر) یا بیشتر تکرار شود. به طور مثال "zab*" یعنی رشته ی ما دارای zaباشد که پس از آن صفر یا چند بار حرف b وجود داشته باشد؛ مثل zabیا zabbیا zabbbیا zaیا zabbbbbbbbbbو الی آخر.
- + نشان دهنده ی این است که فلان قسمت یک بار یا بیشتر تکرار شود. به طور مثال "zab+" یعنی رشته ی ما دارای zaباشد که پس از آن **حداقل یک بار** حرف bوجود داشته باشد؛ مثل zabیا zabbیا zabbbbbbbbو الی آخر (در این حالت za قبول نیست).

مثال

```
import re
pas="zabihi435678"
x=re.search("^za",pas)
if  x:
    print("ok")
else:
    print("try again")
```

مثال

```
import re
pas="zabihi435678"
x=re.search("^43",pas)
if  x:
    print("ok")
else:
    print("try again")
```

مثال

```
import re
pas="zabihi435678"
x=re.search("za$",pas)
if  x:
    print("ok")
else:
    print("try again")
```

مثال

```
import re
pas="zabihi435678"
x=re.search("78$",pas)
if  x:
    print("ok")
else:
    print("try again")
```

- {} تکرار مقداری دقیقا به تعداد ذکر شده باشد. به طور مثال "za{2}" یعنی در رشته ی ما پس از حرف aدقیقا ۲ بار حرف انگلیسی l آمده باشد، نه کمتر و نه بیشتر.
- l یا xباشد یا yبه طور مثال " "falls|staysیعنی رشته ی ما یا دارای Fallsباشد یا دارای staysباشد.

مثال

```
import re
pas="zabihi435678"
x=re.search("za{2}",pas)
if  x:
    print("ok")
else:
    print("try again")
```

مثال

```
import re
pas="zaabihi435678"
x=re.search("za{2}",pas)
if  x:
    print("ok")
else:
    print("try again")
```

- \A این کاراکتر خاص می گوید کاراکترهای پس از آن باید در شروع رشته باشند. به طور مثال "\Azabihi" یعنی رشته باید با zabihi شروع شود.
- \b این کاراکتر می گوید کاراکترهای بعد از آن باید یا در ابتدای **کلمه** یا در انتهای آن باشند.
- \B این کاراکتر می گوید کاراکترهای بعد از آن باید در کلمه وجود داشته باشند اما در انتها یا ابتدای آن **نباشند**.

- \d یعنی رشته باید دارای عدد باشد
- \D یعنی رشته باید دارای کاراکتر غیرعددی باشد.
- \s یعنی رشته باید دارای فضای خالی (اسپیس) **باشد**
- \S یعنی رشته باید دارای کاراکتر غیر فضای خالی (اسپیس) **باشد**
- \w یعنی رشته باید دارای حروف (از a تا z) ویا اعداد (۰ تا ۹) ویا علامت آندرلاین (_) باشد.
- \W یعنی رشته باید دارای کاراکتری غیر از حروف (از a تا z) و اعداد (۰ تا ۹) و علامت آندرلاین (_) باشد.

مثال

```
import re
pas="zabihi435678"
x=re.search("\W",pas)
if  x:
    print("ok")
else:
    print("try again")
```

مثال

```
import re
pas="zabihi435678"
x=re.search("\s",pas)
if  x:
    print("ok")
else:
    print("try again")
```

- \Z یعنی کاراکترهای مشخص شده بعد از خودش در انتهای **رشته** باشند. به طور مثال اگر رشته good day باشد و عبارت جست و جوی ما هم "day\Z" باشد، مقدار صحیح خواهد بود.

