

فصل ۲

ساختارهای تصمیم و تکرار

در برنامه‌هایی که تاکنون نوشته‌ایم، دستورات به صورت پشت سرهم (از اولین دستور به آخرین دستور) اجرا می‌گردیدند. در برنامه‌های واقعی و پیچیده نیاز است بعضی از دستورات تحت شرایط خاصی اجرا شوند، و برخی دیگر از دستورات اجرا نشوند یا بعضی از دستورات چندین بار اجرا گردند. برای پیاده‌سازی چنین برنامه‌هایی از ساختارهای کنترلی استفاده می‌شود. ساختارهای کنترلی دو نوع هستند که عبارت‌اند از:

۱. ساختارهای تصمیم‌گیری
۲. ساختارهای تکرار

۲-۱. ساختارهای تصمیم‌گیری

این ساختارها برای مواقعی به کار می‌روند که بخواهید با برقرار شدن شرط خاصی، مجموعه‌ای از دستورات اجرا شوند یا بعضی از دستورات دیگر اجرا نشوند. در ادامه ساختارهای تصمیم را می‌آموزیم.

ساختار تصمیم if

در این ساختار ابتدا شرطی^۱ ارزیابی می‌شود، اگر نتیجه ارزیابی شرط درست (True) باشد، یک مجموعه از دستورات اجرا می‌شوند، وگرنه، مجموعه دیگری از دستورات اجرا خواهند شد. این ساختار به صورت‌های زیر به کار می‌رود:

شرط if: مجموعه دستورات

۱. ساختار ساده تک انتخابی، این دستور یک دستور مرکب است که به صورت زیر به کار می‌رود:

^۱ Condition

در این ساختار، ابتدا شرط ارزیابی می‌شود، اگر نتیجه ارزیابی شرط True (درست) باشد، مجموعه دستورات اجرا خواهند شد، در غیر این صورت، از اجرای مجموعه دستورات صرف نظر خواهد شد و اولین دستور بعد از if اجرا خواهد شد.

در هنگام استفاده از ساختار if به نکات زیر دقت کنید:

۲. در این ساختار شرط می‌تواند مرکب باشد. یعنی، می‌توان با عملگرهای and، or شرط‌های مرکب را ایجاد نمود. به عنوان مثال، شرط مرکب می‌تواند به صورت زیر بیان گردد.

$x > 10$ and $x < 19$

این شرط بررسی می‌کند که x بین ۱۰ تا ۱۹ است یا خیر؟

۳. برای تست برابری باید از عملگر == استفاده کرد.

مثال ۱-۲. برنامه‌ای که عددی صحیح را خوانده، قدرمطلق آن را نمایش می‌دهد (هدف این برنامه آشنایی با ساختار if ساده است).

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
n = int(input("Enter a number:"))
if n < 0 :
    n = -n
print( n)
```

متغیر	هدف
n	عدد ورودی

۲. ماژول را ذخیره و اجرا کنید. اکنون جلوی number: عدد ۱۰- را وارد نمایید تا خروجی زیر را

ببینید:

Enter a number:-10

10

۲. ساختارهای دو انتخابی if، در ساختار if می‌توان کلمه کلیدی else را به کاربرد، در این صورت

این ساختار به صورت زیر استفاده می‌شود:

```
if شرط:
    مجموعه دستورات ۱
else:
    مجموعه دستورات ۲
```

در این ساختار ابتدا شرط ارزیابی می‌شود. اگر نتیجه ارزیابی شرط درست (True) باشد، مجموعه دستورات ۱ و گرنه (نتیجه ارزیابی شرط False باشد)، مجموعه دستورات ۲ اجرا خواهند شد.

مثال ۲-۲. برنامه‌ای که عددی را خوانده، تشخیص می‌دهد زوج است یا فرد (هدف این برنامه آشنایی با ساختار دو انتخابی if – else است).

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
n = int(input("Enter a number:"))
if n % 2 == 1:
    print("Odd")
else:
    print("Even")
```

متغیر	هدف
n	عدد ورودی

۲. ماژول را ذخیره و اجرا کرده، عدد ۱۵ را وارد کنید تا خروجی زیر را ببینید:

```
Enter a number:15
Odd
```

```
شرط:
    مجموعه دستورات ۱
elif شرط ۲:
    مجموعه دستورات ۲
:
elif شرط n:
    مجموعه دستورات n
else:
    مجموعه دستورات n+1
```

۳. ساختارهای چند انتخابی، در این ساختار دستور if را می‌توان توسعه داد و بخش‌های بیش‌تری را با شرط‌های مختلف ایجاد نمود، این دستور به صورت زیر به کار می‌رود:

در این ساختار ابتدا شرط ۱ ارزیابی می‌شود، چنانچه برابر True (درست) باشد، مجموعه دستورات ۱ اجرا

می‌شوند و سپس دستور بعد از مجموعه دستورات n+1 اجرا خواهند شد، اگر شرط ۱ درست نباشد، شرط ۲ ارزیابی می‌شود، اگر شرط ۲ درست (True) باشد، مجموعه دستورات ۲ اجرا می‌شوند، و سپس دستور بعد از مجموعه دستورات n+1 اجرا می‌گردد، اگر شرط ۲ درست نباشد، شرط ۳ ارزیابی می‌شود و این روند تا شرط n ادامه می‌یابد. اگر هیچ یک از شرط‌های ۱ تا n درست نباشد، مجموعه دستورات n+1 اجرا خواهند شد.

در هنگام استفاده از این ساختار به نکات زیر دقت کنید:

۱. تعدا بخش‌های elif اختیاری است و محدودیتی در آن وجود ندارد.

۲. بخش elif نمی‌تواند قبل از if یا بعد از else قرار بگیرد.

۳. در این ساختار وجود else اختیاری است.

مثال ۳-۲. برنامه‌ای که عددی را خوانده، تشخیص می‌دهد، مثبت، صفر یا منفی است (هدف برنامه آشنایی با if – elif – else است).

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
val = input("Enter a number:");
val = int(val)
if val > 0 :
    print ("Entered value is positive")
elif val == 0:
    print ("Entered value is zero")
else:
    print ("Entered value is negative")
```

متغیر	هدف
val	رشته و عدد
	ورودی

۲. ماژول را ذخیره و اجرا کرده، عدد ۱۲- را وارد کنید تا خروجی زیر را ببینید:

```
Enter a number:-12
Entered value is negative
```

مثال ۴-۲. برنامه‌ای که ضرایب یک معادله درجه ۲ را خوانده، ریشه‌های آن را محاسبه می‌کند و نمایش می‌دهد.

توضیح: برای محاسبه ریشه‌های معادله درجه ۲ ابتدا ضرایب a، b، c را می‌خوانیم. سپس دلتا

(delta) را به صورت زیر محاسبه می‌کنیم:

$$\text{delta} = b^2 - 4*a*c$$

اگر delta کوچک‌تر از صفر باشد، معادله ریشه ندارد.

وگرنه، اگر delta برابر صفر باشد، معادله دو ریشه مساوی دارد و ریشه‌های آن برابر است با:

$$x_1 = x_2 = -b / (2*a)$$

وگرنه (اگر $\text{delta} > 0$) معادله دارای دو ریشه مختلف است و ریشه‌های معادله به صورت زیر

محاسبه می‌شوند:

$$x_2 = \frac{-b - \sqrt{\text{delta}}}{2*a} \text{ و } x_1 = \frac{-b + \sqrt{\text{delta}}}{2*a}$$

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

متغیر هدف

آشنایی با زبان پایتون ۵۱

ضریب x^2	a
ضریب x	b
عدد ثابت معادله	c
مقدار دلتا	delta
ریشه اول	x1
ریشه دوم	x2

```
a = int(input("Enter a:"))
b = int(input("Enter b:"))
c = int(input("Enter c:"))
delta = b ** 2 - 4 * a * c
if delta < 0:
    print("Not root")
elif delta == 0:
    print("x1 = x2 = ", -b / (2.0 * a))
else:
    print("x1 = ", (-b + delta ** 0.5) / (2.0 * a))
    print("x2 = ", (-b - delta ** 0.5) / (2.0 * a))
```

۲. ماژول را ذخیره و اجرا کنید. اکنون جلوی a، b، و c به ترتیب مقادیر ۲، ۴ و ۲ را وارد

کرده تا خروجی زیر را مشاهده نمایید:

```
Enter a:2
Enter b:4
Enter c:2
x1 = x2 = -1.0
```

۵-۲. مسائل حل شده

مثال ۱. برنامه‌ای که نمره عددی دانشجویی را بر مبنای ۱۰۰ خوانده، باتوجه به جدول زیر نمره حرفی دانشجو را نمایش می‌دهد (در این برنامه متغیر grade نمره است):

نمره	پیغام
0-70	Fail
71-80	Good
81-90	Very Good
90-100	Excellent
<0, >100	Invalid Grade

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```

grade = int(input("Enter grade: "))
if (0 <= grade <= 70):
    print("Fail")
elif 71 <= grade <= 80:
    print("Good")
elif 81 <= grade <= 90:
    print("Very good")
elif 91 <= grade <= 100:
    print("Excellent")
else:
    print("Invalid grade")

```

۲. ماژول را ذخیره و اجرا کرده، عدد ۹۰ را وارد نمایید تا خروجی زیر را مشاهده کنید:

```

Enter grade: 90
Very good

```

مثال ۲. برنامه‌ای که n را خوانده، اعداد n تا ۱ نمایش می‌دهد و در پایان حاصل ضرب این اعداد را نمایش خواهد داد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```

n = int(input("Enter n:"))
p = 1
for i in range(n, 0, -1):
    print(i, end = '\t')
    p = p * i
print("\nMultiply is ", p)

```

متغیر	هدف
n	عدد خوانده شده
i	از n تا 1
p	حاصل ضرب اعداد n تا 1

۲. ماژول را ذخیره و اجرا کرده، عدد ۱۲ را وارد نمایید تا خروجی زیر را مشاهده کنید:

```

Enter n:12
12      11      10      9      8      7      6      5      4      3      1
Multiply is  479001600

```

مثال ۳. برنامه‌ای که عددی را خوانده، بزرگ‌ترین رقم آن را نمایش می‌دهد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```

n = int(input("Enter n:"))
max = n % 10
while n > 0:
    if max < n % 10 :
        max = n % 10
    n = n // 10
print("Max is ", max)

```

متغیر	هدف
n	عدد خوانده شده
max	بزرگ‌ترین رقم

آشنایی با زبان پایتون ۵۳

۲. ماژول را ذخیره و اجرا کرده، عدد ۸۵۶۴۹۰۱ را وارد کنید تا خروجی زیر را ببینید:

Enter n:8564901

Max is 9

مثال ۴: برنامه‌ای که عددی را خوانده، مشخص می‌نماید که آیا عدد خوانده شده اول است یا خیر؟ اگر عددی بر یک عدد کوچک‌تر یا مساوی نصف خودش به جز یک بخش پذیر باشد، عدد اول نیست.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
n = int(input("Enter n:"))
isPrime = True
for i in (2, n // 2 + 1):
    if n % i == 0:
        isPrime = False
        break
if (isPrime == True):
    print("Yes")
else:
    print("No")
```

متغیر	هدف
n	عدد خوانده شده
isPrime	آیا n اول است یا نه (اگر اول نباشد isPrime برابر False خواهد شد).
i	شمارنده‌ای که از 2 تا $n/2$ می‌شمارد.

۲. ماژول را ذخیره و اجرا کنید. عدد ۳۷ را وارد کرده تا خروجی زیر را مشاهده نمایید.

Enter n:37

Yes

مثال ۵: برنامه‌ای که رشته‌ای را خوانده، تعداد ارقام رشته را نمایش می‌دهد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
s = input("Enter a string:")
count = 0
for i in s:
    if '0' <= i <= '9':
        count = count + 1
print("Count is ", count)
```

متغیر	هدف
s	رشته دریافتی از ورودی
count	تعداد ارقام موجود در رشته S
i	هر کاراکتر رشته

۲. ماژول را ذخیره و اجرا کرده، اطلاعات زیر را وارد کنید تا خروجی را ببینید:

Enter a string:One equal 1 and Seven equal 7.

Count is 2

مثال ۶: برنامه‌ای که تعدادی عدد را خوانده، اعدادی که همه ارقام آن‌ها برابر باشند را نمایش می‌دهد. در پایان، میانگین اعدادی که تمام ارقام آن‌ها برابر است را نمایش می‌دهد. کاربر برای خروج از برنامه عدد ۹۹- را وارد می‌کند.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
count, sum = 0, 0
while 1:
    n = int(input("Enter a number:"))
    if n == -99:
        break
    r = n % 10
    m = n
    while n > 0:
        if n % 10 != r:
            break
    n = n // 10
    if n == 0:
        count = count + 1
        sum = sum + m
        print(m)
    if count > 0:
        print(sum / count)
```

متغیر	هدف
count	تعداد اعدادی که ارقام آنها برابر است.
sum	مجموع اعدادی که ارقام آنها برابر است.
n	عددی که هر بار می خواند
m	عدد n را خوانده شده را نگه داری می کند، چون با جدا کردن ارقام عدد n، صفر خواهد شد (یا تغییر می یابد)

۲. ماژول را ذخیره و اجرا کرده، اطلاعات زیر را تایپ کرده تا خروجی را ببینید:

```
Enter a number:111
111
Enter a number:156
Enter a number:555
555
Enter a number:889
Enter a number:7779
Enter a number:3444
Enter a number:987
Enter a number:444
444
Enter a number:-99
370.0
```

مثال ۷. برنامه ای که عددی را خوانده، اگر باقی مانده عدد به ۷، صفر بود، شنبه، یک بود، یکشنبه، دو بود، دوشنبه و همین طور اگر ۶ بود، جمعه را نشان می دهد (n عدد خوانده شده است).

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
n = int(input("Enter a number:"))
n = n % 7
if n == 0:
```


آشنایی با زبان پایتون ۵۵

۲. ماژول

را ذخیره و

اجرا کرده،

جلوی n عدد

۱۲ را وارد

کنید تا

خروجی زیر را

بینید:

```
print('Saturday')
elif n == 1:
    print('Sunday')
elif n == 2:
    print('Monday ')
elif n == 3:
    print('Tuesday')
elif n == 4:
    print('Wednesday')
elif n == 5:
    print('Thursday')
elif n == 6:
    print('Friday')
else:
    print('Invalid number ')
```

Enter a number:12

Thursday

فصل ۳

توابع

ساختار یک برنامه خیلی شبیه به ساختار یک سازمان است. یعنی، در هر سازمان ساختار سلسله مراتبی حاکم است. در بالاترین سطح سازمان، مدیریت قرار دارد. هر مدیر می تواند چند معاون داشته باشد. هر یک از معاونین نیز می توانند چندین کارمند داشته باشند. این برنامه ها مانند شرکت های کوچک هستند که مدیر شرکت همه کارهای شرکت را انجام می دهد. ولی، در بسیاری از سازمان ها چنین وضعیتی حاکم نیست. برنامه های واقعی و کاربردی مانند سازمان های بزرگ طولانی و پیچیده هستند.

مدیر برای انجام هر وظیفه اش یکی از توابع (معاونین) خودش را صدا می زند و احتمالاً پارامترهای (پرونده های) را در اختیار او قرار داده، از او می خواهد کار را انجام داده، نتیجه را برگرداند. هر یک از توابع (معاونین) نیز خود توابع دیگر (کارمندان خودش) را صدا می زنند تا بخشی از کار را به آنها محول نمایند و این روند تا انجام کار ادامه دارد. با این تفاسیر، کاربری که از برنامه استفاده می نماید، نقش مشتری را بازی خواهد کرد که می تواند اطلاعاتی را در اختیار سازمان (برنامه) قرار داده، نتایجی را دریافت کند.

استفاده از تابع در برنامه نویسی دارای مزایای زیر است:

۱. برنامه نویسی ساخت یافته را امکان پذیر می کند.
۲. خوانایی برنامه را افزایش می دهد. همچنین تست، اشکال زدایی و خطایابی برنامه نیز آسان تر خواهد شد. چون، برنامه ها به بخش های کوچک تری تبدیل می شوند. لذا، خطایابی و اصلاح برنامه - های کوچک تر آسان تر خواهد بود.

۳. می‌توان توابع مورد نیاز را در یک برنامه نوشت و از آن‌ها در برنامه‌های دیگر نیز استفاده کرد. این امر، استفاده مجدد^۲ نام دارد. بدین ترتیب، کد نویسی کمتر خواهد شد و تولید نرم‌افزار سریع‌تر انجام می‌شود.

۴. توابع، امکان کار گروهی را فراهم می‌کنند. زیرا، پس از این که برنامه به بخش‌های کوچک‌تری تقسیم شدند، هر یک از اعضای گروه وظیفه نوشتن و تست توابع مشخص را بر عهده می‌گیرند. بدین ترتیب، اعضای گروه به صورت هم‌زمان روی بخش‌های مختلف برنامه کار می‌کنند (بدون این که منتظر همدیگر باشند). انجام کار به صورت گروهی موجب می‌شود تا برنامه‌ها سریع‌تر آماده شوند.

