آموزش رایگان پایتون (python)- نحوه وارد کردن ماژول‌ها در برنامه، کار با رشته‌ها

در شماره‌های گذشته آموزش پایتون با اصول و مفاهیم ابتدایی پایتون آشنا شدید. در سه شماره پایانی آموزش مقدماتی پایتون قصد داریم به شکل فشرده مباحث دیگر مرتبط با پایتون را بررسی کنیم.

برای مطالعه بخش بیستم و دوم آموزش رایگان پایتون اینجا کلیک کنید

ساخت برنامه‌هایی که حاوی میلیون‌ها خط کد باشند، تا حدودی رایج است. حال تصور کنید باید با فایلی که حاوی میلیون‌ها خط کد بوده و یک باگ درون آن قرار دارد کار کنید. بدون شک کار طاقت‌فرسایی است. **پایتون** به شما اجازه می‌دهد برای سازمان‌دهی بهتر برنامه‌های خود، کدهای یک برنامه را درون قطعات کوچکی قرار دهید تا مدیریت آن‌ها ساده باشد. راه‌حل پیشنهادی **پایتون** در مفهومی به نام ماژول‌ها مستتر شده است. ماژول‌های معمولی که شامل کدهای منبعی هستند که به شکل عمومی از آن‌ها استفاده می‌شود کتابخانه نام دارند. ماژول‌ها درون فایل‌های جداگانه‌ای قرار می‌گیرند. برای استفاده از ماژول‌ها، باید برای **پایتون** مشخص کنید که باید فایل موردنظر شما را خوانده و آن‌را درون برنامه کاربردی شما قرار دهد. فرآیند خواندن کدهایی که درون یک فایل خارجی قرار دارد وارد کردن (import) نام دارد.

وارد کردن ماژول‌ها در برنامه

برای آن‌که بتوانید از یک ماژول استفاده کنید، ابتدا باید آن‌را وارد کنید. دو راه برای وارد کردن ماژول‌ها درون یک برنامه کاربردی وجود دارد که هر یک تکنیک خاص خود را دارند.

Import: شما از دستور import زمانی که قصد دارید یک ماژول را به‌طور کامل درون برنامه کاربردی خود وارد کنید استفاده می‌کنید. این روش رایج‌ترین تکنیکی است که توسعه‌دهندگان برای صرفه‌جویی در زمان و تنها با نوشتن یک خط کد از آن استفاده می‌کنند. این روش منابع حافظه بیشتری از یک سیستم طلب می‌کند. دقت کنید در زمان فراخوانی دستور import باید در ابتدای برنامه قرار گیرد.

From…import: شما می‌توانید از دستور from…import زمانی که قصد دارید یک خصلت ویژه درون یک ماژول را انتخاب کنید استفاده می‌کنید. این متد باعث صرفه‌جویی در منابع شده اما پیچیدگی خاص خود را دارد و اگر سعی

کنید از خصلتی در برنامه خود استفاده کنید که آن‌را وارد نکرده باشید، **پایتون** پیغام خطایی نشان می‌دهد.

به‌کارگیری فرمان import

برای آشنایی با نحوه عملکرد فرمان import مراحل زیر را انجام دهید.

1. IDLE را باز کرده و از منوی File گزینه New File را انتخاب کنید.

2. در پنجره جدید دستورات زیر را وارد کنید.

```
import random

import math

for i in range(10):

    print(random.randint(1, 25))
```

در قطعه کد بالا ما ماژول random که برای تولید مجموعه‌ای از اعداد تصادفی استفاده می‌شود را درون برنامه خود وارد کرده‌ایم. در ادامه با فراخوانی تابع randint که درون ماژول فوق قرار دارد برای تولید اعداد تصادفی استفاده کرده‌ایم. در فرمان import دوم ماژول math را فراخوانی کردیم. درون این ماژول تابعی به نام pi وجود دارد که مقدار عدد پی را نشان می‌دهد.