مثال

```
import re
pas="zabihi435678"
x=re.search("78\Z",pas)
if  x:
    print("ok")
else:
    print("try again")
```

مثال

```
import re
pas="zabihi435678"
x=re.search("\Aza",pas)
if  x:
    print("ok")
else:
    print("try again")
```

تابع findall()

```
import re
pas="zabihi435678"
x=re.findall("\d",pas)
print(x)
```

نتیجه:

[‘4’,’3’,’5’,’6’,’7’,’8’]

```
import re
pas="zabihi435678"
x=re.findall("\D",pas)
print(x)
```

پاسخ

```
['z', 'a', 'b', 'i', 'h', 'i']
```

تابع split()

```
import re
pas="zabihi435678"
x=re.split("\d",pas)
print(x)
```

نتیجه

```
['zabihi', '', '', '', '', '', '']
```

مثال

```
import re
pas="zabihi435678"
x=re.split("\D",pas)
print(x)
```

نتیجه

```
['۴۳۵۶۷۸' ," ," ," ," ," ," ," ,"]
```

تابع sub()

```
import re
pas="zabihi435678"
x=re.sub("\w","z",pas)
print(x)
```

نتیجه

```
zzzzzzzzzzz
```

```
import re
pas="zabihi435678"
x=re.sub("\W","z",pas)
print(x)
```

نتیجه

```
zabihi435678
```

ماژول re با عبارات منظم (Regular Expressions)

- re.search
  - از این تابع برای بررسی این که آیا یک الگو خاص در یک عبارت وجود دارد یا نه؟ و در صورت موجود بودن، اولین الگوی پیدا شده رو برمیگردونه
- re.findall
  - از این تابع برای پیدا کردن تمام الگوها در یک عبارت استفاده میشه. خروجی و حاصل این تابع یک لیست میباشد
- re.split
  - از این تابع برای مجزا کردن عبارات، بر اساس یک جدا کننده، استفاده میشه. مثلا مجزا کردن عبارت مورد نظر بر اساس "نقطه"، شماره تلفن، ایمیل و ...
- re.sub
  - از این تابع برای جایگزین کردن یک الگو با یک الگوی دیگر در عبارت داده شده، استفاده میشه.

مثال: ساخت پسوردی با ویژگی های زیر

- At least 1 letter between [a-z] and 1 letter between [A-Z].
- At least 1 number between [0-9].
- At least 1 character from [$#@].
- Minimum length 6 characters.
- Maximum length 16 characters.

```
import re
p= input("Input your password")
x = True
while x:
    if (len(p)<6 or len(p)>12):
        break
    elif not re.search("[a-z]",p):
        break
    elif not re.search("[0-9]",p):
        break
    elif not re.search("[A-Z]",p):
        break
    elif not re.search("[$#@]",p):
        break
    elif re.search("\s",p):
        break
    else:
        print("Valid Password")
        x=False
        break

if x:
    print("Not a Valid Password")
```

کار با فایل

```
file_name=“test.txt”
open(file_name,mode_file)
```

- تابع ()open دو آرگومان ورودی می‌گیرد؛ اولی مسیر و نام فایل است و دومین آرگومان مشخص کننده حالت باز شونده فایل است.
-

حالت ها یا mode

- حالت r که مخفف Readو به معنی خواندن است. این مقدار، حالت پیش فرض دسترسی شما می باشد و کارش خواندن اطلاعات از روی فایل است. در صورتی که فایل درخواست شده وجود نداشته باشد یک خطا برمی گرداند.
- حالت a که مخفف Appendو به معنی ضمیمه کردن است. کار این تابع باز کردن یک فایل برای ضمیمه کردن فایل/مقدار دیگری به آن است و چنانچه فایل درخواست داده شده وجود نداشته باشد، یک فایل جدید به همان نام می سازد.
- حالت wکه مخفف write و به معنی نوشتن است. کار این تابع باز کردن فایل برای نوشتن و ایجاد تغییرات در آن است و چنانچه فایل درخواست داده شده وجود نداشته باشد، یک فایل جدید به همان نام می سازد.
- حالت x که نماینده ی createبه معنی ایجاد است. کار این تابع ایجاد کردن یک فایل جدید است و در صورتی که آن فایل از قبل وجود داشته باشد به شما خطایی برمی گرداند.
- tبه معنی دسترسی به صورت متنی است که حالت پیش فرض می باشد.
- bبه معنی دسترسی به صورت باینری (دودویی) است.

