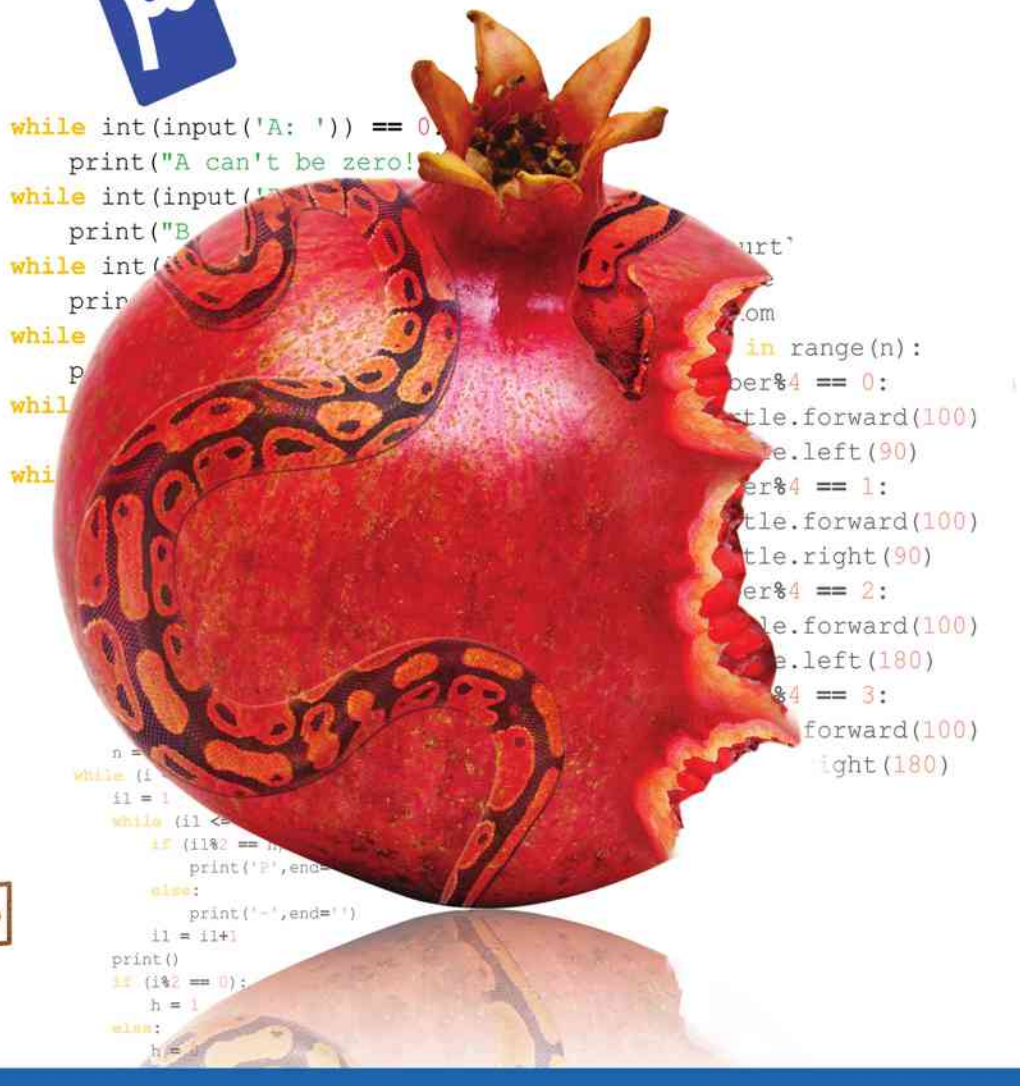
دانش‌آموزان جوان

علامه حلی

برنامه‌نویسی پایتون ۳



ویژه استعدادهای درخشان

امیرعلی عسگری • علی وزیری • محمدرضا جهانگیر

مجموعه کتاب‌های علامه حلی

برنامه‌نویسی پایتون (۳)

- امیرعلی عسگری
- علی وزیری
- محمدرضا جهانگیر

شناسنامه کتاب

سرشناسه : عسگری، امیرعلی، ۱۳۸۶
عنوان و نام پدیدآور : برنامه‌نویسی پایتون(۳)/ امیرعلی عسگری، علی وزیری، محمدرضا جهانگیر.
مشخصات نشر : تهران: انتشارات حلی: دانش‌پژوهان جوان، ۱۴۰۲.
مشخصات ظاهری : ۱۷۶ ص.: تصویر، جدول، نمودار
فروست : مجموعه کتاب‌های علامه‌حلی
شابک : ۹۷۸-۶۰۰-۴۹۶-۲۹۲-۶
وضعیت فهرست نویسی : فیپا
موضوع : پایتون (زبان برنامه‌نویسی کامپیوتر)
موضوع : Python (Computer program language)
شناسه افزوده : وزیری، علی، ۱۳۶۶-
شناسه افزوده : جهانگیر، محمدرضا، ۱۳۵۴-
رده‌بندی کنگره : QA ۷۶/۷۳
رده بندی دیویی : ۰۰۵/۱۳۳
شماره کتابشناسی ملی : ۹۳۰۴۷۲۷

علامه حلی

دانش‌پژوهان جوان

عنوان کتاب : برنامه‌نویسی: پایتون (۳)
ناشر : انتشارات حلی
ناشر همکار : انتشارات دانش‌پژوهان جوان
مؤلف : امیرعلی عسگری، علی وزیری، محمدرضا جهانگیر
مسئول هماهنگی : سمیه‌سادات فاطمی
صفحه‌آرا : راضیه‌سادات فرهانیان
طراح جلد : الهه شرفی
تصویرسازان : محمدحسین صفدریان
سال چاپ : ۱۴۰۲
نوبت چاپ : اول
چاپ و صحافی : واژه‌پرداز اندیشه
شمارگان : ۲۰۰۰ جلد
قیمت : ۱۷۷۰۰۰ تومان
شماره شابک : ۹۷۸-۶۰۰-۴۹۶-۲۹۲-۶

تهران، خیابان انقلاب، میدان فردوسی، ابتدای کوچه براتی، پلاک ۱۶ واحد ۱۴
تلفن دفتر مرکزی: ۹-۶۶۷۴۴۳۸۴

فهرست مطالب

قبل از شروع به مطالعه کتاب، این قسمت را بخوانید:

وقتی شروع به خواندن این کتاب کنید با بخش‌های مختلفی مواجه می‌شوید که غالباً یک لاک‌پشت متفاوت در اول هرکدام وجود دارد. برای هرکدام از این بخش‌ها از شما انتظار داریم کار متفاوتی انجام دهید. این قسمت‌ها بر اساس تئوری‌های نوین آموزش و تجارب موفق تدریس برای آموزش دانش‌آموزان مستعد طراحی شده است. این بخش‌ها شامل:

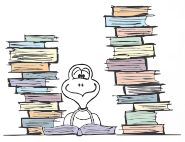
درخت دانش: در صفحه دوم هر فصل، نمودار دایره‌ای شکلی کشیده شده که به ما کمک می‌کند بفهمیم در آن فصل مطالب علمی چطوری تقسیم‌بندی شده و ارتباط آن‌ها با هم چیست. درواقع این بخش نقشه‌ای است برای گم نشدن در موضوعات علمی.

اهداف رفتاری: زیر هر درخت دانش، چند جمله نوشته شده که از اول کار معلوم کند که این فصل را می‌خوانیم که چه بشود. خوب است در آخر فصل هم برگردیم و ببینیم که می‌توانیم کارهایی را که در این بخش گفته انجام دهیم یا نه.

پاسخگو باش: در این قسمت باید پاسخگو باشیم. پاسخگوی سؤالی که پرسیده شده و انتظار می‌رود بعد از خواندن درس تا آن قسمت، بتوانیم باکمی فکر کردن به آن جواب دهیم.

فسفر بسوزان: شاید لازم باشد مقدار بیشتری از مغز خودمان استفاده کنیم و قدری از فسفرهای ذخیره‌شده را بسوزانیم! سؤالاتی که در بخش فسفر بسوزان مطرح می‌شود فقط با خواندن مطالب درسی قابل پاسخگویی نیست و باید کمی بیش از معمول درباره آن‌ها فکر کنیم.

جالب است بدانی: برای افرادی که دوست دارند بیشتر از سطح استاندارد با موضوعات آشنا شوند این قسمت توصیه می‌شود. در این قسمت مطالبی آورده شده که خواندن و یادگرفتن آن الزامی نیست، ولی آن‌قدر جذاب است که نشود به‌راحتی بی‌خیال خواندن آن شد.



لغت‌نامه: ما دانش‌آموزان مستعد و متفاوت (!) دوست داریم بتوانیم علاوه‌بر مطالب درسی، جست‌وجویی هم بکنیم و ببینیم در دنیا درباره موضوع درسی ما چه چیزی وجود دارد، برای همین در پایان هر فصل لغات مهم فصل با معادل انگلیسی آن آورده شده است.

