برنامه نویسی به زبان

پایتون

مرجع کامل

تالیف
مهندس هادی کیامرثی

تمام مثال های موجود در این کتاب با کامپیوتر تست شده اند تا از هر گونه خطا مبرا باشند با این حال ممکن است باز هم خطاهایی در آن وجود داشته باشد از کلیه خوانندگان این کتاب ، اساتید و دانشجویان محترم خواهشمندم برای مطلع کردن مولف از این خطا ها لطفا با ایمیل آدرس زیر تماس بگیرید

hadikiamarsi@gmail.com

فصل یازدهم

در این فصل مطالب زیر را خواهید آموخت

استثناها و مدیریت خطا ها در پایتون (python)

دستور assert

استثنا (Exception) چیست ؟

مدیریت یک استثنا (exception)

مدیریت استثنا (Exception) بدون ذکر نوع استثنا (Exception)

مدیریت استثنا (Exception) با تعیین چند نوع استثنا (Exception)

مدریت خطا (error handling) با try-finally

آرگومان های یک استثنا (Exception)

ایجاد استثنا (Exception)

استثناهای (Exceptions) تعریف شده توسط کاربر

استثناها و مدیریت خطا ها در پایتون (python)

در برنامه نویسی بوجود آمدن خطا و استثنا اجتناب ناپذیر می باشد ولی مهم مدیریت آن ها می باشد در زبان برنامه نویسی پایتون (python) ابزارهایی برای برای مدیریت خطاها و استثناها وجود دارد که در این فصل با آن ها آشنا می گردید

در زیر لیست خطاها و استثناهای پیش فرض در پایتون (python) آورده شده است

استثناها و توضیحات
Exception کلاس پایه برای تمام استثناها
StopIteration این استثنا در حلقه ها رخ می دهد
SystemExit این استثنا در زمان فراخوانی متد sys.exit() ایجاد می گردد
StandardError این استثنا برای همه متد های پیش ساخته رخ می دهد
ArithmeticError برای تمام خطاهایی که در محاسبات اعداد رخ می دهد بکار می رود
OverflowError زمانی که در محاسبات سرریز ایجاد شده باشد رخ می دهد
FloatingPointError زمانی که در محاسبات اعشاری خطا رخ دهد بکار می رود
ZeroDivisionError زمانی که خطای یک عدد تقسیم بر صفر رخ دهد بکار می رود
AssertionError در حالتی بروز می کند که اجرای کد به هر دلیلی شکست بخورد

AttributeError

در صورتی بروز می کند که یک انتصاب اشتباه رخ بدهد

EOFError

زمان استفاده از توابع `raw_input()` و `input()` اگر خطا رخ دهد بکار می آید

ImportError

زمانی که خطایی در دستور `import` رخ دهد بکار می آید

KeyboardInterrupt

زمانی که خطایی در دستور `Ctrl+c` صادر شده از طرف کاربر رخ بدهد بکار می آید

LookupError

زمانی که خطایی در جستجو ایجاد گردد بکار می آید

IndexError

زمانی که اندیس تخصیص داده شده در لیست موجود نباشد بکار می آید

KeyError

زمانی که کلید در دیکشنری وجود نداشته باشد بکار می آید

NameError

زمانی که یک کلمه شناسه وجود نداشته باشد

UnboundLocalError

زمانی که برای یک متغیر محلی مقداری در نظر نگرفته شده باشد

EnvironmentError

زمانی که خطایی مربوط به خارج از محیط پایتون ایجاد گردد بکار می آید

IOError

زمانی که خطایی در باز کردن فایل ها ایجاد گردد بکار می آید

IOError

زمانی که خطایی در ورودی خروجی پیش آید بکار می آید

SyntaxError

زمانی که خطایی در دستورات نحوی پایتون ایجاد گردد بکار می آید

IndentationError

زمانی که تو رفتگی در کدهای پایتون رعایت نشده باشد بکار می آید

SystemError

زمانی که خطایی در مفسر پایتون ایجاد گردد بکار می آید

SystemExit

زمانی که خطایی در متد `sys.exit()` بوجود آید بکار می آید

TypeError

زمانی که یک عملگر برای یک ساختار داده ای غیر معتبر بکار رود بکار می آید

ValueError

زمانی که خطایی در آرگومان یک تابع پیش ساخته ایجاد گردد بکار می آید

RuntimeError

زمانی که خطای ایجاد شده در هیچکدام از گروه خطاهای بالا جای نگیرد بکار می آید

NotImplementedError

زمانی که متد یک کلاس از کلاسی ارث برده شده باشد که آن کلاس وجود نداشته باشد بکار می آید

