Python - Ferramentas/Utilitários

A biblioteca padrão vem com vários módulos que podem ser usados como módulos e como utilitários de linha de comando.

O módulo dis

O módulo **dis** é o desmontador do Python. Ele converte códigos de bytes em um formato um pouco mais apropriado para consumo humano.

Exemplo

```
import dis
def sum():
   vara = 10
   varb = 20

   sum = vara + varb
   print ("vara + varb = %d" % sum)

# Call dis function for the function.
dis.dis(sum)
```

Isso produziria o seguinte resultado -

```
 3         0 LOAD_CONST               1 (10)
           2 STORE_FAST               0 (vara)

 4         4 LOAD_CONST               2 (20)
           6 STORE_FAST               1 (varb)

 6         8 LOAD_FAST                0 (vara)
          10 LOAD_FAST                1 (varb)
          12 BINARY_ADD
          14 STORE_FAST               2 (sum)

 7        16 LOAD_GLOBAL              0 (print)
          18 LOAD_CONST               3 ('vara + varb = %d')
          20 LOAD_FAST                2 (sum)
          22 BINARY_MODULO
          24 CALL_FUNCTION            1
          26 POP_TOP
```

```
        28 LOAD_CONST          0 (None)
        30 RETURN_VALUE
```

O Módulo pdb

O módulo pdb é o depurador padrão do Python. É baseado na estrutura do depurador bdb.

Você pode executar o depurador na linha de comando (digite n [ou próximo] para ir para a próxima linha e ajudar a obter uma lista de comandos disponíveis) -

Exemplo

Antes de tentar executar **pdb.py** , defina seu caminho corretamente para o diretório lib do Python. Então, vamos tentar com o exemplo acima sum.py -

```
$pdb.py sum.py
> /test/sum.py(3)<module>()
-> import dis
(Pdb) n
> /test/sum.py(5)<module>()
-> def sum():
(Pdb) n
>/test/sum.py(14)<module>()
-> dis.dis(sum)
(Pdb) n
  6           0 LOAD_CONST               1 (10)
              3 STORE_FAST               0 (vara)

  7           6 LOAD_CONST               2 (20)
              9 STORE_FAST               1 (varb)

  9          12 LOAD_FAST                0 (vara)
             15 LOAD_FAST                1 (varb)
             18 BINARY_ADD
             19 STORE_FAST               2 (sum)

  10         22 LOAD_CONST               3 ('vara + varb = %d')
             25 LOAD_FAST                2 (sum)
             28 BINARY_MODULO
             29 PRINT_ITEM
             30 PRINT_NEWLINE
             31 LOAD_CONST               0 (None)
             34 RETURN_VALUE
--Return--
> /test/sum.py(14)<module>()->None
-v dis.dis(sum)
```

```
(Pdb) n
--Return--
> <string>(1)<module>()->None
(Pdb)
```

O módulo de perfil

O módulo de perfil é o criador de perfil padrão do Python. Você pode executar o criador de perfil na linha de comando -

Exemplo

Vamos tentar traçar o perfil do seguinte programa -

```
vara = 10
varb = 20
sum = vara + varb
print "vara + varb = %d" % sum
```

Agora, tente executar **cProfile.py** sobre este arquivo sum.py da seguinte maneira -

```
$cProfile.py sum.py
vara + varb = 30
   4 function calls in 0.000 CPU seconds
   Ordered by: standard name
ncalls   tottime  percall  cumtime  percall filename:lineno
 1       0.000    0.000    0.000    0.000 <string>:1(<module>)
 1       0.000    0.000    0.000    0.000 sum.py:3(<module>)
 1       0.000    0.000    0.000    0.000 {execfile}
 1       0.000    0.000    0.000    0.000 {method ......}
```

O módulo tabnanny

O módulo tabnanny verifica os arquivos de origem do Python em busca de recuo ambíguo. Se um arquivo mistura tabulações e espaços de uma forma que elimina o recuo, não importa o tamanho da tabulação que você está usando, a babá reclama.

Exemplo

Vamos tentar traçar o perfil do seguinte programa -

```
vara = 10
varb = 20
```

```
sum = vara + varb
print "vara + varb = %d" % sum
```

Se você tentar um arquivo correto com tabnanny.py, ele não reclamará da seguinte forma -

```
$tabnanny.py -v sum.py
'sum.py': Clean bill of health.
```