Sexta of Python

Profiling

ENACOM
Rodrigo Machado Fonseca

16 de julho de 2021

Agenda

Motivação

O que é?

Profiling

Profiling é uma ferramenta que nos permite fazer uma análise dinâmica de uma execução de um algoritmo gerando como resultado analises de tempo, memória, chamadas e de tudo o que o compõe.

Obtendo assim analises detalhadas de tempo, número de chamadas de função, interrupções, falhas de cache, etc.

Pra que serve?

"Por que meu código demora tanto?"

Ele fará uma análise dinâmica do programa que mede, por exemplo, o espaço (memória) ou tempo de um programa , o uso de instruções específicas ou a frequência e duração das chamadas de função.

Com essas informações será possível identificar formas de melhorar o desempenho do código de acordo com o aspecto desejado.

Como é feito?

1. Perfil Determinístico: Ao rodar o código todos os eventos durante a execução são monitorados. É relativamente lento.

2. Perfil Estatístico: É feito a amostragem do estado de execução em intervalos regulares para calcular indicadores. É relativamente menos preciso.

Profiling

Profiling simples no Python

Algorithm 1 Profiling naive

```
from time import time

start = time()
# your script here
end = time()

print(Execute in {end - start} seconds!')
```

Perfil Determinístico

Python vem com dois módulos de Profiling determinísticos: **cProfile** e **profile**.

1. cProfile: é recomendado para a maioria dos usuários; é uma extensão C com sobrecarga, o que o torna adequado para criar Profilings de programas de longa execução. Baseado em lsprof, contribuição de Brett Rosen e Ted Czotter

2. profile: um módulo Python puro cuja interface é imitada por cProfile. Se você está tentando estender Profiling de alguma forma, a tarefa pode ser mais fácil com este módulo. Originalmente desenhado e escrito por Jim Roskind.

   Fonte: https://docs.python.org/3/library/profile.html

cProfile.run

Algorithm 2 cProfile.run

cProfile.run('function(arg)');

```
 ncalls  tottime  percall  cumtime  percall filename:lineno(function)
      4    0.000    0.000    0.000    0.000 <__array_function__ internals>:2(copyto)
   2000    0.003    0.000    0.031    0.000 <__array_function__ internals>:2(prod)
  15/11    0.000    0.000    0.009    0.001 <frozen importlib._bootstrap>:1017(_handle_fromlist)
     20    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:103(release)
     11    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:143(__init__)
     11    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:147(__enter__)
     11    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:151(__exit__)
     20    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:157(_get_module_lock)
     11    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:176(cb)
      9    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:194(_lock_unlock_module)
```

cProfile

1. ncalls: número de chamadas.
2. tottime: para o tempo total gasto na função dada (e excluindo o tempo feito em chamadas para subfunções);
3. percall: é o quociente de tottime dividido por ncalls;
4. cumtime: é o tempo cumulativo gasto nesta e em todas as subfunções (da chamada até a saída);
5. percall: é o quociente de cumtime dividido por chamadas primitivas

cProfile

Para realizar a análise pelo terminal:

Algorithm 3 Rodar script com cProfile

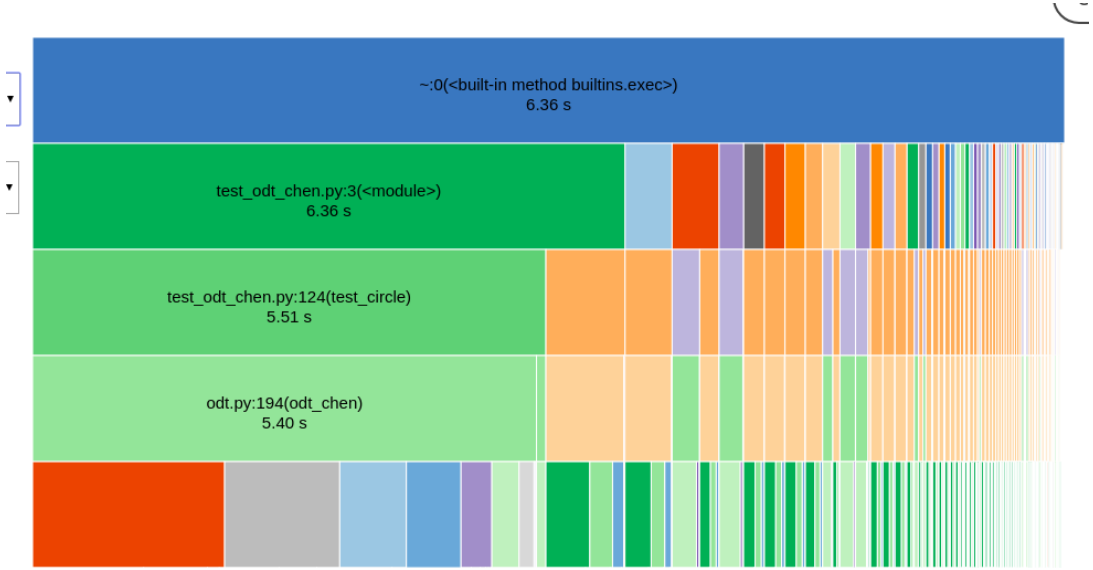
```
python -m cProfile -o temp.dat myscript.py
```

SnakeViz

SnakeViz é um visualizador gráfico baseado em navegador para a saída do módulo cProfile do Python.

