1.Python的硬币两面

1.1 Why Python?

——*Python*为数据而生，但不仅为数据而生

Python很多时候与R/Matlab有着近似的功能：同数据打交道。你可以用它清洗、整理数据，画出各式各样的图形，跑一跑机器学习，三者都能挺好地完成任务。在这种情况下，Python与R/Matlab的差别不是很大，我们会使用各种预定义好的函数，像流水线一样依次对数据进行读取、处理、计算、画图。流水线般的处理过程中，我们关注的核心往往在于每一步该选取何种趁手的工具（选取哪个函数来完成我想要的操作），这也是大部分数据相关工作的使用场景。

但Python不止于此。实际上，强大的数据处理能力让Python受到人们青睐，而广泛的用途、强大的工程适用性才是它热门的真正原因。与R/Matlab不同，Python首先是被看作一门编程语言，而不是一门为数据而生的编程语言来创造，因此拥有完整的面向对象体系，对函数、类有着更加成熟完善的定义和优化。因此，Python允许我们不再用流水线化的思维进行处理问题，而是更加强调模块化、面向对象的思想。这使得Python能够将能力范围扩展到更远的地方，除了数据处理相关应用，Python在网络爬虫、游戏编程、Web服务开发领域都有着重要地位。例如，YouTube的源码就由Python编写。

（参考阅读1 哪些知名项目由Python开发？https://www.zhihu.com/question/19555512）

——*Python*有极佳的第三方模块支持

用Python的开发者常常会称自己的项目为“胶水代码”，称自己为“调包侠”。抛开谦逊和自嘲不谈，这种现象一方面反映了很多Python项目的代码本质就是调用各种第三方模块，完成大任务里各个特定的小任务，然后“缝合”在一起；但另一方面也凸显了Python第三方模块生态的强大，各式各样功能强大、易于使用的模块只要几行代码就可调遣、整合，没有这些强大的模块，“调包侠”们也就无法轻松完成各类看起来极其困难的任务。

经过多年发展，Python的第三方模块呈现极其繁荣的态势，类别横跨数据分析、科学运算、科学绘图、游戏开发、网络爬虫、深度学习、CV、NLP……其中的典型代表有Numpy、Pandas、PyTorch、Scrapy等，这些第三方模块提供了极其方便的功能：一行命令搞定网络视频缓存、一两个函数完成OCR识别、几十行代码即可搭建和训练一个深度学习网络。并且有着各自的官方网站、详细的说明文档和答疑解惑、反馈Bug的社区。除此之外，还有很多稀奇古怪、解决某些特定功能的模块等待大家发现和探索，只需在GitHub上搜索，几乎都能找到满意的项目。

我们生活在一个编程的好时代。

（参考阅读2 一行命令缓存视频 https://github.com/soimort/you-get ）

——*Python*的语法简介优雅，易于上手

因为Python的语法是如此的简单，花时间阐释它有多简单不如直接告诉大家语法规则，因此跳过。

（参考阅读3 Zen of Python https://www.cnblogs.com/charlesblc/p/6108115.html）

1.2 Why not Python?

Python当然不是万能法宝，它也存在一些天生的缺陷。主要是作为动态语言（也称脚本语言）展现出的通病，这个通病就是“过于灵活”。

——动态语言带来的缺陷

Python是一门动态语言。它在运行时，可以改变自己的结构。具体反映为变量的类型可以随时变化，新的函数可以被引进、已有的函数可以被删除，类的属性也可以随意更新。而静态语言在编译间会进行检查，禁止以上情况发生。

Python的动态性带来了以下几个缺点：

1.内存占用大。由于Python程序在运行时会随时改变变量、函数、类，无法像C等语言一样预先估计好内存开销，而是需要动态地进行内存分配。这导致在商业环境下，Python很多时候无法承担底层任务，因为会造成大量的内存开销。

2.运行效率低。动态语言在某些计算任务上会比静态语言耗时十倍甚至一百倍。但这个差距可以用C/C++扩展来解决，如Numpy的底层逻辑就是使用C进行编写的，效率远高于纯Python代码。