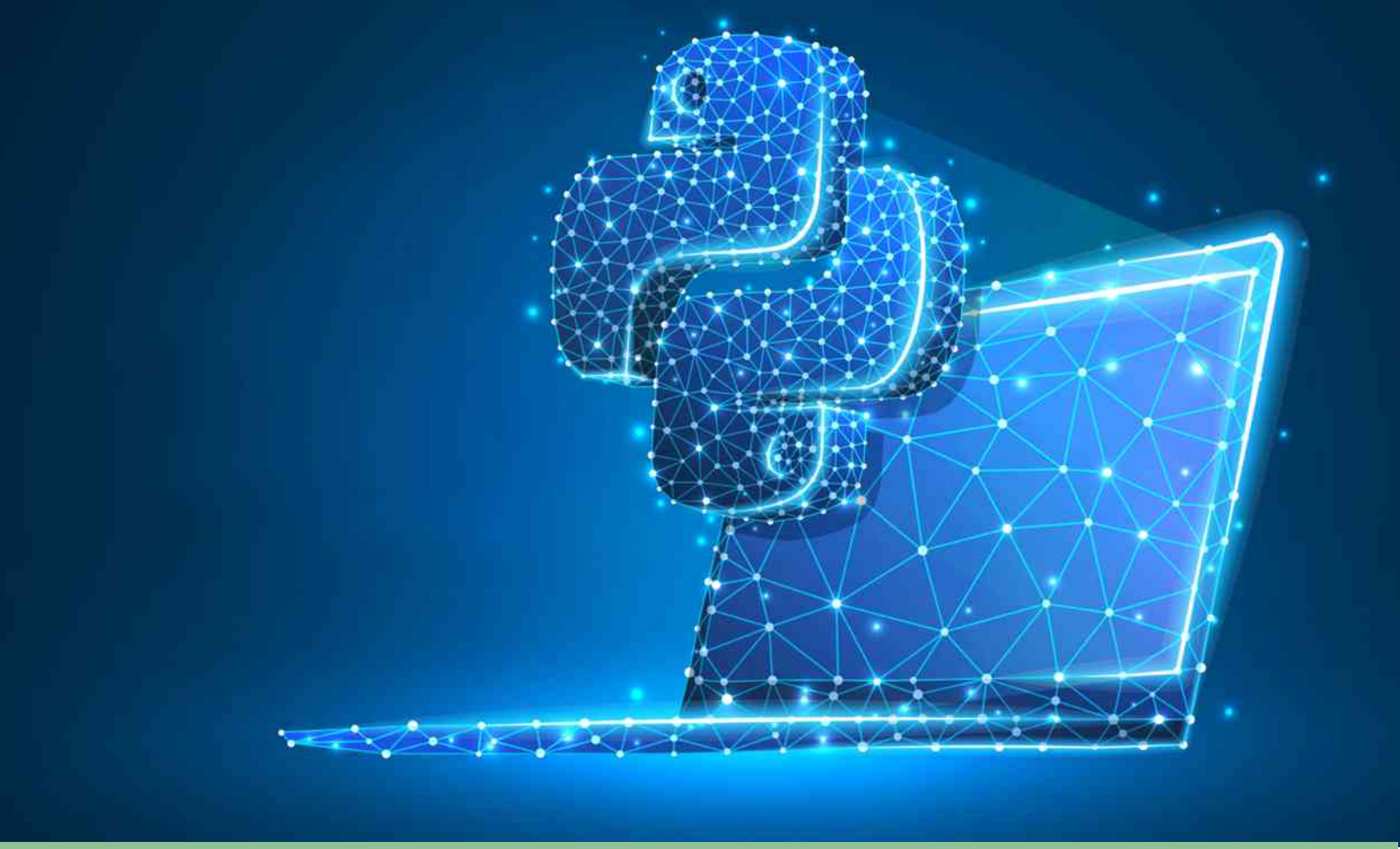
تمرین‌ها: در آخر هر فصل تمرین‌های مرتبط با آن آورده شده است. ازآنجایی‌که مؤلفان کتاب از دبیران باسابقه هستند پس تعداد تمرین‌ها، وقت لازم برای انجام آن‌ها، تعداد سؤالات سخت و آسان و نوع سؤالات با برنامه و محاسبه تعیین شده است پس خیالتان راحت باشد که همه تمرین‌ها را در طول سال می‌شود انجام داد. تمرین‌ها براساس موضوعات هر فصل بخش‌بندی شده؛ بنابراین لازم نیست برای تمرین منتظر پایان فصل باشید، در پایان هر مبحث می‌توانید به بخش تمرین‌ها مراجعه کنید و تمرین‌های همان مبحث را حل کنید.

دست‌به‌کد شو: برنامه‌نویسی درسی کاربردی است که در حین آموزش آن لازم است شما هم دست به کد بشوید. در بخش دست به کد شو از شما خواسته شده تا سعی کنید خودتان برنامه را بنویسید. حواستان باشد این بخش، قسمت مهمی از روند درسی است و نمی‌شود بدون دست‌به‌کد شدن برنامه‌نویسی یاد گرفت.

اشتباهات رایج: همان‌طور که از اسمش مشخص است، در این قسمت اشتباهاتی که ممکن است برای هرکسی پیش آید را برای شما توضیح داده‌ایم تا شما دیگر آن‌ها را تکرار نکنید! می‌توان گفت، اگر قرار باشد در بخش‌های دیگر راه برنامه‌نویسی را یاد بگیرید، در این قسمت با چاه‌های آن آشنا می‌شوید.

چه می‌کنه: در این قسمت، یک برنامه کامل برای شما نوشته‌ایم و از شما انتظار داریم بگویید این برنامه برای چه هدفی نوشته شده، چه کاری انجام می‌دهد و برای ورودی‌های مختلف، چه خروجی تولید می‌کند.

در ضمن شما هم می‌توانید برای ما مطالب و مسئله ارسال کنید! مطالب و مسئله‌هایی که خودتان از آن‌ها لذت برده‌اید به آدرس: ketab.helli@gmail.com



فصل اول لیست، تاپل، دیکشنری

اگر این فصل را به‌خوبی مطالعه کنی و کارهای خواسته شده را به‌دقت انجام دهی:

- با جنبه‌های جدیدی که درباره لیست وجود دارد آشنا می‌شوی.
- با انجام دادن تمرینات مربوط به لیست در این بخش به تسلط کافی بر آن دست پیدا می‌کنی.
- با ساختارهای جدید تاپل و دیکشنری برای ذخیره‌سازی اطلاعات در پایتون آشنا می‌شوی.
- با استفاده از ساختارهای جدید می‌توانی برنامه‌های حرفه‌ای‌تری بنویسی.
- با کاربرد جدیدی از for آشنا می‌شوی.

لیست (List)

در سال قبل در کتاب پایتون ۲ با لیست و نحوه‌ی استفاده از آن در برنامه‌ها آشنا شدید. در فصل ابتدایی از این کتاب با هدف مرور و یادآوری مطالب گذشته، ابتدا به یادآوری مفاهیم کلی لیست و در ادامه به بیان مطالب پیشرفته‌تر در این حوزه می‌پردازیم.

ب.ب.ک (برنامه‌ای بنویسید که) ۳ عدد از کاربر بگیرد و اعداد بالاتر از میانگین را چاپ کند.

همان‌طور که می‌دانید ابتدا پس از دریافت سه عدد در قالب سه متغیر، باید میانگین اعداد را حساب کنیم و سپس در ادامه بررسی و اعلام کنیم که کدام یک از آنها از میانگین بزرگ‌تر هستند:

```
a1 = int(input())
a2 = int(input())
a3 = int(input())
m = (a1 + a2 + a3) / 3
if a1 > m:
    print(a1)
if a2 > m:
    print(a2)
if a3 > m:
    print(a3)
```

ب.ب.ک n را بگیرد، سپس n عدد بگیرد و اعداد بالاتر از میانگین را چاپ کند.

برای نوشتن این برنامه نمی‌توان از روش قبلی استفاده کرد؛ زیرا مقدار n نامشخص است و نمی‌توانیم n متغیر با نام‌های a1, a2, a3, ..., an تعریف کنیم؛ به همین دلیل چاره‌ای جز استفاده از لیست نداریم.

یادآوری لیست

همان‌طور که می‌دانید در پایتون از لیست برای ذخیره‌سازی چند مقدار مختلف در یک متغیر استفاده می‌شود. امکان ایجاد تغییر در محتوای خانه‌های یک لیست وجود دارد. هر خانه از لیست دارای یک پلاک (اندیس) است و شماره پلاک‌ها از 0 شروع می‌شود.

ایجاد لیست

این کد یک لیست شامل بیست صفر متوالی را در متغیر ls ذخیره می‌کند:

```
ls = 20 * [0]
```

همچنین با نوشتن تک تک اعضای لیست نیز می‌توان آن را تعریف کرد:

```
ls = [3, 2, 7]
```

لزومی ندارد که تمام اعضای لیست از یک نوع (برای مثال عدد صحیح) باشند:

```
ls = [6, 3.14, "salam", 'H']
```

دسترسی به خانه‌های لیست

با استفاده از ls[i] می‌توانیم به خانه i ام از لیست ls دسترسی داشته باشیم.