Algorithm 4 Instalação do SnakeViz

```
pip install snakeviz
```



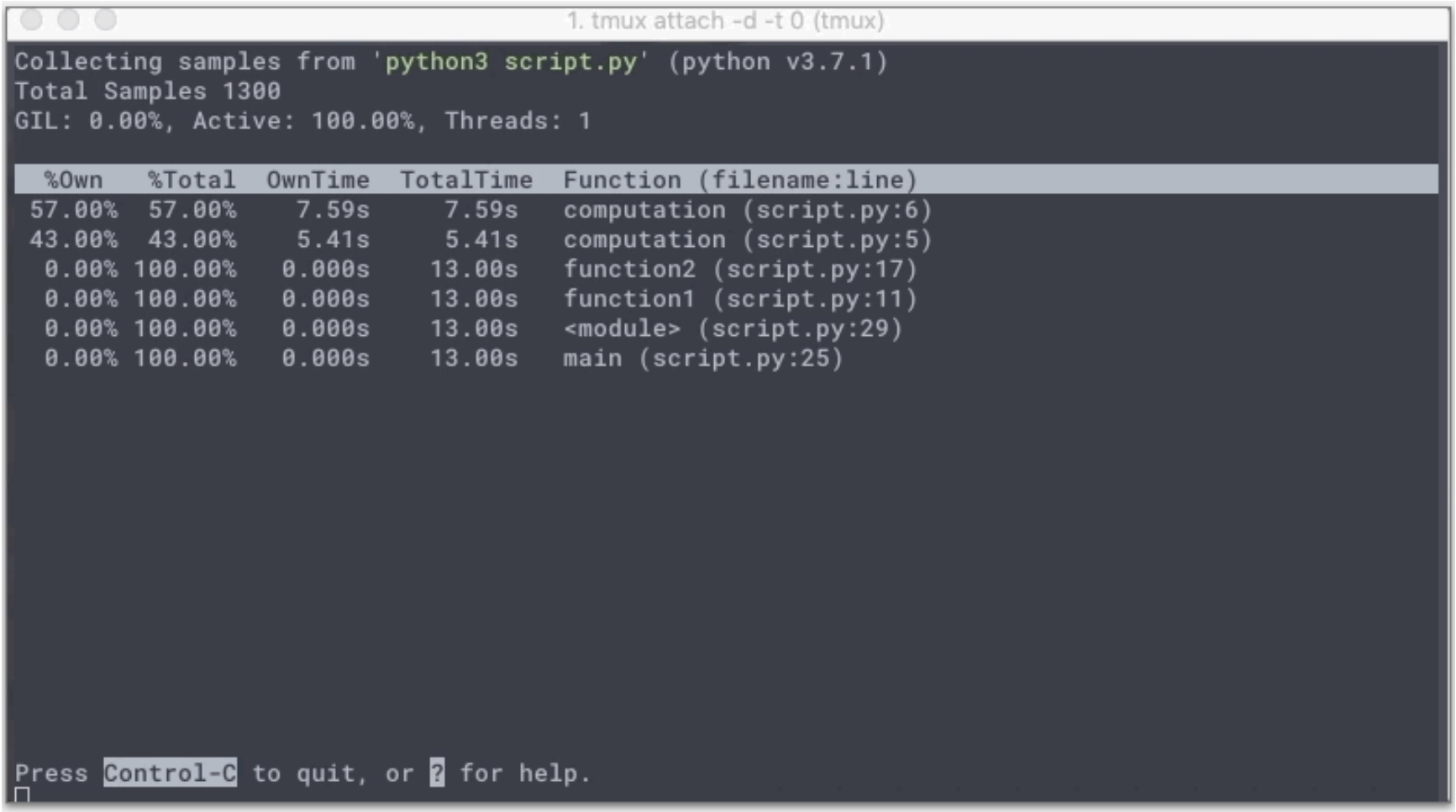
Perfil Estatísticos

Os dois módulos estatísticos mais famosos são: **Py-Spy** e **pyinstrument**.

Py-Spy: é um criador de perfil estatístico que permite visualizar o tempo que cada função consome durante a execução. Não necessita reiniciar o programa ou modificar o código e tem uma baixa sobrecarga.

pyinstrument: É muito semelhante ao cProfile no sentido de que não podemos criar um Profiling em tempo de execução. Mais atraente e gera um relatório HTML.

Py-Spy


1. tmux attach -d -t 0 (tmux)
Collecting samples from 'python3 script.py' (python v3.7.1)
Total Samples 1300
GIL: 0.00%, Active: 100.00%, Threads: 1
%Own %Total OwnTime TotalTime Function (filename:line)
57.00% 57.00% 7.59s 7.59s computation (script.py:6)
43.00% 43.00% 5.41s 5.41s computation (script.py:5)
0.00% 100.00% 0.000s 13.00s function2 (script.py:17)
0.00% 100.00% 0.000s 13.00s function1 (script.py:11)
0.00% 100.00% 0.000s 13.00s <module> (script.py:29)
0.00% 100.00% 0.000s 13.00s main (script.py:25)
Press Control-C to quit, or ? for help.