۵. توابع امکان استفاده از کارهایی که دیگران انجام داده‌اند، را فراهم می‌کنند. یعنی، در برنامه-هایتان می‌توانید از توابعی که دوستانتان آماده کرده‌اند، استفاده کنید.

۶. توابع، امکان ایجاد کتابخانه را فراهم می‌کنند. کتابخانه، مجموعه توابعی هستند که مورد نیازتان می‌باشند (مجموعه توابعی که به هم مرتبط‌اند). بنابراین، می‌توان مجموعه توابع مرتبط به هم را در یک فایل کتابخانه (بسته) قرار داد و در برنامه‌ها از این فایل کتابخانه استفاده نمود.

۱ - ۳. انواع توابع

در هر زبان برنامه‌نویسی دو نوع تابع وجود دارد که عبارت‌اند از:

۱. **توابع کتابخانه‌ای**، توابعی می‌باشند که همراه کامپایلر یا مفسر وجود دارند. این توابع را توابع عمومی نیز می‌نامند. زیرا، کاربردهای زیادی دارند. توابع کتابخانه‌ای را با توجه به کاربرد آن‌ها دسته‌بندی کردند و هر یک از دسته‌ها را در فایل ماژول خاصی قرار داده‌اند. تاکنون با برخی از این توابع نظیر `print()`، `int()` آشنا شدید. در ادامه با بعضی از توابع مهم کتابخانه‌ای به همراه کاربرد آن‌ها آشنا خواهید شد.

۲ - ۳. توابعی که برنامه نویس می‌نویسد

همان‌طور که می‌دانید پایتون شامل ماژول‌های متنوعی است. اما، با این وجود، توابع موجود در ماژول پایتون، پاسخ‌گوی همه در خواست‌های برنامه‌نویس نیستند. لذا، برنامه‌نویس باید بتواند توابعی را نوشته، از آن‌ها استفاده کند. برای این منظور، برنامه‌نویس باید دو کار زیر را انجام دهد:

۱. نوشتن تابع
۲. فراخوانی تابع

۱ - ۲ - ۳. نوشتن تابع

برای نوشتن تابع باید آن را تعریف کرد:

تعریف تابع

قبل از این که تابعی را بنویسید باید تابع را تعریف کنید. تعریف تابع تعیین می‌کند، این تابع چه ورودی‌های دارد، چه چیزی را برمی‌گرداند (خروجی تابع چیست) و چه عملی را انجام می‌دهد. الگوی (امضای) تابع، به صورت زیر است:

def (لیست پارامترها) نام تابع :

بدنه تابع

توابع از لحاظ مقداری که برمی‌گردانند به سه نوع زیر تقسیم می‌شوند:

۱. توابعی که هیچ مقداری را برنمی‌گردانند (مثال ۱ - ۳ را ببینید).
۲. توابعی که فقط یک مقدار را برمی‌گردانند. برخی از این توابع عبارت‌اند از:
 ۱. تابعی که بزرگ‌ترین مقدار بین سه عدد را برمی‌گرداند.
 ۲. تابعی که تعیین می‌کند عددی اول است یا خیر؟
 ۳. تابعی که تعیین می‌کند عددی کامل (تام) است یا خیر؟
 ۴. تابعی که حاصل ضرب دو عدد را برمی‌گرداند.
 ۵. و غیره

توابع برای برگشت مقدار از دستور return استفاده می‌کنند. دستور return به صورت‌های زیر به کار می‌رود:

1. return مقدار;

2. return (عبارت);

3. return;

ساختار اول، یک مقدار را برمی گرداند. به عنوان مثال، دستورات زیر را ببینید:

```
return False;
return 10;
```

دستور اول، مقدار False را برمی گرداند و دستور دوم، مقدار ۱۰ را برگشت خواهد داد.

اما، ساختار دوم، یک عبارت را ارزیابی کرده، نتیجه ارزیابی عبارت را برگشت خواهد داد. به عنوان مثال، دستور زیر را مشاهده کنید:

```
return (2 * i - 3)
```

این دستور، ۲ را در i ضرب کرده، ۳ واحد از این حاصل کم می کند و برمی گرداند.

ساختار سوم، بدون این که تابع مقداری را برگرداند، از تابع برمی گردد.

۳. توابعی که چندین مقدار را برمی گردانند. این نوع توابع را در ادامه می بینید.

🚀 نام تابع، از قانون نام گذاری شناسه ها و متغیرها پیروی می کند و برای دسترسی به تابع به کار می رود.

🚀 پارامترهای تابع، اطلاعاتی هستند که در هنگام فراخوانی تابع باید به آن ارسال گردند. توابع می توانند از لحاظ تعداد پارامترهایی که می پذیرند به گروه های زیر تقسیم گردند:

۱. توابع بدون پارامتر، این توابع معمولاً برای چاپ پیغام مشخصی به کار می روند.
۲. تابع ممکن است یک پارامتر داشته باشد. در این صورت باید نام پارامتر تعیین گردد.

برخی از این توابع عبارت اند از:

🚀 تابعی که عددی را دریافت کرده، تعیین می کند اول است یا خیر؟

🚀 تابعی که عددی را دریافت کرده، تعیین می کند تام است یا خیر؟

🚀 تابعی که عددی را دریافت کرده، فاکتوریل آن را برمی گرداند.

🚀 تابعی که عددی را دریافت کرده، مجموع ارقام آن را برمی گرداند.

🚀 تابع ممکن است چندین پارامتر داشته باشد. در این صورت باید پارامترها با استفاده از کاما (,) از

هم جدا شوند. برخی از این توابع در زیر آمده اند:

🌈 تابعی که دو عدد را دریافت کرده، بزرگ‌ترین مقسوم علیه مشترک آن‌ها را برمی‌گرداند (این تابع دارای دو پارامتر است).

🌈 تابعی که سه عدد را دریافت کرده، کوچک‌ترین عدد را برمی‌گرداند (این تابع سه پارامتر دارد).

🌈 تابعی که دو عدد را دریافت کرده، اولین عدد را به توان عدد دوم رسانده و برمی‌گرداند.

🌈 تابعی که دو عدد را گرفته، محتویات آن‌ها را تعویض می‌کند.

🌈 و غیره.

بدنه تابع

عملی که تابع باید انجام دهد، در بدنه تابع قرار می‌گیرد. بدنه تابع، مجموعه دستوراتی هستند که تابع باید اجرا کند.

۲ - ۳. فراخوانی تابع

دستوری که تابع را صدا زده، از آن استفاده می‌کند، **فراخوانی تابع** نام دارد. فراخوانی تابع به صورت زیر انجام می‌شود:

(لیست آرگومان‌ها) نام تابع

در هنگام نوشتن و استفاده از توابع باید به نکات زیر توجه کنید:

۱. تعداد پارامترها (در هنگام تعریف تابع) باید با تعداد آرگومان‌ها (در هنگام فراخوانی)

یکسان باشد (ممکن است نام آن‌ها یکی نباشد).

۲. در پایتون می‌توان تابعی را در داخل تابع دیگر تعریف کرد.

درک عملکرد تابع

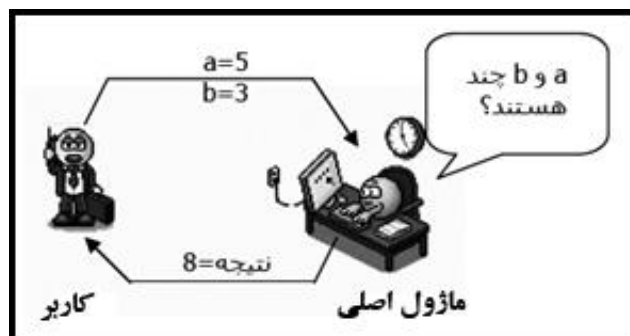
برای درک عملکرد توابع، برنامه‌ای را بدون استفاده از توابع و سپس از طریق توابع پیاده‌سازی می‌کنیم. با همین مثال نیز چگونگی تبدیل یک برنامه معمولی به توابع را می‌آموزیم. فرض کنید، بخواهید برنامه‌ای بنویسید که دو عدد را خوانده، حاصل جمع آن‌ها را نمایش دهد. همان‌طور که قبلاً دیدید، این برنامه به صورت زیر پیاده‌سازی می‌شود (روش اول):

```
a = int(input("Enter a:"))
b = int(input("Enter b:"))
c = a + b
```

آشنایی با زبان پایتون ۶۱

در این `print(a, "+", b, "=", c)`

برنامه، ماژول اصلی همه کاره است. یعنی، دو عدد صحیح را از کاربر (مشتري) گرفته، خودش حاصل جمع دو عدد را محاسبه کرده، چاپ می کند (در اختیار مشتري قرار می دهد). این فرآیند در شکل ۱-۳ آمده است.



شکل ۱-۳ فرآیند اجرای برنامه.

در روش دوم پیاده سازی، ماژول اصلی دو عدد a و b را از کاربر گرفته، در اختیار تابع `addition` (کارمند خودش) قرار می دهد تا حاصل جمع این دو عدد را حساب کند. تابع `addition` پس از محاسبه حاصل جمع دو عدد (مانند کارمند) نتیجه را در اختیار ماژول اصلی (مدیریت) قرار می دهد و ماژول اصلی این نتیجه را به کاربر (مشتري) می دهد. پیاده سازی این روش به صورت زیر است:

```
def addition(a, b):
    return a + b
a = int(input("Enter a:"))
b = int(input("Enter b:"))
c = addition(a, b)
print(a, "+", b, "=", c)
```

مثال ۵-۳. برنامه ای که عدد n را خوانده، اعداد کامل (تام) ۱ تا n را نمایش می دهد (هدف این برنامه به کارگیری مجدد تابع `isPerfect()` نوشته شده در مثال ۴-۳ می باشد).

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

تابع	م	هدف
تغییر		
ماژول	n	عدد ورودی

۶۲ فصل اول

۲. ماژول را ذخیره و اجرا کرده، عدد ۱۰۰۰ را وارد کنید تا خروجی زیر را ببینید:

```
def sum(n):
    s = 0
    for i in range(1, n):
        if(n % i == 0):
            s += i
    return s
def isPerfect(n):
    return(n == sum(n))
n=int(input("Enter n:"))
for i in range(1, n + 1):
    if isPerfect(i) == True:
        print(i)
```

اصلی	i	شمارنده از ۱ تا n
sum	s	مجموع مقسوم علیه‌ها
	i	شمارنده از ۱ تا n
	n	پارامتر ورودی
isPerfect	n	عدد به عنوان پارامتر

Enter n:1000
6
28
496

مثال ۶-۳. برنامه‌ای که x و n را خوانده، حاصل عبارت زیر را نمایش می‌دهد:

توضیح: در این برنامه تابع fact() برای محاسبه فاکتوریل پیاده‌سازی شده است.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

تایم	هدف	تغییر
ماژول اصلی	عدد ورودی	x
	عدد ورودی	n
	شمارنده ۳ تا n با گام افزایش ۳	i
	مجموع سری	sum
	علامت یکی در میان مثبت و منفی	sign
fact	پارامتر ورودی	n
	شمارنده ۱ تا n	i
	فاکتوریل n	f

```
def fact(n):
    f = 1.0
```


آشنایی با زبان پایتون ۶۳

```
for i in range(1, n+1):
    f *= i
    return f
x=int(input("Enter x:"))
n=int(input("Enter n:"))
sum = 0.0
sign = -1
for i in range(3, n + 1, 3):
    sum = sum + (x ** i) / fact(i) * sign
    sign = -sign
print("Sum is ", sum)
```

۲. ماژول را ذخیره و اجرا کرده، اعداد ۳ و ۲۰ را وارد کنید تا خروجی زیر را ببینید:

```
Enter x:3
Enter n:20
Sum is -3.540642507299644
```

مثال ۷-۳. برنامه‌ای که عددی را خوانده، اعداد مربعی ۱ تا آن عدد را نمایش می‌دهد. چند عدد مربعی عبارت‌اند از:

1 4 9 16 25 36 49 64 ...

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
def isSquare(n):
    i = 1
    while i * i <= n:
        if (i * i == n):
            return True
        i = i + 1
    return False
n = int(input("Enter n:"))
for i in range(1, n + 1):
    if isSquare(i) == True:
        print(i, end = '\t')
```

تابع	متغیر	هدف
ماژول اصلی	n	عدد ورودی
	i	شمارنده از ۱ تا n
isSquare	n	پارامتری که باید تعیین شود مربعی است یا نه
	i	شمارنده از ۱ تا جذر n

۲. ماژول را ذخیره و اجرا کرده، عدد ۱۰۰ را وارد کنید تا خروجی زیر را ببینید:

```
Enter n:100
1      4      9      16      25      36      49      64      81      100
```

۳- مسائل حل شده

مثال ۱. برنامه‌ای که عددی را خوانده، رقم اول از سمت چپ را بتوان رقم دوم، نتیجه را به توان رقم سوم و همین روند را ادامه می‌دهد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

تابع	متغیر	هدف
ماژول اصلی	n	مقلوب x
	x	عدد ورودی
Reverse	پارا متر n	عددی که باید مقلوب آن حساب شود
	s	مقلوب عدد n
powDigits	پارا متر n	عددی که باید رقم آن به توان رقم بعدی برسد (از سمت راست)
	p	توان ارقام

```
def reverse(n):
```

```
    s = 0
    while n > 0:
        s = s * 10 + n % 10
        n = n // 10
    return s
```

```
def powDigits(n):
```

```
    p = n % 10
    n = n // 10
    while n > 0:
        p = p ** (n % 10)
        n = n // 10
    return p
```

```
x = int(input("Enter x:"))
```

```
n = reverse(x)
```

```
print("Result:", powDigits(n))
```

ماژول اصلی، ابتدا عدد را خوانده، آن را مقلوب می‌کند و در n قرار می‌دهد، سپس با فراخوانی

تابع powDigits() ارقام مقلوب شده را از سمت راست به توان هم می‌رساند.

تابع reverse() عدد n را به عنوان پارامتر دریافت کرده، آن را مقلوب می‌نماید و در s قرار می‌-

دهد و s را برمی‌گرداند.

آشنایی با زبان پایتون ۶۵

تابع `powDigits()` پارامتر `n` را دریافت کرده، ارقام آن را از سمت راست جدا کرده، هر رقم را به توان رقم دیگر می‌رساند (در `p` قرار می‌دهد) و نتایج را به توان رقم بعدی می‌رساند. این عمل را تا آخرین رقم ادامه می‌دهد و در پایان، `p` را برمی‌گرداند.

۲. پروژه را ذخیره و اجرا کرده، عدد ۲۳۵۱ را وارد کنید تا خروجی را به زیر ببینید:

Enter x:2351

Result: 32768

مثال ۲. برنامه‌ای که عددی را خوانده، تشخیص می‌دهد اول یا نام است. عددی اول است که مجموع مقسوم علیه‌های کوچک‌تر از خودش برابر یک باشد و عددی تام (کامل) است که مجموع مقسوم علیه‌های کوچک‌تر از خودش برابر خودش باشد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

تابع	متغیر	هدف
ماژول اصلی	x	عدد ورودی
sumDivided	پارا متر n	عددی که باید مجموع مقسوم علیه‌های آن حساب شود.
	s	مجموع مقسوم علیه‌ها
	i	شمارنده از ۱ تا n-1
isPrimary	پارا متر k	عددی که باید تعیین شود اول است یا خیر؟
isPerfect	پارا متر k	عددی که باید تعیین شود کامل است یا خیر؟

```
def sumDivided(n):
```

```
    s = 0
```

```
    for i in range(1, n):
```

```
        if n % i == 0:
```

```
            s += i
```

```
    return s
```

```
def isPrimary(n):
```

```
    return sumDivided(n) == 1
```

```
def isPerfect(n):
```

```

    return sumDivided(n) == n
x = int( input("Enter a number:"))
if isPrimary(x) == True :
    print("Yes primary")
else:
    print("No primary")
if isPerfect(x) == True :
    print("Yes perfect")
else:
    print("No perfect")

```

ماژول اصلی، عدد x را خوانده، با فراخوانی تابع isPrimary()، تعیین می‌کند اول است یا نه؟ در

ادامه با فراخوانی تابع isPerfect() تعیین می‌کند عدد خوانده شده تام است یا نه؟

تابع sumDivided() پارامتر n را دریافت کرده، مجموع مقسوم علیه‌های آن را محاسبه کرده،

برمی‌گرداند.

تابع isPrimary() پارامتر k را دریافت کرده، اگر مجموع مقسوم علیه‌های آن برابر با یک باشد،

مقدار True، و گرنه مقدار False را برمی‌گرداند.

تابع isPerfect() پارامتر k را دریافت کرده، اگر مجموع مقسوم علیه‌های آن برابر k باشد، True،

و گرنه False را برمی‌گرداند.

۲. ماژول را ذخیره و اجرا کنید. اکنون، عدد ۷ را وارد کرده تا خروجی زیر را ببینید:

```

Enter a number:7
Yes primary
No perfect

```

مثال ۳. برنامه‌ای که مجموع کلیه اعداد چهار رقمی بدون رقم صفر را محاسبه می‌نماید که بر ۷ بخش پذیر

هستند.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

تابع		متغ	هدف
ماژول اصلی	sum	مجموع اعدادی که ص در فر رقم آن‌ها نباشد و بر ۷ بخش پذیرند	
		اعداد ۱۰۰۰ تا ۹۹۹۹	

عددی که باید تعیین شود آیا رقم صفر در آن وجود دارد یا نه؟	n	nonZero
---	---	---------

```
def nonZero(n):
    while n > 0:
        if n % 10 == 0:
            return False
        n = n // 10
    return True
sum = 0
for i in range(1000, 10000):
    if i % 7 == 0 and nonZero(i) == True:
        sum += i
print("Sum:", sum)
```

🚩 **ماژول اصلی**، ابتدا sum را برابر صفر قرار داده، با استفاده از یک حلقه تکرار اعداد ۱۰۰۰ تا ۹۹۹۹

که در ارقام آن‌ها صفر نباشد و بر ۷ بخش پذیرند را با sum جمع کرده، در پایان، sum را نمایش می‌دهد.

🚩 **تابع nonZero()** پارامتر n را دریافت کرده، اگر در ارقام آن صفر نباشد، مقدار True، وگرنه مقدار False را برمی‌گرداند.

۲. ماژول را ذخیره و اجرا کرده تا خروجی زیر را ببینید:

Sum: 5211759

فصل

۴

آرایه‌ها و بسته NumPy

تاکنون برنامه‌هایی که نوشته شده‌اند، هر متغیر حداکثر یک مقدار را در یک لحظه نگه‌داری می‌کرد. یعنی، داده جدید جایگزین داده فعلی در یک مکان حافظه می‌شود. به عنوان مثال، دستورات زیر را در نظر بگیرید:

```
sum = 0
for i in range(5):
    num = int(input("Enter a number:"))
    sum += num
print(sum / 5)
```

این دستورات ۵ عدد را خوانده، میانگین آن‌ها را نمایش می‌دهند. حال، اگر بخواهید اعدادی که بزرگ‌تر از میانگین هستند را نمایش دهید، به داده‌های خوانده‌شده قبلی نیاز دارید. در این برنامه داده‌های خوانده‌شده در `num` قرار می‌گیرند. بنابراین، پس از خواندن ۵ عدد، فقط پنجمین عدد در `num` قرار دارد. جهت حل این مشکل باید داده‌های خوانده‌شده را نگه‌داری نمود. برای این منظور، می‌توان به دو طریق زیر عمل کرد:

۱. می‌توان پنج متغیر (با نام‌های متفاوت) در نظر گرفت و هر داده را در یک متغیر ذخیره کرد. این روش دو مشکل عمده دارد که عبارت‌اند از:

✚ اگر تعداد مقادیر زیاد شود، تعداد متغیرها نیز زیاد خواهد شد.

✚ از حلقه تکرار نمی‌توان برای پردازش مقادیر استفاده نمود.

لذا، این روش معقول نمی‌باشد.

۲. می‌توان از ساختار داده جدیدی به نام **آرایه**^۱ استفاده کرد. آرایه، مجموعه‌ای از عناصر است که دارای ویژگی‌های زیر باشند:

^۱.Array

چند خانه از حافظه که دارای یک نام باشند. اگر چند نام برای خانه‌های حافظه در نظر گرفته شود، همان مشکل تعریف متغیر و غیرقابل پردازش بودن توسط حلقه‌های تکرار وجود خواهد داشت. دارای یک نوع باشند و به صورت پشت سرهم در حافظه ذخیره شوند. چون، اگر عناصر آرایه از یک نوع نباشند و به صوت پشت سرهم ذخیره نشوند، پیمایش عناصر مشکل خواهد شد. بنابراین، عناصر آرایه باید دارای یک نوع باشند و به صورت پشت سرهم ذخیره گردند تا با داشتن آدرس شروع آرایه (همان نام آرایه)، بتوان آدرس هر خانه را به سادگی حساب کرد. این عمل به صورت زیر انجام می‌شود:

$$\text{نوع آرایه} * \text{sizeof} + i = \text{آدرس شروع آرایه} = \text{آدرس خانه } i$$

به عنوان مثال، اگر آدرس شروع آرایه a با نوع صحیح، ۱۰۰۰ باشد، آدرس شروع خانه سوم به صورت زیر محاسبه می‌گردد:

$$\text{آدرس شروع خانه سوم} = 1000 + 3 * \text{sizeof}(\text{int}) = 1000 + 12 = 1012 = 100\text{CH}$$

برای دسترسی به عناصر آرایه از اندیس^۱ (شماره خانه) استفاده می‌شود. به همین دلیل، نام دیگر آرایه‌ها متغیرهای اندیس دار است. آرایه‌ها می‌توانند با توجه به تعداد اندیس آن‌ها چند نوع باشند که عبارت‌اند از:

۱. آرایه‌های یک‌بعدی، دارای یک اندیس هستند.

۲. آرایه‌های دوبعدی، دارای دو اندیس هستند.

۳. آرایه‌های چندبعدی، دارای چند اندیس هستند.

۱-۴. آرایه‌های یک‌بعدی

همان‌طور که بیان گردید، آرایه‌های یک‌بعدی، یک اندیس دارند. برای استفاده از آرایه‌ها باید دو

عمل زیر انجام شود:

۱-۴-۱. تعریف آرایه

^۱.Index

برای تعریف آرایه می‌توانید از کلاس array استفاده کنید. این کلاس در ماژول array قرار دارد. لذا، برای استفاده از این کلاس ابتدا باید ماژول array را با دستور زیر به برنامه اضافه کنید:

```
from array import *
```

اکنون می‌توانید از کلاس آرایه به صورت زیر استفاده کنید:

```
array(typecode [, initializer]) -> array
```

این کلاس دو پارامتر را دریافت می‌کند. پارامتر typecode، نوع آرایه را مشخص می‌کند. مقادیری که این پارامتر می‌پذیرد در جدول ۱ - ۴ آمده‌اند و پارامتر [initializer]، مقادیر اولیه آرایه را تعیین می‌کند.

جدول ۱-۴ مقادیر پارامتر TypeCode برای تعریف آرایه.					
مقدار	نوع C	حداکثر اندازه	مقدار	نوع C	حداکثر اندازه
'b'	عدد صحیح با علامت	۱ بایت	'B'	عدد صحیح بدون علامت	۱ بایت
'u'	کاراکتری یونیکد	۲ بایت	'h'	عدد صحیح با علامت	۲ بایت
'H'	عدد صحیح بدون علامت	۲ بایت	'i'	عدد صحیح با علامت	۲ بایت
'T'	عدد صحیح بدون علامت	۲ بایت	'l'	عدد صحیح با علامت	۴ بایت
'L'	عدد صحیح بدون علامت	۴ بایت	'q'	عدد صحیح با علامت	۸ بایت
'Q'	عدد صحیح بدون علامت	۸ بایت	'f'	عدد اعشاری	۴ بایت
'd'	عدد اعشاری	۸ بایت			

به عنوان مثال، دستورات زیر را ببینید:

```
>>> from array import *
>>> a = array('i', [])
```


آشنایی با زبان پایتون ۷۱

دستور اول، ماژول array را به برنامه اضافه می‌کند، دستور دوم، آرایه‌ای به نام a بدون هیچ عضوی با نوع صحیح ۱۶ بیتی تعریف می‌کند.

پس از ایجاد آرایه، اکنون می‌توانید با متدهای آن اعمالی را روی آرایه انجام دهید. برخی از این متدها و خواص کلاس array عبارت‌اند از:

متد **append()** برای اضافه کردن مقداری به انتهای آرایه به کار می‌رود. به‌عنوان مثال، دستورات زیر را ببینید:

```
>>> from array import *
>>> a = array('i', [])
>>> a.append(12)
>>> print(a)
```

این دستورات ابتدا، آرایه‌ای به نام a با نوع صحیح ۱۶ بیتی تعریف کرده، ۱۲ را به انتهای آن اضافه می‌کند و دستور آخر مقدار آرایه a (یعنی، array('i', [12])) را نمایش می‌دهد.

متد **buffer_info()** تاپلی را برمی‌گرداند که آدرس و طول آرایه می‌باشد.

متد **tobytes()** مقادیر آرایه را به بایت تبدیل می‌کند. به‌عنوان مثال، دستورات زیر را ببینید:

```
>>> from array import *
>>> a = array('I', [])
>>> for i in range(65, 80):
>>>     a.append(i)
>>> print(a.tobytes())
```

این دستورات، آرایه‌ای به نام a با نوع عدد صحیح بدون علامت تعریف کرده، مقادیر ۶۵ تا ۷۹ به

آرایه اضافه می‌کند و در پایان، اعضای آرایه را به بایت تبدیل می‌کند و نمایش می‌دهد (خروجی زیر):

```
b'A\x00\x00\x00B\x00\x00\x00C\x00\x00\x00D\x00\x00\x00E\x00\x00\x00F\x00\x00\x00G\x00\x00\x00H\x00\x00\x00I\x00\x00\x00J\x00\x00\x00K\x00\x00\x00L\x00\x00\x00M\x00\x00\x00N\x00\x00\x00O\x00\x00\x00'
```

متد **extend()** یک سری عناصر را به انتهای آرایه اضافه می‌کند. به‌عنوان مثال، دستورات زیر را

ببینید:

```
>>> from array import *
>>> a = array('i', [1, 2, 3])
>>> a.extend([3, 4])
>>> print(str(a))
```

این دستورات، ابتدا آرایه‌ای با نوع صحیح به نام `a` با مقادیر اولیه `[1, 2, 3]` ایجاد کرده، سپس مقادیر `[3, 4]` را به انتهای آن اضافه کرده و در پایان آرایه `a` را به صورت زیر نمایش می‌دهند:

```
array('i', [1, 2, 3, 3, 4])
```

متد `count()`، تعداد تکرار مقداری را در آرایه شمارش می‌کند. به عنوان مثال، دستورات زیر را

بینید:

```
>>> from array import *
>>> a = array('b', [1, 2, 3, 1, 4, 1])
>>> print(a.count(1))
```

این دستورات، ابتدا آرایه‌ای به نام `a` با نوع بایت با مقادیر اولیه `[1, 2, 3, 1, 4, 1]` تعریف می‌کنند و در پایان، تعداد عناصر آرایه را شمارش کرده، (یعنی ۳) نمایش می‌دهد.

متد `index()` مکان اولین وقوع مقداری را برمی‌گرداند. اگر مقدار در آرایه موجود نباشد، پیغام

خطای صادر خواهد شد. به عنوان مثال، دستورات زیر را بینید:

```
>>> from array import *
>>> a = array('b', [1, 2, 3, 1, 4, 2])
>>> print(a.index(2))
>>> print(a.index(5))
```

این دستورات، ابتدا آرایه‌ای به نام `a` با نوع بایت تعریف کرده، مقادیر اولیه `[1, 2, 3, 1, 4, 2]` را به آن تخصیص می‌دهند، سپس مکان اولین وقوع مقدار ۲ (یعنی، ۱) را نمایش می‌دهند و در پایان، چون مقدار ۵ در آرایه وجود ندارد، پیغام خطای زیر را صادر می‌کنند:

```
Traceback (most recent call last):
  File "<pyshell#57>", line 1, in <module>
    print(a.index(5))
ValueError: array.index(x): x not in list
```

متد `insert()`، عنصری را در مکان خاصی از آرایه اضافه می‌کند. این متد دو پارامتر را می‌گیرد،

پارامتر اول، مکانی است که مقدار پارامتر دوم باید قبل از آن درج شود. اگر مقدار پارامتر اول، بیش-

تر از تعداد عناصر آرایه باشد، مقدار پارامتر دوم را به انتهای آرایه اضافه می‌کند. دستورات زیر را

بینید:

```
>>> from array import *
>>> a = array('B', [1, 2, 3, 1, 4, 2])
>>> a.insert(1, 45)
```

آشنایی با زبان پایتون ۷۳

```
>>> print(a)
>>> a.insert(10, 45)
>>> print(a)
```

این دستورات، ابتدا آرایه `a` را با نوع بدون علامت تعریف می‌کنند، سپس مقادیر `[1, 2, 3, 1, 4]` را به آن تخصیص می‌دهند، در ادامه، مقدار `۴۵` را به قبل از مقدار `۲` (اندیش یک) اضافه کرده، آرایه `a` (یعنی `array('B', [1, 45, 2, 3, 1, 4, 2])`) را نمایش می‌دهند و در پایان، مقدار `۴۵` را به انتهای آرایه `a` اضافه نموده و آرایه `a` (یعنی `array('B', [1, 45, 2, 3, 1, 4, 2, 45])`) را نمایش می‌دهند.

متد `remove()`، مقداری را از آرایه حذف می‌کند (اولین وقوع مقدار را از آرایه حذف می‌کند). اگر مقدار در آرایه وجود نداشته باشد، پیغام خطای صادر خواهد شد. به‌عنوان مثال، دستورات زیر را در نظر بگیرید:

```
>>> from array import *
>>> a = array('d', [1, 2, 3, 1])
>>> a.remove(1)
>>> print(a)
>>> a.remove(4)
```

این دستورات، ابتدا آرایه‌ای به نام `a` با نوع اعشاری با دقت مضاعف کرده، مقادیر `[1, 2, 3, 1]` را به اعضای آن تخصیص می‌دهند، سپس اولین مقدار `۱` را حذف کرده، آرایه `a` (یعنی `array('d', [2.0, 3.0, 1.0])`) را نمایش می‌دهند. دستور آخر، می‌خواهد مقدار `۵` را از آرایه `a` حذف کند، چون در آرایه وجود ندارد، پیغام خطای زیر را صادر می‌کند:

```
Traceback (most recent call last):
  File "<pyshell#71>", line 1, in <module>
    a.remove(4)
```

ValueError: array.remove(x): x not in list

متد `pop()`، مقدار عنصری از آرایه را با توجه به اندیس آن برمی‌گرداند و آن عنصر را از آرایه حذف می‌کند. اگر اندیس که برای برگرداندن عناصر آرایه به کار رود بزرگ‌تر یا مساوی تعداد عناصر آرایه و کوچک‌تر از صفر باشد، پیغام خطای صادر می‌شود. به‌عنوان مثال، دستورات زیر را ببینید:

```
>>> from array import *
>>> a = array('d', [1, 2, 3, 4])
>>> print(a.pop(2))
```

```
>>> print(a.pop(5))
>>> print(a)
```

دستور دوم، آرایه‌ای به نام `a` با نوع عدد اعشاری با دقت مضاعف تعریف کرده، مقادیر `1.0, 2.0, 3.0, 4.0` را به آن تخصیص می‌دهد. دستور سوم، عنصر سوم (یعنی، عنصر با اندیس ۲) آرایه را برمی‌گرداند و از آرایه حذف می‌نماید (مقدار `3.0` را نمایش می‌دهد)، دستور چهارم، می‌خواهد عنصری با اندیس ۵ را برگرداند و از آرایه حذف کند، چون این عنصر در آن وجود ندارد، پیغام خطای زیر را صادر می‌نماید:

```
Traceback (most recent call last):
  File "<pyshell#75>", line 1, in <module>
    print(a.pop(5))
IndexError: pop index out of range
```

در پایان، عناصر آرایه `a` را نمایش می‌دهد (خروجی زیر):

```
array('d', [1.0, 2.0, 4.0])
```

متد `reverse()` عناصر آرایه را معکوس می‌کند. به عنوان مثال، دستورات زیر را در نظر بگیرید:

```
>>> from array import *
>>> a = array('d', [1, 2, 3, 4])
>>> a.reverse()
>>> print(a)
```

این دستورات، آرایه‌ای به نام `a` با نوع اعشاری با دقت مضاعف و مقادیر اولیه `[1.0, 2.0, 3.0, 4.0]` تعریف کرده، آن را معکوس می‌نمایند و نمایش می‌دهند (خروجی زیر):

```
array('d', [4.0, 3.0, 2.0, 1.0])
```

خاصیت `itemsize` اندازه هر عنصر آرایه را برمی‌گرداند. به عنوان مثال، دستورات زیر را ببینید:

```
>>> from array import *
>>> a = array('d', [1, 2, 3, 4])
>>> print(a.itemsize)
```

این دستورات، آرایه‌ای به نام `a` با نوع اعداد اعشاری با دقت مضاعف و مقادیر `1.0, 2.0, 3.0, 4.0` تعریف کرده، اندازه هر عنصر آرایه `a` (یعنی، ۸) را نمایش می‌دهند.

خاصیت `typecode` مقدار typecode آرایه‌ای را برمی‌گرداند. به عنوان مثال، دستورات زیر را

ببینید:

```
>>> from array import *
>>> a = array('I', [1, 2, 3, 4])
```

```
>>> print(str(a.typecode))
```

این دستورات، آرایه‌ای به نام `a` با نوع عددی صحیح ۱۶ بیتی بدون علامت تعریف کرده، مقادیر `[1, 2, 3, 4]` را به آن تخصیص می‌دهند، در پایان، مقدار `typecode` آرایه (یعنی، همان `I`) را نمایش می‌دهند.

۲-۱-۴. دسترسی به عناصر آرایه

همان‌طور که بیان گردید، برای دسترسی به عناصر آرایه از اندیس آن به‌صورت زیر استفاده می‌شود:

[اندیس] نام آرایه

اندیس، شماره خانه آرایه را تعیین می‌کند. اندیس آرایه در پایتون از صفر شروع می‌شود. بنابراین، حداکثر مقداری که اندیس می‌تواند بپذیرد، برابر با `[۱-تعداد عناصر آرایه]` است. یعنی، آرایه‌ای با ۵ عنصر، به‌صورت زیر نمایش داده می‌شود:

a[0]	a[1]	a[2]	a[3]	a[4]
------	------	------	------	------

همان‌طور که در شکل می‌بینید، آخرین خانه آرایه دارای اندیس ۴ است. اکنون دستورات زیر را ببینید:

```
a[2] = int(input("Enter a number:"))
print(a[2])
```

دستور اول، عنصر سوم آرایه (`a[2]`) را می‌خواند و دستور دوم، مقدار عنصر سوم آرایه (`a[2]`) را نمایش می‌دهد.

مثال ۲-۴. برنامه‌ای که ابتدا `n` را خوانده، `n` عدد تصادفی بین ۲۰- تا ۲۰ تولید کرده، در آرایه‌ای قرار می‌دهد. سپس اعمال زیر را انجام می‌دهد:

۱. بزرگ‌ترین عدد و مکان آن را نمایش می‌دهد.
۲. دومین عدد از لحاظ کوچکی و مکان آن را نمایش می‌دهد.
۳. عناصر آرایه را معکوس می‌نماید. یعنی، جای اولین عدد و آخرین عدد، یکی مانده به آخرین عدد و دومین عدد را تعویض می‌کند و همین روند را ادامه می‌دهد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

تابع	متغیر	هدف
main	a	آرایه‌ی ورودی
	n	تعداد عناصر آرایه
	max	بزرگ‌ترین عنصر آرایه
	lmax	مکان بزرگ‌ترین عنصر آرایه
	min2	دومین عدد کوچک
	lmin	مکان دومین عدد کوچک
createRandom	n	تعداد عناصر آرایه (تعداد اعداد تصادفی)
	i	شمارنده‌ای که از صفر تا n-1 شمارش می‌کند.
printArray	a	آرایه‌ای که اعضای آن باید نمایش داده شود.
	i	هر یک از عناصر آرایه a
findMax	A	آرایه‌ای که بزرگ‌ترین عنصر آن باید پیدا شود.
	N	تعداد عناصر آرایه a
	Max	بزرگ‌ترین عنصر
	Lmax	مکان بزرگ‌ترین عنصر
	I	اندیس عناصر از ۱ تا n-1 شمارش می‌کند.
findSecondMin	A	آرایه‌ای که دومین عدد از لحاظ کوچکی بزرگ‌ترین آن باید پیدا شود.
	N	تعداد عناصر آرایه a
	min1	کوچک‌ترین عدد
	lmin1	مکان کوچک‌ترین عدد
	min2	دومین عدد از لحاظ کوچکی
	lmin2	مکان دومین عدد از لحاظ کوچکی
	I	شمارنده‌ای که از ۲ تا n-1 را شمارش می‌کند

```

from array import *
import random
def createRandom(n):
    a = array('i', [])

```

```

        for i in range(0, n):
            a.append(random.randint(-20, 20))
        return a
def printArray(a):
    for i in a:
        print(i, end = '\t')
    print()
def findMax(a, n):
    max = a[0]
    lMax = 0
    for i in range(1, n):
        if a[i] > max:
            max = a[i]
            lMax = i
    return max, lMax
def findSecondMin(a, n):
    if a[0] < a[1]:
        min1 = a[0]
        min2 = a[1]
        lMin1 = 0
        lMin2 = 1
    else:
        min1 = a[1]
        min2 = a[0]
        lMin1 = 1
        lMin2 = 0
    for i in range(2, n):
        if a[i] < min1:
            min2 = min1
            lMin2 = lMin1
            min1 = a[i]
            lMin1 = i
        elif a[i] < min2 :
            min2 = a[i]
            lMin2 = i
    return min2, lMin2
def main():
    a = array('i', [])
    n = int(input("Enter n:"))
    a = createRandom(n)
    print("Orginal array ")
    printArray(a)

```

```

max, lMax = findMax(a, n)
print("Max is ", max, " Location max is ", lMax)
min2, lMin2 = findSecondMin(a, n)
print("Second min is ", min2, " Location second min is ", lMin2)
a.reverse()
print("Reverse array ")
printArray(a)
main()

```

متد **createRandom()**، تعداد عناصر (n) را به عنوان پارامتر دریافت کرده، n عدد تصادفی بین

۲۰- تا ۲۰۰ ایجاد می نماید و در آرایه a قرار می دهد. در پایان، آرایه a را برمی گرداند.

متد **printArray()**، آرایه a را به عنوان پارامتر دریافت کرده، عناصر آن را نمایش می دهد.

متد **findMax()**، آرایه a و تعداد عناصر آن (n) را به عنوان پارامتر دریافت کرده، بزرگ ترین عنصر و مکان آن را برمی گرداند. برای این منظور، ابتدا عنصر اول آرایه a[0] را در max و 0 (مکان اولین عنصر) را در lMax قرار می دهد. سپس از دومین عنصر تا عنصر nام، عناصر را یکی، یکی با max مقایسه می کند (داخل for)، اگر عنصری بزرگ تر از max باشد، آن عنصر (یعنی، a[i]) را در max و مکان آن (یعنی، i) را در lMax قرار می دهد. در پایان، max و lMax را برمی گرداند.

متد **findSecondMin()**، آرایه a و تعداد عناصر آن (n) را به عنوان پارامتر دریافت کرده، دومین عدد از لحاظ کوچکی و مکان آن را برمی گرداند. برای این منظور، ابتدا اولین عدد آرایه (a[0]) با دومین عدد آرایه (a[1]) مقایسه می کند، عدد کوچک تر را در min1 و مکان آن را در lMin1 و عدد بزرگ تر را در min2 و مکان آن را در lMin2 قرار می دهد. در ادامه، با یک حلقه for از سومین عدد تا nامین عدد، ابتدا عدد iام را با min1 مقایسه می کند، اگر کوچک تر از min1 باشد، عدد min1 را در min2، مکان (lMin1)min1 را در lMin2 قرار داده، عدد iام را در min1 و i را در lMin1 قرار می دهد. وگرنه، اگر عدد iام کوچک تر از min2 باشد، عدد iام را در min2 و i را در lMin2 قرار می دهد. در پایان، min2 و lMin2 را برمی گرداند.

۲. ماژول را ذخیره و اجرا کنید تا خروجی زیر را مشاهده نمایید:

```

Enter n:8
Orginal array

```



```

15  6  -1  -7  8  -11  -7  18
Max is 18 Location max is 7
Second min is -7 Location second min is 3
Reverse array
18  -7  -11  8  -7  -1  6  15

```

۵-۴. بسته NumPy

عملیات پایه‌ای مورد استفاده در برنامه‌نویسی علمی، آرایه‌ها، ماتریس‌ها، حل‌کننده‌های معادله‌های دیفرانسیل، آمار و غیره را در برمی‌گیرد که به صورت پیش فرض به جز **شماری** از عملگرهای ساده ریاضیاتی که تنها بر روی یک متغیر، و نه آرایه‌ها و ماتریس، قابل استفاده‌اند. پایتون چنین قابلیت‌هایی را به طور ذاتی در اختیار ندارد. با بسته NumPy می‌توان برخی از این قابلیت‌ها را به زبان پایتون اضافه نمود. NumPy مختص پردازش‌های عددی به وسیله‌ی آرایه‌های چندبعدی `ndarrays` است که در آن آرایه‌ها می‌توانند محاسبات درایه به درایه را انجام دهند. بدون این که نیاز به تغییر یا اصلاح آرایه‌های NumPy باشد، می‌توان از فرمول‌های جبر خطی نیز استفاده کرد. علاوه بر این اندازه‌ی آرایه را می‌توان به صورت پویا تغییر داد.

دقت داشته باشید که بسته NumPy یک افزونه است که باید بر روی مفسر پایتون نصب گردد. یعنی، در هنگام نصب نرم‌افزار پایتون، این بسته به طور خودکار بر روی پایتون نصب نمی‌شود. فرآیند نصب NumPy بر روی سیستم عامل مختلف، متفاوت است. لذا، می‌توانند فرآیند نصب ماژول NumPy را از سایت انتشارات فناوری نوین دانلود کنید.

۱-۵-۴. آرایه‌های NumPy

NumPy، یک بسته بنیادی پایتون برای محاسبات علمی است. این بسته قابلیت‌های آرایه `N`بعدی، عملیات درایه به درایه، عملیات اصلی مانند جبر خطی توانایی فراخوانی کدها `Fortran`، `C` و `C++` را جمع‌آوری کرده است. عملیات بر روی آرایه‌ها از طریق NumPy تقریباً حدود ۲۵ برابر سریع‌تر از حالت معمولی انجام می‌شود. به همین دلیل، در این بخش بسته NumPy را می‌آموزیم.

تعریف آرایه NumPy

مهم‌ترین شیء تعریف‌شده در Numpy، یک نوع آرایه N بعدی که ndarray نامیده می‌شود، است. آرایه ndarray، مجموعه‌ای از عناصر هم نوع هستند. اندیس شروع عناصر صفر است. اندازه همه عناصر یکی است. هر عنصر در ndarray یک شیء از شیء نوع داده (dtype نامیده می‌شود)، است.

Numpy پایه با استفاده از تابع array در Numpy به صورت زیر ایجاد می‌شود:

Numpy.array

برای ایجاد آرایه می‌توانید از متد array به صورت زیر استفاده کنید:

Numpy.array(object, dtype = none, copy = tyue, order = none, subok = false, ndmin=0)

🚩 پارامتر object، شیء مربوطه را تعیین می‌کند.

🚩 پارامتر dtype، نوع عناصر آرایه را تعیین می‌کند.

🚩 پارامتر copy، تعیین می‌کند آیا شیء کپی شود یا نه؟

🚩 پارامتر order، روش ذخیره آرایه را تعیین می‌کند، 'C'، ذخیره‌سازی به صورت سطری، 'F'،

ذخیره‌سازی به صورت ستونی و 'A' (هر دو) می‌باشد.

🚩 پارامتر ndmin، حداقل تعداد ابعاد آرایه را مشخص می‌کند.

```
>>> import numpy as np
>>> a = np.array([1, 2, 3, 4])
>>> print(a)
```

به عنوان مثال، دستورات زیر را ببینید:

دستور اول، ماژول np را از بسته numpy به برنامه اضافه می‌کند، دستور دوم، آرایه یک بعدی با مقادیر ۱، ۲، ۳ و ۴ ایجاد می‌نماید و دستور سوم، عناصر آرایه ایجادشده a را نمایش می‌دهد (یعنی، [1 2 3 4]). اکنون دستورات زیر را ببینید:

```
>>> import numpy as np
>>> a = np.array([[1, 2], [3, 4]])
>>> print(a)
```

آشنایی با زبان پایتون ۸۱

دستور اول ماژول numpy را با نام np به برنامه اضافه می کند، دستور دوم، آرایه دوبعدی با دو سطر و دو ستون به برنامه اضافه می کند و دستور سوم، اطلاعات آرایه دوبعدی را نمایش می دهد (خروجی زیر):

```
[[1 2] [3 4]]
```

دستورات زیر یک آرایه ایجاد می کنند که نوع هر عنصر آن مختلط می باشد و سپس

خروجی (یعنی [1.+0.j 2.+0.j 3.+0.j 4.+0.j]) را نمایش می دهند:

```
>>> import numpy as np
>>> a = np.array([1, 2, 3, 4], dtype = complex)
>>> print(a)
```

توجه داشته باشید که نوع هر نوع آرایه (dtype) می تواند bool (مقادیر true یا false)، int (نوع صحیح پیش فرض ۳۲ یا ۶۴ بیتی)، intc (نوع صحیح زبان C) intp (عدد صحیح که برای شاخص گذاری استفاده می شود)، int8 (بایت)، int16 (عدد صحیح ۱۶ بیتی)، int32 (عدد صحیح ۳۲ بیتی) int64 (عدد صحیح ۶۴ بیتی) float (عدد اعشاری ۶۴ بیتی)، float16 (عدد اعشاری ۱۶ بیتی)، float32 (عدد اعشاری ۳۲ بیتی)، float64 (عدد اعشاری ۶۴ بیتی)، complex (عدد مختلط ۱۲۸ بیتی)، complex64 (عدد مختلط ۶۴ بیتی (دو عدد ۳۲ بیتی)) و complex128 (عدد مختلط ۱۲۸ بیتی) باشد.

متد dtype() این متد یک شیء نوع داده ایجاد می کند که به صورت زیر به کار می رود:

numpy.dtype (object, align, copy)

پارامتر object، برای تبدیل شیء به نوع داده به کار می رود.

پارامتر align، اگر true باشد، لایه گذاری اضافه می کند تا ساختار شبیه C را به وجود آورد.

پارامتر copy، اگر true باشد، یک کپی جدید از شیء dtype ایجاد می کند، وگرنه ارجاع

ایجاد شده در شیء نوع داده را برمی گرداند.

به عنوان مثال، دستورات زیر را ببینید:

```
>>> import numpy as np
>>> dt = np.dtype(np.int32)
>>> print(dt)
>>> dt1 = np.dtype('i4')
>>> print(dt1)
```

```
>>> dt2 = np.dtype([('age', np.int8)])
>>> print(dt2)
```

دستور اول، ماژول numpy را با نام np به برنامه اضافه می‌کند، دستور دوم، dt را با نوع int32 (صحیح ۳۲ بیتی) در نظر می‌گیرد، دستور سوم، مقدار dt را نمایش می‌دهد (همان int32)، دستور چهارم، dt1 را با نوع int32 در نظر می‌گیرد (زیرا به جای int8، int16، int32 و int64 به ترتیب می‌توان معادل آن‌ها 'i1'، 'i2'، 'i3' و 'i4' را جایگزین کرد)، دستور پنجم، مقدار dt1 (int32) را نمایش می‌دهد، دستور ششم، یک نوع داده ساخته یافته ایجاد کرده، در dt2 قرار می‌دهد و دستور هفتم، مقدار dt2 (['age', 'i1']) را نمایش می‌دهد.

اکنون دستورات زیر را ببینید:

```
>>> import numpy as np
>>> dt = np.dtype([('age', np.int8)])
>>> a = np.array([(5, ), (10, ), (15, )], dtype = dt)
>>> print(a['age'])
```

دستور اول، ماژول numpy را با نام np به برنامه اضافه می‌کند، دستور دوم، ساختاری از نوع دیکشنری ایجاد کرده، در dt قرار می‌دهد، دستور سوم، آرایه‌ای با سه عنصر به نام a با نوع dt ایجاد می‌نماید و دستور چهارم، اطلاعات آرایه a ([5 10 15]) را نمایش می‌دهد.

✚ **خاصیت ndarray.shape** این خاصیت تاپلی را برمی‌گرداند که شامل ابعاد آرایه است. از این

خاصیت می‌توان برای تغییر اندازه آرایه نیز استفاده کرد. به عنوان مثال، دستورات زیر را ببینید:

```
>>> import numpy as np
>>> a = np.array([[1, 3, 5], [2, 4, 6]])
>>> print(a.shape)
>>> a.shape = (3, 2)
>>> print(a)
```

دستور اول، ماژول numpy را با نام np به برنامه اضافه می‌کند، دستور دوم، آرایه‌ای دوبعدی 3×2 به نام a ایجاد می‌کند و عناصر آن را مقدار می‌دهد، دستور سوم، تعداد ابعاد آرایه a (۳، ۲) را نمایش می‌دهد، دستور چهارم، ابعاد آرایه a را به 3×2 (سه سطر و دو ستون) تغییر می‌دهد و دستور پنجم، محتوی آرایه a (3×2) را پس از تغییر ابعاد نمایش می‌دهد (خروجی زیر):

```
[[1 3]
 [5 2]]
```

[4 6]]

متد `reshape()`، برای تغییر ابعاد آرایه به کار می‌رود. این متد به صورت زیر استفاده می‌شود:

`reshape (shape, order = 'c')` نام آرایه

پارامتر `shape`، تعداد ابعاد را مشخص می‌کند. عملکرد پارامتر `order` را در متد `array` دیدید.

به عنوان مثال، دستورات زیر را ببینید:

```
>>> import numpy as np
>>> a = np.array([[1, 3, 5], [2, 4, 6]])
>>> b = a.reshape(3, 2)
>>> print(b)
```

دستور اول، ماژول `numpy` را با نام `np` به برنامه اضافه می‌کند، دستور دوم، آرایه‌ای 2×3 به نام `a` ایجاد می‌کند، دستور سوم، از طریق آرایه `a` یک آرایه 3×2 ایجاد کرده، در آرایه‌ای به نام `b` قرار می‌دهد و دستور چهارم، اطلاعات آرایه `b` را نمایش می‌دهد (خروجی زیر):