3. برنامه را ذخیره کرده و از منوی run گزینه Rum Module را انتخاب کنید. خروجی این برنامه همانند تصویر زیر است:

```
========== RESTART: C:\Node\myimport.py =======
12
8
21
24
13
1
17
19
5
18
3.141592653589793
>>>
```

4. اکنون دو فرمان import که ابتدای برنامه قرار گرفته‌اند را پاک کرده و برنامه را دومرتبه اجرا کنید. پیغام خطایی را با این مضموم که توابع فوق تعریف نشده‌اند را دریافت خواهید کرد.

فراخوانی ماژول‌ها با دستور from...import

همان‌گونه که گفتیم این ترکیب نحوی باعث صرفه‌جویی در منابع یک سیستم شده، اما به‌کارگیری آن کمی پیچیده است. برای روشن شدن بحث به مثال زیر دقت کنید.

1. IDLE را باز کرده و از منوی File گزینه New File را انتخاب کنید.

2. دستورات زیر را درون پنجره جدید وارد کنید.

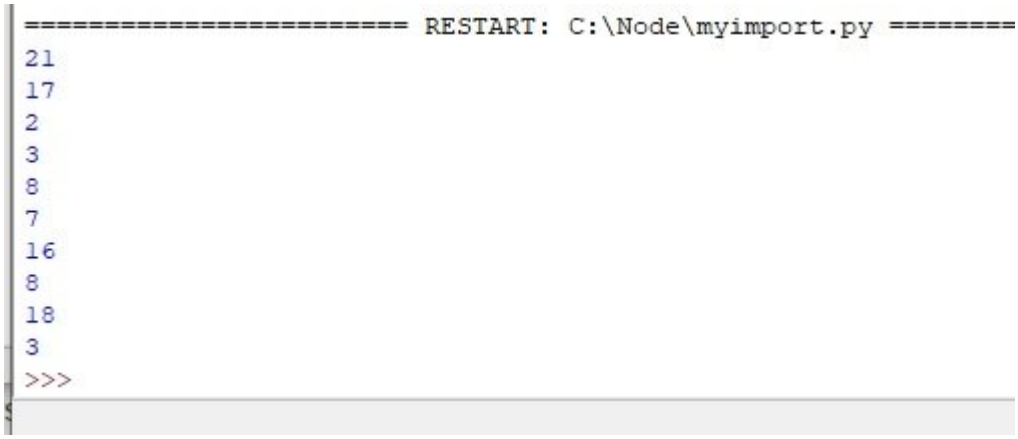
```
from random import randint

for i in range(10):

    print(randint(1, 25))
```

در قطعه کد بالا ما ابتدا کلمه کلیدی from و سپس تابع random را فراخوانی کردیم. در ادامه به سراغ کلمه کلید import رفته و نام تابع خاصی که قصد استفاده از آن‌را داریم را مشخص کردیم. زمانی که خصلت‌ها و توابع درون یک ماژول را به این شکل فراخوانی می‌کنید در زمان به‌کارگیری توابع کافی است تنها نام تابع را بنویسیم. به عبارت دیگر به جای آن‌که همانند قطعه کد قبل از عبارت random.randint() استفاده کنیم تنها از نام تابع randint استفاده کردیم.

3. برنامه را ذخیره کرده و از منوی Run گزینه New Module را اجرا کنید. خروجی این برنامه همانند تصویر زیر است.



پیدا کردن یک ماژول روی دیسک

برای آن‌که بتوانید از کدهایی که درون ماژول‌ها قرار دارد استفاده کنید، در ابتدا باید برای **پایتون** مشخص کنید در چه مکانی به دنبال ماژول باشد. این اطلاعات مکانی درون مسیرهای **پایتون** ذخیره شده‌اند. زمانی که از **پایتون** درخواست می‌کنید تا یک ماژول را وارد کند، **پایتون** ابتدا به فایل‌هایی که درون مسیر از پیش تعیین شده قرار دارد نگاه می‌کند. اطلاعات مسیر از یکی از سه منبع متغیرهای محیطی همچون pythonpath که به **پایتون** اعلام می‌دارند ماژول‌ها در چه مکانی ذخیره شده‌اند، پوشه جاری برنامه که ممکن است ماژول‌های خود را درون پوشه جاری ذخیره کرده باشید و پوشه‌های پیش‌فرض که زمانی که هیچ متغیر محیطی را تعریف نکرده‌اید و پوشه جاری حاوی ماژول‌های مدنظر نیست، استفاده خواهد شد.