دستور assert

این دستور همانند یک دستور شرطی عمل می نماید و عبارتی را بررسی می نماید و در صورت `false` بودن خطایی را صادر می نماید

دستور نحوی آن را در زیر می بینید

```
assert Expression[, Arguments]
```

برای آشنایی بیشتر با این دستور به مثال زیر توجه نمایید

```
#!/usr/bin/python
def KelvinToFahrenheit(Temperature):
   assert (Temperature >= 0),"Colder than absolute zero!"
   return ((Temperature-273)*1.8)+32
print KelvinToFahrenheit(273)
print int(KelvinToFahrenheit(505.78))
print KelvinToFahrenheit(-5)
```

اجرای کد بالا نتیجه زیر را در صفحه خروجی ظاهر خواهد نمود

```
32.0
451
Traceback (most recent call last):
File "test.py", line 9, in <module>
```

```
print KelvinToFahrenheit(-5)
File "test.py", line 4, in KelvinToFahrenheit
assert (Temperature >= 0),"Colder than absolute zero!"
AssertionError: Colder than absolute zero!
```

استثنا (Exception) چیست ؟

استثنا (exception) یک رویداد می باشد که در روند اجرای یک برنامه یا کد رخ می دهد رخ دادن یک استثنا اگر مدیریت نشود باعث توقف اجرای برنامه یا کد می گردد

مدیریت یک استثنا (exception)

مدیریت یک استثنا (exception) و خطا به این معنا می باشد که عملی انجام بدهیم که استثنا و خطا موجب توقف روند اجرای برنامه نگردد برای انجام این کار در دنیای برنامه نویسی دو روش وجود دارد اولین روش استفاده از بلاک های تصمیم گیری می باشد و دومین روش استفاده از ابزارهای مدیریت خطا و استثنایی که به طور پیش فرض در یک زبان برنامه نویسی وجود دارد یکی از این ابزارها بلاک try .. except می باشد

راهنمای نحوی

```
try:
   You do your operations here;
   ......................
except ExceptionI:
   If there is ExceptionI, then execute this block.
except ExceptionII:
   If there is ExceptionII, then execute this block.
   ......................
else:
   If there is no exception then execute this block.
```

در زیر نکات مهمدر مورد بلاک try آورده شده است

هر دستور try می تواند چندتا except داشته باشد

هر دستور except می تواند در درون خودش چند دستور except داشته باشد

بعد از دستور except می توانید از دستور else استفاده نمایید

مثال

در مثال زیر اگر فایل testfile وجود نداشته باشد پیغام خطای مناسب ظاهر می گردد

```
#!/usr/bin/python

try:
   fh = open("testfile", "w")
   fh.write("This is my test file for exception handling!!")
except IOError:
   print "Error: can\'t find file or read data"
else:
   print "Written content in the file successfully"
   fh.close()
```

اجرای کد بالا نتیجه زیر را در صفحه خروجی ظاهر خواهد نمود

```
Written content in the file successfully
```

مثال

```
#!/usr/bin/python

try:
   fh = open("testfile", "r")
   fh.write("This is my test file for exception handling!!")
except IOError:
   print "Error: can\'t find file or read data"
else:
   print "Written content in the file successfully"
```

اجرای کد بالا نتیجه زیر را در صفحه خروجی ظاهر خواهد نمود

```
Error: can't find file or read data
```

مدیریت استثنا (Exception) بدون ذکر نوع استثنا (Exception)

شما می توانید به مدیریت خطاها و استثناها (exception) بپردازید بدون اینکه نوع خطا یا استثنا (exception) را بدانید

```
try:
   You do your operations here;
   ......................
except:
```

```
   If there is any exception, then execute this block.
   ......................
else:
   If there is no exception then execute this block.
```

اگر در جلوی except نوع خطا نوشته نگردد به صورت خودکار هرنوع خطا و استثنایی (exception) را پوشش می دهد

مدیریت استثنا (Exception) با تعیین چند نوع استثنا (Exception)