```
[[1 3]
 [5 2]
 [4 6]]
```

خاصیت `ndarray.ndim`، تعداد ابعاد آرایه را برمی‌گرداند.

خاصیت `ndarray.itemsize` اندازه عناصر آرایه را برمی‌گرداند. به عنوان مثال، دستورات زیر را

ببینید:

```
>>> import numpy as np
>>> a = np.array([[1, 3, 5], [2, 4, 6], [3, 7, 8]], dtype=np.float32)
>>> print(a.ndim)
>>> print(a.itemsize)
```

دستور اول، ماژول `numpy` را با نام `np` به برنامه اضافه می‌کند، دستور دوم، آرایه‌ای 3×3 به نام `a` با نوع `float32` ایجاد می‌کند، دستور سوم، تعداد ابعاد آرایه `a` (یعنی ۲) را نمایش می‌دهد، دستور چهارم، اندازه هر عنصر (یعنی ۴ برای `float32`) را نمایش می‌دهد.

متد `numpy.empty()` یک آرایه بدون مقدار اولیه ایجاد می‌کند که به صورت زیر به کار می‌-

رود:

`numpy.empty (shape, dtype, order)`

پارامترهای shape، dtype و order همانند پارامترهای متد array() هستند. به عنوان مثال،

دستورات زیر را ببینید:

```
>>> import numpy as np
>>> a = np.empty([2,2], dtype = np.int8)
>>> print(a)
```

دستور اول، ماژول numpy را با نام np به برنامه اضافه می کند، دستور دوم، یک آرایه 2×2 به

نام a که هر عنصر آن از نوع بایت است، ایجاد می کند و عناصر آن را به صورت تصادفی مقدار می -

دهد و دستور سوم، محتوی آرایه a را نمایش می دهد (خروجی زیر):

```
[[6 0] [0 0]]
```

متد **numpy.zeros()** آرایه ای را ایجاد کرده و مقادیر اولیه صفر به عناصر آن تخصیص

می دهد. این متد به صورت زیر به کار می رود:

numpy.zeros (shape, dtype, order)

عملکرد پارامترهای shape، dtype و order را در متد array() دیدید. به عنوان مثال، دستورات

زیر، آرایه ای یک بعدی ۴ عنصری با مقادیر صفر ایجاد کرده، نمایش می دهد (مقادیر [0. 0. 0. 0.]

: (0.)

```
>>> import numpy as np
>>> a = np.zeros (4)
>>> print(a)
```

متد **numpy.ones()** آرایه ای ایجاد کرده و عناصر آن را به یک پر می کند. این متد به صورت

زیر به کار می رود:

numpy.ones (shape, dtype, order)

عملکرد پارامترهای shape، dtype و order را در متد array() مشاهده کردید. دستورات زیر

آرایه ای به نام a با ابعاد 2×3 ایجاد کرده، عناصر آن را با ۱ پر می کند و در پایان، آرایه a را

نمایش می دهد.

```
>>> import numpy as np
>>> a = np.ones((2,3), dtype = np.float32)
>>> print(a)
```

آشنایی با زبان پایتون ۸۵

متد `numpy.asarray()` آرایه‌ای ایجاد کرده، عناصر آن را با داده‌ای موجود پر می‌کند. این متد به صورت زیر به کار می‌رود.

`numpy.asarray(a, dtype, order)`

`a`، پارامتری است که داده ورودی را تعیین می‌کند که آرایه باید با آن پر شود. این داده می‌تواند لیست تاپلی، تاپلی از تاپل یا تاپلی از لیست باشد. با پارامترهای `dtype` و `order` در متد `array()` آشنا شدید. به عنوان مثال، دستورات زیر اطلاعات لیست `x` را در آرایه `a` قرار می‌دهند و آرایه `a` (4. 1. [6. 8.] را نمایش می‌دهند:

```
>>> import numpy as np
>>> x = [1, 4, 6, 8]
>>> a = np.asarray(x, dtype = np.float64)
>>> print(a)
```

فصل ۵ رشته‌ها

رشته، یکی از پرکاربردترین انواع داده در پایتون است. رشته‌ها، دنباله‌ای از کاراکترها هستند که در داخل تک کتیشن (') یا جفت کتیشن (") قرار می‌گیرند. به عنوان مثال، دستوارت زیر را ببینید:

```
>>> url = 'www.fanavarienovin.net'
>>> language = "Python programming"
```

دستور اول، رشته‌ای به نام url تعریف کرده، مقدار 'www.fanavarienovin.net' را در آن قرار

می‌دهد، دستور دوم، متغیری به نام language با مقدار "Python.Programming" ایجاد می‌کند.

۱-۵. عملگرهای رشته‌ای

جدول ۱-۵ عملگرهای رشته‌ای.		
عملگر	نام	هدف
+	اتصال	بین دو رشته قرار گرفته، رشته‌ای را به رشته دیگر متصل می‌کند.
*	تکرار	بین یک رشته و یک عدد قرار گرفته، رشته را به تعداد عدد تکرار می‌کند و رشته جدیدی ایجاد می‌نماید.
[]	برش	کاراکتر با اندیس خاصی از رشته را برمی‌گرداند.
[:]	برش در یک محدوده مشخص	این عملگر دو اندیس می‌پذیرد که اندیس شروع (مشخص کننده اولین کاراکتر) و اندیس پایان (آخرین کاراکتر) هستند و کاراکترهای بین این دو اندیس را برمی‌گرداند.
In	عضویت	عضویت کاراکتری را در رشته بررسی می‌کند، در صورت موجود بودن کاراکتری در رشته True، وگرنه False را برمی‌گرداند.
not in	عضو نبودن	در صورت موجود نبودن کاراکتری در رشته True، وگرنه False را برمی‌گرداند.

r/R	رشته‌ی خام	عملکرد و معنی اصلی کاراکتر را لغو می‌کند.
%	فرمت دهی	یک رشته را قالب‌بندی می‌کند.

عملگرهایی وجود دارند که برای کار بر روی رشته‌ها استفاده می‌شوند. خلاصه این عملگرها در

جدول ۵-۱ آمده‌اند. در ادامه شرح کامل این عملگرها را می‌بینید.

عملگر +، برای اتصال دو رشته به کار می‌رود. به عنوان مثال، دستورات زیر را ببینید:

```
>>> name = "fanavarienovin"
>>> url = "www." + name + ".net"
>>> print(url)
```

دستور اول، رشته‌ای به نام name با مقدار "fanavarienovin" ایجاد می‌کند، دستور دوم، رشته‌های

"www." و ".net" را به ترتیب به ابتدا و انتهای رشته name اضافه می‌کند تا رشته url را تولید کند و

دستور سوم، محتوی url (یعنی، "www.fanavarienovin.net") را نمایش می‌دهد.

عملگر *، رشته‌ای را چندین بار تکرار می‌کند. به عنوان مثال، دستورات زیر را ببینید:

```
>>> s1 = "Python" * 5
>>> print(s1)
```

دستور اول، رشته "Python" را پنج بار تکرار کرده، در متغیر رشته‌ای به نام s1 قرار می‌دهد،

دستور دوم، محتوی رشته s1 (مقدار PythonPythonPythonPythonPython) را نمایش می‌دهد.

عملگر []، کاراکتر خاصی از رشته را برمی‌گرداند. به عنوان مثال، دستورات زیر را ببینید:

```
>>> s = "Python"
>>> print(s[3], s[4], s[1])
```

دستور اول، رشته s را با مقدار "Python" تعریف می‌کند. اکنون کاراکترهای رشته به صورت زیر

اندیس گذاری می‌شوند:

P	y	t	h	o	n
s[0]	s[1]	s[2]	s[3]	s[4]	s[5]

دستور دوم، مقادیر اندیس‌های 3 (h)، 4 (o) و 1 (y) را به ترتیب نمایش می‌دهد (مقدار

hoy).

دقت کنید که اگر بخواهید به اندیسی دسترسی داشته باشید که وجود ندارد، از طرف مفسر پایتون

خطا صادر می‌شود. به عنوان مثال، دستور زیر را ببینید:

```
>>> print(s[10])
```

چون اندیس ۱۰ برای رشته s وجود ندارد، پیغام خطای زیر صادر خواهد شد:

Traceback (most recent call last):

File "<pyshell#13>", line 1, in <module>

print(s[10])

IndexError: string index out of range

عملگر [n:m] (برش رشته‌ها)، برای برش رشته به کار می‌رود، به طوری که از کاراکتر n رشته تا

کاراکتر m آن را جدا می‌کند. قبل از این که به برش رشته پردازیم، نحوه ذخیره‌سازی و اندیس

گذاری رشته را شرح می‌دهیم. به عنوان مثال، دستور زیر را مشاهده کنید:

```
>>> s = "www.fanavarienovin.net"
```

این دستور رشته s را به صورت زیر اندیس گذاری می‌کند:

-22	-21	-20	-19	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
w	w	w	.	f	a	n	a	v	A	r	i	e	n	o	v	i	n	.	n	e	t
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

در هنگام برش رشته، اندیس‌ها می‌توانند مثبت یا منفی باشند. اگر اندیس‌ها مثبت باشند، با توجه به

مقادیر n و m کاراکترهای اندیس n تا m رشته را برمی گرداند (کاراکتر اندیس m را بر نمی گرداند).

به عنوان مثال، دستور زیر را ببینید:

```
>>> print(s[5:12])
```

```
>>> print(s[8:19]).
```

دستور اول، کاراکترهایی از اندیس ۵ تا ۱۱ رشته s را نمایش می‌دهد (یعنی، از کاراکتر a تا

کاراکتر i همان رشته "anavari") و دستور دوم، از کاراکتر ۸ تا کاراکتر ۱۸ رشته s (یعنی، همان

رشته "varienovin." را برمی گرداند. در هنگام برش رشته می‌توان m را حذف کرد (یعنی به صورت

[n:] استفاده نمود). در این صورت، از n امین کاراکتر تا انتهای رشته را برمی گرداند. به عنوان مثال،

دستورات زیر را ببینید:

```
>>> print(s[15:])
```

```
>>> print(s[13:])
```

دستور اول از کاراکتر ۱۵ رشته s تا انتهای آن (یعنی، ۷ کاراکتر آخر رشته همان "vin.net") را نمایش می‌دهد و دستور دوم، از کاراکتر ۱۳ رشته s تا انتهای آن (یعنی، رشته "novin.net") را نمایش می‌دهد.

در هنگام برش رشته می‌توان n را حذف کرد (یعنی به صورت [:m] استفاده کرد). در این صورت، از شروع رشته تا قبل از کاراکتر m آن را برمی‌گرداند. به عنوان مثال، دستورات زیر را ببینید:

```
>>> print(s[:8])
>>> print(s[:4]).
```

دستور اول، از کاراکتر شروع تا کاراکتر ۸ رشته (یعنی، همان "www.fana") را نمایش می‌دهد و دستور دوم، از کاراکتر شروع تا کاراکتر چهارم (یعنی "www.") را نمایش می‌دهد. در هنگام برش رشته می‌توان m و n را دو عدد منفی وارد کرد. در این صورت، به جای استفاده از اندیس‌های پایین رشته از اندیس‌های بالای رشته شکل فوق استفاده می‌کند. به عنوان مثال، دستورات زیر را ببینید:

```
>>> print(s[-4:-1])
>>> print(s[-17:-10])
```

دستور اول، کاراکترهای از اندیس 4- تا اندیس 2- رشته را جدا کرده (یعنی "ne.") و نمایش می‌دهد و دستور دوم، کاراکترهای از اندیس 17- تا 11- را جدا کرده (یعنی، "anavari") و نمایش می‌دهد. مانند حالت‌های قبل می‌توان اندیس‌های m و n را حذف کرد. در صورتی که اندیس m حذف شود و n منفی باشد، از اندیس n- تا انتهای رشته را برمی‌گرداند. به عنوان مثال، دستور زیر را ببینید:

```
>>> print(s[-4:])
```

این دستور از اندیس 4- تا انتهای رشته (یعنی ".net") را نمایش می‌دهد.

اگر اندیس n حذف شود، از کاراکترهای ابتدای رشته تا اندیس 1 - m را برمی‌گرداند.

به عنوان مثال، دستور زیر را ببینید:

```
>>> print(s[:-10])
```

این دستور از ابتدای رشته (یعنی، اندیس 22-) تا اندیس 11- آن را نمایش می‌دهد (یعنی، رشته

"www.fanavari" را نمایش می‌دهد).

در هنگام برش رشته، چنانچه اندیس اول بزرگ تر یا مساوی اندیس دوم باشد، نتیجه برش یک رشته تهی خواهد بود. به عنوان مثال، دستورات زیر را ببینید:

```
>>> print(s[12:11])
>>> print(s[-11:-12])
>>> print(s[14:14])
```

دستور اول، می خواهد از اندیس ۱۲ تا ۱۱ را برش دهد، چون ۱۲ بزرگ تر از ۱۱ است، یک رشته تهی (خالی) را نمایش می دهد، دستور دوم، نیز یک رشته تهی را برمی گرداند. چون ۱۱- بزرگ تر از - ۱۲ است و دستور سوم، همچنین یک رشته تهی را برمی گرداند، چون ۱۴ برابر ۱۴ است. **عملگر in** رشته ای را در رشته دیگر جست و جو می کند. چنانچه رشته اول در رشته دوم باشد، True، و گرنه False را برمی گرداند. به عنوان مثال، دستورات زیر را ببینید:

```
>>> s = "Python"
>>> print("th" in s)
```

دستور اول، رشته s را با مقدار 'Python' تعریف می کند و دستور دوم، رشته 'th' را در s جست و جو می کند و نتیجه را نمایش می دهد، چون 'th' در s وجود دارد، نتیجه True نمایش داده می شود.

عملگر not in رشته ای را در رشته دیگر جست و جو می کند، چنانچه رشته اول در رشته دوم موجود نباشد، True، و گرنه False را برمی گرداند. دستورات زیر را مشاهده کنید:

```
>>> s = "Python"
>>> print("th" not in s)
```

دستور اول، رشته s را با مقدار 'Python' تعریف می کند و دستور دوم، مقدار False را نمایش می دهد، چون رشته th در رشته s وجود دارد.

مثال ۱-۵. برنامه ای که رشته ای را خوانده، اعمال زیر را انجام می دهد (هدف این برنامه آشنایی با عملگرهای رشته است):

۱. قبل از رشته عبارت "Hello" را قرار داده و با رشته خوانده شده جایگزین می کند.
۲. n را خوانده، کاراکتر n رشته را برمی گرداند.
۳. رشته ای دیگری را خوانده، اگر رشته خوانده شده دوم در رشته اول وجود داشت، "Yes"، و گرنه "No" را چاپ می کند.
۴. n و m را خوانده از کاراکتر m تا m رشته را نمایش می دهد (عمل ۴ را ۵ بار انجام دهد).

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات زیر را تایپ کنید:

متغیر	هدف
s	رشته ورودی
n	اندیس کاراکتری که باید برگردانده شود یا اندیس شروع کاراکتری که باید برگردانده شود.
s ₁	رشته‌ای که باید جست‌وجو گردد.
i	شمارنده ۱ تا ۵
m	اندیس پایان کاراکتری که باید برگردانده شود.