ایده خوبی است که درباره مسیر فعلی برنامه خود اطلاعاتی به دست آورید، زیرا عدم آگاهی از یک مسیر ممکن است برنامه را با مشکل روبرو کند. مراحل زیر به شما نشان می‌دهد که چگونه اطلاعات مسیرها را به دست آورید.

1. IDLE را باز کنید.

2. در پنجره اصلی دستور زیر را تایپ کرده و کلید اینتر را فشار دهید.

```
Import sys
```

3. فرمان زیر را تایپ کرده و کلید اینتر را فشار دهید.

```
For p in sys.path:
```

4. برای چاپ مسیرهای اطلاعاتی از فرمان print زیر استفاده کنید.

Print(p)

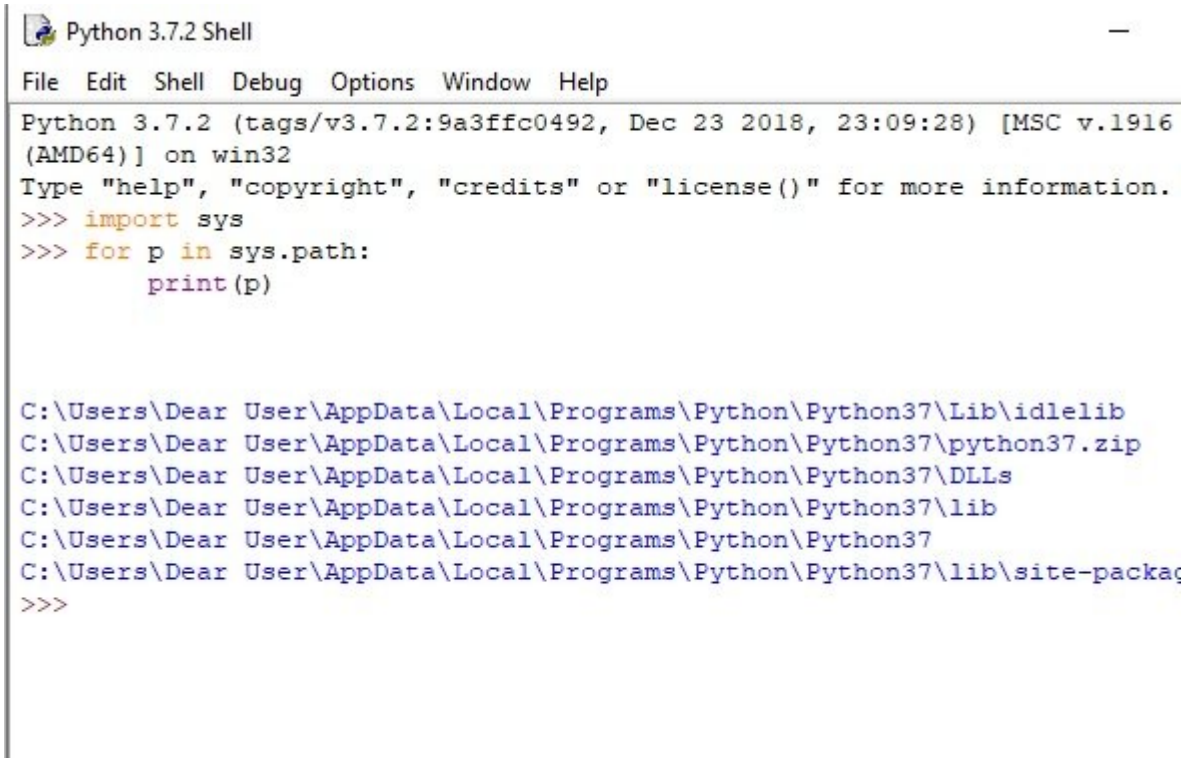
5. با فشار کلید اینتر مسیرهایی که **پایتون** از آن ها استفاده می کند را مشاهده می کنید.