اگر اندیس مورد نظر در لیست وجود نداشته باشد، با خطای IndexError مواجه می‌شویم:

```
ls = [30, 20, 40, 20, 60]
print(ls[5])
```

خروجی:

```
IndexError: list index out of range
```


```
ls = 10 * [0]
for i in range(10):
    ls[i] = int(input())
print(ls)
```

با توجه به نکات گفته شده، برنامه سؤال اعداد بالاتر از میانگین را می‌توانیم به شکل زیر بنویسیم:

```
n = int(input())
a = n * [0]
s = 0
for i in range(n):
    a[i] = int(input())
    s += a[i]
m = s / n
for i in range(n):
    if a[i] > m:
        print(a[i])
```

فرض کنید متغیری به نام a دارید که می‌خواهید مقدار فعلی آن را با b جمع کنید و حاصل را در خودش ذخیره کنید. این کار را می‌توانید با استفاده از عبارت a = a + b انجام دهید؛ اما کار بهتر این است که از عبارت a += b استفاده کنید.

البته جالب است بدانید این کار برای تمام عملیات‌های ریاضی قابل انجام است و فقط برای جمع نیست:

```
a = a ** b      →      a **= b
a = a * b       →      a *= b
a = a / b       →      a /= b
a = a + b       →      a += b
a = a - b       →      a -= b
```

ب.ب.ک n را بگیرد، سپس n عدد بگیرد و آن را به صورت برعکس در یک لیست ذخیره کند و در انتها لیست را چاپ کند.

بعد از تعریف لیست n تایی و نوشتن یک for از 0 تا n-1، باید ببینیم که هر ورودی را در کدام اندیس از لیست ذخیره کنیم. برای این کار می‌توانیم یک جدول رسم کنیم:

```
    print(':)')
tp1 = (1, 3, 2)
tp2 = (1, 2, 3)
if tp1 != tp2:
    print(':(')
tp = (4, 3, 6, 5)
print(tp)
if 6 in tp:
    print('6 hast')
if 2 not in tp:
    print('2 nist')
```

خروجی:

```
(1, 2, 5, 6) (5, 6, 1, 2)
(1, 2, 1, 2, 1, 2)
:)
:(
(4, 3, 6, 5)
6 hast
2 nist
```

انتساب چندتایی

فرض کنید یک تاپل با ۳ عضو داریم و می‌خواهیم عضو اول آن را در متغیر a، عضو دوم آن را در متغیر b و عضو سوم آن را در متغیر c ذخیره کنیم. با دستوراتی که تا الان خوانده‌ایم می‌توانیم این کار را به این شکل انجام دهیم:

```
tp = (4, 5, 3)
a = tp[0]
b = tp[1]
c = tp[2]
```

اما کار بهتر و تمیزتر استفاده از قابلیت انتساب چندتایی (Tuple Assignment) در پایتون است. با استفاده از این قابلیت می‌توانیم مقدار چند متغیر را با استفاده از مقادیر موجود در یک تاپل تعیین کنیم. به مثال زیر توجه کنید:

```
a, b, c = (2, 4, 5)
print(a, b, c)
```

خروجی:

```
2 4 5
```

از آنجایی که گذاشتن پرانتز در تاپل‌هایی که حداقل یک عضو دارند اختیاری است، می‌توانیم به این شکل متغیرها را تعریف کنیم:

```
a, b, c = 2, 4, 5
```

اگر تعداد متغیرها با تعداد اعضای تاپل برابر نباشد، با خطای ValueError مواجه می‌شویم.

```
a, b, c = 2, 4, 5, 3
```

خروجی:

```
ValueError: too many values to unpack (expected 3)
```

یک کاربرد خیلی مهم این قابلیت، زمانی است که می‌خواهیم مقدار دو متغیر را با هم جا به جا کنیم. در حالت عادی اگر بخواهیم مقدار دو متغیر a و b را با هم جابه‌جا کنیم، به یک متغیر کمکی مانند c نیز نیاز داریم تا مقدار اولیه‌ی متغیر a را در آن ذخیره کنیم:

```
c = a
a = b
b = c
```

اما با استفاده از انتساب چندتایی خیلی راحت‌تر می‌توانیم این کار را انجام دهیم:

```
a, b = b, a
```

این قابلیت فقط برای تاپل‌ها نیست و با هر موجودیت قابل پیمایش (به این شرط که تعداد اعضای آن با تعداد متغیرها برابر باشد) می‌توانیم این کار را انجام دهیم:

```
a, b, c = [2, 4, 5]
print(a, b, c)
a, b, c = range(1, 4)
print(a, b, c)
a, b = 'hi'
print(a, b)
```

خروجی:

```
2 4 5
1 2 3
h i
```

تاپل به چه دردی می‌خورد؟

یکی از مهم‌ترین کاربردهای تاپل، تعریف مقداری ثابت در برنامه است. برای مثال هنگامی که می‌خواهید رنگ‌های سیاه و سفید را به صورت RGB در برنامه‌ی خود تعریف کنید و در یک برنامه‌ی گرافیکی از آن‌ها استفاده کنید، می‌توانید آن‌ها را در قالب تاپل ذخیره کنید. (در فصل پایگیم به RGB و رنگ‌ها خواهیم پرداخت، فعلاً در همین حد بدانید که رنگ سیاه دارای سه مؤلفه رنگی 0 و رنگ سفید نیز دارای سه مؤلفه رنگی 255 است.)

```
black = (0, 0, 0)
white = (255, 255, 255)
```

دیکشنری (Dictionary)

انتخابات شورای دانش‌آموزی در پیش است و پنج دانش‌آموز نامزد انتخابات شده‌اند. مسئولین مدرسه تصمیم گرفته‌اند که در راستای استفاده کمتر از کاغذ و در نتیجه کمتر قطع شدن درختان، انتخابات امسال را به‌صورت الکترونیکی برگزار کنند. به همین دلیل از شما که یک برنامه‌نویس حرفه‌ای هستید کمک می‌خواهند. ب.ب.ک در ابتدا نام پنج نامزد را بگیرد، سپس انتخابات را به شکل زیر انجام دهد:

هر نفر برای رأی دادن پشت کامپیوتر می‌آید و نام نامزد مورد نظر خود را وارد می‌کند. پس از اتمام زمان رأی‌گیری مدیر کلمه end را وارد می‌کند و برنامه تعداد آرای هر نامزد را چاپ می‌کند.

برای حل این تمرین، می‌توانیم دو لیست پنج‌تایی در برنامه تعریف کنیم. به این شکل که لیست اول برای ذخیره کردن نام نامزدها و لیست دوم برای ذخیره کردن آرای نامزدها باشد. در ابتدای برنامه محتوای لیست اول را از کاربر ورودی می‌گیریم و تمام مقادیر لیست دوم را برابر با 0 می‌گذاریم.

namzad =	"Mahdi"	"Ahmad"	"Mohammad"	"Reza"	"Ali"
	0	1	2	3	4
ray =	0	0	0	0	0

در ادامه به ازای هر اسمی که شرکت‌کنندگان وارد می‌کنند، به دنبال آن اسم در لیست namzad می‌گردیم و به مقدار آرای متناظر با آن اسم که در لیست ray قرار دارد یکی اضافه می‌کنیم. به طور دقیق‌تر اگر اسمی که شرکت‌کننده وارد می‌کند در لیست namzad دارای اندیس i باشد، کافی است تا مقدار ray[i] را یکی زیاد کنیم.

کد این سؤال را می‌توانیم به این شکل بنویسیم:

```
namzad = []
for i in range(5):
    name = input()
    namzad.append(name)
ray = 5 * [0]
name = input()
while name != "end":
    if name in namzad:
        i = namzad.index(name)
        ray[i] += 1
    name = input()
for i in range(5):
    print(namzad[i], ':', ray[i])
```

به نحوه ورودی گرفتن آرای شرکت‌کنندگان دقت کنید. در بیرون از حلقه while یکی از آرا را ورودی گرفتیم و در ادامه بقیه‌ی آرا را داخل حلقه‌ی while می‌گیریم. در واقع هنگامی که اولین بار name را ورودی می‌گیریم، در صورتی وارد while شده و رأی اعمال می‌شود که نامساوی با 'end' بوده باشد. در آخر حلقه‌ی while نیز دوباره name را ورودی می‌گیریم تا این روند تکرار شود و هر دفعه با بررسی اینکه ورودی نامساوی با 'end' است آن رأی را اعمال کنیم. فایده‌ی این کار این است که با شرط while نامساوی بودن ورودی با 'end' نیز بررسی می‌شود و دیگر نیاز به بررسی کردن این قضیه داخل while نداریم.