```
s = input("Enter a string:")
s = "Hello " + s
print(s)
n = int(input("Enter n:"))
if n < len(s) :
    print(s[n])
s1 = input("Enter a string for search:")
if s1 in s:
    print("Yes")
else:
    print("No")
for i in range(1, 6):
    n = int(input("Enter n:"))
    m = int(input("Enter m:"))
    print(s[n:m])
```

۲. ماژول را ذخیره و اجرا کنید. نمونه خروجی این ماژول در زیر آمده است:

```
Enter a string:python
Hello python
Enter n:7
y
Enter a string for search:llo
Yes
Enter n:1
Enter m:3
el
Enter n:-7
Enter m:-4
```

```

py
Enter n:3
Enter m:9
lo pyt
Enter n:-4
Enter m:-6
Enter n:6
Enter m:6

```

جدول ۵-۲ کاراکترهای چاپ نشدنی که با \ شروع می‌شوند.

متغیر	معادل مبنای 16	هدف
\a	0x07	بوق یا هشدار سیستم را به صدا در می‌آورد.
\b	0x08	کاراکتر برگشت به عقب (Backspace) را تایپ می‌کند.
\cx		کاراکتر ctrl + x می‌باشد.
\c-x		کاراکتر ctrl + x است.
\e	0x1b	کاراکتر Escape می‌باشد.
\f	0x0c	برای ایجاد صفحه جدیدی (formfeed) می‌باشد.
\n	0x0a	برای ایجاد سطر جدیدی (new line) می‌باشد.
\nnn		نشان‌گذاری هشت را انجام می‌دهد که در آن هر n عدد بین ۰ تا ۷ است.
\r	0x0d	به سر سطر فعلی برمی‌گردد.
\s	0x20	جای خالی (Space) ایجاد می‌کند.
\t	0x09	برای انتقال مکان‌نما به تب بعدی به کار می‌رود.
\v	0x0b	چندین سطر خالی به صورت عمودی ایجاد می‌کند.
\x		کاراکتر x را نمایش می‌دهد.
\xnn		نشان‌گذاری مبنای ۱۶ را انجام می‌دهد که در آن n بین ۰ تا ۹، A`، B`، C`، D`، E` یا F` می‌باشد.

**عملگر **، کاراکترهای غیرقابل چاپ (چاپ نشدنی) را ارائه می‌دهد که هریک از کاراکترها مفهوم

خاصی دارند. تاکنون با نمونه این کاراکترها '\t' آشنا شدید. در جدول ۵-۲ برخی از کاراکترهای

چاپ نشدنی و مفهوم آن‌ها که با کاراکتر \ شروع می‌شوند، آمده‌اند.

عملگر٪، یکی از عملگرهای جالب پایتون است که برای قالب‌بندی رشته به کار می‌رود. کاراکترهای که بعد از این عملگر قرار می‌گیرند، هریک برای قالب‌بندی داده‌های خاصی به کار می‌روند. برخی از این کاراکترها و عملکردهای آن‌ها در جدول ۳-۵ آمده‌اند.

جدول ۳-۵ کاراکترهایی که برای قالب‌بندی به کار می‌روند.	
متغیر	هدف
%c	برای قالب‌بندی کاراکترها به کار می‌رود.
%C	برای قالب‌بندی رشته به کار می‌رود.
%i , %d	برای قالب‌بندی عدد صحیح علامت‌دار در مبنای ۱۰ به کار می‌رود.
%v	برای قالب‌بندی عدد صحیح بدون علامت در مبنای ۱۰ به کار می‌رود.
%o	برای قالب‌بندی عدد در مبنای ۸ به کار می‌رود.
%x	برای قالب‌بندی عدد صحیح در مبنای ۱۶ به کار می‌رود و حروف 'A' تا 'Z' را با حروف کوچک نمایش می‌دهد.
%X	برای قالب‌بندی عدد صحیح در مبنای ۱۶ به کار می‌رود و حروف 'A' تا 'Z' را با حروف بزرگ نمایش می‌دهد.
%e	برای قالب‌بندی نماد توانی (نمایی) با حرف e به کار می‌رود. یعنی، حروف 'A' تا 'F' را با حروف کوچک نمایش می‌دهد.
%E	برای قالب‌بندی نماد توانی (نمایی) با حرف E به کار می‌رود. یعنی، حروف 'A' تا 'F' را با حروف بزرگ نمایش می‌دهد.
%f	برای قالب‌بندی اعداد حقیقی با ممیز شناور به کار می‌رود.
%g	برای قالب‌بندی با فرم کوتاه‌تر %f و %e می‌باشد.
%G	برای قالب‌بندی با فرم کوتاه‌تر %f و %E می‌باشد.

کاراکترهای قالب‌بندی را می‌توان با برخی کاراکترهای دیگر به کار برد که هر یک از این کاراکترها کار خاصی را انجام می‌دهند. برخی از این کاراکترها و عملکردهای آن‌ها در جدول ۴-۵ آمده‌اند.

به عنوان مثال، دستورات زیر را مشاهده کنید:

```
>>> a = 20
>>> print( %10i, %18d, %E" %(a, -a, float(a)))
```

با اجرای این دستورات a برابر ۲۰ شده و خروجی زیر نمایش داده می‌شود:

20, -20, 2.000000E+01

همان‌طور که در این خروجی می‌بینید، قبل از عدد ۲۰ هشت فاصله نمایش داده شده است. چون،

۲۰ با فرمت ۱۰۱٪ چاپ شد که در چاپ به شکل زیر عمل می کند:

[illegible]

بیست را به سمت چپ انتقال داده و ۸ فاصله قبل از آن نمایش داده است. سپس، یک کاما نمایش

داده است و عدد a- (20-) را با فرمت 18d % چاپ کرده است که ۱۵ فاصله قبل از آن قرار می گیرد

(به صورت زیر عمل می کند):

[illegible]

سیس یک علامت کا ما نمایش داده است و معادل عدد اعشاری a را به صورت $20000000E+01$ با

فرمت E% نمایش داده است.

اکنون دستورات زیر را در نظر بگیرید:

```
>>> a = 125
>>> print("%d\t%o\t%x\t%X\n%08i" % (a, a, a, a, a))
```

دستور اول، a را برابر ۱۲۵ قرار می‌دهد و دستور دوم، خروجی زیر را نمایش می‌دهد:

125	175	7d	7D
00000125			

همان‌طور که در این خروجی ملاحظه می‌شود، عدد 125 با فرمت %d نمایش داده می‌شود، سپس

چون کاراکتر t آمده است، مکان نما به تب بعدی انتقال می یابد و عدد صحیح 125 به مبنای ۸

تبدیل شده و خروجی 175 نمایش داده می شود، با کاراکتر '\t' مکان نما به تب بعدی می رود و مقدار

125 یا فرمت x% به عدد مبنای ۱۶ یعنی 7d تبدیل شده و نمایش داده می‌شود، در ادامه کاراکتر '\t'

مکان‌نما را به تب بعدی انتقال می‌دهد و عدد 125 با فرمت %x به منای ۱۶ تبدیل شده و مقدار 7D

نمایش داده می‌شود، در پایان، کاراکتر n مکان‌نما را به سطر بعدی انتقال می‌دهد و عدد 125 با

فرمت %08i به صورت 00000125 نمایش داده می شود که به جای کاراکتر Space قبل از عدد با صفر پر می شود.

۶-۵. مسائل حل شده

مثال ۱. برنامه ای که با تابعی که تعداد تکرار رشته ای را در رشته دیگر پیدا می کند و نمایش می دهد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

ماژول	متغیر	هدف
ماژول اصلی	mainStr	رشته ورودی
	subStr	رشته ای که باید جستجو شود
countSubstr	len ₁	طول رشته مورد جستجو
	len ₂	طول رشته اصلی
	i	شمارنده هر کاراکتر رشته
	count	تعداد تکرار رشته مورد جستجو

```
def countSubStr(str, substr):
```

```
    len1 = len(substr)
    len2 = len(str)
    i = -1
    count = 0
    if len1 > len2:
        return ("error:>>> substr is len1onger than str")
    while i <= len2 - len1 + 1:
        i=i+1
        if substr == str[ i : i + len1]:
            count=count + 1
    return count
```

```
mainStr = input("Enter a string:")
```

```
subStr = input("Enter a string for find:")
```

```
print("Count is ", countSubStr(mainStr, subStr))
```

🔗 تابع countSubStr()، رشته اصلی (str) و رشته مورد جستجو (substr) را به عنوان پارامتر

دریافت کرده، طول رشته مورد جستجو را به len₁ و طول رشته اصلی (str) را در len₂ قرار

می دهد. اندیس پیمایش رشته i را برابر 1- قرار می دهد و تعداد تکرار رشته را برابر صفر قرار

می‌دهد، اگر طول رشته مورد جست‌وجو (len_1) بیش از طول رشته اصلی (len_2) باشد، یک پیغام را برمی‌گرداند، وگرنه، تا زمانی که i کوچک‌تر یا مساوی $len_2 - len_1 + 1$ باشد، ابتدا به i یک واحد اضافه می‌کند و `substr` را با `str[i : i + len1]` (یعنی مکان فعلی رشته تا طول رشته مورد جست‌وجو جلو می‌رود. بدین معنی که به طول رشته مورد جست‌وجو از رشته اصلی جدا می‌کند) مقایسه کرده و اگر این دو مقدار برابر باشند، به `count` یک واحد اضافه می‌کند، سپس شرط حلقه را بررسی می‌کند. عمل مقایسه را برای رشته‌های بعدی که از مکان‌های بعدی رشته اصلی جدا می‌گردد، انجام می‌دهد. در پایان، `count` را برمی‌گرداند.

۲. ماژول را ذخیره و اجرا کرده، اطلاعات زیر را وارد کنید تا خروجی را ببینید:

Enter a string: C++ is a very good language. Python is a good language.

Enter a string for find: good

Count is 2

مثال ۲. برنامه‌ای که رشته‌ای را خوانده، تمام کاراکترهای کوچک رشته را به بزرگ و کاراکترهای بزرگ را به کوچک تبدیل می‌کند.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

هدف	متغیر	ماژول
رشته ورودی	str	ماژول اصلی
آرگومان که رشته ورودی را دریافت می‌کند	str	charChange
رشته تغییر یافته	s	
اندیس هر کاراکتر رشته str	i	

def charChange(str):

s = ""

for i in range(0, len(str)):

if str[i].islower() == True:

s += str[i].upper()

else:

s += str[i].lower()

return s

str = input("Enter a string:")

str = charChange(str)

print("Result is ", str)

🚩 تابع `charChange()` رشته `str` را به عنوان پارامتر دریافت کرده، ابتدا رشته `s` (رشته تغییر یافته) را خالی می‌کند، سپس با حلقه `for` هر کاراکتر رشته `str` را پیمایش می‌کند، اگر کاراکتر رشته حرف کوچک باشد، آن را به حرف بزرگ تبدیل کرده، به انتهای رشته `s` اضافه می‌کند، در غیر این صورت، حرف کوچک `str[i]` را به انتهای رشته `s` اضافه می‌کند. در پایان، رشته `s` را برمی‌گرداند.

۲. ماژول را ذخیره و اجرا کرده، نمونه خروجی برنامه را به صورت زیر ببینید:

```
Enter a string:Python Is A gooD lanGuage.
Result is pYTHON iS a GOOd LANgUAGE.
```

فصل ۶

لیست‌ها، چندتایی‌ها، دیکشنری‌ها و مجموعه‌ها

تاکنون در فصل‌های ۱ تا ۵ با انواع داده‌ها از قبیل اعداد، رشته‌ها و آرایه‌ها آشنا شدید. در این فصل انواع داده‌ای جدیدی از قبیل لیست‌ها، چندتایی‌ها^۱، دیکشنری‌ها و مجموعه‌ها را می‌آموزیم.

۶-۱. لیست‌ها

یکی از انواع آماده دیگر در پایتون نوع لیست^۲ است. لیست در پایتون مجموعه‌ای از مقادیر هستند که در یک متغیر قرار می‌گیرند. یعنی، لیست مانند رشته از دنباله‌ای از مقادیر تشکیل می‌شود، اما برخلاف رشته، یک نوع تغییرپذیر^۳ است. هر عضو لیست می‌تواند هر نوع داده باشد. یعنی، اعضای لیست می‌توانند انواع مختلف داشته باشند. به عبارت دیگر، یک عضو رشته‌ای باشد و عضو دیگر عددی باشد. حتی گاهی اوقات اعضای آن می‌توانند از نوع لیست باشند. اعضای لیست با کاما (,) از یکدیگر جدا می‌شوند. برای تعریف لیست از عملگرهای `[]` استفاده می‌شود. به عنوان مثال، دستور زیر را ببینید:

```
>>> list1 = []
```

^۱ Tuples

^۲ List

^۳ Mutable

این دستور یک لیست خالی به نام list1 ایجاد می‌کند. به جای این دستور می‌توان از دستور زیر استفاده کرد:

```
>>> list1 = list()
```

اکنون دستورات زیر را ببینید:

```
>>> list1 = ["Fanavarienovin", 2, True]
>>> list1[0]
```

دستور اول، یک لیست به نام list1 ایجاد می‌کند و به اعضای آن را به ترتیب "Fanavarienovin"، ۲ و True تخصیص می‌دهد، دستور دوم، محتوی عضو اول آن (list1[0]) یعنی، همان 'Fanavarienovin' را نمایش می‌دهد.

اندیس لیست از صفر شروع می‌شود و با استفاده از اندیس می‌توان به اعضای لیست دسترسی یافت. علاوه بر اندیس با استفاده از عملگر slice می‌توان به بخشی از لیست دسترسی یافت. به عنوان مثال، دستورات زیر را ببینید:

```
>>> list1 = [1, "Python", "Program language", True, [3, "Ali"]]
>>> list1[2:4]
```

دستور اول، لیستی به نام list1 با پنج عضو تعریف کرده، مقادیری را به آن‌ها تخصیص می‌دهد، دستور دوم، اعضای سوم و چهارم list1 را نمایش می‌دهد (خروجی زیر):

```
['Program language', True']
```

مثال ۱-۶. برنامه‌ای که یک رشته را خوانده، کلمات رشته را جدا می‌نماید و در یک لیست قرار می‌دهد. در پایان، اطلاعات لیست را نمایش می‌دهد (هدف این برنامه آشنایی با ایجاد لیست و چاپ اعضای آن است).

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
words=list()
s=input("Enter a string:")
words=s.split()
print(words)
```

متغیر	هدف
words	لیستی از کلمات
s	رشته خوانده شده


دستور اول، یک لیست خالی به نام words ایجاد می‌کند، دستور دوم، با یک پیغام یک‌رشته را می‌خواند، در s قرار می‌دهد، دستور سوم، کلمات رشته را جدا کرده، در لیست words قرار می‌دهد و دستور چهارم، محتوای لیست words را نمایش می‌دهد.

۲. ماژول را ذخیره و اجرا کنید. اکنون جلوی string عبارت `www fanavarienovin net` را وارد کرده تا خروجی زیر را ببینید:

```
Enter a string:www fanavarienovin net
['www', 'fanavarienovin', 'net']
```

۱-۱-۶. عملگرهای کار بر روی لیست

عملگرهای زیادی برای کار بر روی لیست به کار می‌روند. برخی از این عملگرها عبارت‌اند از:

 **عملگر +**، برای پیوند (اتصال) دو لیست به کار می‌رود. به عنوان مثال، دستورات زیر را ببینید:


```
>>> list1 = [1, 2]
>>> list2 = [3, 4]
>>> list1 = list1 + list2
>>> list1
```

دستور اول، شی‌ای به نام list1 با نوع لیست و مقادیر ۱ و ۲ ایجاد می‌کند، دستور دوم، شیء

دیگری به نام list2 با نوع لیست و مقادیر ۳ و ۴ ایجاد می‌نماید، دستور سوم، اعضای list2 را به انتهای

list1 پیوند می‌زند و دستور چهارم، اعضای list1 را نمایش می‌دهد (خروجی زیر):

```
[1, 2, 3, 4]
```

 **عملگر ***، برای تکرار اعضای لیست به کار می‌رود و به صورت زیر به کار می‌رود:

```
>>> تعداد تکرار * متغیر لیست
```


به عنوان مثال، دستورات زیر را مشاهده کنید:

```
>>> list1 = ['Python'] * 4
>>> list1
```

دستور اول، شی‌ای به نام list1 ایجاد کرده، چهار عضو با مقدار "Python" در آن قرار می‌دهد و

دستور دوم، اعضای list1 را نمایش می‌دهد (خروجی زیر):

```
['Python', 'Python', 'Python', 'Python']
```

 **عملگر ==**، برای مقایسه برابری دو لیست به کار می‌رود. به عنوان مثال، دستور زیر را ببینید:

```
>>> ['Python', 'Program'] == ['Python', 'Program']
```

این دستور اعضای دو لیست را نظیر به نظیر را باهم مقایسه کرده، چون برابرند، خروجی زیر را نشان می‌دهد:

```
True
```

اکنون دستورات زیر را ببینید:

```
>>> ['Python', 'Program'] == ['PYTHON', 'Program']
```

این دستور دو لیست را باهم مقایسه می‌کند، چون عضو `Python` از لیست اول با عضو PYTHON از لیست دوم برابر نیست، پس خروجی زیر را نمایش می‌دهد:

```
False
```

🔗 **عملگر in** برای بررسی عضویت یک مقدار در یک لیست به کار می‌رود. چنانچه مقدار عضو لیست باشد، این عملگر True، وگرنه False را برمی‌گرداند. به‌عنوان مثال، دستورات زیر را ببینید:

```
>>> list1 = [1, [2, 3], [4,5]]
>>> [4, 5] in list1
>>> 5 in list1
```

دستور اول، یک لیست به نام list1 را تعریف می‌کند و اعضای آن را مقدار می‌دهد، دستور دوم، تعیین می‌کند [4, 5] در لیست list1 وجود دارد یا نه؟ چون، این عضو در list1 وجود دارد، خروجی این دستور True می‌باشد. دستور سوم، تعیین می‌کند که آیا 5 در list1 وجود دارد یا نه؟ چون 5 عضو list1 نیست، خروجی False نمایش داده می‌شود (۵ به‌تنهایی عضو list1 نیست، بلکه لیست [4, 5] عضو list1 است).