3.Bug难以发现。由于变量、函数、类可以自由地变化，出现Bug时想要弄清问题来源会变得尤为困难。很多时候，变量并没有变成我们想要的样子，但由于Python的动态性，代码依然可以正常运行，然后输出离谱的结果，Debug时往往会感觉无从下手。

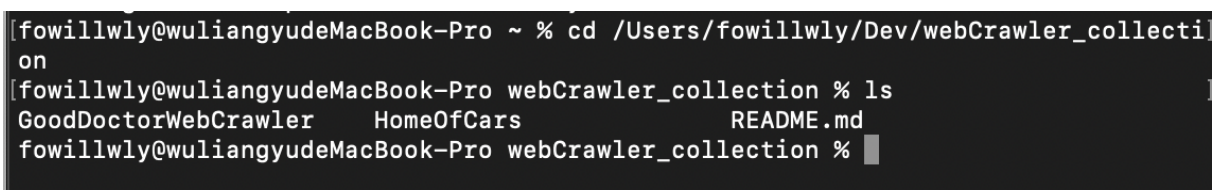
（参考阅读4 动态语言与静态语言 https://www.zhihu.com/question/316509027）

2.Python的开发环境配置

2.1 命令行的基本知识

与Python打交道，很多时候我们需要通过命令行工具进行操作，因此有必要对其先做介绍。

命令行工具有很多，常用选项中，Windows下主要为CMD、Windows Terminal，Mac下为Terminal。各自的启动方法也比较多样。一般来说，CMD可以通过按住Windows徽标键+R，在弹出的对话框中输入 *cmd* 回车启动；Terminal可以在Mac的控制台中找到“终端”图标打开。以下是一张常见的命令行界面（Mac的Terminal）：



我们可以在命令行中输入各种指令，让系统执行对应的操作。注意⚠️，每一条命令都会在命令行当前的工作路径下执行，因此我们常常需要切换当前的工作路径，在不同文件夹间进行跳转。

事实上，这一操作方式正类似于多年前DOS系统的操作方式，也是目前类UNIX系统（Linux、MacOS）的基本操作方式。

在上图中，我们首先执行了cd命令（change directory，切换工作目录），格式为 *cd* + 空格+路径（绝对路径和相对路径皆可），切换到了*/Users/fowillwly/Dev/webCrawler_collection*目录。可以在第三行中看到，%前显示，执行*cd*命令后，我们处于了*webCrawler_collection*目录下。我们继续执行 *ls* 命令，作用为列出当前工作目录下的文件名。可以看到，命令行给出了正确的答案。

Windows命令行的其他常用命令如下表：

CD	切换目录
MKDIR	创建一个目录
DEL	删除
HELP	显示帮助
RD	删除目录
EXIT	退出

Windows系统中的命令行逻辑与例子中基本相同，但存在一些语法上的差异，如路径中的斜杠应该为反斜杠“\”而不是正斜杠“/”；又如路径中若存在空格，应该用双引号括起。

一个基本操作示例：https://blog.csdn.net/ruanwei_0317/article/details/81459698

具体规则参见：https://blog.csdn.net/qq_23994787/article/details/79498299?utm_medium=distribute.pc_relevant.none-task-blog-title-3&spm=1001.2101.3001.4242

2.2 Python的安装

——Step1:安装

Windows用户参照：https://www.runoob.com/python/python-install.html

Mac用户较为简单，参照：http://c.biancheng.net/view/4164.html

Ps：Windows环境下，跟随Python安装指引程序，不断点击“下一步”时，注意“添加到环境变量”或“Add to path”类似的选项，请勾选！同时，也请记住自己将Python安装到了哪个具体路径中，后续步骤会用到。

——Step2:环境变量配置

如果在Step1中勾选了“添加到环境变量”或“Add to path”选项，可以跳过这一步。

如果安装后，用Step3介绍的方法检测时，提示“python不是内部或外部命令，也不是可运行的程序或批处理文件”时建议着重检查这一步。

为什么要配置环境变量？我们已经介绍过，在命令行中，每一条命令都是在当前工作目录下运行的，我们输入的 *python* 命令本质上是让系统在当前工作目录下寻找名为 *python* 的程序并启动。然而，在启动一个新的命令行窗口时，其默认的工作目录往往是系统预设的用户文件夹，其中并没有叫做 *python* 的程序可以启动，这时尝试启动就会报错。环境变量的作用就是告诉系统，当我没有在当前工作目录下找到符合要求的文件时，还可以去哪些其他目录下寻找。因此，将Python的安装路径添加到系统的环境变量，就可以在任何工作路径下顺利启动Python。