اگر بیرون از حلقه while اولین ورودی را نمی‌گرفتیم چه اتفاقی می‌افتاد؟

اگر name = input() را ابتدای حلقه while می‌گذاشتیم، چه مشکلی پیش می‌آمد؟

برنامه‌ای که نوشتیم را طوری تغییر دهید که تعداد آرای باطله را نیز چاپ کند.

آیا راه ساده‌تری برای نوشتن این برنامه وجود دارد؟ آیا ساختاری در پایتون وجود دارد که مانند لیست باشد منتها به جای اندیس بتوانیم با یک مقدار (اسم نامزد) به هر خانه از آن دسترسی داشته باشیم؟
خوشبختانه بله، دیکشنری دقیقاً همان چیزی است که ما به آن نیاز داریم!

دیکشنری چیست؟

List					
خانه‌ها	20	10	40	20	70
پلاک‌ها (اندیس‌ها)	0	1	2	3	4

Dictionary					
خانه‌ها	20	10	40	20	70
کلیدها	“Ali”	“Reza”	“Mohammad”	“Ahmad”	“Mahdi”

دیکشنری بسیار شبیه به لیست است. مهم‌ترین تفاوت دیکشنری با لیست این است که در لیست هر خانه با یک پلاک (اندیس) شناخته می‌شود منتها در دیکشنری هر خانه با یک کلید قابل دسترسی است؛ ضمن این که دیکشنری بر خلاف لیست ترتیب ندارد.

ایجاد دیکشنری

برای ایجاد دیکشنری، هر کلید (key) و مقدار آن کلید (value) را مشخص می‌کنیم و به شکل key: value می‌نویسیم؛ سپس بین همه آن‌ها , قرار می‌دهیم.

```
dic = {“Ali”: 20, “Reza”: 10, “Mohammad”: 40, “Ahmad”: 20,
“Mahdi”: 70}
```

معمولاً برای تمیزتر شدن کد، هر کلید و مقدار آن را در یک خط می‌نویسند:

```
dic = {
    “Ali”: 20,
    “Reza”: 10,
    “Mohammad”: 40,
    “Ahmad”: 20,
    “Mahdi”: 70,
}
```

دیکشنری خالی را نیز می‌توانیم به این شکل تعریف کنیم:

```
dic = {}
```

کلیدهای دیکشنری علاوه‌بر رشته می‌توانند عدد یا تاپل نیز باشند. همچنین لزومی ندارد که همه کلیدها از یک نوع (برای مثال رشته) باشند.

```
dic = {"π": 3.14, 1.5: 2, 25: "z", (1, 3): 5}
```

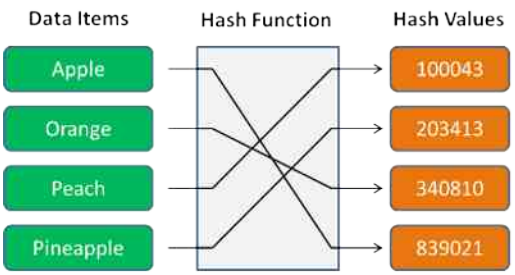
آیا لیست نیز می‌تواند کلید یک دیکشنری باشد؟

برنامه زیر را در محیط پایتون اجرا کنید.

```
dic = {[1, 3]: 5}
print(dic)
```

همان طور که دیدید اگر از لیست به عنوان کلید دیکشنری استفاده کنیم با `TypeError` مواجه می‌شویم.

به طور کلی هر شیء که قابلیت تغییرناپذیر بودن را داشته باشد (hashable باشد)، می‌تواند کلید یک دیکشنری باشد.از بین مواردی که تا الان خوانده‌ایم، فقط لیست و دیکشنری این قابلیت را ندارند.



نکته: با توجه به اینکه تاپل hashable است و لیست hashable نیست، به نظر شما تاپلی که حداقل یکی از اعضای آن لیست باشد، hashable است؟

ابتدا کمی فکر کنید، سپس با نوشتن یک برنامه (مانند برنامه‌ای که برای استفاده از لیست به عنوان کلید یک دیکشنری نوشتیم) درستی پاسخ خود را بررسی کنید.

برای کسب اطلاعات بیشتر به این پیوند مراجعه کنید:

https://stackoverflow.com/questions/2671376/hashable-immutable

دسترسی به خانه‌های دیکشنری

با استفاده از `dic[key]` می‌توانیم به خانه‌ای از دیکشنری `dic` که کلید آن `key` است دسترسی داشته باشیم. اگر کلید `key` در دیکشنری وجود نداشته باشد با خطای `KeyError` مواجه خواهیم شد.

```
dic1 = {"Ali": 20, "Reza": 10, "Mohammad": 40, "Ahmad": 20,
"Mahdi": 70}
print(dic1["Ali"], dic1["Mahdi"])
dic2 = {2: 30, 1: 40, 5: 20}
print(dic2[2], dic2[5])
print(dic1["Amir"])
```

خروجی:

```
20 70
30 20
KeyError: 'Amir'
```

تغییر محتوای دیکشنری

با `dic[key] = value` می‌توانیم خانه‌ای با کلید `key` در دیکشنری `dic` بسازیم که مقدار آن برابر با `value` باشد. اگر کلید `key` از قبل وجود داشته باشد صرفاً مقدار آن تغییر می‌کند، در غیر این صورت کلید جدید به دیکشنری اضافه خواهد شد.

```
dic = {"Ali": 20, "Reza": 10}
dic["Ali"] = 30
print(dic)
dic["Amir"] = 5
print(dic)
```

خروجی:

```
{'Ali': 30, 'Reza': 10}
{'Ali': 30, 'Reza': 10, 'Amir': 5}
```

آیا دیکشنری مانند لیست ارجاعی است؟

برنامه زیر را در محیط پایتون اجرا کنید.

```
dic1 = {"Ali": 20, "Reza": 10}
dic2 = dic1
dic2["Ali"] = 30
print(dic1, dic2)
```

همان‌طور که دیدید دیکشنری نیز مانند لیست ارجاعی است. برای ذخیره کردن یک کپی از دیکشنری `dic1` می‌توانیم از `dic1.copy()` استفاده کنیم.

```
dic1 = {"Ali": 20, "Reza": 10}
dic2 = dic1.copy()
dic2["Ali"] = 30
print(dic1, dic2)
```

خروجی:

```
{'Ali': 20, 'Reza': 10} {'Ali': 30, 'Reza': 10}
```

توابع و دستورات درون دیکشنری

۱) حذف یک مقدار از دیکشنری با کلید (pop)

این تابع یک کلید را به عنوان ورودی می‌گیرد و خانه‌ای از دیکشنری که دارای آن کلید است را حذف می‌کند و مقدار آن را برمی‌گرداند.

```
dic = {'Ali': 30, 'Reza': 10}
value = dic.pop('Ali')
print(value)
print(dic)
```

خروجی:

```
30
{'Reza': 10}
```

اگر کلید ورودی تابع pop در دیکشنری وجود نداشته باشد چه اتفاقی می‌افتد؟

۲) دریافت تمام کلیدها (keys)

این تابع کلیدهای موجود در یک دیکشنری که توسط آن صدا زده شده را در قالب یک dict_keys (که قابل پیمایش است) برمی‌گرداند.

```
dic = {'Ali': 30, 'Reza': 10}
print(dic.keys(), max(dic.keys()))
ls = []
for key in dic.keys():
    print(key)
    ls.append(key)
print(ls)
```

خروجی:

```
dict_keys(['Ali', 'Reza']) Reza
Ali
Reza
['Ali', 'Reza']
```

۳) دریافت تمام مقادیر (values)

این تابع مقادیر موجود در یک دیکشنری که توسط آن صدا زده شده را در قالب یک dict_values (که قابل پیمایش است) برمی‌گرداند.

```
dic = {'Ali': 30, 'Reza': 10}
print(dic.values(), sum(dic.values()))
ls = []
for value in dic.values():
    print(value)
    ls.append(value)
print(ls)
```

خروجی:

```
dict_values([30, 10]) 40
30
10
[30, 10]
```

۴) دریافت تمام کلیدها و مقادیر به صورت یک تاپل دو عضوی (items)

این تابع کلیدها و مقادیر موجود در یک دیکشنری که توسط آن صدا زده شده را به صورت یک تاپل دو عضوی در قالب یک items_dict (که قابل پیمایش است) برمی‌گرداند.

```
dic = {'Ali': 30, 'Reza': 10}
print(dic.items())
ls = []
for item in dic.items():
    key, value = item
    print(item, key, value)
    ls.append(item)
print(ls)
```

خروجی:

```
dict_items([('Ali', 30), ('Reza', 10)])
('Ali', 30) Ali 30
('Reza', 10) Reza 10
[('Ali', 30), ('Reza', 10)]
```

تمیزتر کد بزن

هنگام استفاده از dict_items به جای اینکه به این شکل key و value را به دست آوریم:

```
dic = {'Ali': 30, 'Reza': 10}
for item in dic.items():
    key, value = item
    print(key, value)
```

