Python入门和应用分享

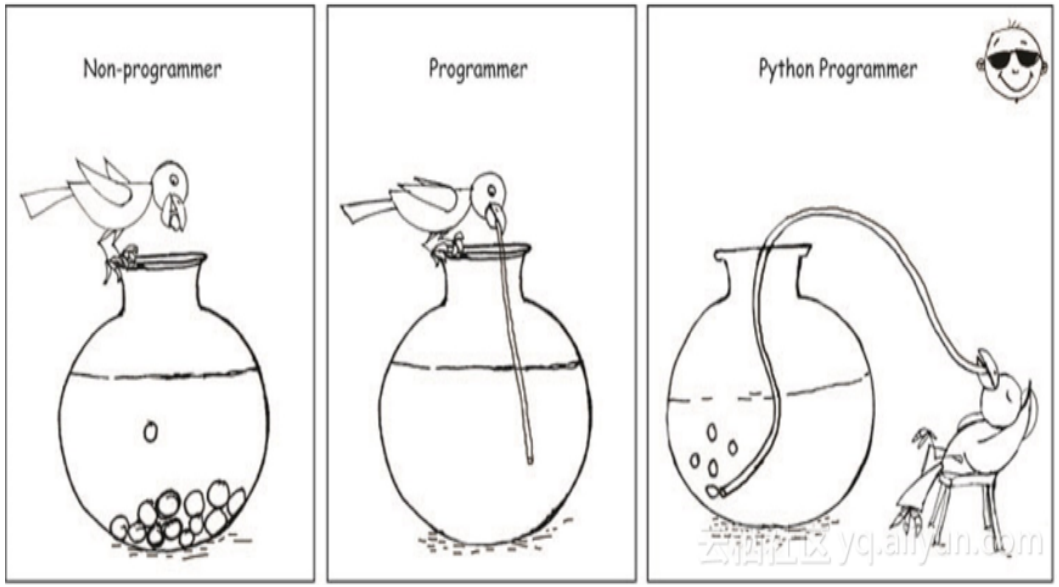
有品 · 数据中台 · 冯申杰

2019/3/29

OUTLINE

- **1. Python是什么**
  - **优缺点**
  - **流行度**
- 2. Hello World
  - 数据类型、运算符、输入输出、函数
  - 控制流：顺序、条件、循环、异常处理
  - 模块、标准库、第三方库
  - 一张图入门
- 3. 两个栗子
  - 爬虫：房天下二手房数据
  - ETL：数据处理
- 4.更多领域
  - 数据科学：Numpy、Pandas
  - web
- 5. 更多资料

什么是Python？



- 简洁高效、新手友好的编程语言
- 一种脚本语言、"解释型"语言、面向对象的高级语言。

Python is powerful... and fast;

plays well with others;

runs everywhere;
is friendly & easy to learn;

