برنامه نویسی به زبان

پایتون

مرجع کامل

تالیف
مهندس هادی کیامرثی

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

لازم به ذکر است کلیه حقوق مادی و معنوی این اثر برای مولف محفوظ می باشد و هرگونه کپی برداری و استفاده از محتویات این کتاب به هر نوعی تحت پیگرد قانونی قرار می گیرد

هادی کیامرثی

فصل پنجم

هادی پیامرثی

در این فصل مطالب زیر را خواهید آموخت

شی گرایی در پایتون

ایجاد کلاس

ایجاد اشیا

دسترسی به صفات

صفت های پیش ساخته کلاس ها

نابود کردن اشیا (Garbage Collection)

وراثت (Inheritance)

تعریف مجدد تابع ها (Overriding Methods)

متدهای پیش ساخته کلاس ها

تعریف مجدد عملگرها (Overloading Operators)

پنهان سازی اطلاعات (Data Hiding)

هادی کرامتی

شی گرایی در پایتون

در برنامه نویسی دو سبک عمده برنامه نویسی وجود دارد یکی برنامه نویسی ساخت یافته می باشد و یکی برنامه نویسی شی گرا (object-oriented) می باشد زبان برنامه نویسی پایتون (python) از هر دوسبک برنامه نویسی پشتیبانی می نماید و در این فصل شما با سبک برنامه نویسی شی گرا (object-oriented) در زبان برنامه نویسی پایتون (python) آشنا می گردید . مفاهیم شی گرایی برای کسانی که با این سبک برنامه نویسی آشنایی ندارند شاید کمی سخت باشد ولی در این کتاب سعی شده تا حد ممکن مفاهیم ساده بیان گردند . شاید یک برنامه نویس احساس نماید به برنامه نویسی شی گرا Object-Oriented Programming (OOP) نیازی ندارد ولی موقعیت هایی پیش می آید که تنها این سبک برنامه نویسی راه گشاست .

ایجاد کلاس

در زبان برنامه نویسی پایتون (python) برای ایجاد یک کلاس از کلمه کلیدی class استفاده می گردد ساختار یک کلاس در زبان برنامه نویسی پایتون (python) در زیر نشان داده شده است

```
class ClassName:
   'Optional class documentation string'
   class_suite
```

بیاد داشته باشید یک توضیح اول هر کلاس قرار می گیرد که به صورت زیر قابل دستیابی می باشد

ClassName.__doc__.

به عبارتی برای دستیابی به این توضیح باید نام کلاس را به همراه __doc__ بکار برید .

برای آشنایی بیشتربه مثال زیر توجه نمایید

```
class Employee:
   'Common base class for all employees'
   empCount = 0
```

```
   def __init__(self, name, salary):
      self.name = name
      self.salary = salary
      Employee.empCount += 1

   def displayCount(self):
     print "Total Employee %d" % Employee.empCount

   def displayEmployee(self):
      print "Name : ", self.name,  ", Salary: ", self.salary
```

در مثال بالا __init__() یک متد پیش ساخته می باشد که به سازنده (constructor) معروف است کدهای نوشته شده در این بخش زمانی که کلاس ایجاد می گردد اجرا می گردند به این معنا که کدهای موجود در این بخش پیش از همه متدهای دیگر کلاس اجرا می گردند

ایجاد اشیا

برای استفاده از کلاس ها باید اشیایی از روی آن ها بسازید در زبان برنامه نویسی پایتون (python) این کار بوسیله عملگر تساوی (=) انجام می پذیرد برای آشنایی بیشتر با این درس به مثال زیر توجه نمایید

```
"This would create first object of Employee class"
emp1 = Employee("Zara", 2000)
"This would create second object of Employee class"
emp2 = Employee("Manni", 5000)
```

