

Myanmar Security Forum

HOME

MEMBERS

HELP DOCS

UPGRADE

AWARDS

BLACKLIST

GROUPS

Myanmar Security

NEW POSTS

Forum - MSF > Programming >
Python > [Explanation]
Generator, Yield, Lambdas

Search Poly...

Search

TODAY'S POSTS

Thread Rating:

New Reply

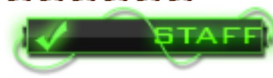
[Explanation] Generator, Yield, Lambdas

Thread Modes



Anubis

Moderator



Posts:
501
Threads:
0
Thanks
Received:
422 in
272 posts
Thanks
Given: 70
Joined:
Jun 2013
Reputation:
169



03-12-2016, 10:44 PM

#1

Python (Yield and Generators & Lambda)

Generators, lambda တွေနဲ့ yield တွေကတော့ နည်းနည်း advance ဖြစ်တဲ့ subtopic တွေဖြစ်သွားပါပြီ။ ဒီတော့ python basic ရသွားမှ နားလည်ရလွယ်မှာပါ။ Generator တွေဆိုတာ python object တစ်ခုဖြစ်ပြီး yield ကတော့ generator တွေနဲ့ တွဲသုံးလေ့ရှိတဲ့ statement တစ်မျိုးပါ။

Generators တွေကို အလွယ်ပြောရရင် item လေးတွေထုတ်ပေးဖို့ object တစ်မျိုးပါပဲ။ Generator တွေကို အသုံးပြုမယ်ဆိုရင် Generator function ရေးရပါမယ်။ Generator function ကတော့ တစ်ခြားမဟုတ်ပါဘူး။ ဘယ် def မဆို yield statement ပါလို့ရှိရင် အဲဒါ generator function လို့ခေါ်တာပါပဲ။

နောက်ပြီးတော့ အဲဒီလို function တွေက တစ်ခါ Generator iterator ဒါမှမဟုတ် တခြားနာမည်တစ်ခုဆိုရင် generator object တွေထုတ်ပေးပါတယ်။ (Iterator ကိုတော့ သိပြီးပြီလို့ ထင်ပါတယ်။ Basic မှာအဲဒါတွေအရမ်းသုံးရတယ်လေ။) Generator ပစ္စည်းတွေမှန်သမျှက next() ဆိုတဲ့ built-in function ပါပါတယ်။ အဲဒီကောင်လေးတွေကတော့ Generator တွေထဲကထုတ်ပေးတဲ့ နောက်ထပ် item တစ်ခုကို လှမ်းခေါ်လိုက်တဲ့သဘောပါပဲ။

Example တွေကို ကြည့်လိုက်ရင် သဘောပေါက်ပါလိမ့်မယ်။ ကျွန်တော်လဲ post မရေးတာကြာပြီဆိုတော့ ရေးတာ သိပ်မရှင်းဘူးဖြစ်သွားရင် ဆောရီးပါ။

အရင်ဆုံး Generator function တစ်ခုတည်ဆောက်ပါမယ်။

Code:

```
def simple_generator():
    for i in range(1,11):
        yield i
```

ဒါကတော့ function define လုပ်ပြီးသွားပါပြီ။ ဒါဆို ပြောစရာဖြစ်လာတယ်။ 1 ကနေ 10 အထိ generate လုပ်တာများ အောက်ကလိုလဲရတာပဲဥစ္စာ။

Code:

```
numList = [i for i in range(1,11)]
```

အမေး: ဘာလို့များ function တစ်ခုထပ်ဆောက်ပြီး လုပ်ရတာလဲ?
အဖြေ: အဲဒီလို numList Define လုပ်လိုက်ရင် Variable က RAM မှာ 1 ကနေ 10 အထိသွားသိမ်းပါလိမ့်မယ်။ နံပါတ်နည်းရင် ပြဿနာမရှိပေမယ့် နံပါတ်က အရမ်းကြီးသွားရင် RAM မှာအများကြီးသွားစားမယ်လေ။