می‌توانیم خیلی راحت‌تر و در همان حلقه for این کار را انجام دهیم:

```
dic = {'Ali': 30, 'Reza': 10}
for key, value in dic.items():
    print(key, value)
```

dict_values، dict_keys و dict_items مثال‌هایی برای موجودیت‌های قابل پیمایش بدون اندیس به شمار می‌آیند.

```
dic = {'Ali': 30, 'Reza': 10}
items = dic.items()
print(items[0])
```

خروجی:

```
TypeError: 'dict_items' object is not subscriptable
```

۵) خالی کردن دیکشنری (clear)

این تابع یک دیکشنری که توسط آن صدا زده شده را به یک دیکشنری خالی تبدیل می‌کند.

```
dic = {'Ali': 30, 'Reza': 10}
dic.clear()
print(dic)
```

خروجی:

```
{}
```

۶) کپی از مقادیر دیکشنری (copy)

این تابع یک کپی از یک دیکشنری که توسط آن صدا زده شده را ساخته و آن را برمی‌گرداند.

```
dic = {'Ali': 30, 'Reza': 10}
a = dic
b = dic.copy()
a['Ali'] = 20
print(dic)
print(a)
print(b)
```

خروجی:

```
{'Ali': 20, 'Reza': 10}
{'Ali': 20, 'Reza': 10}
{'Ali': 30, 'Reza': 10}
```

عملگرهای دیکشنری

۱) بررسی مساوی بودن دو دیکشنری (==)

این شرط در صورتی درست است که دو دیکشنری از لحاظ محتوا کاملاً مانند یکدیگر باشند.

```
dic1 = {'Ali': 20, 'Reza': 10}
dic2 = {'Reza': 10, 'Ali': 20}
if dic1 == dic2:
    print(':)')
```

خروجی:

```
:)
```

۲) بررسی مساوی نبودن دو دیکشنری (=!)

این شرط در صورتی درست است که دو دیکشنری از لحاظ محتوا مانند یکدیگر نباشند.

```
dic1 = {'Ali': 20, 'Reza': 10}
dic2 = {'Reza': 10, 'Mahdi': 30}
if dic1 != dic2:
    print(':(')
```

خروجی:

```
:(
```

۳) بررسی بودن یک کلید در دیکشنری (in)

این شرط در صورتی درست است که آن کلید در دیکشنری باشد.

```
dic = {'Ali': 20, 'Reza': 10}
if 'Ali' in dic:
    print('Ali hast')
```

خروجی:

```
Ali hast
```

۴) بررسی نبودن یک کلید در دیکشنری (not in)

این شرط در صورتی درست است که آن کلید در دیکشنری نباشد.

```
dic = {'Ali': 20, 'Reza': 10}
if 'Mahdi' not in dic:
    print('Mahdi nist')
```

خروجی:

```
Mahdi nist
```

مثال اول این بخش (انتخابات شورای دانش‌آموزی) را با دیکشنری پیاده‌سازی کنید.

در ابتدا یک دیکشنری با نام ray می‌سازیم و کلیدهای آن را برابر با اسامی نامزدها و مقدار هر کلید را برابر با 0 قرار می‌دهیم. سپس بعد از دریافت هر ورودی، رأی شرکت کننده آن را در دیکشنری اعمال می‌کنیم:

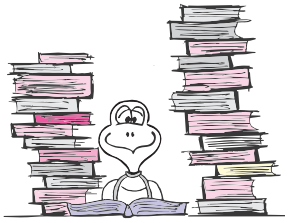
```
ray = {}
for i in range(5):
    name = input()
    ray[name] = 0
name = input()
while name != "end":
    if name in ray:
        ray[name] += 1
    name = input()
for key, value in ray.items():
    print(key, ':', value)
```

ب.ب.ک یک کلمه بگیرد و تعداد تکرار هر حرف در آن کلمه را در قالب یک دیکشنری چاپ کند.

پس از دریافت کلمه در ورودی، یک دیکشنری تعریف می‌کنیم. سپس از حلقه for برای حرکت روی حروف کلمه استفاده می‌کنیم و برای هر حرف اگر کلید مربوط به آن از قبل در دیکشنری وجود نداشت، آن را می‌سازیم تا برنامه با خطا مواجه نشود. در آخر حلقه for هم مقدار آن حرف در دیکشنری را یکی زیاد می‌کنیم:

```
kalame = input()
dic = {}
for harf in kalame:
    if harf not in dic:
        dic[harf] = 0
    dic[harf] += 1
print(dic)
```

لغت‌نامه



ترجمه	واژه علمی	ترجمه	واژه علمی
دیکشنری	Dictionary	برش لیست	List Slicing
کلید	Key	قابل پیمایش	Iterable
مقدار	Value	الحاق	Concatenate
تغییرناپذیر	Immutable	تاپل	Tuple
خطای کلید	Key Error	انتساب چندتایی	Tuple Assignment
خطای مقدار	Value Error	خطای اندیس	Index Error

جمع بندی کن

در پایتون از ..... برای ذخیره‌سازی چند مقدار مختلف در یک متغیر استفاده می‌شود.
امکان ..... در محتوای خانه‌های یک لیست وجود دارد.

یک کاربرد خیلی مهم ....., زمانی است که بخواهیم مقدار دو متغیر را با هم جا به جا کنیم.
یکی از مهم‌ترین کاربردهای تاپل، ..... در برنامه است.
مهم‌ترین تفاوت دیکشنری با لیست این است که در لیست هر خانه با یک ..... شناخته می‌شود منتها در دیکشنری هر خانه با یک ..... قابل دسترسی است.
دیکشنری بر خلاف لیست ..... ندارد.
کلیدهای دیکشنری علاوه بر ..... می‌توانند ..... یا ..... نیز باشند.

هر خانه از لیست دارای یک ..... است.
شماره اندیس‌ها از ..... شروع می‌شود.
در پایتون امکان اندیس‌دهی ..... نیز وجود دارد.

از هر داده‌ای که ..... باشد می‌توان در حلقه‌ی for استفاده کرد.
تاپل بسیار شبیه به لیست است، منتها بر خلاف لیست ..... نیست.

```
def majmoo(ls):
    output = 0
    for num in ls:
        output += num
    return output

def zarb(ls):
    output = 0
    for num in ls:
        output *= num
    return output

n = int(input())
ls = n * [0]
for i in range(n):
    ls[i] = int(input())

output = (majmoo(ls) / len(ls)) + (zarb(ls) ** (1 / len(ls)))
print(output)
```

این برنامه برای ورودی روبه‌رو:

```
4
9
4
18
2
```

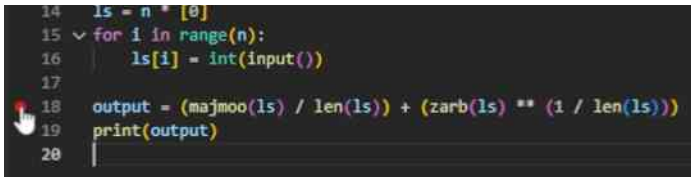
عدد ۸/۲۵ را خروجی می‌دهد در صورتی که خروجی صحیح ۱۴/۲۵ است.

برای اشکال‌زدایی برنامه، در محیط VSCode از بخش سمت راست بالای صفحه بر روی گزینه Debug Python File کلیک کنید.

شکل ۲۲- گام اول اشکال‌زدایی

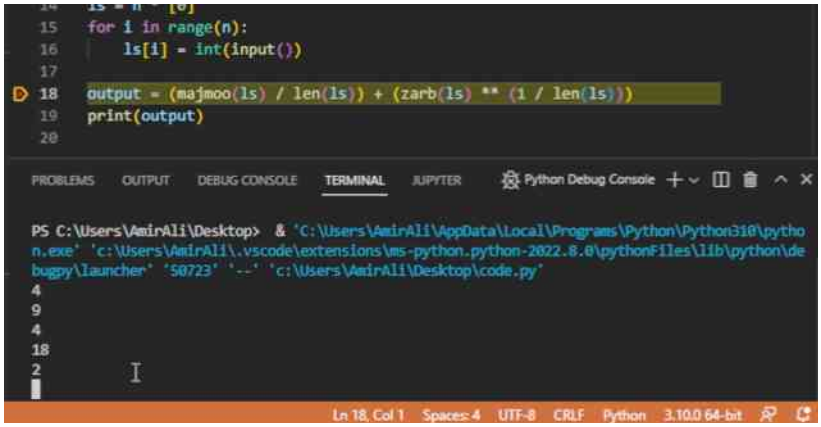
البته، در حالت عادی پس از انجام این کار اتفاق خاصی نمی‌افتد و برنامه مانند گذشته اجرا می‌شود. به همین دلیل باید با کلیک کردن بر روی حاشیه سمت چپ بخش‌هایی از کد، تعدادی نقطه توقف (Breakpoint) در مکان‌هایی از برنامه که می‌خواهیم بررسی کنیم، برجسته (Highlight) نماییم. (با کلیک کردن دوباره

بر روی نقطه توقف می‌توان آن را غیرفعال کرد). برای مثال در برنامه بالا، زمان مقداردهی output می‌توانیم یک نقطه توقف قرار دهیم تا وضعیت متغیر output را بررسی کنیم.