دسترسی به صفات

در زبان برنامه نویسی پایتون (python) برای دسترسی به صفات یک کلاس از عملگر نقطه (dot) استفاده می گردد برای آشنایی بیشتر با این درس به مثال زیر توجه نمایید

```
emp1.displayEmployee()
```

```
emp2.displayEmployee()
print "Total Employee %d" % Employee.empCount
```

برای آشنایی بیشتربه مثال زیر توجه نمایید

```
#!/usr/bin/python

class Employee:
   'Common base class for all employees'
   empCount = 0

   def __init__(self, name, salary):
      self.name = name
      self.salary = salary
      Employee.empCount += 1

   def displayCount(self):
     print "Total Employee %d" % Employee.empCount

   def displayEmployee(self):
      print "Name : ", self.name,  ", Salary: ", self.salary

"This would create first object of Employee class"
emp1 = Employee("Zara", 2000)
"This would create second object of Employee class"
emp2 = Employee("Manni", 5000)
emp1.displayEmployee()
emp2.displayEmployee()
print "Total Employee %d" % Employee.empCount
```

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

```
Name :  Zara ,Salary:  2000
Name :  Manni ,Salary:  5000
Total Employee 2
```

شما در زبان برنامه نویسی پایتون (python) هر زمان که بخواهید می توانید صفت ها را اضافه ، حذف یا تغییر دهید

```
emp1.age = 7  # Add an 'age' attribute.
emp1.age = 8  # Modify 'age' attribute.
del emp1.age  # Delete 'age' attribute.
```

در زبان برنامه نویسی پایتون (python) برای دسترسی به صفات علاوه بر استفاده از نقطه (dot) تعدادی تابع نیز وجود دارد که برنامه نویس را در این زمینه یاری می دهد نام و کاربرد این توابع در زیر آورده شده است

- **getattr(obj, name[, default])** – دسترسی به صفت از یک شی
- **hasattr(obj,name)** – بررسی وجود یا عدم وجود یک صفت
- **setattr(obj,name,value)** – مقدار دهی به یک صفت اگر وجود داشته باشد

- **delattr(obj, name)** – حذف یک صفت

```
hasattr(emp1, 'age')    # Returns true if 'age' attribute exists
getattr(emp1, 'age')    # Returns value of 'age' attribute
setattr(emp1, 'age', 8) # Set attribute 'age' at 8
delattr(empl, 'age')    # Delete attribute 'age'
```

صفت های پیش ساخته کلاس ها

در زبان برنامه نویسی پایتون (python) یک سری صفت های پیش ساخته وجود دارد که کار با کلاس ها را برای برنامه نویس راحتر می نماید لیست و کاربرد این صفت ها در زیر آورده شده است

- **__dict__** – یک دیکشنری شامل فضای نام کلاس
- **__doc__** – توضیحات کلاس را درخود نگهداری می نماید
- **__name__** – نام کلاس
- **__module__** – نام ماژول مورد نظر که کلاس در آن تعریف شده است
- **__bases__** – یک تاپل حاوی اطلاعات کلاس را در خود نگه می دارد

برای آشنایی بیشتربه مثال زیر توجه نمایید

```
#!/usr/bin/python

class Employee:
   'Common base class for all employees'
   empCount = 0

   def __init__(self, name, salary):
      self.name = name
      self.salary = salary
      Employee.empCount += 1

   def displayCount(self):
     print "Total Employee %d" % Employee.empCount

   def displayEmployee(self):
      print "Name : ", self.name,  ", Salary: ", self.salary

print "Employee.__doc__:", Employee.__doc__
print "Employee.__name__:", Employee.__name__
print "Employee.__module__:", Employee.__module__
print "Employee.__bases__:", Employee.__bases__
print "Employee.__dict__:", Employee.__dict__
```

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

```
Employee.__doc__: Common base class for all employees
Employee.__name__: Employee
Employee.__module__: __main__
Employee.__bases__: ()
Employee.__dict__: {'__module__': '__main__', 'displayCount':
<function displayCount at 0xb7c84994>, 'empCount': 2,
'displayEmployee': <function displayEmployee at 0xb7c8441c>,
'__doc__': 'Common base class for all employees',
'__init__': <function __init__ at 0xb7c846bc>}
```