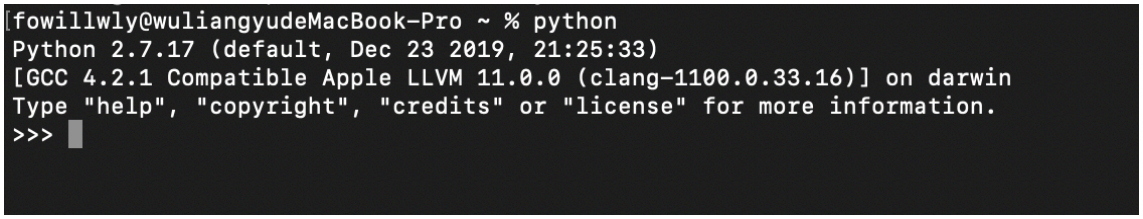
环境变量的配置方法见：https://www.cnblogs.com/dangeal/p/5455005.html

注意⚠️，Step1中的安装路径就是要添加的环境变量。

——Step3:检测Python是否安装成功

我们往往会在命令行工具中启动/检测Python的状况。

在命令行中敲入 *python* 命令，回车，如果电脑上Python环境正确，即可看到相关信息。



如图所示，启动的Python版本为2.7.17（从学术需求出发，Python2已经不建议使用，可以使用 *python3* 命令启动Python3版本），最下一行为输入提示符，表示我们可以在当前窗口中输入Python命令。

在输入提示符（>>>）处输入 *quit()* 命令，回车，退出Python。

如果无法正常显示，说明机器内未成功配置环境，需要安装配置。

2.3 常用开发工具推荐

解决了环境问题，我们就可以开始愉快地写Python代码了。

Python代码的载体为.py文件，而.py文件理论上是可以被记事本这样的软件编写和储存的（虽然可能有些小问题）。因此，写Python代码很简单，你可以在任何一个你喜欢的文本编辑器里编写Python代码（显然，Word除外）。但专业的人做专业的事，有些文本编辑器提供了针对Python的文本高亮、自动缩进等功能，可以更好地帮助各位进行开发。

在这里我们推荐三种不同的编辑器，分别面向不同的需求。在编写类似R/Matlab的数据相关任务时，Jupyter Notebook以其代码块的输入输出方式、富有交互性的界面、对绘图模块的良好支持成为首选；在编写爬虫等需要反复调试的任务时，VSCode的断点、变量检查等功能可以提供良好的支持；在开发由多个.py文件构成的大项目，如网站搭建时，Sublime Text是一款轻量化、界面优美、纯粹的文本编辑器。

Jupyter Notebook的配置方法见 https://www.jianshu.com/p/91365f343585

VSCode：https://www.jianshu.com/p/cbf500c22154

Sublime Text：https://www.cnblogs.com/022414ls/p/12488860.html

建议一定要先配置好Jupyter Notebook，以后的材料都会以Jupyter提供给大家。VSCode作为知名的编辑器，可定制的插件、各种功能过于复杂，可能造成困难。Sublime Text同理。

对于PyCharm、Spyder等更加笨重、商业化的IDE，暂时不做推荐。

2.4 Python代码的运行方法

找到了合适地方来写Python代码，下一步就是运行。这里以Hello World为例，介绍运行Python代码的三种方法。

——Jupyter Notebook

如图，Jupyter中，代码是以代码块的形式，分为In和Out来进行输入和呈现的。我们在In文本框中输入一行或多行代码，保持输入光标在文本框中，点击“运行”（或Shift+Enter），即可得到一个Out，展现了对应的In中代码的运行结果。因此，Jupyter中的代码是可以分段运行的，且不受代码块之间的顺序影响，非常适合数据分析这样的任务。