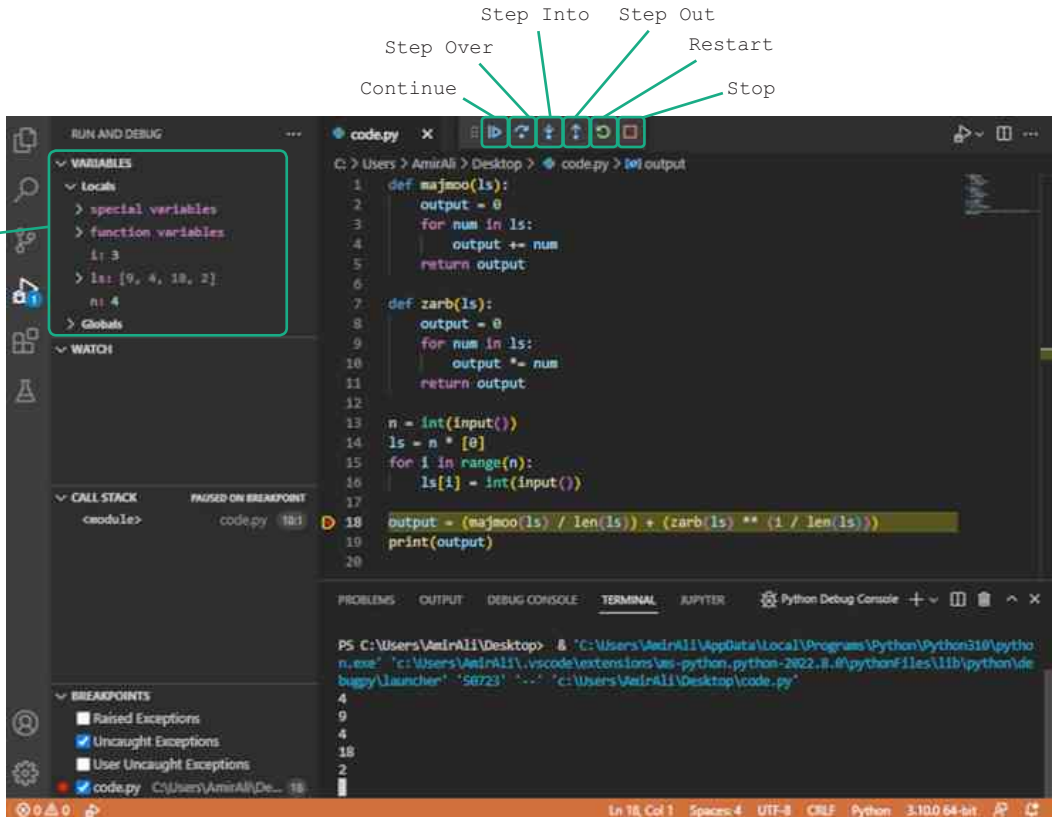


شکل ۲۳- گام دوم اشکال‌زدایی

پس از تنظیم نقطه توقف، دوباره بر روی گزینه Debug Python File کلیک کنید و ورودی‌ها را در کنسول وارد کنید.



شکل ۲۴- گام سوم اشکال‌زدایی



شکل ۲۵- انتخاب نحوه اشکال‌زدایی

- دستور Continue

این دستور تا زمانی که به نقطه توقف بعدی نرسد، برنامه را اجرا می‌کند، اگر نقطه توقف دیگری وجود نداشته باشد برنامه تا انتها اجرا می‌شود. کلید میان‌بر این دستور F5 است.

- دستور Step Over

این دستور خط برجسته (Highlight) شده به رنگ زرد را اجرا می‌کند سپس برجسته بودن آن خط از بین می‌رود و خط بعدی برجسته می‌شود. نکته موجود در این دستور این است که اگر خط برجسته شده دارای تابع باشد، بخش اشکال‌زدای VSCode وارد آن تابع نمی‌شود و در همان مرحله آن را به طور کامل اجرا می‌کند. کلید میان‌بر این دستور F10 است.

- دستور Step Into

این دستور بسیار شبیه به Step Over است، منتها اگر خط برجسته شده دارای تابع باشد و این تابع را خودمان در برنامه تعریف کرده باشیم (برای مثال print نباشد)، بخش اشکال‌زدای VSCode وارد آن می‌شود و در خط اول آن متوقف می‌شود. کلید میان‌بر این دستور F11 است.

- دستور Step Out

این دستور تمام دستورات موجود در تابعی که در آن قرار دارد را تا زمانی که به نقطه توقف یا انتهای تابع نرسد اجرا می‌کند. کارکرد این دستور در صورتی که داخل تابعی نباشیم مانند دستور Continue خواهد بود. کلیدهای میان‌بر این دستور Shift + F11 است.

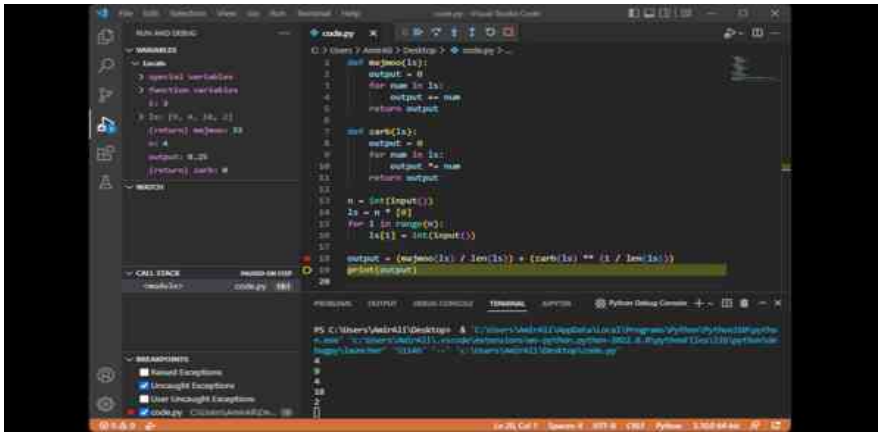
- دستور Restart

این دستور عملیات اشکال‌زدایی را متوقف می‌کند و آن را دوباره از اول آغاز می‌کند. کلیدهای میان‌بر این دستور Ctrl + Shift + F5 است.

- دستور Stop

این دستور عملیات اشکال‌زدایی را متوقف می‌کند. کلیدهای میان‌بر این دستور Shift + F5 است.

در ادامه از دستور Step Over استفاده می‌کنیم تا وضعیت خروجی توابع مختلف را مشاهده کنیم. (حتماً خودتان یک‌بار در این بخش به جای دستور Step Over از دستور Step Into استفاده کنید تا متوجه تفاوت این دو دستور شوید).



شکل ۲۶- نمایی از روند اشکال‌زدایی

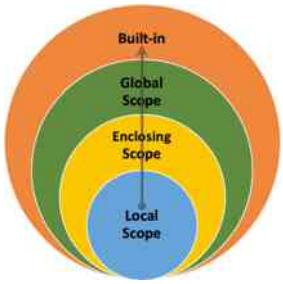
خروجی تابع majmoo درست است؛ اما تابع zarb مقدار 0 را به‌عنوان خروجی برگردانده است و همین باعث خروجی نادرست برنامه ما شده است، اگر به تابع zarb دقت کنیم، می‌بینیم که مقدار اولیه متغیر output به جای 1 برابر با 0 است و درنتیجه با ضرب شدن اعداد، مقدار آن همچنان 0 باقی می‌ماند.
تابع zarb تصحیح شده به شکل زیر است:

```
def zarb(ls):
    output = 1
    for num in ls:
        output *= num
    return output
```

به‌عنوان تمرین، خودتان برنامه زیر را با استفاده از هر روشی که دوست داشتید اشکال‌زدایی کنید.
- ب.ب.ک n و سپس محتویات یک آرایه n عضوی را ورودی بگیرد و درنهایت تعداد اعداد اول موجود در آرایه را برگرداند.

```
n = int(input())
ls = n * [0]
for i in range(n):
    ls[i] = int(input())

for num in ls:
    output = 0
    flag = 1
    for i in range(2, n):
        if num % i == 0:
            flag = 0
    output += flag
print(output)
```

محدوده‌ها در پایتون (scope)

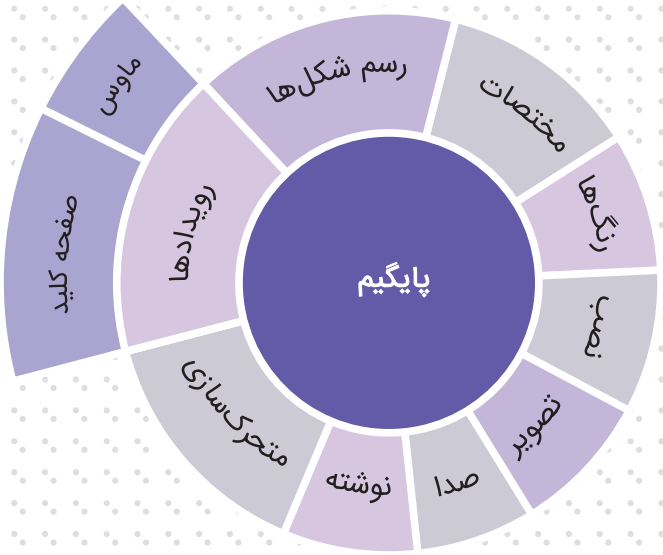


یکی از مفاهیمی که آشنا نبودن با آن در خیلی از مواقع باعث ایجاد مشکل در عملکرد برنامه ما می‌شود، مفهوم محدوده‌های موجود در پایتون است. در کتاب پایتون ۲ با محدوده محلی در پایتون آشنا شدید و دانستید که متغیرهای داخل یک تابع به‌صورت محلی تعریف می‌شوند. به طور کلی چهار نوع محدوده در پایتون وجود دارد که در این بخش به معرفی سه مورد از آن‌ها می‌پردازیم.