အမေး: ဪ ဒါဆို Generator object ကရော RAM မစားဘူးပေါ့?
အဖြေ: မစားတာမဟုတ်ပါဘူး။ စားပါတယ်။ ဒါပေမယ့် အရမ်းနည်းတာပေါ့။ အောက်က Example ကိုကြည့်ပါ။

Code:

```
numList = [i for i in range(1000) if i%2 == 0]
```

ဒီလိုလုပ်ရင် CPU ကတစ်ခါထဲ 1 ကနေ 1000 အထိ တစ်ခုချင်းစီသွားပြီး စုံကိန်းဖြစ်မဖြစ်ကို တန်းပြီး တွက်ချက်ပြီး သိမ်းရပါမယ်။ Generator အားသာချက်က လိုအပ်မှပဲ တွက်ချက်ပြီး ထုတ်ပေးလိုက်တာပဲ။ အဲဒီကုဒ်ကို Generator နဲ့ရေးကြည့်မယ်။

Code:

```
def gen_even_num():
    for i in range(1000):
        if i%2 == 0:
            yield i

num = gen_even_num()
num.next()
num.next()
```

ဒီလိုဆိုရင် သူက num.next() ခေါ်လိုက်မှ function ကို execute လုပ်ပြီး စုံကိန်းကို ပြန် yield လုပ်ပေးတာပါ။ num.next() က python 2.x ဗားရှင်းမှာ ရေးနည်းပါ။ ဗားရှင်း 3 မှာဆိုရင် next(num) ဆိုပြီး လုပ်ရပါတယ်။

နောက်ပြီး generator function နဲ့ သာမန် function မတူတာတစ်ချက်က သာမန် function ဆိုရင် function ကို execute လုပ်ပြီးရင် သူ့အထဲမှာရှိတဲ့ local variable ရှေ့တန်ဖိုးတွေအားလုံးကို မေ့ပစ်လိုက်တယ်။ ဒါပေမယ့် generator function တွေမှာတော့ အဲဒီလိုမဟုတ်ဘူး။ သူက မှတ်ထားသေးတယ်။ အဲဒါကြောင့် ဒီလို next() နဲ့ခေါ်လိုရတာပါ။

အောက်မှာတော့ memory profile လုပ်ထားတာကို ပြပါမယ်။

```
Filename: normal_func.py

Line #      Mem usage      Increment      Line Contents
=====
     4         20.4 MiB         0.0 MiB      @profile
     5         20.4 MiB         0.0 MiB      def my_func():
     6         24.3 MiB         3.8 MiB          num = [i for i in range(100000) if i%2 == 0]

Duration : 4.719994
-----
Filename: generator_func.py

Line #      Mem usage      Increment      Line Contents
=====
    10         20.2 MiB         0.0 MiB      @profile
    11         20.2 MiB         0.0 MiB      def main():
    12         20.2 MiB         0.0 MiB          num = []
    13         23.6 MiB         3.5 MiB          for value in generator_func():
    14         23.6 MiB         0.0 MiB              num.append(value)

Duration : 4.843883
```

Normal function code

Code:

```
from memory_profiler import profile
import time

@profile
```

```
def my_func():
    num = [i for i in range(100000) if i%2 == 0]

if __name__ == "__main__":
    start = time.time()
    my_func()
    end = time.time()
    print"Duration : %f" % (float(end)-float(start))
```

Generator function code

Code:

```
from memory_profiler import profile
import time

def generator_func():
    for i in range(100000):
        if i%2 == 0:
            yield i

@profile
def main():
```

Memory ကွာဟချက်လေးနဲ့ duration ကွာဟချက်လေးက သိပ်တွေ့မကွာပါဘူး။ ဒါပေမယ့် generator က လိုမှသုံးတဲ့နေရာမှာ အသုံးတည့်တယ်။ နောက်ပြီး item ဘယ်လောက်လိုလဲ အတိအကျမသိတဲ့ case မှာဆို လိုသလောက် ထုတ်သုံးယုံပဲ။

Lambda (λ)