نابود کردن اشیا (Garbage Collection)

زبان برنامه نویسی پایتون (python) به صورت خودکار اشیایی را که نیاز ندارد را ازبین می برد تا فضای حافظه آزاد گردد که به این عمل نابودکردن اشیا به صورت خودکار (Garbage Collection) گفته می شود در زبان برنامه نویسی پایتون (python) برای نابود کردن ارجاع ها (reference) از تابع del استفاده می گردد مشابه همین روش برای اشیا نیز اتفاق می افتد برای آشنایی بیشتر به مثال زیر توجه نمایید

```
a = 40      # Create object <40>
b = a       # Increase ref. count  of <40>
c = [b]     # Increase ref. count  of <40>

del a       # Decrease ref. count  of <40>
b = 100     # Decrease ref. count  of <40>
c[0] = -1   # Decrease ref. count  of <40>
```

اما بیاد داشته باشید عمل نابودکردن اشیا به صورت خودکار (Garbage Collection) را شما می توانید به وسیله متد پیش ساخته ای به نام مخرب (destructor) به صورت دستی هم انجام بدهید برای این کار لازم است که __del__() را در کلاس خود تعریف نمایید

برای آشنایی بیشتربه مثال زیر توجه نمایید

```
#!/usr/bin/python

class Point:
   def __init__( self, x=0, y=0):
      self.x = x
      self.y = y
   def __del__(self):
```

```
      class_name = self.__class__.__name__
      print class_name, "destroyed"

pt1 = Point()
pt2 = pt1
pt3 = pt1
print id(pt1), id(pt2), id(pt3) # prints the ids of the obejcts
del pt1
del pt2
del pt3
```

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

```
3083401324 3083401324 3083401324
Point destroyed
```

وراثت (Inheritance)

خیلی اوقات برای اینکه برنامه نویس بتواند از متدهایی که در کلاس های دیگر وجود دارد استفاده نماید و دیگر نیاز نباشد آن ها را مجددا بنویسد از قابلیتی استفاده می گردد به نام وراثت (Inheritance) به این معنا که یک کلاس که اولاد یک کلاس دیگر قرار می گیرد تمام صفت ها و متد های کلاس والد خود را به ارث می برد و می تواند از آن ها استفاده نماید برای این کار در زبان برنامه نویسی پایتون (python) باید اسم کلاس والد را در جلوی نام کلاس اولاد در داخل پرانتز بیاوریم

ساختار نحوی

ساختار نحوی یک کلاس اولاد در زیر آورده شده است

```
class SubClassName (ParentClass1[, ParentClass2, ...]):
   'Optional class documentation string'
   class_suite
```

برای آشنایی بیشتربه مثال زیر توجه نمایید

```
#!/usr/bin/python

class Parent:        # define parent class
   parentAttr = 100
   def __init__(self):
```

```
      print "Calling parent constructor"

   def parentMethod(self):
      print 'Calling parent method'

   def setAttr(self, attr):
      Parent.parentAttr = attr

   def getAttr(self):
      print "Parent attribute :", Parent.parentAttr

class Child(Parent): # define child class
   def __init__(self):
      print "Calling child constructor"

   def childMethod(self):
      print 'Calling child method'

c = Child()           # instance of child
c.childMethod()       # child calls its method
c.parentMethod()      # calls parent's method
c.setAttr(200)        # again call parent's method
c.getAttr()           # again call parent's method
```

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

```
Calling child constructor
Calling child method
Calling parent method
Parent attribute : 200
```

بیاد داشته باشید در زبان برنامه نویسی پایتون (python) یک کلاس اولاد می تواند چند والد داشته باشد مانند مثال زیر

```
class A:        # define your class A
.....

class B:         # define your class B
.....

class C(A, B):   # subclass of A and B
.....
```

