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 چهارم

در این فصل مطالب زیر را خواهید آموخت

تعریف یک تابع

اجرای یک تابع

پاس دادن آرگومان با ارجاع در مقابل پاس دادن با مقدار

آرگومان های تابع

استفاده از نام آرگومان ها

آرگومان های پیش فرض

طول متغیرهای آرگومان ها

توابع ناشناس

دستور `return`

محدوده متغیرها

متغیرهای محلی در برابر متغیرهای عمومی

توابع در پایتون

تابع یک بلاک از کدهای زبان برنامه نویسی می باشد که به طور مستقل اجرا می گردد و تاثیر زیادی در خوانایی کد برنامه و تسهیل برنامه نویسی گروهی دارد . توابع در زبان برنامه نویسی پایتون (python) به دو دسته توابع پیش ساخته (built-in) و توابع ساخت کابر (user-defined) تقسیم می گردند

توابع پیش ساخته مانند تابع print که از قبل در زبان برنامه نویسی پایتون (python) تعبیه شده است که در فصل های قبل با آن آشنا شدید

تعریف یک تابع

شما براحتی می توانید توابع را در زبان برنامه نویسی پایتون (python) تعریف نمایید به شرطی که قوانین زیر را در آن رعایت نمایید

بلاک توابع در زبان برنامه نویسی پایتون (python) با کلمه کلیدی def به همراه نام تابع و پرانتز باز و بسته شروع می شوند

پارامترها یا آرگومان ها درون همین پرانتز جلوی نام تابع قرار می گیرند

در انتهای خط تعریف تابع بعد از پرانتزها از علامت کولون (colon) : استفده می گردد

اولین خط از بلاک تابع به صورت اختیاری می تواند توضیحات قرار بگیرد که اصطلاحا (docstring) نامیده می شود

ساختار نحوی

```
def functionname( parameters ):
   "function_docstring"
   function_suite
   return [expression]
```

به صورت اختیاری هرتابع می تواند آرگومان داشته باشد یا نداشته باشد.

مثال

مثالی از یک تابع پیاده سازی شده در زبان برنامه نویسی پایتون (python) در زیر نشان داده شده است

```
def printme( str ):
   "This prints a passed string into this function"
   print str
   return
```

اجرای یک تابع

برای اجرای یک تابع یا به عبارت دیگر صدا زدن یک تابع در زبان برنامه نویسی پایتون (python) می توانید از نام تابع استفاده نمایید . به عبارتی نام تابع را به همراه پرانتز باز بسته بنویسید . برای آشنایی بیشتر با این مبحث به مثال زیر توجه نمایید

```
#!/usr/bin/python

# Function definition is here
def printme( str ):
   "This prints a passed string into this function"
   print str
   return;

# Now you can call printme function
printme("I'm first call to user defined function!")
printme("Again second call to the same function")
```

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

```
I'm first call to user defined function!
Again second call to the same function
```

پاس دادن آرگومان با ارجاع در مقابل پاس دادن با مقدار

تمام آرگومان ها به طور پیش فرض در زبان برنامه نویسی پایتون (python) به روش ارجاع به توابع پاس داده می شوند به مثال زیر توجه نمایید

```
#!/usr/bin/python

# Function definition is here
```

```
def changeme( mylist ):
   "This changes a passed list into this function"
   mylist.append([1,2,3,4]);
   print "Values inside the function: ", mylist
   return

# Now you can call changeme function
mylist = [10,20,30];
changeme( mylist );
print "Values outside the function: ", mylist
```

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

```
Values inside the function:  [10, 20, 30, [1, 2, 3, 4]]
Values outside the function:  [10, 20, 30, [1, 2, 3, 4]]
```

به مثال دیگری در این زمینه توجه نمایید

```
#!/usr/bin/python

# Function definition is here
def changeme( mylist ):
   "This changes a passed list into this function"
   mylist = [1,2,3,4]; # This would assig new reference in mylist
   print "Values inside the function: ", mylist
   return

# Now you can call changeme function
mylist = [10,20,30];
changeme( mylist );
print "Values outside the function: ", mylist
```

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

```
Values inside the function:  [1, 2, 3, 4]
Values outside the function:  [10, 20, 30]
```

آرگومان های تابع

هر تابع می تواند آرگومان ها یا پارامترهایی داشته باشد که آنها درون پرانتز باز و بسته نوشته می شوند به یاد داشته باشید که اگر برای تابعی آرگومانی تعریف نمایید ولی مقداری را به آن پاس ندهید پیام خطایی ظاهر می گردد برای آشنایی بیشتر با این مبحث به مثال زیر توجه نمایید

```
#!/usr/bin/python

# Function definition is here
def printme( str ):
   "This prints a passed string into this function"
```

```
   print str
   return;

# Now you can call printme function
printme()
```

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

```
Traceback (most recent call last):
   File "test.py", line 11, in <module>
      printme();
TypeError: printme() takes exactly 1 argument (0 given)
```

استفاده از نام آرگومان ها

در زبان برنامه نویسی پایتون (python) باید به همان ترتیبی که آرگومان ها در تابع تعریف شده اند ، آن ها مقدار دهی گردند ولی در بعضی مواقع برنامه نویس نیاز دارد تا ترتیب انتقال مقدار به آرگومان ها را رعایت نکند دراین گونه مواقع در زبان برنامه نویسی پایتون (python) باید در جایی که می خواهید تابع را صدا بزنید نام آرگومان به همراه مقدار آن ذکر گردد برای آشنایی بیشتر با این درس به مثال زیر توجه نمایید

```
#!/usr/bin/python

# Function definition is here
def printme( str ):
   "This prints a passed string into this function"
   print str
   return;

# Now you can call printme function
printme( str = "My string")
```