خصلت sys.path در بیشتر موارد قابل اعتماد است، اما در برخی از موارد ممکن است همه مسیرهایی که **پایتون** از آن‌ها استفاده می‌کند را نشان ندهند. اگر مسیر پیشنهادی خود را در فهرست بالا مشاهده نمی‌کنید ممکن است این مسیر در مکان دیگری قرار داشته باشد. برای مشاهده مسیرهای احتمالی دیگر مراحل زیر را دنبال کنید:

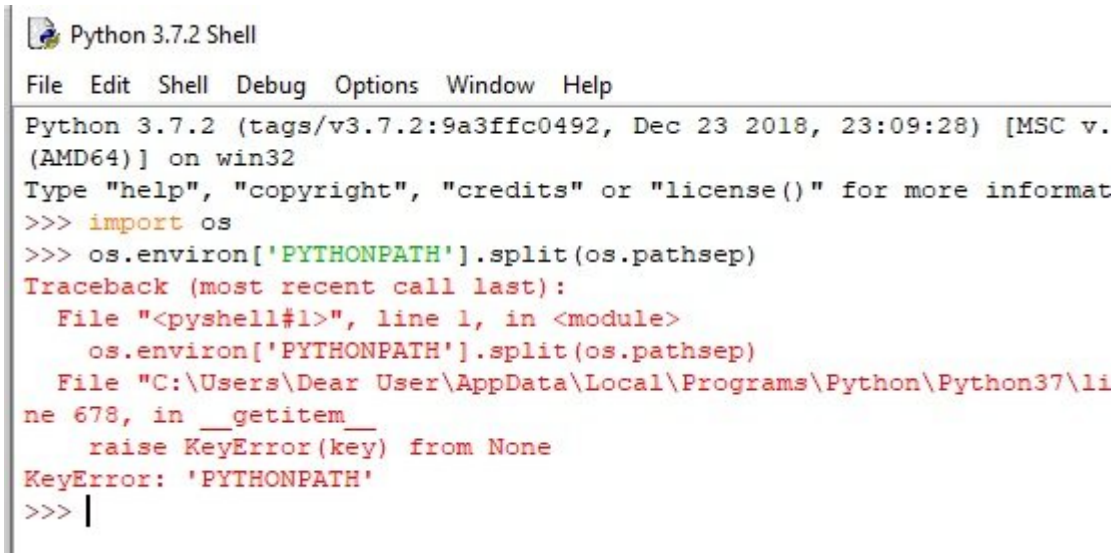
1. import os را تایپ کرده و کلید اینتر را فشار دهید.

2. دستور os.environ['PYTHONPATH'].split(os.pathsep) را تایپ کرده و کلید اینتر را فشار دهید.

زمانی که متغیر محلی PYTHONPATH تعریف شده باشد، شما فهرستی از مسیرهایی که در شکل زیر مشخص شده‌اند را مشاهده می‌کنید.



اما اگر هیچ متغیر محیطی را تعریف نکرده باشید با اجرای فرمان بالا پیغام خطایی همانند شکل زیر مشاهده خواهید کرد.



مشاهده محتوای یک ماژول

پایتون به شما اجازه می‌دهد به روش‌های مختلفی محتوای یک ماژول را مشاهده کنید. متدی که بیشتر توسعه‌دهندگان از آن استفاده می‌کنند تابع dir است. این متد به شما اطلاعات جالبی درباره خصلت‌هایی که یک درون یک ماژول قرار دارند نشان می‌دهد. برای درک بهتر این موضوع به مثال زیر دقت کنید. فرض کنید که نیاز داریم محتوای کتابخانه random را مشاهده کنیم. برای مشاهده محتوای این کتابخانه با استفاده از تابع dir به شکل زیر عمل می‌کنیم.

```
Import random

Print(“The contents of random library are::”)

   Print(dir(random))
```

خروجی قطعه کد بالا همانند شکل زیر است:

ما بحث مربوط به ماژول‌ها را در همین نقطه به پایان می‌رسانیم، البته نکات جالب توجه دیگری نیز در مورد

ماژول‌ها وجود دارد که پیشنهاد می‌کنیم از منابعی همچون https://www.learnpython.org برای کسب اطلاعات بیشتر استفاده کنید.

