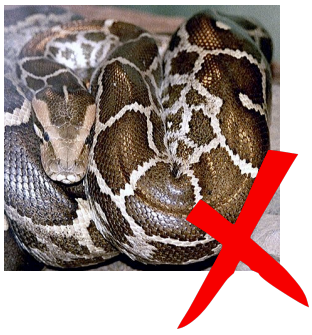
实用python编程 第1讲

认识python

2017-09-11

什么是python?

- 编程语言



谁发明了Python?

- Guido van Rossum

1982 UvA 硕士毕业

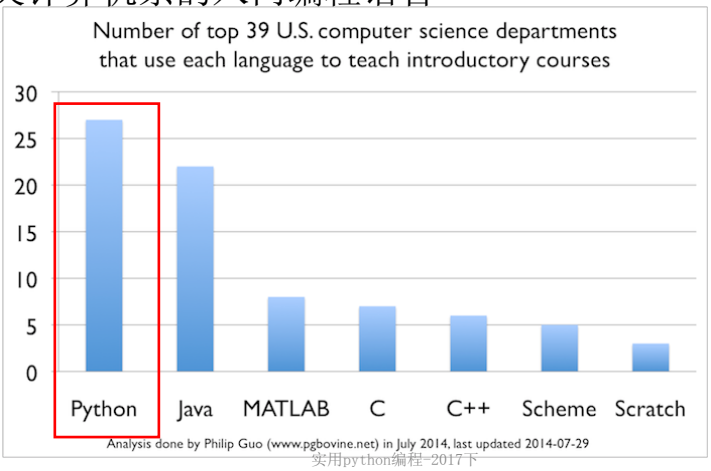
Over six years ago, in December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office ... would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus).

为什么要学python?

- an easy and intuitive language just as powerful as major competitors
- open source, so anyone can contribute to its development
- code that is as understandable as plain English
- suitability for everyday tasks, allowing for short development times

为什么要学python?

- 美国顶尖计算机系的入门编程语言



谁可以学python?

- MUST
  - 中学英语
  - 算术
  - 日常计算机使用
- PLUS
  - 编程（其他语言）

怎么学python?

- 纸上得来终觉浅
- 随用随学

一个简单的文本分析任务

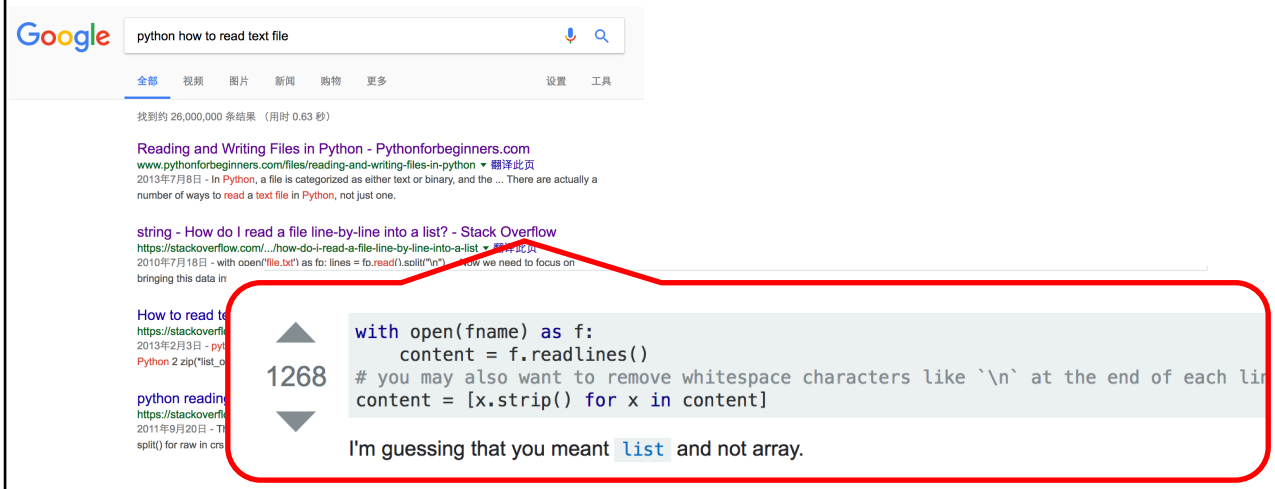
- 文件中有多少个不同的标签?
- 出现频率最高的前100个标签?

```
cedar tree grecia greece sunset sombra shadow
black stork
fall autumn olympus e1 nature landscape 14-54 favorite
colbert flag gloves cascade 220 wool christmas red atl
northfreedom mid-continentrailwaymuseum train snow snc
mountrobson bc britishcolumbia mountain snow top peak
sx-70 polaroid landcamera instantfilm timezero pronto
lca belgium lomography coke machine opened vandalised
processed xpro film 35mm
wilson chia saints for christ christian cca saint secc
spider egg sack white halloween
martinmere wildfowlandwetlandstrust wildfowl ice noven
anasacuta
snow stone post barbed wire
```