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

```
My string
```

به مثال دیگری در زیر توجه نمایید

```
#!/usr/bin/python

# Function definition is here
def printinfo( name, age ):
   "This prints a passed info into this function"
   print "Name: ", name
   print "Age ", age
   return;

# Now you can call printinfo function
```

```
printinfo( age=50, name="miki" )
```

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

```
Name:  miki
Age  50
```

آرگومان های پیش فرض

در زبان برنامه نویسی پایتون (python) یک برنامه نویس می تواند به صورت پیش فرض مقداری را به یک آرگومان اختصاص دهد که در این صورت اگر برنامه نویس مقداری را به آرگومان اختصاص ندهد این مقدار پیش فرض مورد استفاده قرار می گیرد برای آشنایی بیشتر با این مبحث به مثال زیر توجه نمایید

```
#!/usr/bin/python

# Function definition is here
def printinfo( name, age = 35 ):
   "This prints a passed info into this function"
   print "Name: ", name
   print "Age ", age
   return;

# Now you can call printinfo function
printinfo( age=50, name="miki" )
printinfo( name="miki" )
```

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

```
Name:  miki
Age  50
Name:  miki
Age  35
```

طول متغیرهای آرگومان ها

در بعضی موقعیت ها نیاز می باشد که تعداد زیادی مقدار به یک آرگومان از تابع پاس داده شود در این حالت از شکل زیر استفاده می نماییم

```
def functionname([formal_args,] *var_args_tuple ):
   "function_docstring"
```

```
   function_suite
   return [expression]
```

اگر در قبل از نام یک آرگومان علامت (asterisk) یا همان علامت * قرار گیرد این آرگومان به عنوان آرگومان آخر در نظر گرفته می شود و هر تعداد مقدار پاس داده شود به این تابع همگی در همین آرگومان قرار می گیرند برای آشنایی بیشتر با این درس به مثال زیر توجه نمایید

```
#!/usr/bin/python

# Function definition is here
def printinfo( arg1, *vartuple ):
   "This prints a variable passed arguments"
   print "Output is: "
   print arg1
   for var in vartuple:
      print var
   return;

# Now you can call printinfo function
printinfo( 10 )
printinfo( 70, 60, 50 )
```

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

```
Output is:
10
Output is:
70
60
50
```

توابع ناشناس

این توابع ناشناس (anonymous) نامیده می شوند چرا که به صورت روال معمول زبان برنامه نویسی پایتون (python) پیاده سازی نمی شوند برای تعریف این نمونه از توابع از کلمه کلیدی Lambda استفاده می گردد

ساختار نحوی

بیاد داشته باشید توابع ناشناس (anonymous) فقط در یک خط پیاده سازی می گردند در زیر ساختار نحوی این نمونه از توابع نشان داده شده است

```
lambda [arg1 [,arg2,.....argn]]:expression
```

برای آشنایی بیشتر با این نمونه توابع به مثال زیر توجه نمایید

```
#!/usr/bin/python

# Function definition is here
sum = lambda arg1, arg2: arg1 + arg2;

# Now you can call sum as a function
print "Value of total : ", sum( 10, 20 )
print "Value of total : ", sum( 20, 20 )
```

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

```
Value of total :  30
Value of total :  40
```

دستور return

موقعیت هایی پیش می آید که برنامه نویس نیاز دارد تا نتیجه عملیات یک تابع به بدنه اصلی برنامه بر گردانده بشه که در این صورت در زبان برنامه نویسی پایتون (python) از دستور return استفاده می گردد برای آشنایی با نحوه کاربرد این دستور به مثال زیر توجه نمایید

```
#!/usr/bin/python

# Function definition is here
def sum( arg1, arg2 ):
   # Add both the parameters and return them."
   total = arg1 + arg2
   print "Inside the function : ", total
   return total;

# Now you can call sum function
total = sum( 10, 20 );
print "Outside the function : ", total
```

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

```
Inside the function :  30
Outside the function :  30
```

محدوده متغیرها

همه متغیرهای موجود در یک برنامه در همه قسمت های برنامه در دسترس نیستند و این وابسته به جایی هست که متغیر تعریف شده است .

در زبان برنامه نویسی پایتون (python) بر اساس محل تعریف شدن متغیر ، متغیرها به دو دسته محلی و عمومی تقسیم می گردند

متغیرهای محلی در برابر متغیرهای عمومی

متغیری که در درون یک تابع تعریف می شود متغیر محلی (local) نامیده می شود و فقط در درون همان تابع قابل استفاده می باشد و متغیری که خارج از بدنه تابع تعریف می شود متغیر عمومی (global) نامیده می شود و در تمام توابع قابل دسترسی می باشد . برای آشنایی بیشتر با این مبحث به مثال زیر توجه نمایید

```
#!/usr/bin/python

total = 0; # This is global variable.
# Function definition is here
def sum( arg1, arg2 ):
   # Add both the parameters and return them."
   total = arg1 + arg2; # Here total is local variable.
   print "Inside the function local total : ", total
   return total;

# Now you can call sum function
sum( 10, 20 );
print "Outside the function global total : ", total
```

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

```
Inside the function local total :  30
Outside the function global total :  0
```