Lambda ဆိုတာ Radioactive Decay ကိုတွက်ချက်ရင်သုံးတဲ့ constant ကိုပြောတာမဟုတ်ပါဘူး။ python မှာဆို တစ်ခြား အဓိပ္ပာယ်ရှိပါတယ်။ lambda ဆိုတာ mini-function လေးတွေ ဒါမှမဟုတ် function on-the-fly / on-the-run ပါ။ code statement လေးတွေကလဲ တစ်ကြောင်းထဲရှိသောအခါမျိုးမှာ ကိုယ်ကလဲ နောက်ထပ်သုံးချင်တယ်၊ ဒါပေမယ့် def လဲမရေးချင်ဘူး။ ဒါဆို lambda သုံးလို့ရပါပြီ။

Code:

```
>>> def f(x): return x**2
```

f ဆိုတဲ့ function ကခေါ်လိုက်ရင် ထည့်လိုက်တဲ့ argument ကို power 2 တင်ပြီး ပြန်ထုတ်ပေးမှာပါ။

Code:

```
>>> print f(8)
64
```

ဒါဆို lambda နဲ့ရေးကြည့်ရအောင်။ အရင်ဆုံး lambda မှာ return statement မလိုပါဘူး။ သူက နဂိုထဲက auto return လုပ်ပါတယ်။ ဒါသိရင် စရေးလို့ရပါပြီ။

Code:

```
>>> g = lambda x: x**2
```

code ကိုခွဲခြမ်းကြည့်ရင် g ဆိုတာ variable define လုပ်တာနဲ့ အတော်ကိုတူပါတယ်။ ဒါပေမယ့် lambda ပါနေတဲ့အတွက် သူ့ကို variable လို့သတ်မှတ်လို့မရပါဘူး။ ပြန်ခေါ်မယ်ဆိုရင် function ခေါ်တဲ့အတိုင်းပါပဲ။

Code:

```
>>> print g(8)
64
```

g = lambda x ဆိုတဲ့အထဲက x ကတော့ g(10) လို့ခေါ်ရင် x = 10 ဖြစ်သွားပါတယ်။ သူက function ကို parameter ထည့်ပေးတဲ့အခါသုံးတာပါ။ သုံးတာများသွားရင် သိသွားပါလိမ့်မယ်။
နောက်တစ်ခုအနေနဲ့ expression တစ်ခုပဲထည့်လို့ရတာကို အနောက်ကနေ ; semi-colon ခံပြီး နောက်တစ်ခုသွားထည့်လဲမရပါဘူး။

ကျွန်တော်ရေးထားတဲ့ lambda example function တစ်ခုကိုပြပါမယ်။

Code:

```
>>> sumTwo = lambda num: min(list(num)) + min([i for i in num if i != min(list(num))])
```

lambda num: ဆိုတာ lambda function ကို define လုပ်ပြီး num ကတော့ function argument ကိုသိမ်းထားပေးမယ့် variable နာမည်ပါပဲ။

min(list(num)) ဆိုတာ num ကိုအရင်ဆုံး list datatype အဖြစ်ပြောင်းမယ်။ နောက်ပြီးရင် min() ဆိုတဲ့ built-in function ကိုခေါ်သုံးပြီး အသေးဆုံး ဂဏန်းကို ထုတ်မယ်။
+ ပေါင်းတာ
min([i for i in num if i != min(list(num))]) ဆိုတာကို တစ်ဆင့်ချင်း ရှင်းပြပါမယ်။ i for i in num ဆိုတာ num ထဲမှာပါတဲ့ element တွေအားလုံးကို အသုံးချမယ်လို့ပြောတာ။ နောက်ပြီး if i != min(list(num)) ဆိုတာတော့ တကယ်လို့ i က min(list(num)) အသေးဆုံးကိန်းနဲ့ မတူတာပေါ့။ အရင်ဆုံး num ထဲကဟာကို အကုန်ထုတ်တယ် အသေးဆုံးကိန်းကလွဲရင် နောက်တစ်ဆင့်က min function ကိုပြန်ခေါ်လိုက်တယ်။ ဒုတိယအသေးဆုံးကို ပြန်ခေါ်ထုတ်တာပါ။ နောက်ပြီးတော့ တစ်ခါ ပထမ(အသေးဆုံးကိန်း)နဲ့ ဒုတိယ(အသေးဆုံးကိန်း) ကိုပေါင်းပြီး return ပြန်လုပ်တာပါ။