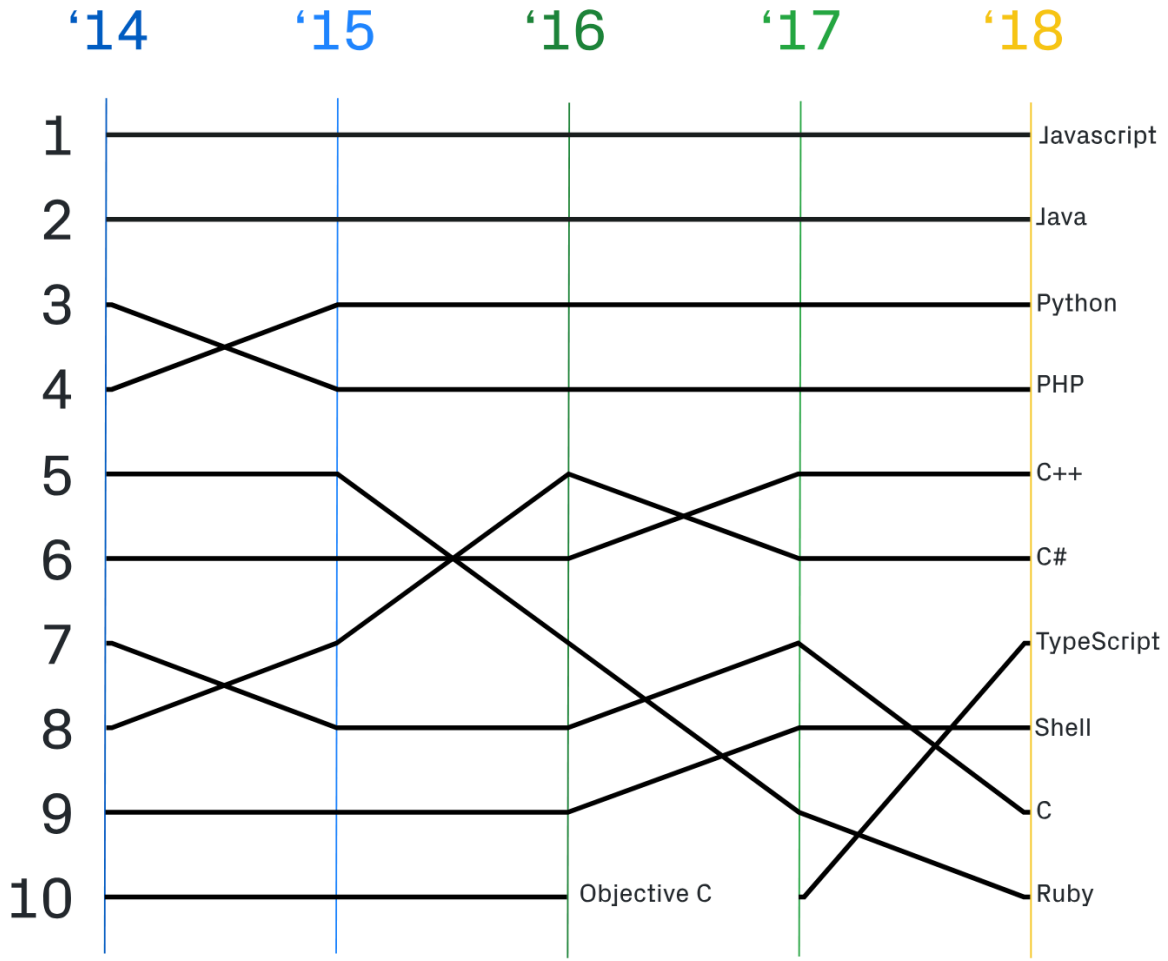
is Open.

- 代码缩进决定了代码的逻辑关系
- 功能丰富的自带的库以及种类和数量繁多且强大的第三方库
- 面向对象、过程式、函数式
- 好用的交互环境

作为解释性语言

- 优点：不需要编译即可运行
- 缺点：性能相对不强

国外的Youtube，Instagram，Pinterest，Reddit，Dropbox，Disqus， Quora等知名应用一开始都是基于Python构建，国内的豆瓣，知乎，今日头条，果壳，饿了么，搜狐等也是Python应用的典型。



Top languages over time

https://octoverse.github.com/projects

OUTLINE

- 1. Python是什么
  - 优缺点
  - 流行度
- **2. Hello World**
  - **数据类型、运算符、输入输出、函数**
  - **控制流：顺序、条件、循环、异常处理**
  - **模块、标准库、第三方库**
  - **一张图入门**
- 3. 两个栗子
  - 爬虫：房天下二手房数据
  - ETL：数据处理
- 4.更多领域
  - 数据科学：Numpy、Pandas
  - web
- 5. 更多资料

首先按照官网指引搭建好环境，然后就可以愉快的运行Python程序了。

```
$ python  # 在交互环境运行
Python 3.6.2 (v3.6.2:5fd33b5926, Jul 16 2017, 20:11:06)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print('hello world')
hello world

$ cat hello.py
print('hello world')
$ python hello.py  # 运行py脚本
hello world
```

一张图入门

http://coffeeghost.net

Quick Python Script Explanation for Programmers

Load other code modules.