Pyinstrument

Caso de Estudo

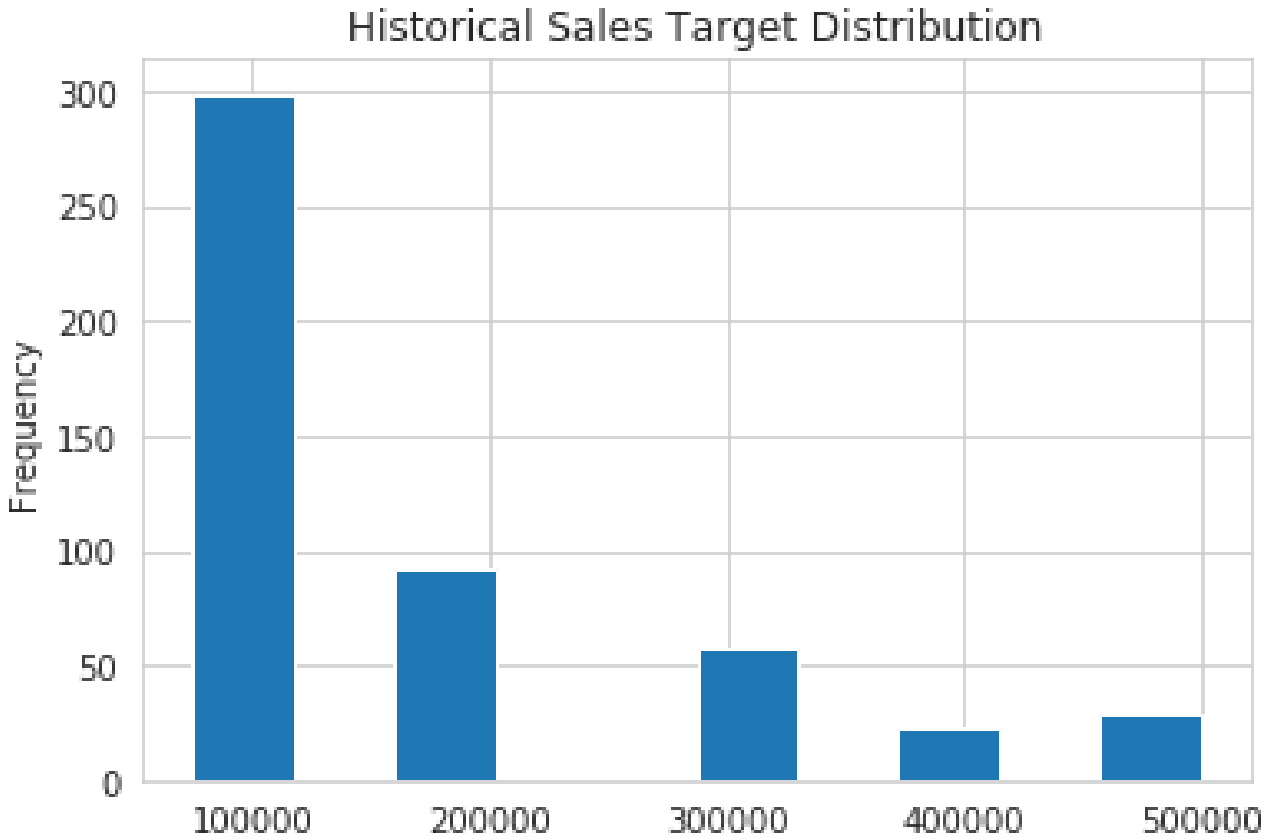
Motivação

- **Método de Monte Carlo**
  Em seu nível mais simples, uma análise de Monte Carlo envolve a execução de muitos cenários com diferentes entradas aleatórias e o resumo da distribuição dos resultados.

- **Problema**
  Uma empresa tem um determinado número de vendedores. Baseado na venda anual de cada vendedor, iremos calcular a venda total da empresa e o montante da taxa de comissão paga pela empresa aos funcionários.