برای بررسی رابطه بین دو کلاس در زبان برنامه نویسی پایتون (python) می توانید از تابع issubclass() و isinstance() استفاده نمایید روش استفاده از این دو تابع در زیر آورده شده است

- **issubclass(کلاس اولاد, کلاس والد)**
- **isinstance(نام کلاس, یک شی از یک کلاس)**

تعریف مجدد تابع ها (Overriding Methods)

در زبان برنامه نویسی پایتون (python) می توان متدهای کلاس والد (parent class) را مجددا تعریف کرد و از آن ها استفاده کرد بدون اینکه تداخلی ایجاد گردد برای آشنایی بیشتر به مثال زیر توجه نمایید

```
#!/usr/bin/python

class Parent:        # define parent class
   def myMethod(self):
      print 'Calling parent method'

class Child(Parent): # define child class
   def myMethod(self):
      print 'Calling child method'

c = Child()          # instance of child
c.myMethod()         # child calls overridden method
```

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

```
Calling child method
```

متدهای پیش ساخته کلاس ها

برای کلاس ها در زبان برنامه نویسی پایتون (python) یک سری متد پیش ساخته وجود دارد که در زیر لیست آن ها آورده شده است

متدهای پیش ساخته کلاس ها

__init__ (self [,args...])

سازنده

__del__(self)

مخرب

__repr__(self)

اطلاعاتی در مورد یک شی بر می گرداند

__str__(self)

چاپ رشته

__cmp__ (self, x)

مقایسه اشیا

تعریف مجدد عملگرها (Overloading Operators)

خیلی وقت ها پیش می آید که شما خواسته ای که از یک عملگر دارید برآورده نمی شود بنابر این نیاز دارید نحوه عملکرد عملگر را طبق نیاز خود تغییر دهید این کار که تعریف مجدد عملگرها نامیده می شود براحتی در زبان برنامه نویسی پایتون (python) قابل پیاده سازی می باشد به عنوان مثال فرض کنید نیاز داشته باشید دو عدد Vector را با هم جمع نمایید اگر از عملگر + استفاده نمایید نتیجه ای را که نیاز دارید برآورده نمی شود پس نیاز به تعریف مجدد عملگر + برای Vector داریم که این عمل در مثال زیر نشان داده شده است

```
#!/usr/bin/python

class Vector:
   def __init__(self, a, b):
      self.a = a
      self.b = b

   def __str__(self):
      return 'Vector (%d, %d)' % (self.a, self.b)

   def __add__(self,other):
      return Vector(self.a + other.a, self.b + other.b)

v1 = Vector(2,10)
v2 = Vector(5,-2)
print v1 + v2
```

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

```
Vector(7,8)
```

پنهان سازی اطلاعات (Data Hiding)

پنهان سازی اطلاعات یکی از ویژگی های مهم برنامه نویسی شی گرا (object-oriented) می باشد به این معنا که برنامه نویس می تواند صفت ها را در یک کلاس به گونه ای تعریف نماید که کاربران دیگر نتوانند به آن ها دسترسی داشته باشند به این صفت ها خصوصی (private) گفته می شود برای تعریف چنین صفت هایی در زبان برنامه نویسی پایتون (python) باید در اول نام یک صفت از دو تا علامت _ یا همان underscore استفاده نمایید

برای آشنایی بیشتربه مثال زیر توجه نمایید

```
#!/usr/bin/python

class JustCounter:
   __secretCount = 0

   def count(self):
      self.__secretCount += 1
      print self.__secretCount

counter = JustCounter()
counter.count()
counter.count()
print counter.__secretCount
```

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

```
1
2
Traceback (most recent call last):
   File "test.py", line 12, in <module>
      print counter.__secretCount
AttributeError: JustCounter instance has no attribute '__secretCount'
```

مثالی دیگر از روش دسترسی به صفت های خصوصی

```
print counter._JustCounter__secretCount
```

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

```
1
2
2
```