۱) محدوده درونی (Built-in)

این محدوده توابع و دستورات پیش‌فرض در پایتون (مانند print) همچنین مقادیر و دستورات import شده از کتابخانه‌ها را شامل می‌شود. دستوراتی که در محدوده درونی برنامه قرار دارند، در هر جایی از کد قابل دسترسی هستند. برای مثال شما چه در بیرون تابع و چه در درون تابع می‌توانید از دستور print استفاده کنید.

فصل چهارم پایگیم



اگر این فصل را به‌خوبی مطالعه کنی و کارهای خواسته شده را به‌دقت انجام دهی:

- با کتابخانه قوی پایگیم در زمینه بازی‌سازی آشنا می‌شوی.
- یاد می‌گیری که چطور در با این کتابخانه انواع شکل‌ها را بکشی، تصاویر مختلف را بارگذاری کنی و صداهای متنوع را پخش کنی.
- یاد می‌گیری که چگونه تصاویر و اشکال را به حرکت درآوری.
- یاد می‌گیری که ورودی‌های ماوس و صفحه‌کلید را بخوانی و واکنش مناسبی نشان دهی.
- یاد می‌گیری که با ترکیب کارهای بالا بازی‌های زیبا بسازی.

پایگیم (Pygame)

مقدمه‌ای بر پایگیم

پیش از این در کتاب پایتون ۱ با کتابخانه ترتل و مقدمات دنیای گرافیک در برنامه‌نویسی آشنا شدید. در این فصل با کتابخانه‌ای پیشرفته‌تر در این حوزه به نام پایگیم آشنا می‌شویم سپس سراغ مبحث بسیار جذاب بازی‌سازی می‌رویم. پایگیم کتابخانه‌ای محبوب برای نوشتن بازی‌های ویدئویی در پایتون است که از مهم‌ترین ویژگی‌های آن می‌توان به رایگان بودن، همچنین قابل‌اجرابودن بر روی پلتفرم‌ها و سیستم عامل‌های مختلف اشاره کرد.

نصب پایگیم

کتابخانه پایگیم به‌صورت پیش‌فرض در پایتون وجود ندارد و به همین دلیل برای استفاده از آن، باید آن را نصب کنیم. راه‌های متفاوتی برای نصب پایگیم وجود دارد که یکی از ساده‌ترین آن‌ها دنبال کردن مراحل دستورالعمل زیر است:

۱) وارد محیط `Command Prompt` در ویندوز شوید.

۲) دستور `pip install pygame` یا `python -m pip install pygame` را بنویسید و کلید `Enter` را فشار دهید.

پس از مدتی نصب پایگیم بر روی سیستم شما انجام می‌شود.

```
C:\Users\AmirAli>pip install pygame
Collecting pygame
  Downloading pygame-2.1.2-cp310-cp310-win_amd64.whl (8.4 MB)
     ---------------------------------------- 8.4/8.4 MB 1.1 MB/s eta 0:00:00
Installing collected packages: pygame
Successfully installed pygame-2.1.2

C:\Users\AmirAli>
```

شکل ۱- پیام موفقیت‌آمیز بودن نصب پایگیم بر روی سیستم

جالب است بدانید حتی در زمانی که به اینترنت دسترسی ندارید، می‌توانید کتابخانه‌های موجود در `pip` را با استفاده از فایل `whl` مربوط به آن نصب کنید. فایل `whl` کتابخانه پایگیم در `CD` همراه کتاب موجود است و برای نصب آن کافی است در دستور `pip install` به جای نام کتابخانه آدرس فایل `whl` را وارد کنید. برای مثال:

```
python -m pip install C:\Users\AmirAli\Desktop\pygame-2.1.2-
cp310-cp310-win_amd64.whl
```

احتمال اینکه در فرایند نصب پایگیم مطابق با روش دوم، با خطا مواجه شوید بسیار زیاد است. در صورت بروز خطا، عبارت خطایی که با آن برخورد کرده‌اید را در اینترنت جست‌وجو کنید و راه‌حل آن را پیدا کنید.

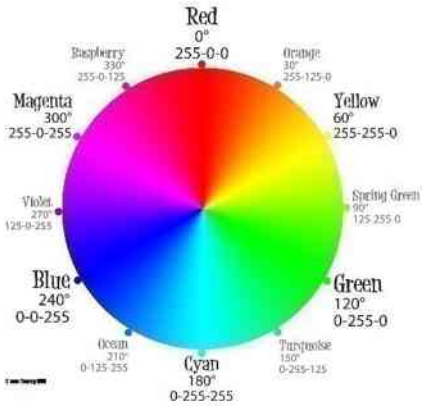
یکی از خطاهای رایج در هنگام نصب یک کتابخانه، خطای مربوط به وجود نداشتن python است:

```
'python' is not recognized as an internal or external command, oper-
able program or batch file.
```

برای رفع این مشکل راه‌های زیادی وجود دارد. ساده‌ترین راه حذف کردن پایتون و نصب کردن دوباره آن است، فقط حواستان باشد که هنگام نصب گزینه Add Python to PATH را فعال کنید.

رنگ‌ها در پایگیم

قبل از آنکه برنامه‌نویسی با پایگیم را شروع کنیم، لازم است تا با ساختار رنگ‌ها در پایگیم آشنا شویم. رنگ‌ها در پایگیم بر اساس مدل رنگی استاندارد RGB مقداردهی می‌شوند. در این ساختار هر رنگ سه مؤلفه دارد که مؤلفه اول مربوط به رنگ قرمز (Red)، مؤلفه دوم مربوط به رنگ سبز (Green) و مؤلفه سوم مربوط به رنگ آبی (Blue) است. (نام RGB نیز بر اساس حرف اول هرکدام از این سه رنگ به‌وجود آمده است)، همچنین مقدار هر مؤلفه عددی بین ۰ تا ۲۵۵ است، درنهایت هر رنگ با استفاده از ترکیب نورهای قرمز، سبز و آبی با توجه به مقدار مؤلفه هرکدام از این سه رنگ، ساخته می‌شود برای مثال، اگر مقدار هر سه مؤلفه برابر با ۰ باشد، هیچ نوری وجود ندارد و خروجی رنگ سیاه می‌شود و برعکس هنگامی که مقدار همه آن‌ها برابر با ۲۵۵ باشد رنگ سفید تشکیل می‌گردد. یا هنگامی که مقدار مؤلفه رنگ قرمز برابر با ۲۵۵ باشد و بقیه مؤلفه‌ها مقدار ۰ را داشته باشند، رنگ قرمز روشن به وجود می‌آید و در ادامه هر چه مقدار مؤلفه اول کمتر شود، این رنگ تیره‌تر خواهد شد.



شکل ۲- طیف رنگی در ازای مؤلفه‌های مختلفR و G و B

شروع کار با پایگیم

اولین کاری که باید انجام دهیم، import کردن کتابخانه پایگیم و صدا زدن تابع init آن است:

```
import pygame

pygame.init()
```

در ادامه لازم است تا یک صفحه با ابعاد مشخصی (مثلاً اینجا ۸۰۰ در ۶۰۰) بسازیم و آن را در متغیری مانند disp ذخیره کنیم تا در آینده کارهای گرافیکی خود را بر روی آن انجام دهیم:

```
disp_size = (800, 600)
disp = pygame.display.set_mode(disp_size)
```

ورودی دستور set_mode یک تاپل دو عضوی است که عضو اول آن طول و عضو دوم آن عرض صفحه را مشخص می‌کند. بهتر است تا اندازه صفحه را در یک متغیر مانند disp_size ذخیره کنیم سپس آن را به‌عنوان پارامتر ورودی به set_mode بدهیم.