حالت های مختلف باز کردن فایل

- rخواندن (شروع از ابتدای فایل)
- +rخواندن و نوشتن (شروع از ابتدای فایل)
- wایجاد فایل برای نوشتن (شروع از ابتدا)
- +wایجاد فایل برای نوشتن و خواندن (شروع از ابتدا)
- aخواندن (شروع از انتها):
- +aخواندن و نوشتن (شروع از انتها)

- در دو حالت w و w+ در صورتی که فایلی از قبل وجود داشته باشد، فایل را پاک کرده و یک فایل جدید ایجاد می‌کند.
- در دو حالت a و a+ در صورتی که فایلی وجود داشته باشد، آن‌را باز کرده و از انتهای آن شروع به انجام عملیات می‌کند و اگر فایلی وجود نداشته باشد، یک فایل خالی ایجاد می‌کند.
- اگر تابع open() برای فایلی که وجود ندارد فراخوانی کنیم، آن فایل را ایجاد کرده . میتوانیم اطلاعات را در آن ذخیره کنیم.
- در هنگام فراخوانی open() اگر مد مشخص نشود، بصورت پیش فرض مد را r در نظر می گیرد.

مثال

```
open("a.txt","r")
```

خواندن فایل به read()

- این تابع که می‌بایست روی متغیر شی فایل صدا زده شود، یک رشته متنی بزرگ برمی‌گرداند که حاوی تمام محتویات فایل است.

- اگر تابع read() را دو بار پشت سر هم صدا بزنیم، در دفعه دوم، هیچ مقداری برای ما چاپ نمی‌شود! به دلیل اینکه تمام فایل تا انتها بررسی و خوانده شده و دیگر محتوای بیشتری برای نمایش نداریم.
- تابع read() می‌تواند یک آرگومان ورودی به صورت دلخواه داشته باشد؛ این آرگومان از نوع عددی بوده و مشخص می‌کند که چه تعداد کاراکتر خوانده شود.
- مقدار 1- به معنای کل فایل است.

```
name="a.txt"
name2=open(name,"r")
print(name2.read(4))
```

```
name1= "a.txt"
file=open(name1,"r")
for x in file:
  print(x)
file.close()
```

تابع readlineبرای خواندن فایل

- این تابع در هر بار صدا زده شدن روی فایل، یک خط از فایل را به ما بر می‌گرداند. یعنی می‌توان آن را به تعداد خطوط موجود در فایل (با احتساب خطوط خالی) فراخوانی کنیم.

فایل a را بصورت یک فایل txt در نظر بگیرید که در سطر اول آن عدد ۱ ، سطر دوم عدد ۲ .....، سطر نهم عدد 9 تایپ شده باشد.

```
name="a.txt"
name2=open(name,"r")
print(name2.readline())
print(name2.readline())
print(name2.readline())
```

```
name="a.txt"
name2=open(name,"r")
For I in range(8)
   print(name2.readline())
```

- فایل a را بصورت txt در نظر بگیرید که اعداد 1 تا 9 پشت سرهم و بدون هیچ فاصله در سطر اول قرار گرفته باشند. میانگین یک تا نه را می خواهیم بدست بیاوریم.

```
f1=open("a.txt","r")
s=f1.read(-1)
n=len(s)
p=0
for i in s:
   ss=int(i)
   print(i)
   p=p+ss
mean=p/n
print(n,p,mean)
```

فایل a.txt شامل ۶ سطر شامل داده های ۱۲، ۲۴، ۳۵، ۷۲، ۲۲،۴۶

```
name="a.txt"
name2=open(name,"r")
n=6
s=0
for i in range(n):
  x=name2.readline(-)
  if x not in " ":
    x=int(x)
    print(x)
    s=s+x
print("-------------------")
mean=s/n
print("mean=",mean)
```

```
f1=open("a.txt","r")
t=f1.readlines()
print(t)
n=len(t)
f1.close()
f2=open("a.txt","r")
s=0
for i in range(n):
  x=f2.readline()
  if x not in " ":
    x=float(x)
    print(x)
    s=s+x
print("----------")
print("s=",s)
mean=s/n
print("mean=",mean)
f2.close()
```

تابع readlines

- هنگامی که تابع readlines() را روی فایل استفاده می‌کنیم، به عنوان خروجی تابع، یک لیست داریم که شامل خطوط موجود در فایل هست. یعنی هر عضو این لیست معادل یک خط در درون فایل خواهد بود

```
name="a.txt"
name2=open(name,"r")
print(name2.readlines())
```

در خروجی هر سطر از فایل a را بصورت یک عضو (بصورت رشته) از لیست چاپ می کند.