Module name. This refers to "os.py"

```
import os

def main():
    print 'Hello world!'

    print "This is Alice's greeting."
    print 'This is Bob\'s greeting.'

    foo(5, 10)

    print '=' * 10
    print 'Current working directory is ' + os.getcwd()

    counter = 0
    counter += 1

    food = ['apples', 'oranges', 'cats']

    for i in food:
        print 'I like to eat ' + i

    print 'Count to ten:'
    for i in range(10):
        print i


def foo(param1, secondParam):
    res = param1 + secondParam

    print '%s plus %s is equal to %s' % (param1, secondParam, res)

    if res < 50:
        print 'foo'

    elif (res >= 50) and ((param1 == 42) or (secondParam == 24)):
        print 'bar'

    else:
        print 'moo'

    return res # This is a one-line comment.
    '''A multi-
line string, but can also be a multi-line comment.'''


if __name__ == '__main__':
    main()
```

The name "main" is just a convention, not a requirement. See the very bottom of this script.

Newline automatically added to print statements. Also, there are no semicolons at the end of the line.

No curly braces! Instead, the block starts when you add 4 spaces of indentation. (or any other amount, but 4 spaces is the standard)

I prefer single-quotes, but either is fine. Either way, you don't have to escape the other kind of quote inside the string.

Function call.

String replication. Evaluates to '=========='

String concatenation.

Call a function in the os module.

Variables MUST be instantiated first.

Lists can contain different data types in the same list, including other lists.

One-line block. When the indentation goes back down, the block has ended.

For loop. "i" takes on each value in the list "food" in order.

The range() function returns a list like [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Don't forget the colon at the end!

Function definition. Don't forget the colon at the end.

String interpolation works basically the same way as it does in C.

The comparison operators are the same as C.

End of indentation, not a } brace, signifies the end of the block.

Boolean operators are words, not && and ||.

Colons come after def, for, while, if, elif, and else statements.

Comments.

Multi-line strings don't affect block indendation, though, only the indentation at the START of the statement or expression.

We put a call to main() at the bottom so that each def statement is executed by the time we call main(). This script's __name__ variable has the value '__main__' only when the script is run, not imported. With this check, the main() function won't run if another script imports this script.

数据类型、运算符、输入输出、函数

- **数据类型**
  - 不可变数据：**Number**（数字）、**String**（字符串）、**Tuple**（元组）；
  - 可变数据：**List**（列表）、**Dictionary**（字典）、**Set**（集合）。
- **运算符**：算术、比较、赋值、逻辑等运算符
- **函数**：将程序拆分为多个部分，使得代码的结构更加合理。
- **值类型和引用类型**

```
>>> a = [1, 2, 3] # a now references the list [1, 2, 3]
>>> b = a # b now references what a references
>>> a.append(4) # this changes the list a references
>>> print(b) # if we print what b references,
>>> [1, 2, 3, 4] # SURPRISE! It has changed…
```