🔗 **عملگر is** همان‌طور که بیان گردید، هر شیء در پایتون شامل شناسه^۱، نوع^۲ و مقدار^۳ است. عملگر == دو شیء را از لحاظ یکسان بودن مقدار آن‌ها بررسی می‌کند. درحالی‌که عملگر is دو شیء را از لحاظ یکسان بودن شناسه (خروجی تابع id()) مقایسه می‌کند. به‌عنوان مثال، دستورات زیر را ببینید:

```
>>> list1 = [1, 2, 3]
>>> list2 = list1
>>> list1 == list2
```

^۱. Identity

^۲. Type

^۳. Value

```
>>> list2 is list2
```

دستور اول، شی‌ای به نام list1 ایجاد کرده، مقادیر ۱، ۲ و ۳ را به اعضای آن تخصیص می‌دهد، دستور دوم، آدرس list1 را در list2 قرار می‌دهد، دستور سوم، مقادیر اعضای list1 و list2 را باهم مقایسه می‌کند، چون نظیر به نظیر باهم برابرند، خروجی زیر را نمایش می‌دهد:

```
True
```

دستور چهارم، شناسه list1 و list2 را باهم مقایسه می‌نماید، چون برابر هستند، خروجی زیر را

نمایش می‌دهد:

```
True
```

اکنون، دستورات زیر را ببینید:

```
>>> list1 = [1, 2]
>>> list2 = list1[:]
>>> list1 == list2
>>> list2 is list1
```

دستور اول، شی‌ای به نام list1 با اعضای ۱ و ۲ تعریف می‌کند، دستور دوم، اعضای list1 را در list2 کپی می‌کند، در دستور سوم، چون مقادیر اعضای list1 و list2 برابر هستند، عملگر == خروجی زیر را نمایش می‌دهد:

```
True
```

اما، در دستور چهارم، چون شناسه شیء list1 و list2 باهم برابر نیستند، عملگر is خروجی زیر را

برمی‌گرداند:

```
False
```

مثال ۳-۶. برنامه‌ای که ابتدا اعضای یک لیست را خوانده، سپس از طریق نمایش یک منو و انتخاب گزینه‌ای توسط کاربر، یکی از اعمال زیر را انجام می‌دهد:

۱. عضوی را به لیست اضافه می‌کند.
۲. عضوی را در مکانی خاص از لیست درج می‌کند.
۳. عضوی را از لیست حذف می‌کند.
۴. اعضای لیست را مرتب می‌کند.
۵. اعضای لیست را وارد می‌کند.
۶. تعداد تکرار مقداری را در لیست شمارش می‌کند.
۷. مکان یک مقدار را در لیست پیدا می‌کند.

هدف این برنامه آشنایی با متدهای `append()`، `insert()`، `remove()`، `sort()`، `reverse()` و `count()` و `index()` بر روی لیست است.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

تابع	متغیر	هدف
ماژول اصلی	list1	متغیری از نوع لیست که اطلاعات لیست را نگهداری می کند.
	s	رشته دریافتی که کلمات آن جدا می شوند.
	x	عنصری که باید اضافه، درج، حذف شود یا مکان آن و تعداد تکرار آن پیدا گردد.
	index	مکانی که باید x درج شود یا اندیسی که x در آن قرار دارد.
	choose	مقدار برگشتی تابع <code>menu()</code>
menu	choose	گزینه ای که کاربر انتخاب کرده است.

```
def menu():
    print ( "1: Append ")
    print ( "2: Insert")
    print ( "3: Remove")
    print ( "4: Sort")
    print ( "5: Reverse")
    print ( "6: Count")
    print ( "7: Index")
    print ( "8: Exit")
    choose = int(input("Select 1 to 8:"))
    return choose

list1= []
s=input("Enter list elements:")
list1=s.split()
print(list1)
while 1:
    choose=menu()
    if choose == 8 :
        break;
    elif choose == 1:
        x = input("Enter a element for append:")
        list1.append(x)
        print("List is ", list1)
    elif choose == 2:
        x = input("Enter a element for insert:")
```



```

index = int(input("Enter insert index:"))
if index < len(list1):
    list1.insert(index, x)
    print("List is ", list1)
elif choose == 3:
    x = input("Enter a remove element:")
    if x in list1:
        list1.remove(x)
        print("List is ", list1)
    elif choose == 4:
        list1.sort()
        print("List is ", list1)
    elif choose == 5:
        list1.reverse()
        print("List is ", list1)
    elif choose == 6:
        x = input("Enter a element for count:")
        print("Count is ", list1.count(x))
    elif choose == 7:
        x = input("Enter a element for find index:")
        if x in list1:
            index = list1.index(x)
        else:
            index = 0
        print("Index is ", index)
    else:
        print("Invalid choose")
print("-"*40)

```

۲. ماژول را ذخیره و اجرا کنید. نمونه‌ای از خروجی برنامه در زیر آمده است:

```

Enter list elements:Ali Reza
['Ali', 'Reza']
1: Append
2: Insert
3: Remove
4: Sort
5: Reverse
6: Count
7: Index
8: Exit
Select 1 to 8:2
Enter a element for insert:Book
Enter insert index:1

```

List is ['Ali', 'Book', 'Reza']

1: Append

2: Insert

3: Remove

4: Sort

5: Reverse

6: Count

7: Index

8: Exit

Select 1 to 8:4

List is ['Ali', 'Book', 'Reza']

1: Append

2: Insert

3: Remove

4: Sort

5: Reverse

6: Count

7: Index

8: Exit

Select 1 to 8:8

بقیه گزینه‌ها را خودتان آزمایش نمایید

فصل ۷ کلاس‌ها و وراثت

برنامه‌هایی که در فصل‌های ۱ تا ۶ نوشته‌ایم، همه شامل کلاس‌های آماده بودند که قبلاً توسط مفسر زبان ایجاد گردیدند. گاهی نیاز است کلاس‌های جدیدی ایجاد کنید. در این فصل روش ایجاد کلاس جدید و استفاده از آن را می‌آموزیم.

۱ - ۷. کلاس‌ها

دنیای واقعی پر از اشیا است. کافی است به اطراف خودتان نگاه کنید، اطرافتان از اشیایی نظیر دانشگاه‌ها، هواپیماها، انسان‌ها، حیوانات، ماشین‌ها، ساختمان‌ها، موبایل‌ها، رایانه‌ها و غیره تشکیل شده است. در زبان‌های برنامه‌نویسی ساخت‌یافته نظیر C، پاسکال، برنامه‌نویسان بر روی اعمال^۱ (کارها یا وظایف (Tasks)) به جای شیء^۲ تمرکز می‌کردند (برنامه‌نویسان در دنیای واقعی زندگی می‌کنند، اما تفکر آن‌ها شیء‌گرایی نیست). این امر موجب می‌گردد تا برنامه‌های نوشته‌شده دارای مشکلات زیر - باشند:

۱. واحدهای تشکیل‌دهنده برنامه نمی‌توانند به‌سادگی نشان‌دهنده اشیای دنیای واقعی باشند. لذا، بخش‌های تشکیل‌دهنده برنامه قابلیت استفاده مجدد^۳ مناسبی نخواهند داشت.

۲. بخش‌های تشکیل‌دهنده برنامه اشیای دنیای واقعی را مدل‌سازی نمی‌کنند. لذا، پیچیدگی افزایش می‌یابد.

۳. امکان تغییرپذیری داده‌ها و توابع سخت است.

۴. نگهداری و پشتیبانی برنامه‌ها مشکل‌تر خواهد شد.

اما، در برنامه‌نویسی شیء‌گرا، کلاس‌ها ایجاد می‌شوند. چون کلاس‌ها، اشیای دنیای واقعی را مدل‌سازی می‌کنند. لذا، مزایای زیر را در پی خواهند داشت:

^۱.Actions

^۲.Object

^۳.Reuse

۱. قابلیت استفاده مجدد افزایش می‌یابد، زیرا، اشیاء را به راحتی می‌توان در برنامه‌های مختلف استفاده نمود.

۲. به کارگیری اشیاء راحت‌تر خواهد بود، زیرا، این اشیاء، اشیای دنیای واقعی را مدل‌سازی خواهند کرد. بنابراین، موجب کاهش پیچیدگی خواهند شد.

۳. قابلیت نگهداری اشیاء راحت‌تر است. در ادامه خواهید دید که هر یک از اشیاء به‌طور مستقل نگه‌داری می‌شوند. لذا، تغییرپذیری، توسعه و نگه‌داری آن‌ها آسان‌تر خواهد شد.

همان‌طور که بیان گردید، کلاس‌ها برای مدل‌سازی اشیای دنیای واقعی به کار می‌روند. اشیای دنیای واقعی یکسری ویژگی‌هایی از قبیل رنگ، وزن، اندازه، نام، جنس و غیره دارند که شکل ظاهری آن‌ها را تعیین می‌کنند. این ویژگی‌ها صفات اشیاء نام دارند. صفات اشیاء همان اعضای داده‌ای آن‌ها می‌باشند. اشیاء علاوه بر صفات، یکسری رفتارها نیز از خودشان نشان می‌دهند. این رفتارها، تابع عضو (متد) نام دارند. متدها، عملیاتی هستند که بر روی اعضای داده‌ای قابل اجرا هستند.

برای درک بهتر این موضوعات به طرز کار کلاس‌ها می‌پردازیم. در حالت عادی هر شیء در دنیای واقعی به سه جنبه زیر شناخته می‌شود:

۱. نام شیء ۲. وضعیت شیء ۳. رفتار شیء

به عنوان مثال، شیء پخش‌کننده CD را در نظر بگیرید. نام این شیء پخش‌کننده CD است. پخش‌کننده CD می‌تواند CD را پخش کند، بین آهنگ‌ها سوئیچ کند و اعمالی دیگری انجام دهد. این‌ها رفتارهای یک پخش‌کننده CD هستند. منظور از وضعیت این شیء این است که پخش‌کننده میزان بلندی صدا را کنترل می‌کند یا می‌داند که اکنون چه مقدار از ادامه آهنگ باقی مانده است.

کلاس‌ها، الگوهایی برای اشیاء با صفات مشترک و رفتارهای یکسان می‌باشند. به عنوان مثال، کلاس ماشین خواصی مانند رنگ، وزن، تعداد درها، تعداد چرخ‌ها و غیره دارد که شکل ظاهری ماشین را تعیین می‌کنند. ماشین‌ها علاوه بر صفات یکسری متدهای از قبیل روشن شدن، خاموش شدن، سرعت گرفتن و ترمز کردن دارند. بنابراین، برای استفاده از ماشین باید از متدهایش استفاده نمود. برای استفاده از کلاس‌ها دو کار باید انجام گردد که عبارت‌اند از:

۱. تعریف کلاس‌ها ۲. نمونه‌سازی از کلاس‌ها

۱-۱-۷. تعریف کلاس‌ها

بیان گردید که کلاس‌ها برای مدل‌سازی اشیاء به کار می‌روند. برای تعریف کلاس ابتدا باید اعضای داده‌ای آن را مشخص کرد. سپس اعضای تابعی آن را پیاده‌سازی نمود. نام کلاس، از قانون نام‌گذاری شناسه‌ها پیروی می‌کند و اعضای تشکیل‌دهنده کلاس، متغیرها و متدهایی هستند که کلاس از آن‌ها تشکیل می‌شود. اعضای تشکیل‌دهنده کلاس عبارت‌اند از:

۱. فیلدها
۲. سازنده‌ها
۳. مخرب‌ها
۴. متدها

به‌عنوان مثال، دستورات زیر را ببینید:

```
>>> class Circle:  
    PI = 3.1415
```

این دستور کلاس Circle با یک عضو به نام PI ایجاد می‌کند.

در ادامه در مورد هر یک از اعضای کلاس به‌طور مفصل بحث خواهیم کرد.

۲-۱-۷. نمونه‌سازی کلاس‌ها

همان‌طور که در فصل اول دیدید، برای استفاده از انواعی نظیر int، double، char و ... می‌بایست نمونه‌ای از این انواع تعریف می‌گردید. برای استفاده از کلاس باید نمونه‌ای (شی‌ایی) از کلاس ایجاد کنید. برای این منظور، به‌صورت زیر عمل می‌شود:

() نام کلاس = نام نمونه

نام نمونه، از قانون نام‌گذاری متغیرها پیروی می‌کند.

دسترسی به اعضای کلاس به صورت‌های زیر می‌باشد:

۱. نام عضو . نام نمونه

۲. نام عضو . نام کلاس

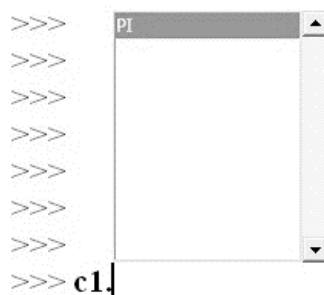
ساختار اول، برای دسترسی به اعضای معمولی کلاس به کار می‌رود و ساختار دوم، برای دسترسی

به اعضای نوع static کلاس استفاده می‌شود که در ادامه خواهید دید.

اکنون دستورات زیر را ببینید:

```
>>> c1 = Circle()
```

این دستور نمونه‌ای به نام c1 از نوع Circle ایجاد می‌کند. اکنون c1 را تایپ کرده و کلید `ctrl + space` را فشار دهید تا شکل زیر ظاهر شود:



همان‌طور که در این شکل می‌بینید، این کلاس فقط یک عضو PI دارد.

۲-۷. اعضای کلاس

همان‌طور که دیدید، برای استفاده از کلاس باید اعضای آن را پیاده‌سازی کرد. قبل از پیاده‌سازی اعضای کلاس باید آن‌ها را شناخت. یعنی، این که تعیین نمود که کلاس از چه اعضای تشکیل شده است. برای این منظور می‌توان از واژه **HAS A** (دارد یک) استفاده نمود. به‌عنوان مثال، کلاس دایره را در نظر بگیرید:

۱. کلاس دایره یک شعاع دارد.

۲. کلاس دایره یک عدد ثابت π دارد.

۳. کلاس دایره یک قطر دارد.

۴. کلاس دایره یک محیط دارد.

۵. کلاس دایره یک مساحت دارد.

اما، کلاس استاد را در نظر بگیرید.

۱. کلاس استاد، یک شماره استادی دارد.

۲. کلاس استاد، یک نام دارد.

۳. کلاس استاد، یک نام خانوادگی دارد.

۴. کلاس استاد، ساعت حق‌التدریس دارد.

۵. کلاس استاد، یک مبلغ به ازای هر ساعت حق‌التدریس دارد.

۶. کلاس استاد، مبلغ دریافتی حق التدریس دارد.

همان‌طور که در مثال‌های کلاس دایره و کلاس استاد دیده‌اید، مقدار ثابت π ، شعاع دایره، قطر دایره، محیط و مساحت دایره اعضای کلاس دایره هستند. اما، کد استادی، نام، نام خانوادگی، تعداد ساعت حق التدریس، مبلغ حق التدریس به ازای هر ساعت و مبلغ دریافتی حق التدریس، اعضای کلاس استاد هستند. پس برای تعیین اعضای کلاس باید ببینیم که یک کلاس چه اطلاعاتی دارد.

۲-۱ - ۷. دسترسی به اعضای کلاس

قبل از این که به تعریف اعضای کلاس پردازیم، ابتدا به چگونگی دسترسی اعضای کلاس می‌پردازیم. به دو روش می‌توان به اعضای کلاس دسترسی داشت:

۱. دسترسی به اعضای کلاس از درون کلاس: در حالت معمولی در درون کلاس می‌توان از طریق عبارت زیر به اعضای کلاس دسترسی یافت:

نام عضو. self

۲. دسترسی به اعضای کلاس در بیرون کلاس: برای دسترسی به اعضای عمومی کلاس دو روش وجود دارد که عبارت‌اند از:

➤ دسترسی عضو از طریق نام کلاس، نام کلاس برای دسترسی به اعضای static کلاس به کار می‌رود. به‌عنوان مثال، دستور زیر به عضو Sqrt (عضو static) کلاس Math به کار می‌رود:

Math.Sqrt (1.5);

➤ دسترسی از طریق نام نمونه، برای دسترسی به اعضای غیر static کلاس باید از آن نمونه‌سازی کرد. سپس از طریق نمونه ایجادشده به اعضای آن دسترسی نمود. قبلاً نمونه c1 از کلاس Circle را دیدید.

۲-۲ - ۷. انواع اعضای کلاس

کلاس معمولاً از دو نوع عضو تشکیل می‌شود که عبارت‌اند از:

➤ اعضای داده‌ای، اعضای هستند که مقادیر موردنیاز کلاس را نگهداری می‌کنند. به‌عنوان مثال، کلاس دایره دارای دو عضو داده‌ای است. عضو داده‌ای اول PI است که عدد 3.14159 می‌باشد و عضو داده‌ای دوم شعاع (r) است که شعاع دایره در آن قرار می‌گیرد.