حلقه برای خواندن فایل

- یک روش ساده برای خواندن خط به خط یک **فایل در پایتون**، استفاده از حلقه for و یا for each است

بستن فایل پس از کار

- پس ازاتمام کار با فایل باز شده، با صدا زدن تابع close() فایل را می بندیم.
- پس از اتمام اجرای برنامه پایتون، فایل به صورت خودکار بسته می‌شود، اما استفاده از تابع close() به آزاد شدن زودتر منابع کمک خواهد کرد

```
name="a.txt"
name2=open(name,"r")
print(name2.readlines())
name2.close()
```

نوشتن یا ذخیره اطلاعات در فایل با تابع write

- تابع ()write یک رشته را به عنوان ورودی گرفته و آن را درون فایل می‌نویسد.
-

```
f1=open("a.txt","r+")
s="zabiholah zabihi"
f1.write(s)
F1.close()
```

اطلاعات را جایگزین کاراکترهای قبلی از ابتدای فایل می کند. و مابقی کارکترهای قبلی موجود در فایل را نگه می دارد.

```
f1=open("a.txt",“w+")
s="zabiholah zabihi"
f1.write(s)
F1.close()
```

اطلاعات قبلی موجود در فایل را پاک می کند و اطلاعات جدید را ذخیره می کند.

```
f1=open("a.txt",“a+")
s="zabiholah zabihi"
f1.write(s)
f1.close()
```

اطلاعات قبلی موجود در فایل را نگه می دارد و اطلاعات جدید را به انتهای اطلاعات قبلی اضافه می کند.

ذخیره یا نوشتن اطلاعات در فایل با تابع writelines

- تابع writelines() یک لیست از رشته‌های متنی گرفته و آن ها را به ترتیب بصورت یک رشته منفرد درون فایل مورد نظر می‌نویسد.
-

```
f1=open(“a.txt","a+")
s=["zabiholah", "zabihi","32355433"]
f1.writelines(s)
f1.close()
-----------------------------------------------
f1=open(“a.txt","a+")
s=["zabiholah\n", "zabihi\n","32355433"]
f1.writelines(s)
f1.close()
```

دستور ”\n“ نقش enter را ایفا میکند.

اضافه کردن محتوا به محتوای موجود در فایل پایتون

- همان‌طور که در هنگام معرفی حالت باز کردن یک فایل گفته شد، اگر فایلی که از قبل وجود داشته را در حالت w باز کنیم، تمام محتویات قبلی حذف شده و فایل آماده نوشته شدن محتوای جدید می‌شود. در صورتی که بخواهیم به محتوای فعلی یک فایل، محتوای جدیدی اضافه کنیم، می‌بایست فایل را در حالت a ابتدای کلمه (append) باز کرده و سپس عملیات نوشتن در فایل را انجام دهیم.

حذف فایل

- برای حذف کردن یک فایل در زبان پایتون، نیاز به یک ماژول سیستمی داریم. این ماژول یا کتابخانه به طور پیش‌فرض در پایتون وجود دارد و می‌توان با اضافه کردن ماژولی به نام os آن را به برنامه خودمان اضافه کنیم. با تابع remove() در ماژول os می توان با دادن نام فایل به عنوان ورودی، فایل مورد نظر حذف می‌شود.

دایرکتوری

تا الان فایل کد و فایلی که قرار بود از ان داده فراخوانی شود یا ذخیره شود در یک پوشه بودند. برای فراخوانی فایل از پوشه یا درایو دیگر بصور زیر عمل می کنیم

```
f1=open(“C:\\NEW_Folder_name\a.txt","a+")
s=["zabiholah", "zabihi","32355433"]
f1.writelines(s)
f1.close()
```

تمرین

- تمام مثال ها را با یک پسورد انتخابی خودتان انجام دهید.

تمرین

- فایل a را بصورت txt در نظر بگیرید که اعداد 10 تا 20 پشت سرهم و بدون هیچ فاصله در سطر اول قرار گرفته باشند. میانگین ده تا بیست را می خواهیم بدست بیاوریم.