控制流：顺序、条件、循环、异常处理

- if-else
- for/while
- try-except

模块、标准库、第三方库

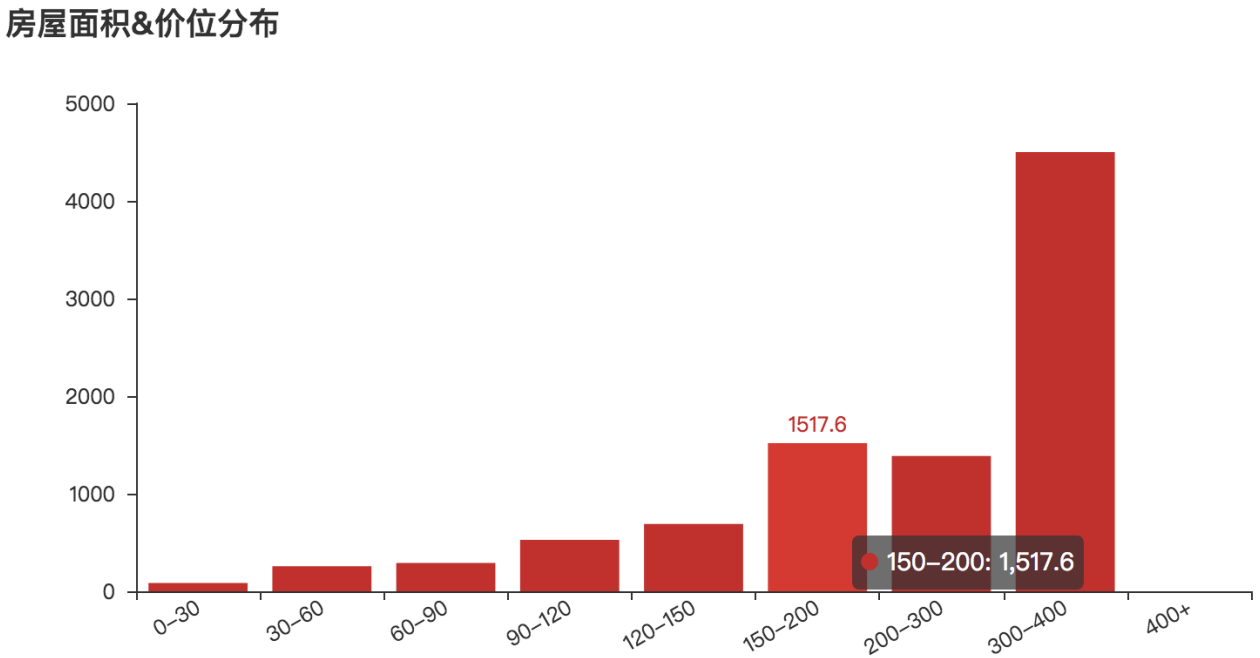
- web 框架：Django/Flask/Tornado
- ORM: sqlalchemy, Peewee
- 表单验证：WTForms
- 数据处理和分析：Numpy, Pandas, Matplotlib
- 异步：celery, asyncio, tornado
- 并发：gevent, threading, concurrent.futures
- 部署：uwsgi, gunicorn
- html 处理: lxml, beautifulsoup
- 爬虫：requests, Scrapy
- 单元测试：unittest, nose, pytest
- 图片处理：pillow
- python2/3 兼容：six, 2to3
- 代码检测：autopep8, pylint, flake8, mypy(python3)
- 调试：Ipython, Ipdb, pdbpp

OUTLINE

- 1. Python是什么
  - 优缺点
  - 流行度
- 2. Hello World
  - 数据类型、运算符、输入输出、函数
  - 控制流：顺序、条件、循环、异常处理
  - 模块、标准库、第三方库
  - 一张图入门
- **3. 两个栗子**
  - **爬虫：房天下二手房数据**
  - **ETL：数据处理**
- 4.更多领域
  - 数据科学：Numpy、Pandas
  - web
- 5. 更多资料

爬虫：房天下二手房数据

```
.
├── area_price_plot.html
├── fsjutils
|   ├── __init__.py
|   └── conf.py
├── main.py
├── readme.md
├── view.py
└── wcplot.jpg
```