شما می توانید در جلوی except چند نوع خطا و استثنا (exception) را ذکر نمایید

```
try:
   You do your operations here;
   ......................
except(Exception1[, Exception2[,...ExceptionN]]]):
   If there is any exception from the given exception list,
   then execute this block.
   ......................
else:
   If there is no exception then execute this block.
```

مدریت خطا (error handling) با try-finally

بلاک finally را می توانید در موقعیت هایی که دوست ندارید نوع خطا و استثنا (exception) را تعیین نمایید بکار ببرید

ساختار نحوی آن در زیر آورده شده است

```
try:
   You do your operations here;
   ......................
   Due to any exception, this may be skipped.
finally:
   This would always be executed.
   ......................
```

بیاد داشته باشید شما نمی توانید از else در بدنه بلاک finally استفاده نمایید

مثال

```
#!/usr/bin/python

try:
   fh = open("testfile", "w")
   fh.write("This is my test file for exception handling!!")
finally:
   print "Error: can\'t find file or read data"
```

اجرای کد بالا نتیجه زیر را در صفحه خروجی ظاهر خواهد نمود

```
Error: can't find file or read data
```

مثالی دیگر

```
#!/usr/bin/python

try:
   fh = open("testfile", "w")
   try:
      fh.write("This is my test file for exception handling!!")
   finally:
      print "Going to close the file"
      fh.close()
except IOError:
   print "Error: can\'t find file or read data"
```

شما می توانید بلاک های مدیریت خطا و استثنا (exception) را به صورت تو در تو بکار ببرید همانطور که در مثال بالا دیده می شود

آرگومان های یک استثنا (Exception)

بلاک exception می تواند یک ورودی یا همان آرگومان بپذیرد که توضیحات اضافه ایجاد خطا و استثنا (exception) در آن قرار می گیرد برای آشنایی بیشتر با این درس به مثال زیر توجه نمایید

```
try:
   You do your operations here;
   ......................
except ExceptionType, Argument:
   You can print value of Argument here...
```

لازم به ذکر است این آرگومان می تواند به هر اسمی باشد برای آشنایی بیشتر به مثال زیر توجه نمایید

```
#!/usr/bin/python

# Define a function here.
def temp_convert(var):
   try:
      return int(var)
   except ValueError, Argument:
      print "The argument does not contain numbers\n", Argument

# Call above function here.
temp_convert("xyz");
```

اجرای کد بالا نتیجه زیر را در صفحه خروجی ظاهر خواهد نمود

```
The argument does not contain numbers
invalid literal for int() with base 10: 'xyz'
```

ایجاد استثنا (Exception)

در زبان برنامه نویسی پایتون (python) برنامه نیس می تواند یک استثنا را به صورت دستی ایجاد نماید این کار با تابع پیش فرض raise انجام می پذیرد

راهنمای نحوی

```
raise [Exception [, args [, traceback]]]
```

در دستور نحوی بالا exception نوع استثنا می باشد args آرگومان می باشد که اختیاری می باشد و traceback میزان سطح بررسی استثنا را مشخص می نماید که اختیاری می باشد.

مثال

```
def functionName( level ):
   if level < 1:
      raise "Invalid level!", level
      # The code below to this would not be executed
      # if we raise the exception
```

برای استفاده از کد بالا باید بوسیله کد زیر خطا را گیر بیاندازید

```
try:
   Business Logic here...
except "Invalid level!":
```

```
   Exception handling here...
else:
   Rest of the code here...
```

استثناهای (Exceptions) تعریف شده توسط کاربر

در زبان برنامه نویسی پایتون (python) شما به عنوان برنامه نویس می توانید یک استثنا را خودتان تعریف نمایید . البته برای انجام این کار باید یک کلاس طراحی نمایید که این کلاس باید از یکی از خطاها و استثناهای از پیش تعریف شده در جدول اول این فصل مشتق شده باشد . برای آشنایی بیشتر به مثال زیر توجه نمایید در مثال زیر کلاس تعریف شده از کلاس پیش فرض RuntimeError مشتق شده است

```
class Networkerror(RuntimeError):
   def __init__(self, arg):
      self.args = arg
```

کلاس استثنا تعریف شده در بالا را به شکل زیر می توانید به صورت زیر صدا بزنید یا اجرا نمایید

```
try:
   raise Networkerror("Bad hostname")
except Networkerror,e:
   print e.args
```