🚩 **اعضای متدی، اعمالی را تعیین می کنند که کلاس می خواهد انجام دهد.** اعضای متدی می توانند سازنده ها، مخرب ها و متدهای کلاس باشند. به عنوان مثال، دایره دارای متدهایی جهت محاسبه محیط و مساحت می باشد. بنابراین کلاس می تواند اعضای زیر را داشته باشد:

۱. **فیلدها** ۲. **سازنده ها** ۳. **مخرب ها** ۴. **متدها** ۵. **و غیره**

فیلدها

فیلدها، اعضای از کلاس هستند که مقدار داده ای کلاس را نگه داری می کنند (مانند شعاع دایره). اعضای فیلد کلاس، همان متغیرهای کلاس هستند. به عنوان مثال، دستورات زیر را ببینید:

```
>>> import math
>>> class Circle:
    PI = math.pi
    r = 0
```

دستور اول، ماژول `math` را به برنامه اضافه می کند و دستور دوم، کلاسی به نام `Circle` ایجاد

می-نماید که دو عضو داده ای `PI` و `r` دارد. `PI`، عدد ثابت π و `r`، شعاع دایره می باشد.

متدها

همان طور که بیان گردید، کلاس علاوه بر اعضای داده ای می تواند دارای متد نیز باشد. متدها، اعمالی هستند که می خواهید بر روی داده انجام دهید. متد به صورت زیر تعریف می شود (به فصل سوم مراجعه کنید):

def (لیست پارامترها و `self`) نام متد :

دستورات پیاده سازی متد

return [مقدار]

به عنوان مثال، دستورات زیر را ببینید:

```
class Circle:
    PI = math.pi
    r = 0
    def Area(self) :
        return (self.PI* self.r ** 2)
    def Perime(self):
        return 2 * self.PI * self.r
    def __str__(self):
        s = "R : " + str(self.r)
```



```
s += "\tArea : " + str(self.Area())
s += "\tPerime : "+str(self.Perime())
return s
```

این دستورات، کلاسی به نام Circle با دو عضو داده‌ای PI (عدد ثابت π)، r (شعاع دایره) و سه متد زیر را تعریف می‌کنند:

متد **Area()**، این متد مساحت دایره را محاسبه می‌کند. همان‌طور که در این متد مشاهده می‌گردد، یک پارامتر self دارد و برای دسترسی به اعضاء کلاس در متد Area از نام عضو self استفاده شده است.

متد **Perime()**، برای محاسبه محیط دایره به کار می‌رود.

متد **__str__()**، متد str() را برای این کلاس پیاده‌سازی می‌کند. این متد شعاع، مساحت و محیط را با پیغام مناسب برمی‌گرداند.

مثال ۱-۷. برنامه‌ای که کلاس دایره را پیاده‌سازی کرده، نمونه‌ای از آن ساخته و استفاده می‌نماید (هدف برنامه آشنایی با ایجاد کلاس و نمونه‌سازی آن می‌باشد):

مراحل طراحی و اجرا

۱. ماژول جدیدی را ایجاد کرده، دستورات آن را به‌صورت زیر تایپ کنید:

```
import math
class Circle:
    PI = math.pi
    r = 0
    def Area(self):
        return (self.PI* self.r ** 2)
    def Perime(self):
        return (2 * self.PI * self.r)
    def __str__(self):
        s = "R : " + str(self.r)
        s += "\tArea : " + str(self.Area())
        s += "\tPerime : "+str(self.Perime())
        return s

c = Circle()
c.r = int(input("Enter r:"))
print(str(c))
```

۲. ماژول را ذخیره و اجرا کرده، جلوی R عدد ۱۰ را وارد کنید تا خروجی زیر را ببینید:

Enter r:10

R : 10 Area : 314.1592653589793 Perime : 62.83185307179586

۳-۷. سازنده‌ها و مخرب‌ها

سازنده‌ها برای مقداردهی اولیه به اعضای کلاس به کار می‌روند. با دو روش زیر می‌توان به اعضای کلاس مقدار اولیه تخصیص داد:

۱. مقداردهی به اعضای داده‌ای کلاس در بدنه آن، در هنگام تعریف کلاس در بدنه کلاس می‌توان به اعضای داده‌ای آن مقدار اولیه داد. این عمل به صورت زیر انجام می‌شود:

مقدار اولیه = نام عضو داده‌ای

به عنوان مثال، در کلاس Circle به PI مقدار اولیه `math.pi` تخصیص داده شد و به `r` مقدار اولیه صفر تخصیص داده شد.

۲. مقداردهی اولیه به اعضای کلاس از طریق متد سازنده، در پایتون هر کلاس یک متد سازنده به نام `__init__()` دارد که برای مقداردهی اولیه به اعضای داده‌ای کلاس به کار می‌رود. این متد به صورت زیر پیاده‌سازی می‌شود:

def `__init__`(self, پارامترها):

دستورات بدنه متد

```
def __init__(self, r):
    self.r = r
```

به عنوان مثال، دستورات زیر را ببینید:

این دستور سازنده کلاس Circle را ایجاد می‌کند

۱۰-۷. مسائل حل شده

مثال ۱. برنامه‌ای که کلاس دانشجو را پیاده‌سازی می‌کند. هر دانشجو دارای اعضای کد (شماره دانشجویی)، نام، نام خانوادگی، معدل و آیا مشروط شده است یا خیر؟ می‌باشد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی را ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

ماژول	متغیر	هدف
ماژول	code	شماره دانشجویی

نام دانشجو	fname	اصلی
نام خانوادگی دانشجو	lname	
معدل	average	
نمونه‌ای از کلاس Student	s	
متدی که مخرب کلاس Student را پیاده‌سازی می‌کند.	__del__	کلاس Student
متدی که سازنده نمونه‌ای از کلاس Student را با متد Str به رشته تبدیل می‌کند.	__str__	
متدی که سازنده کلاس Student را پیاده‌سازی می‌کند.	__init__	
متدی که تعیین می‌کند که آیا دانشجو مشروط شده است یا خیر؟	isPass	

class Student:

def __init__(self, code, fname, lname, average):

```
self.code = code
self.fname = fname
self.lname = lname
self.average = average
```

def __del__(self):

```
print("object is deleted")
```

def isPass(self):

```
if self.average >= 12 :
    return "Yes"
else :
    return "No"
```

def __str__(self):

```
s = "Code : " + self.code
s += "\t\tFirst name : " + self.fname
s += "\t\tLast name : " + self.lname
s += "\nAverage : " + str(self.average)
s += "\t\tPassed : " + str(self.isPass())
return s
```

code = input("Enter code:")

fname = input("Enter First name:")

lname = input("Enter Last name:")

average = float(input("Enter average:"))

s = Student(code, fname, lname, average)

print(str(s))

del s

۲. ماژول را ذخیره و اجرا کرده، اطلاعات زیر وارد را نمایید تا خروجی را ببینید:

Enter code:931216049

Enter First name:Ramin

Enter Last name:Baboli

Enter average:12.75

Code : 931216049

First name : Ramin

Last name : Baboli

Average : 12.75

Passed : Yes

object is deleted

پیوست مسائل تکمیلی

۱. برنامه‌ای که تاریخ و زمان فعلی را نمایش می‌دهد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
import datetime
now = datetime.datetime.now()
print ("Current date and time : ", now.strftime("%Y-%m-%d %H:%M:%S"))
```

۲. ماژول را ذخیره و اجرا کرده تا نمونه خروجی را به صورت زیر ببینید:

Current date and time : 2017-06-10 20:22:32

۲. برنامه‌ای که نام فایلی را از کاربر خوانده، توسعه (پسوند) فایل را نمایش می‌دهد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
filename = input("Input the Filename: ")
f_extns = filename.split(".")
print ("The extension of the file is : " + repr(f_extns[-1]))
```

۲. ماژول را ذخیره و اجرا کرده تا نمونه خروجی را به صورت زیر ببینید:

Input the Filename: D:\1.zip
The extension of the file is : 'zip'

۳. برنامه‌ای که عدد صحیح n را خوانده، حاصل عبارت $n+nn+nnn$ را نمایش می‌دهد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
a = int(input("Input an integer : "))
n1 = int( "%s" % a )
n2 = int( "%s%s" % (a,a) )
n3 = int( "%s%s%s" % (a,a,a) )
print (n1, " + ", n2, " + ", n3, " = ", n1+ n2 +n3)
```

۲. ماژول را ذخیره و اجرا کرده تا نمونه خروجی را به صورت زیر ببینید:

Input an integer : 8
8 + 88 + 888 = 984.

۴. برنامه‌ای که سال و ماه را خوانده، تقویم را نشان می‌دهد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
import calendar
y = int(input("Input the year : "))
m = int(input("Input the month : "))
print(calendar.month(y, m))
```

۲. ماژول را ذخیره و اجرا کرده تا نمونه خروجی را به صورت زیر ببینید:

```
Input the year : 2017
Input the month : 8
August 2017
Mo Tu We Th Fr Sa Su
1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

۵. برنامه‌ای که اختلاف بین دو تاریخ را محاسبه کرده، نمایش می‌دهد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
from datetime import date
f_date = date(2014, 7, 2)
l_date = date(2014, 7, 11)
delta = l_date - f_date
print('Day is ', delta.days)
```

۲. ماژول را ذخیره و اجرا کرده تا نمونه خروجی را به صورت زیر ببینید:

```
Day is 9
```

۶. برنامه‌ای که یک لیست را دریافت کرده، با ستاره و مقادیر لیست هیستوگرام را رسم می‌کند.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
def histogram( items ):
    for n in items:
        output = ""
        times = n
        while( times > 0 ):
            output += '*'
            times = times - 1
        print(output)

histogram([3, 6, 4, 3, 6, 5])
```

۲. ماژول را ذخیره و اجرا کرده تا نمونه خروجی را به صورت زیر ببینید:

```
***
*****
****
***
*****
*****
```

۷. برنامه‌ای که دو عدد را خوانده، بزرگ‌ترین مقسوم‌علیه مشترک (GCO) آن‌ها را نمایش می‌دهد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
def gcd(x, y):
    gcd = 1
    if x % y == 0:
        return y
    for k in range(int(y / 2), 0, -1):
        if x % k == 0 and y % k == 0:
            gcd = k
            break
    return gcd
x = int(input("Enter x:"))
y = int(input("Enter y:"))
print("gcd is ", gcd(x, y))
```

۲. ماژول را ذخیره و اجرا کرده تا نمونه خروجی را به صورت زیر ببینید:

```
Enter x:12
Enter y:20
gcd is 4
```

۸. برنامه‌ای که دو عدد را خوانده، کوچک‌ترین مضرب مشترک آن‌ها (LCM) را نمایش می‌دهد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
def lcm(x, y):
    if x > y:
        z = x
    else:
        z = y
    while(True):
        if((z % x == 0) and (z % y == 0)):
            lcm = z
            break
        z += 1
    return lcm
x = int(input("Enter x:"))
```

```
y = int(input("Enter y:"))
print("lcm is ", lcm(x, y))
```

۲. ماژول را ذخیره و اجرا کرده تا نمونه خروجی را به صورت زیر ببینید:

```
Enter x:8
Enter y:12
lcm is 24
```

۹. برنامه‌ای که مبلغ فعلی، نرخ بهره و تعداد سال را خوانده و ارزش پول را برای چند سال آینده که از ورودی خوانده، محاسبه می‌کند.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
amt = int(input("Enter amount:"))
rate = float(input("Enter rate:"))
years = int(input("Enter years:"))
future_value = amt*((1+(0.01*rate)) ** years)
print("Future_value is %12.0f" %(future_value))
```

۲. ماژول را ذخیره و اجرا کرده تا نمونه خروجی را به صورت زیر ببینید:

```
Enter amount:1000000
Enter rate:7.5
Enter years:10
Future_value is 2061032
```

۱۰. برنامه‌ای که چک می‌کند آیا فایلی وجود دارد یا خیر؟

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
import os.path
fileName = input("Enter filename:")
open(fileName, 'w')
print(os.path.isfile(fileName))
```

۲. ماژول را ذخیره و اجرا کرده تا نمونه خروجی را به صورت زیر ببینید:

```
Enter filename:1.txt
True
```

۱۱. برنامه‌ای که تعیین می‌کند آیا پوسته اجرا ۳۲ یا ۶۴ بیتی است؟

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
import struct
print(struct.calcsize("P") * 8)
```


۲. ماژول را ذخیره و اجرا کرده تا نمونه خروجی را به صورت زیر ببینید:

64

۱۲. برنامه‌ای که نام سیستم عامل، اطلاعات پلت فرم release را نمایش می‌دهد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
import platform
import os
print("Name is ", os.name)
print("Platform system is ", platform.system())
print("Platform release is ", platform.release())
```

۲. ماژول را ذخیره و اجرا کرده تا نمونه خروجی را به صورت زیر ببینید:

```
Name is nt
Platform system is Windows
Platform release is 7
```

۱۳. برنامه‌ای که نام سایت بسته‌های پایتون را نمایش می‌دهد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
import site;
print("Site package is ", site.getsitepackages())
```

۲. ماژول را ذخیره و اجرا کرده تا نمونه خروجی را به صورت زیر ببینید:

```
Site package is ['C:\\Users\\Fansno\\AppData\\Local\\Programs\\Python\\Python35',
'C:\\Users\\Fansno\\AppData\\Local\\Programs\\Python\\Python35\\lib\\site-packages']
```

۱۴. برنامه‌ای که یک برنامه اجرایی خارجی را در پایتون اجرا می‌کند.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
from subprocess import call
call(["calc.exe"])
```

۲. ماژول را ذخیره و اجرا کرده تا نمونه خروجی را ببینید.

۱۵. برنامه‌ای که نام و مسیر فایل فعلی در حال اجرا را نمایش می‌دهد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
import os
print("Current File Name : ",os.path.realpath(__file__))
```

۲. ماژول را ذخیره و اجرا کرده تا نمونه خروجی را به صورت زیر ببینید:

Current File Name : D:\BookCSharp\python-book\program\site\py\15.py

۱۶. برنامه‌ای که تعداد پردازنده‌های رایانه را نمایش می‌دهد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
import multiprocessing
print("Count cpu is ", multiprocessing.cpucount())
```

۲. ماژول را ذخیره و اجرا کرده تا نمونه خروجی را به صورت زیر ببینید:

Count cpu is 2

۱۷. برنامه‌ای که به متغیرهای محیطی دست‌یابی پیدا می‌کند.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
import os
# Access all environment variables
print('*-----*')
print("Environment is ", os.environ)
print('*-----*')
# Access a particular environment variable
print("home is ", os.environ['HOME'])
print('*-----*')
print("Path is ", os.environ['PATH'])
print('*-----*')
```

۲. ماژول را ذخیره و اجرا کرده تا نمونه خروجی را به صورت زیر ببینید:

```
*-----*
Environment is environ({'COMSPEC': 'C:\\Windows\\system32\\cmd.exe',
'PROCESSOR_ARCHITECTURE': 'AMD64', 'LOGONSERVER': '\\\\FANSNO-PC',
'HOME': 'C:\\Users\\Fansno', 'LOCALAPPDATA':
'C:\\Users\\Fansno\\AppData\\Local', 'PROGRAMDATA': 'C:\\ProgramData',
'PROGRAMFILES': 'C:\\Program Files', 'OS': 'Windows_NT', 'PROGRAMW6432':
'C:\\Program Files', 'PROCESSOR_LEVEL': '20', 'COMMONPROGRAMFILES(X86)':
'C:\\Program Files' ....
```

کتاب شامل ۳۸۱ صفحه است که فایل الکترونیکی آن را می‌توانید از سایت کتابراه تهیه کنید.

<http://ktbr.ir/b۲۹۹۸۴>

طراحی سیستم‌های شی گرا با زبان C#

برای رشته‌های:

مهندسی کامپیوتر، فناوری اطلاعات

مهندس رمضان عباس نژادووری

تالیف: مهندس باقر رحیم پورکاسی

مهندس ابراهیم هاشمیان



برخی از عناوین مهم

- آشنایی با زبان C#
- برنامه‌نویسی تحت کنسول
- ساختارهای کنترل
- پیاده‌سازی متدها
- ارزیابی و رشته‌ها
- کلاس‌ها
- وراثت و واسط‌ها
- برنامه‌نویسی تحت فرم
- برنامه‌نویسی بانک اطلاعاتی
- پروژه‌های متعدد با C#
- ارائه و حل مسائل مختلف

حل مسائل C++

(آزمایشگاه کامپیوتر مرجع کامل)



دکتر رمضان عباس نژادورزی
مؤلفین: مهندس عمران یونسی
مهندس یوسف عباس نژادورزی

برخی از عناوین مهم

مقدمه‌ای بر C++
ساختار تصمیم و حلقه تکرار
توابع در C++
آرایه‌ها و رشته‌ها
حل بیش از ۳۲۵ مسئله برنامه نویسی
حل بیش از ۱۴ پروژه برنامه نویسی

درس و کنکور پایگاه داده پیشرفته



دکتر رمضان عباس نژادورزی
مؤلفین: دکتر مرتضی بابا زاده
دکتر میر سعید حسینی

برخی از عناوین مهم

تراکنش ها
تئوری پی در پی پذیری
قفل گذاری ها
زمان بندی بدون قفل
ترمیم پایگاه داده
امنیت در پایگاه داده
حل تشریحی بیش از ۲۵۰ پرسش های چهار گزینه ای
حل تشریحی بیش از ۱۰۰ مسئله پایگاه داده پیشرفته