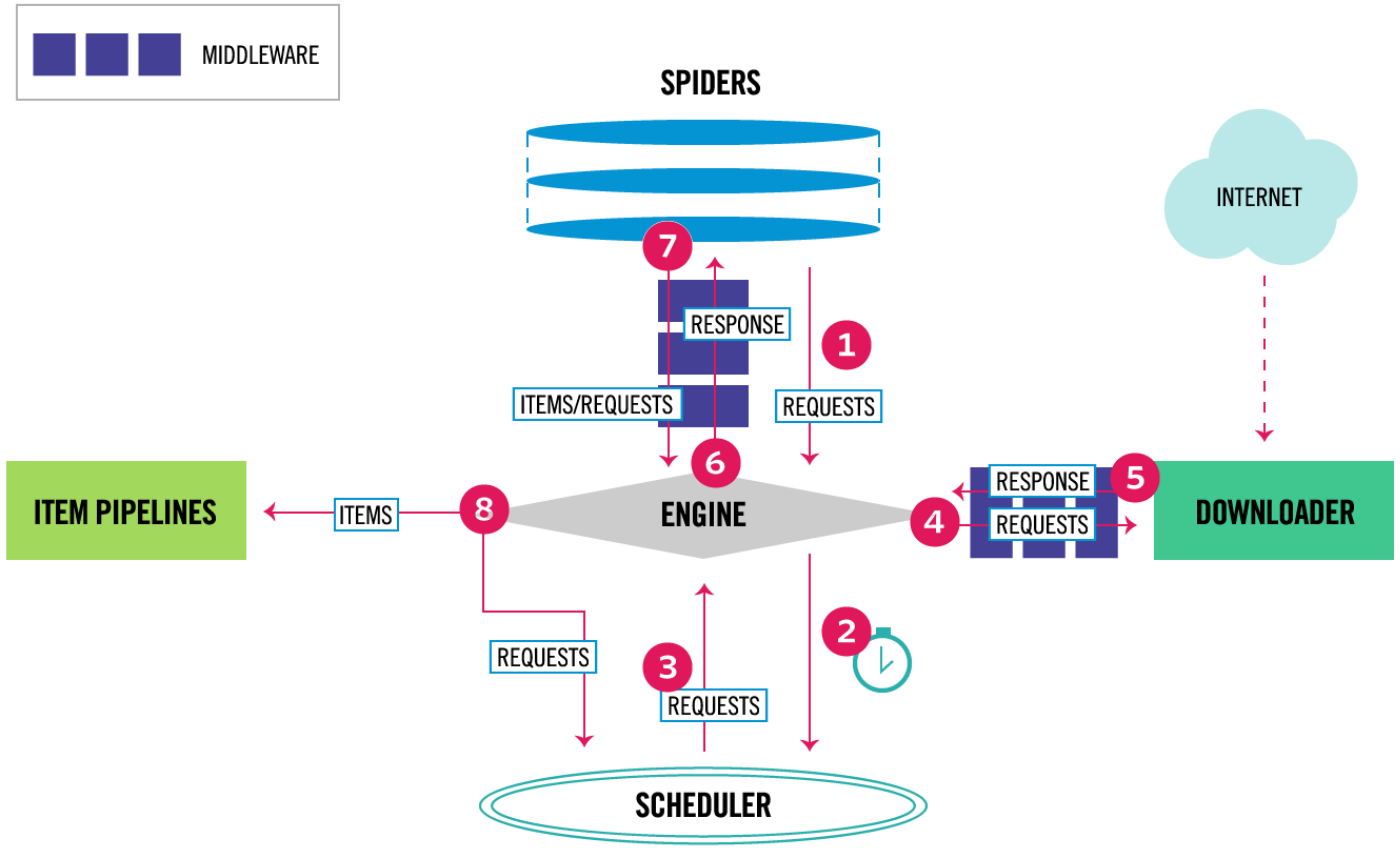


http://v9.git.n.xiaomi.com/fengshenjie/fangSpider

爬虫：房天下二手房数据

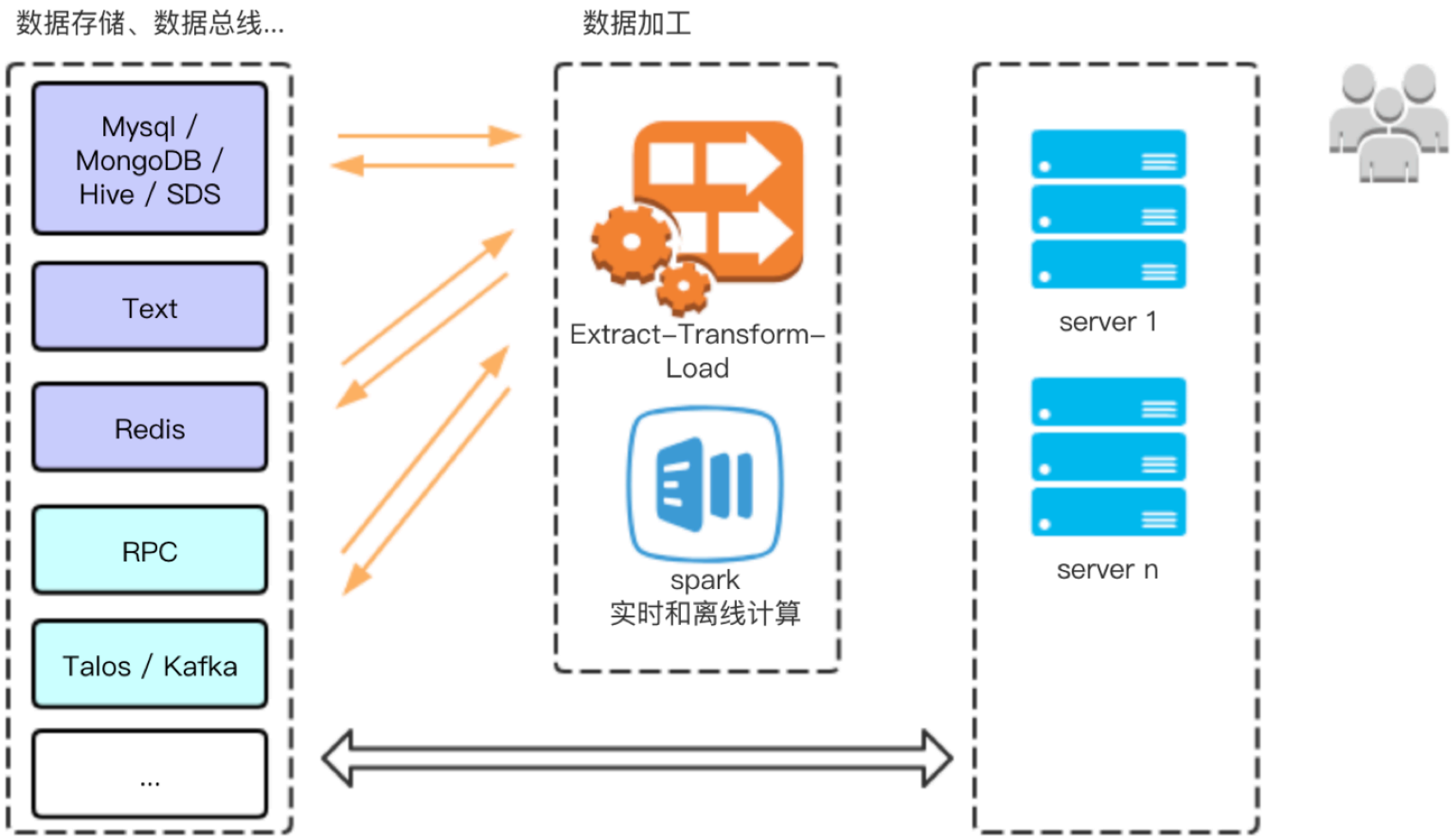
生产环境爬虫复杂得多，需要处理包括路由、存储、分布式计算等很多问题。

比如scrapy爬虫框架



ETL：数据处理

Python在数据处理流程中的位置



ETL：数据处理

minimum-template.py

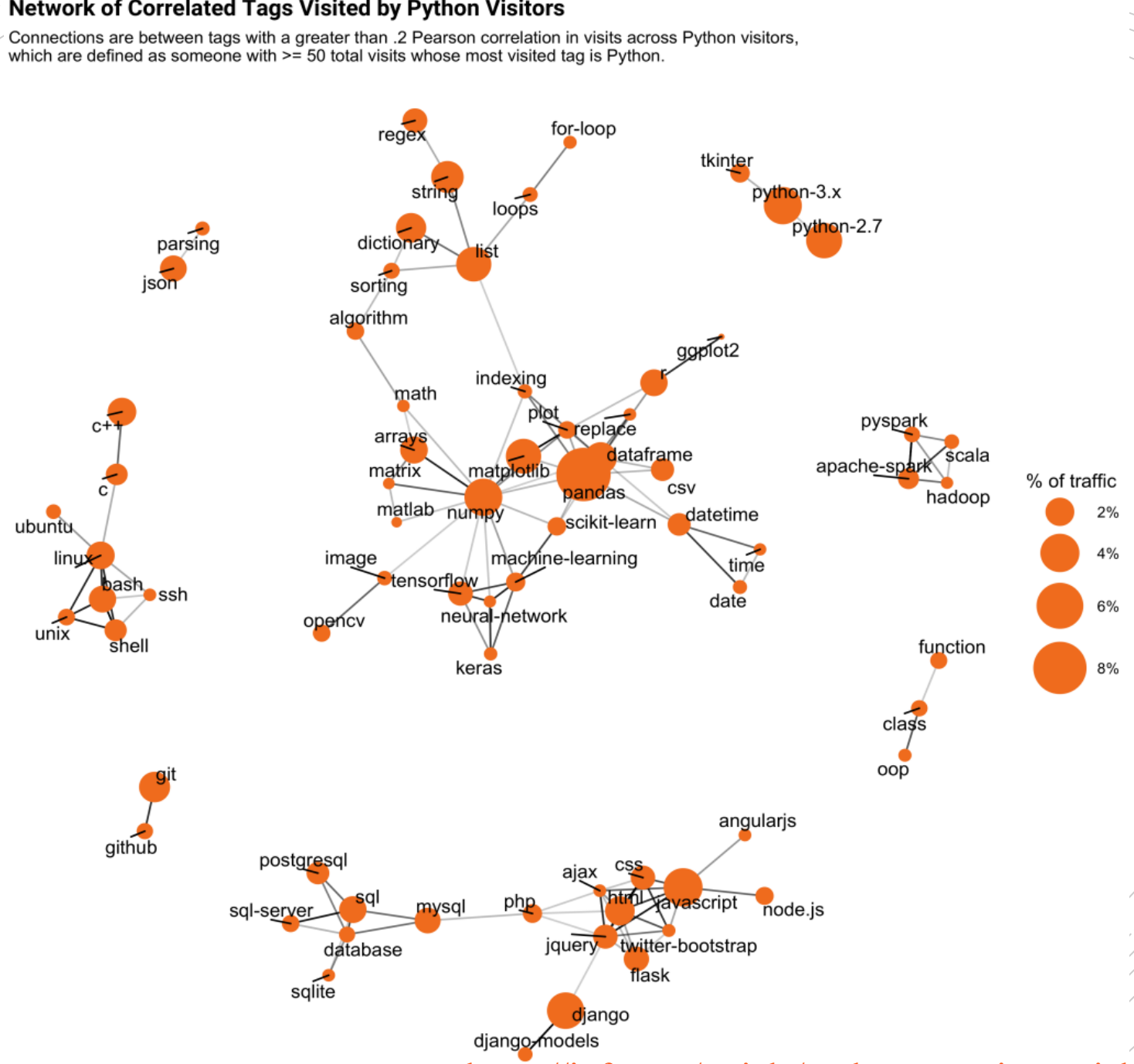
```
# encoding: utf-8
from utils.DBEngines import engineName
from utils.base import getLogger, errHander
import getopt, os, sys

targetEngine = engineName('engineShopstat')
PROJ_NAME = os.path.basename(__file__)
log = getLogger(PROJ_NAME, logPath="/home/work/log/dataProcess/{}/".format(__file__))

@errHander(PROJ_NAME)
def main():
    opts, args = getopt.getopt(sys.argv[1:], "b:e:h", ['db='])
    for op, value in opts:
        pass
    pass

if __name__ == '__main__':
    main()
```

OUTLINE

- 1. Python是什么
  - 优缺点
  - 流行度
- 2. Hello World
  - 数据类型、运算符、输入输出、函数
  - 控制流：顺序、条件、循环、异常处理
  - 模块、标准库、第三方库
  - 一张图入门
- 3. 两个栗子
  - 爬虫：房天下二手房数据
  - ETL：数据处理
- **4.更多领域**
  - **数据科学：Numpy、Pandas**
  - **web**
- 5. 更多资料