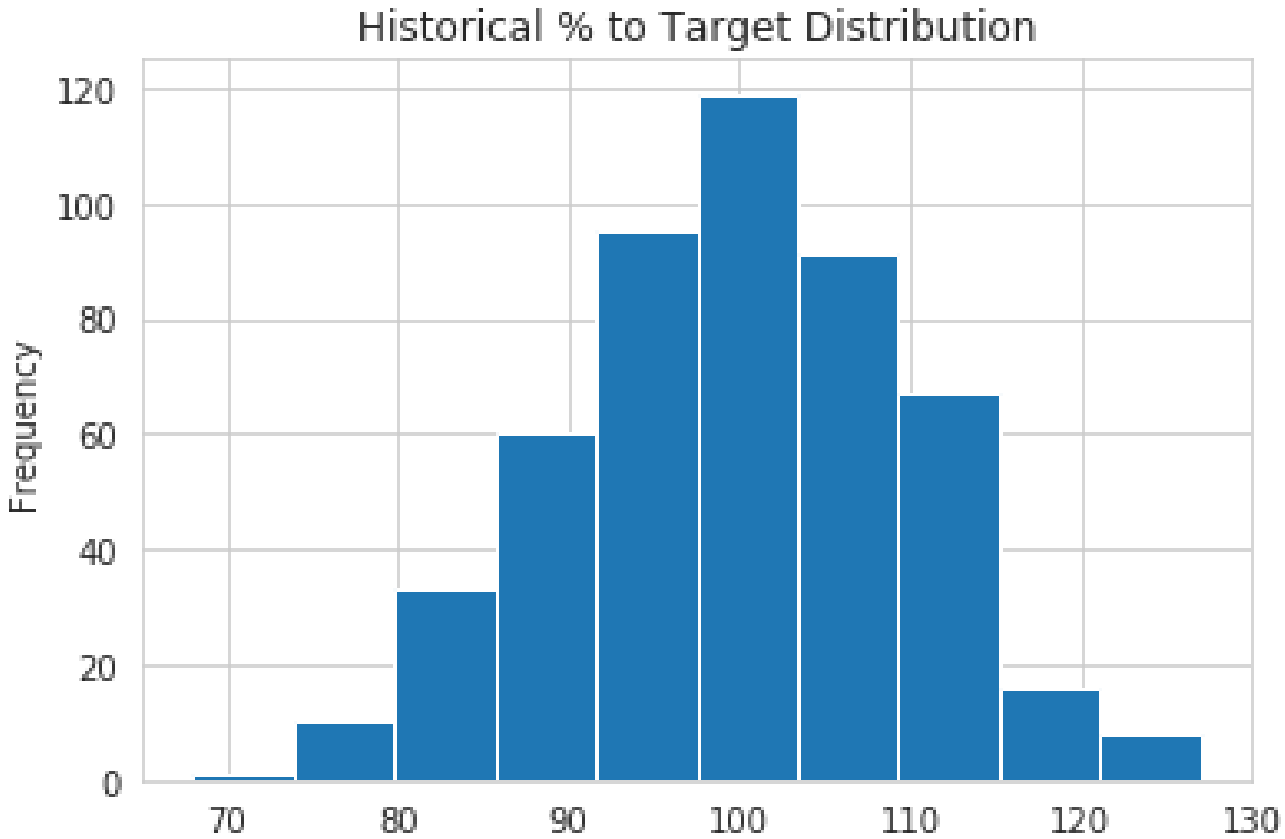
Metas de venda

Vamos gerar valores aleatórios de meta de venda para cada vendedor de acordo com a seguinte distribuição:



Porcentagem de Venda

A porcentagem de venda para cada vendedor será estimada a partir de uma gaussiana de média 1 e desvio padrão 0.1:



Comissão

A taxa de comissão é obtida à partir da porcentagem de venda cumprida por cada vendedor.

Porcentagem de vendas	Taxa de Comissão
X <= 0.9	0.02
0.9 < X <= 1	0.03
1 < X	0.04

Cálculos

$$Venda = Meta\ de\ vendas * Porcentagem\ de\ venda \tag{1}$$

$$Comissao = Venda * Taxa\ de\ Comissão \tag{2}$$

$$Montante\ Vendas = \sum_{i=1}^{N\ de\ vendedores} Venda_i \tag{3}$$

$$Montante\ Comissoes = \sum_{i=1}^{N\ de\ vendedores} Comissao_i \tag{4}$$

Perguntas

Conclusão

Referências Bibliográficas

Referências

- https://docs.python.org/3/library/profile.html
- https://www.machinelearningplus.com/python/cprofile-how-to-profile-your-python-code/
- https://jiffyclub.github.io/snakeviz/
- https://medium.com/fintechexplained/advanced-python-learn-how-to-profile-python-code-10680554
- https://pbpython.com/monte-carlo.html