کار با رشته‌ها در پایتون

همان‌گونه که در شماره‌های قبل به آن اشاره کردیم، کامپیوترها مفهوم رشته‌ها را درک نکرده و تنها اعداد برای آن‌ها قابل فهم هستند. شما روی صفحه‌نمایش یک رشته متنی را مشاهده می‌کنید، در حالی که در پس‌زمینه کامپیوتر با مجموعه‌ای از اعداد در حال کار است. اما برای آن‌که رشته‌ها قابل فهم شوند، باید به شکلی آن‌ها را دستکاری کنید که بدون مشکل قابل خواندن باشند. درست است که یک رشته را مجموعه‌ای از کاراکترها به وجود می‌آورند، اما در پس‌زمینه کاراکترها نیز اعداد قرار دارند. هر کاراکتری یک مقدار عددی مختص خود دارد. به‌طور مثال، کاراکتر A دارای مقدار 65 است، در حالی که کاراکتر a کوچک کد اسکی 97 را دارد. همین مسئله باعث می‌شود تا این دو کاراکتر برای یک کامپیوتر متفاوت از دیگری باشند. برای اجتناب از طولانی شدن بحث اجازه دهید نحوه کار با رشته‌ها را در قالب مثال‌های کوچکی بررسی کنیم.

رشته مجموعه‌ای از کاراکترها

همان‌گونه که اشاره کردیم یک رشته مجموعه‌ای از کاراکترها است. **پایتون** اجازه می‌دهد از علامت کروشه برای دسترسی به کاراکترهایی که درون یک رشته قرار دارد استفاده کنید:

```
Car=”Benz”

Letter=Car[1]
```

در قطعه کد بالا کاراکتر اول متغیر Car انتخاب شده و درون متغیر Letter قرار می‌گیرد. مقداری که درون کروشه بالا قرار می‌گیرد به نام اندیس شناخته می‌شود. اندیسی که به کاراکتر درون یک رشته اشاره دارد. دقت کنید در متغیر Letter کاراکتر e قرار خواهد گرفت، زیرا در یک دنبال کاراکتری مقادیر از 0 شروع می‌شوند. در مثال بالا اگر به جای مقدار 1 مقدار 0 را قرار دهید کاراکتر B درون متغیر Letter قرار خواهد گرفت. نکته مهم دیگری که باید به آن دقت کنید به مقداری باز می‌گردد که برای یک اندیس استفاده می‌شود. مقدار اندیس باید از نوع صحیح (integer) باشد، اگر از یک مقدار اعشاری یا هر نوع داده دیگری استفاده کنید **پایتون** پیغام خطا نشان خواهد داد. به‌طور مثال قطعه کد زیر باعث نشان دادن یک پیغام خطا می‌شود.

```
Letter=Car[1.5]
```

اندازه یک رشته

پایتون تابعی به نام len دارد که این تابع تعداد کاراکترهای دورن یک رشته را باز می‌گرداند.

```
Car=”Benz”

Len(Car)
```

خروجی تابع بالا برابر با مقدار 4 خواهد بود.

به‌طور مثال، زمانی که قصد دارید آخرین حرف یک رشته را به دست آورید از تابع فوق به شکل زیر استفاده کنید:

```
Length= len(Car)

Last=Car[Length-1]
```

به قطعه کد بالا دقت کنید. اگر در مثال بالا شما مقدار 1- را حذف کنید، **پایتون** پیغام خطایی به شما نشان خواهد داد. زیرا همان‌گونه که اشاره کردیم، شروع خواندن با مقدار 0 آغاز می‌شود و در نتیجه **پایتون** اندازه رشته بالا را برابر با 3 کاراکتر تصور می‌کند.