数据科学：Numpy、Pandas

Python可能是通用编程语言中最灵活的技术。

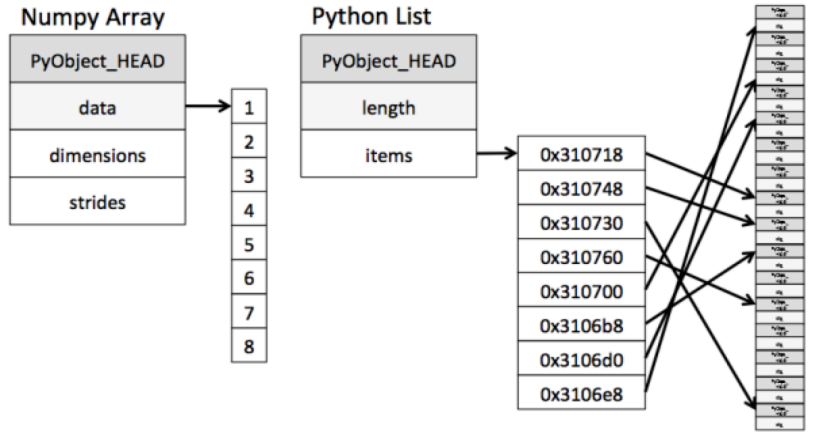


https://infoq.cn/article/python-growing-quickly

数据科学：Numpy、Pandas

```
$ cat cars.csv
,cars_per_cap,country,drives_right
US,809,United States,True
AUS,731,Australia,False
JAP,588,Japan,False
RU,200,Russia,True

In [1]: import pandas as pd

In [2]: cars = pd.read_csv('cars.csv', index_col = 0)

In [3]: cars
Out[3]:
     cars_per_cap        country  drives_right
US            809  United States          True
AUS           731      Australia         False
JAP           588          Japan         False
RU            200         Russia          True
```

```
In [6]: # Create 2 new lists height and weight
   ...: height = [1.87,  1.87, 1.82, 1.91, 1.90, 1.85]
   ...: weight = [81.65, 97.52, 95.25, 92.98, 86.18, 88.45]
   ...:
   ...: import numpy as np
   ...: np_height = np.array(height) # Create 2 numpy arrays from height and weight
   ...: np_weight = np.array(weight)
   ...: print(type(np_height))
   ...:
<class 'numpy.ndarray'>

In [7]: bmi = np_weight / np_height ** 2 # Calculate bmi
   ...: print(bmi)
   ...:
[23.34925219 27.88755755 28.75558507 25.48723993 23.87257618 25.84368152]

In [8]: bmi[bmi > 23] # Print only those observations above 23
   ...:
Out[8]:
array([23.34925219, 27.88755755, 28.75558507, 25.48723993, 23.87257618,
       25.84368152])
```



Web框架

比如 Django、Flask、Tornado 等.

- 知乎到 2010 年 12 月份上线时，工程师是四个。初期的架构选型使用的是Tornado框架。
  - 简单轻量，学习成本低
  - 社区支持，有 **FriendFeed** 的成熟案例
  - 实时推送 **Feed** 和通知需要跟浏览器端建立一个长连接

运行效率较低。机房机柜空间已经不足；
语言过于灵活，多人协作和项目维护成本较高。

从 0 到 100——知乎架构变迁史, 2014
知乎社区核心业务 Golang 化实践, 2018

OUTLINE

- 1. Python是什么
  - 优缺点
  - 流行度
- 2. Hello World
  - 数据类型、运算符、输入输出、函数
  - 控制流：顺序、条件、循环、异常处理
  - 模块、标准库、第三方库
  - 一张图入门
- 3. 两个栗子
  - 爬虫：房天下二手房数据
  - ETL：数据处理
- 4.更多领域
  - 数据科学：Numpy、Pandas
  - web
- **5. 更多资料**

Python代码的执行由Python 虚拟机(也叫解释器主循环)来控制。对Python 虚拟机的访问由全局解释器锁（GIL）来控制。

PVM是Python的运行引擎，是迭代运行字节码指令的一个大循环，一个接一个的完成操作。

main.py

```
def foo():
    x = 1
    def bar(y):
        z = y + 2  # <--- (3) ... and the interpreter is here.
        return z
    return bar(x)  # <--- (2) ... which is returning a call to bar ...
foo()              # <--- (1) We're in the middle of a call to foo ...
```

call stack / block stack / data stack :

```
c   ---------------------------
a  | bar Frame                 | -> block stack: []
l  |     (newest)              | -> data stack: [1, 2]
l   ---------------------------
   | foo Frame                 | -> block stack: []
s  |                           | -> data stack: [<Function foo.<locals>.bar at
0x10d389680>, 1]
t   ---------------------------
a  | main (module) Frame       | -> block stack: []
c  |       (oldest)            | -> data stack: [<Function foo at 0x10d3540e0>]
k   ---------------------------
```

谢谢

更多资料

- https://cn.udacity.com/course/intro-to-python-nanodegree--nd302-cn-basic
- https://www.learnpython.org/
- https://morvanzhou.github.io/tutorials/data-manipulation/np-pd/
- 人们对Python在企业级开发中的10大误解
- Python解释器简介（5）：深入主循环
- 书单：https://python-web-guide.readthedocs.io/zh/latest/base/basics.html