ပုံမှန်အတိုင်းဆို အရင် sort လုပ်ပြီး ပထမတစ်ခုနဲ့ ဒုတိယတစ်ခုပေါင်းယုံပဲ။ ဒါပေမယ့် lambda ကတစ်ခုပဲလက်ခံတယ်လေ။ နောက်ပြီး num.sort() function က ဘာမှ return မလုပ်ပါဘူး။ သူ့နဂို list ကိုပဲ sort လုပ်လိုက်တာပါ။

Code:

```
>>> sumTwo([5,8,12,19,22])
```

```
13
>>> sumTwo([15,28,4,2,43])
6
```

Lambda

* တကယ်လို့ statement ဒါမှမဟုတ် Expression က ဘာမှ return မလုပ်ရင် lambda မှာသုံးလို့မရပါဘူး။ None လို့ return လုပ်ရင်တောင် သုံးလို့ရပါတယ်။ ဘာ return လုပ်လဲသိချင်ရင် code ကို type() function နဲ့စစ်ကြည့်လို့ရပါတယ်။

Code:

```
>>> num = [3,2,1]
>>> type(num.sort())
<type 'NoneType'>
```

* Assignment statement (example x=10) ဆိုတာမျိုးက ဘာမှ return မလုပ်ပါဘူး။ None တောင် return မလုပ်ပါဘူး။ အဲဒီတော့ lambda မှာသုံးလို့မရပါဘူး။

Code:

```
>>> x = lambda y: y = 10
File "<stdin>", line 1
SyntaxError: can't assign to lambda
```


* Function call တွေကို lambda မှာသုံးလို့ရပါတယ်။ lambda ကနေတစ်ခါ တစ်ခြား function ကိုလှမ်းခေါ်တာပါ။

Code:

```
lambda: a if some_contition() else b
```

* Print က python 3.x version မှာဆိုရင် function call ဖြစ်သွားတာကြောင့် lambda မှာအသုံးပြုလို့ရပါတယ်။

အကျယ်သိချင်ရင် reference ထဲက အောက်ဆုံးဟာကို ဝင်ဖတ်နိုင်ပါတယ်။ မသိတာ မရှင်းတာတွေလဲ မေးမြန်းနိုင်ပါတယ်။

Regards, 
Anubis

Reference

Code:

```
http://www.secnexis.de/olli/Python/lambda_functions.hawk
https://jeffknupp.com/blog/2013/04/07/improve-your-python-yield-and-generators-explained/
http://stackoverflow.com/questions/890128/why-are-python-lambdas-useful
https://pythonconquerstheuniverse.wordpress.com/2011/08
```

/29/lambda_tutorial/

Being able to crack security, doesn't make you a hacker
As being able to hotwire cars, doesn't make you automotive engineer.

☐ The following 1 user says Thank You to **Anubis** for this post:

• **NyanTunAung**

PM Find

Add Thank You

Reply

Quote

Report



0M3G4_4P3X

Senior Member



Posts: 60

Threads:

0

Thanks

Received:

2 in 2

posts

Thanks

Given: 25

Joined:

Jan 2016

Reputatio

n: 0

03-13-2016, 10:56 AM

#2

Python ကိုလေ့လာတုန်းဘဲ သေချာမရသေးဘူး 🤖🤖 ရှင်းပြပေးတာကျေးဇူးပါ bro 🤖



PM Find

Add Thank You

Reply

Quote

Report

« Next Oldest | Next Newest »

Enter Keywords

Search Thread

New Reply

Quick Reply



Quick Reply

Message

Type your reply to this message here.

☐ Disable Smilies

Post Reply

Preview Post

[View a Printable Version](#)

[Subscribe to this thread](#)

Forum Jump:

-- Python

Go

Users browsing this thread: Hacke3erDD

Myanmar Security Forum (MSF) © 2013 - 2016 - All Rights Reserved.

Powered By MyBB, © 2002-2016 MyBB Group. — Theme by FlatInk LLC.
[Contact Us](#) — [Return to Top](#) — [Lite \(Archive\) Mode](#) — [RSS Syndication](#) | [Awards](#)