پیمایش یک رشته با حلقه while

بیشتر فعالیت‌های مرتبط با رشته‌ها به پردازش نیاز دارند، زیرا در برخی موارد مجبور می‌شوید با کاراکترهای مشخصی کار کرده و در هر بار کاراکتر خاصی را استفاده کنید. این الگو به نام الگوی پردازش پیمایش شهرت داشته و با حلقه‌های while و for انجام می‌شود. قطعه کد زیر نحوه پیمایش با حلقه while را نشان می‌دهد.

```
index = 0

car="Benz"

while index < len(car):

    Letter=car[index]

    print(Letter)

    index+=1
```

حلقه فوق رشته را پیمایش کرده و کاراکترهای درون رشته را یک‌به‌یک خوانده و درون سطرهای جداگانه‌ای چاپ می‌کند. زمانی که متغیر index برابر با اندازه رشته می‌شود شرط حلقه False شده و اجرای حلقه پایان پیدا می‌کند. البته در این‌گونه موارد از حلقه for استفاده می‌شود.

برش‌زنی یک رشته

فرآیند مشخص کردن یک بخش یا بند از یک رشته به نام برش‌زنی یک رشته معروف است. برش‌زنی در اصل به معنای انتخاب مجموعه‌ای از کاراکترها است. برای روشن شدن بحث به مثال زیر دقت کنید:

```
Mystr="This is a good day"

print(Mystr[0:5])
```

خروجی قطعه کد بالا برابر با واژه this است. در قطعه کد بالا عملگر [n:m] بخشی از یک رشته که با کاراکتر n مشخص شده است را باز می‌گرداند. این بخش شامل اولین برشی است که عدد n آن‌را مشخص کرده، اما کاراکتر m را شامل نمی‌شود. در این ترکیب نحوی اگر از اندیسی که قبل : قرار دارد صرنظر کنید، برش‌زنی از ابتدای رشته آغاز می‌شود و برعکس قضیه اگر از اندیس دوم صرفنظر کنید، برش‌زنی تا انتهای رشته انجام می‌شود. به طور مثال، اگر قطعه کد بالا را به شکل زیر ویرایش کنید، خروجی برابر با is a good day خواهد بود.

```
Mystr="This is a good day"

print(Mystr[5:])
```

رشته‌ها غیرقابل تغییر هستند

زمانی که فرآیند انتساب مقدار رشته‌ای به متغیری انجام می‌شود، دیگر این امکان ندارد تا مقدار کاراکتری که درون یک متغیر ذخیره شده است را تغییر داد. اگر سعی در انجام چنین کاری کنید، پایتون پیغام خطایی به شما نشان خواهد داد. به‌طور مثال، اگر قطعه کد زیر را اجرا کنید پایتون پیغام خطایی نشان خواهد داد.

```
Mystr=”It’s a good Day”

Mystr[0]=’J’
```

در شماره آینده آموزش **پایتون** مبحث رشته‌ها را ادامه خواهیم داد.

تاریخ انتشار:
19 اسفند 1397

نشانی منبع:
https://www.shabakeh-mag.com/workshop/14781/%D8%A2%D9%85%D9%88%D8%B2%D8%B4-%D8%B1%D8%A7%DB%8C%DA%AF%D8%A7%D9%86-%D9%BE%D8%A7%DB%8C%D8%AA%D9%88%D9%86-python-%D9%86%D8%AD%D9%88%D9%87-%D9%88%D8%A7%D8%B1%D8%AF-%DA%A9%D8%B1%D8%AF%D9%86-%D9%85%D8%A7%DA%98%D9%88%D9%84%E2%80%8C%D9%87%D8%A7-%D8%AF%D8%B1-%D8%A8%D8%B1%D9%86%D8%A7%D9%85%D9%87%D8%8C-%DA%A9%D8%A7%D8%B1-%D8%A8%D8%A7-%D8%B1%D8%B4%D8%AA%D9%87%E2%80%8C%D9%87%D8%A7