问题分析

- 读取数据
  - 需要从文件中读取标签
- 分析数据
  - 统计每个标签出现了多少次
  - 对标签根据其频率降序排列
- 展示结果
  - 在屏幕中打印
  - 保存到文件中
  - 绘制图表

如何用python读取文本文件的数据？



两行代码搞定！

```
datafile = 'flickr-tags.txt'
lines = open(datafile).readlines()
```

基础概念：
+ 字符串（string）：'flickr-tags.txt' 用单引号或双引号括起来
+ 变量（variable）：datafile / lines 存储数据，它的值可以改变，变量名命名规则
+ 函数（function）：open / readlines 需要重复做的事情用函数实现
+ 函数输入值: datafile 打开哪个文件，有的函数不需要输入值，如readlines
+ 函数返回值: open / readlines 分别返回什么？
…

https://github.com/ruc-python/2017fall/blob/master/data/flickr-tags.txt

变量

小数（float）、列表（list）、词典（dict）、…

```
a = 2.93
b = []
c = range(10)
d = {'first':'bill', 'last':'clinton'}

for x in [a, b, c, d]:
    print x
```

```
2.93
[]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
{'last': 'clinton', 'first': 'bill'}
```

for /if / def 等语句以冒号结尾

The colon is required primarily to enhance readability (one of the results of the experimental ABC language). Consider this:

```
if a == b
    print(a)
```

versus

```
if a == b:
    print(a)
```

Notice how the second one is slightly easier to read. Notice further how a colon sets off the example in this FAQ answer; it's a standard usage in English.

Another minor reason is that the colon makes it easier for editors with syntax highlighting; they can look for colons to decide when indentation needs to be increased instead of having to do a more elaborate parsing of the program text.

https://docs.python.org/3/faq/design.html#why-are-colons-required-for-the-if-while-def-class-statements

函数

- 注意代码的缩进

```
def example(str):
    print("The string is %s." %(str))
    return 1

flag = example("an apple")
print flag
```

```
The string is an apple.
1
```

条件语句 if / if else / if elif

```
def is_us_president(name):
    if name == 'hillary':
        return False
    if name == 'trump':
        return True
    # we are not done yet
```


问题分析

- 读取数据
  - 需要从文件中读取标签

```
datafile = 'flickr-tags.txt'
lines = open(datafile).readlines()
```

- 分析数据
  - 统计每个标签出现了多少次
  - 对标签根据其频率降序排列
- 展示结果
  - 在屏幕中打印
  - 保存到文件中
  - 绘制图表

统计标签出现频率（5行代码）

```
datafile = 'flickr-tags.txt'
lines = open(datafile).readlines()
tag2count = {}

for line in lines:
    tokens = line.strip().split()

    for tag in tokens:
        tag2count[tag] = tag2count.get(tag,0)+1

print len(tag2count)
```

```
42851
```


统计标签出现频率并排序（6行代码）

```
datafile = 'flickr-tags.txt'
lines = open(datafile).readlines()
tag2count = {}

for line in lines:
    tokens = line.strip().split()

    for tag in tokens:
        tag2count[tag] = tag2count.get(tag,0)+1

#print len(tag2count)
sorted_list = sorted(tag2count.iteritems(), key=lambda v:v[1], reverse=True)
print sorted_list[:5]
print sorted_list[-5:]
```

```
[('car', 583), ('nature', 535), ('canon', 512), ('street', 503), ('water', 490)]
[('thebeatles', 1), ('hessle', 1), ('telemetrique', 1), ('mosaics', 1), ('jawbone', 1)]
```