اکنون، اگر برنامه را اجرا کنیم، یک صفحه سیاه تشکیل می‌شود. برای تغییر رنگ این صفحه می‌توانیم از دستور fill استفاده کنیم که یک رنگ (مثلاً در اینجا رنگ سفید) را به‌عنوان ورودی می‌گیرد و آن صفحه را با آن رنگ پر می‌کند:

```
white = (255, 255, 255)
disp.fill(white)
```

اما بعد از نوشتن این کد نیز صفحه پایگیم، همچنان سیاه خواهد بود؛ زیرا بعد از اعمال تغییرات بر روی صفحه پایگیم باید حتماً آن را به‌روز کنیم تا تغییرات اعمال شده در صفحه نمایش ظاهر شود. برای به‌روز کردن صفحه از pygame.display.update() استفاده می‌کنیم.

همچنین برای بسته شدن صفحه و آزاد شدن منابع تخصیص یافته توسط پایگیم، می‌توانیم از دستور pygame.quit() استفاده کنیم.

حلقه اصلی برنامه

تا به امروز، برنامه‌هایی که در محیط پایتون می‌نوشتیم، شروع و پایان مشخصی داشتند و پس از دریافت ورودی‌ها و چاپ کردن خروجی‌ها به اتمام می‌رسیدند؛ اما برنامه‌ها و بازی‌هایی که در کامپیوتر، گوشی یا ... اجرا می‌کنیم به این شکل نیستند و دائماً در حال اجرا هستند. به طور کلی تمام برنامه‌هایی که ما می‌بینیم از یک حلقه اصلی تشکیل شده‌اند و تمام اتفاقات برنامه داخل آن حلقه رخ می‌دهد سپس هنگامی که کاربر دکمه خروج را می‌زند، آن حلقه به پایان می‌رسد و برنامه تمام می‌شود. این ساختار در برنامه‌هایی که با استفاده از کتابخانه پایگیم، نوشته می‌شوند نیز وجود دارد و اکثر برنامه‌های پایگیم دارای یک حلقه اصلی `while` هستند. پس از این به بعد لازم است یک حلقه `while` در برنامه قرار دهیم تا کارهای اصلی برنامه را در داخل این حلقه انجام دهیم:

```
running = True
while running == True:
    disp.fill(white)

    pygame.display.update()
```

بهتر است به جای `while running == True:` از `while running:` استفاده کنیم.

درنهایت کد این بخش به شکل زیر می‌شود:

```
import pygame
pygame.init()

# display
disp_size = (800, 600)
disp = pygame.display.set_mode(disp_size)

# colors
white = (255, 255, 255)

# variables
running = True

# main loop
while running:
    # draws
    disp.fill(white)

    pygame.display.update()
```

مختصات در پایگیم

اکنون نیاز است تا با ساختار مختصات در پایگیم آشنا شویم. در پایگیم نقطه بالا-چپ صفحه دارای مختصات (۰,۰) است و هر چه به سمت راست برویم، مقدار x (طول) بیشتر می‌شود (مانند دنیای ریاضی)؛ و هر چه به سمت پایین برویم، مقدار y (عرض) افزایش می‌یابد (برخلاف دنیای ریاضی). برای مثال، اگر ابعاد صفحه ۸۰۰ در ۶۰۰ باشد، مختصات نقطه بالا-راست برابر با(۷۹۹,۰) و مختصات نقطه پایین-چپ برابر با (۰,۵۹۹) است، همچنین نقطه پایین-راست نیز مختصات (۷۹۹,۵۹۹) را دارد.

رسم اشکال هندسی در صفحه

برای رسم اشکال هندسی می‌توانیم از زیرمجموعه pygame.draw استفاده کنیم. در تمام توابع موجود در این زیرمجموعه، پارامتر ورودی اول صفحه نمایشی است که قرار است شکل بر روی آن رسم شود و پارامتر ورودی دوم نیز رنگ شکل است.

۱) رسم خط (pygame.draw.line)

پارامترهای ورودی سوم و چهارم این تابع تاپل‌های دو عضوی هستند که به ترتیب مختصات نقاط شروع و پایان خط را مشخص می‌کنند. پارامتر ورودی پنجم نیز ضخامت خط را مشخص می‌کند که به طور پیش‌فرض برابر با ۱ است.

۲) رسم مستطیل (pygame.draw.rect)

پارامتر ورودی سوم این تابع یک تاپل چهار عضوی است که دو عضو اول آن مختصات نقطه بالا چپ مستطیل و دو عضو دوم آن طول و عرض مستطیل را مشخص می‌کند. پارامتر ورودی چهارم نیز ضخامت مستطیل است که مقدار آن به طور پیش‌فرض برابر با ۰ (به معنای توپُر بودن شکل) است.

۳) رسم دایره (pygame.draw.circle)

پارامتر ورودی سوم این تابع، مختصات مرکز دایره را در قالب یک تاپل دو عضوی مشخص می‌کند و پارامتر ورودی چهارم نیز شعاع دایره است، همچنین پارامتر ورودی پنجم ضخامت دایره است که مقدار آن به طور پیش‌فرض برابر با ۰ (به معنای توپُر بودن شکل) است.

۴) رسم بیضی (pygame.draw.ellipse)

پارامتر ورودی سوم این تابع یک تاپل چهار عضوی است که اطلاعات مربوط به مستطیلی فرضی که بیضی داخل آن قرار می‌گیرد را در خود دارد، به‌گونه‌ای که دو عضو اول آن مختصات نقطه بالا چپ مستطیل و دو عضو دوم آن طول و عرض مستطیل را مشخص می‌کند. پارامتر ورودی چهارم نیز ضخامت بیضی است که مقدار آن به طور پیش‌فرض برابر با ۰ (به معنای توپُر بودن شکل) است.

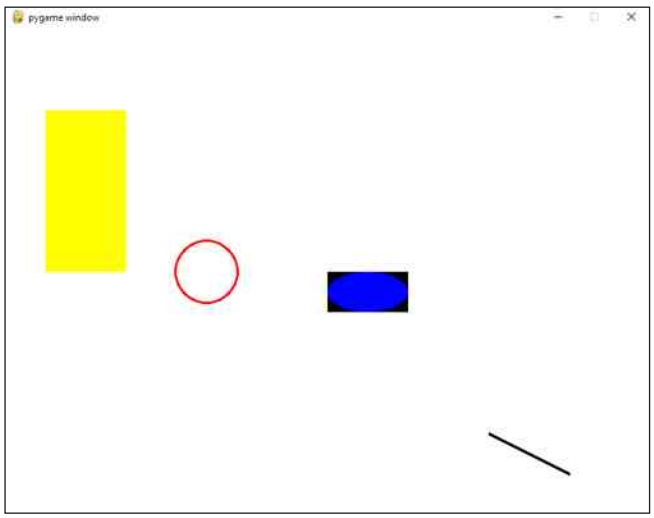
در مورد توابع دیگر موجود در pygame.draw اطلاعات بیشتری پیدا کنید.

مثال:

```
import pygame
pygame.init()
```

```
# display
disp_size = (800, 600)
disp = pygame.display.set_mode(disp_size)

# colors
black = (0, 0, 0)
white = (255, 255, 255)
red = (255, 0, 0)
blue = (0, 0, 255)
yellow = (255, 255, 0)

# variables
running = True

# main loop
while running:
    # draws
    disp.fill(white)
    pygame.draw.rect(disp, yellow, (50, 100, 100, 200), 0)
    pygame.draw.circle(disp, red, (250, 300), 40, 3)
    pygame.draw.rect(disp, black, (400, 300, 100, 50), 0)
    pygame.draw.ellipse(disp, blue, (400, 300, 100, 50), 0)
    pygame.draw.line(disp, black, (600, 500), (700, 550), 4)

    pygame.display.update()
```

خروجی:



نکته مهمی که در مورد خروجی این برنامه وجود دارد، این است که شکل‌ها به ترتیب کشیده می‌شوند و از آنجایی که مستطیل سیاه قبل از بیضی آبی رسم شده است، بیضی آبی بر روی مستطیل سیاه قرار می‌گیرد.

چه می‌کند؟

```
import pygame
pygame.init()

# display
disp_size = (800, 600)
disp = pygame.display.set_mode(disp_size)

# variables
running = True
b_val = 0

# main loop
while running:
    # draws
    if b_val < 256:
        pygame.draw.line(disp,(0, 0, b_val),\
        (300, b_val + 100), (500, b_val + 100), 1)
        b_val += 1

    pygame.display.update()
```

دست به کار شو

ب.ب.ک یک مستطیل در صفحه بکشد که با سرعتی ثابت در حال حرکت است.

برای نوشتن ساده‌تر برنامه بهتر است دو متغیر برای مختصات مستطیل و دو متغیر هم برای سرعت آن تعریف کنیم و داخل حلقه، مختصات مستطیل را با توجه به سرعت تغییر دهیم:

```
import pygame
pygame.init()

# display
disp_size = (800, 600)
disp = pygame.display.set_mode(disp_size)

# colors
black = (0, 0, 0)
white = (255, 255, 255)
# variables
running = True
rect_size = (120, 100)
```