更多排序的例子

```
a = [5, 2, 3, 1, 4]
a.sort() 与 sorted(a) 有何不同?

students = [
('john', 'A', 15),
('jane', 'B', 10),
('dave', 'B', 12), ]
```

https://wiki.python.org/moin/HowTo/Sorting

统计标签出现频率并排序

- 用Counter实现

```
from collections import Counter

datafile = 'flickr-tags.txt'
lines = open(datafile).readlines()

cnt = Counter()

for line in lines:
    tokens = line.strip().split()
    for tag in tokens:
        cnt[tag] = cnt[tag] + 1

sorted_list = cnt.most_common(5)

print sorted_list
```

```
[('car', 583), ('nature', 535), ('canon', 512), ('street', 503), ('water', 490)]
```

课堂练习

- 实现一个比较两个整数大小的函数
  - 输入值：x，y
  - 输出：如果x大于或等于y，返回1，否则返回0

```
def compare_int(x, y):
    …
```

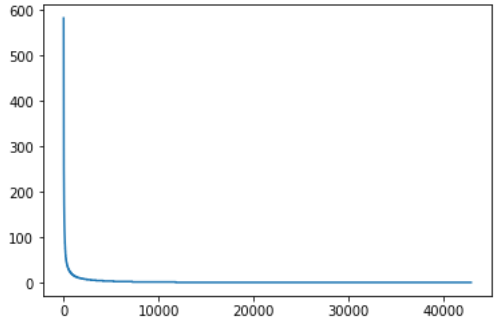

问题分析

- 读取数据
  - 需要从文件中读取标签
- 分析数据
  - 统计每个标签出现了多少次
  - 对标签根据其频率降序排列
- 展示结果
  - 在屏幕中打印
  - 保存到文件中
  - 绘制图表

```
datafile = 'flickr-tags.txt'
lines = open(datafile).readlines()

cnt = Counter()
for line in lines:
    tokens = line.strip().split()
    for tag in tokens:
        cnt[tag] = cnt[tag] + 1

sorted_list = cnt.most_common()
```

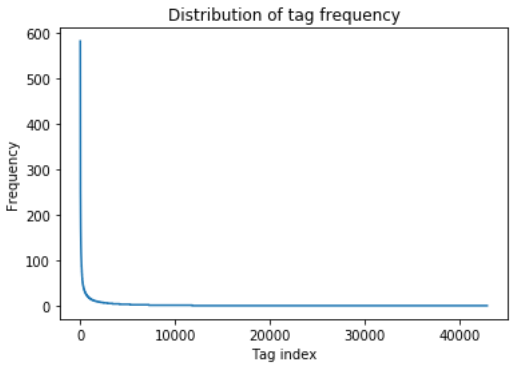
matplotlib绘图

```
%matplotlib inline
import matplotlib.pyplot as plt

n = len(sorted_list)
x = range(n)
y = [c for w,c in sorted_list]
plt.plot(x, y)
```



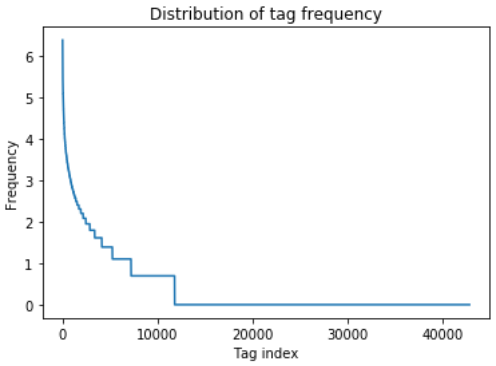
matplotlib绘图

```
%matplotlib inline
import matplotlib.pyplot as plt

n = len(sorted_list)
x = range(n)
y = [c for w,c in sorted_list]
plt.plot(x, y)

plt.xlabel('Tag index')
plt.ylabel('Frequency')
plt.title('Distribution of tag frequency')
```



matplotlib绘图

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

n = len(sorted_list)
x = range(n)
y = [c for w,c in sorted_list]
plt.plot(x, np.log(y))

plt.xlabel('Tag index')
plt.ylabel('Frequency')
plt.title('Distribution of tag frequency')
```



matplotlib绘图

```
ax = plt.gca()


word_index = 20
ax.annotate(sorted_list[word_index][0],
xy=(word_index, y1[word_index]),
xytext=(10000, y1[word_index]),
arrowprops=dict(facecolor='green',
shrink=0.01, width=0.5),)
```