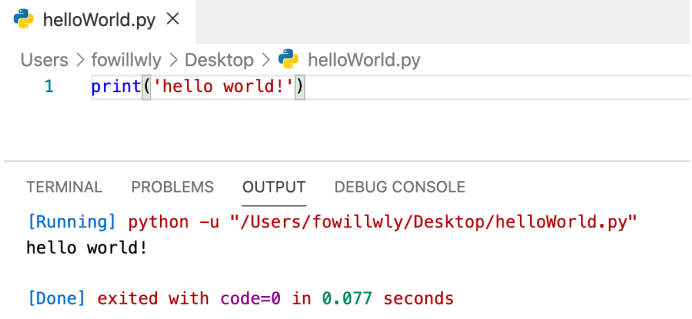
——VSCode/Sublime Text

以VSCode为例，在VSCode中，我们有多种方式运行代码。

首先确保自己已经在VSCode中成功安装了Python Extension，这样VSCode才能认出Python代码，并提供对应的文本高亮等功能。

新建一个文件，另存为helloWorld.py，VSCode会提醒你选择Python Interpreter（解释器，可以让用户自由选择某个特定版本的Python以满足不同需求，这里我们选择Python3）、安装pylint（一个VSCode的Python语法检查插件），顺利完成后就可以写代码了。

第一种方法（最常使用）：用Debug模式运行代码，点击绿色的三角图标，或按F5运行，结果如下：



第二种方法：使用VSCode的内置终端运行，右键点击代码编写窗体，选择”Run Python File in Terminal”。

第三种方法（不常用）：光标移动到某一行代码，按Shift+Enter，单独执行该行代码。

——在命令行中执行

对于部分大型项目，有时习惯在命令行中执行Python文件。

编写并存储好helloWorld.py文件，打开命令行工具，使用cd命令切换到helloWorld.py的父文件夹。执行 *python3 helloWorld.py* 命令（或者 *python helloWorld.py* ），运行整个.py文件中的代码。

2.5 Python的包管理工具——pip

前文提到，Python强大的原因是拥有众多第三方模块的支持，要安装各种第三方模块，需要我们有统一的包管理工具，常用pip实现。

如果正常安装了Python2.7.9+或Python3.4+版本，系统会自带pip工具，其中Python2对应pip，Python3对应pip3，我们一般使用后者。

打开命令行，输入 *pip3 --version* 命令检查pip版本。

安装某个包，输入 *pip3 install xxx* 命令。

升级某个包，输入 *pip3 install -- upgrade xxx* 命令。

卸载某个包，输入 *pip3 uninstall xxx* 命令。

```
fowillwly@wuliangyudeMacBook-Pro ~ % pip3 --version
pip 19.0.3 from /usr/local/lib/python3.7/site-packages/pip (python 3.7)
fowillwly@wuliangyudeMacBook-Pro ~ % pip3 install pandas
Looking in indexes: https://mirrors.aliyun.com/pypi/simple
Requirement already satisfied: pandas in /usr/local/lib/python3.7/site-packages
(0.24.2)
Requirement already satisfied: numpy>=1.12.0 in /usr/local/lib/python3.7/site-pa
ckages (from pandas) (1.16.2)
Requirement already satisfied: python-dateutil>=2.5.0 in /usr/local/lib/python3.
7/site-packages (from pandas) (2.8.0)
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.7/site-pack
ages (from pandas) (2019.1)
Requirement already satisfied: six>=1.5 in ./Library/Python/3.7/lib/python/site-
packages (from python-dateutil>=2.5.0->pandas) (1.12.0)
fowillwly@wuliangyudeMacBook-Pro ~ % pip3 install --upgrade pandas
Looking in indexes: https://mirrors.aliyun.com/pypi/simple
Collecting pandas
  Downloading https://mirrors.aliyun.com/pypi/packages/4b/11/af80c1f40bd17af2594
5ad5f27d57e4514db53b8370d2dc54ff3d23c35c4/pandas-1.1.2-cp37-cp37m-macosx_10_9_x8
6_64.whl (10.4MB)
    100% |████████████████████████████████| 10.4MB 1.6MB/s
Requirement already satisfied, skipping upgrade: pytz>=2017.2 in /usr/local/lib/
python3.7/site-packages (from pandas) (2019.1)
```

详细的pip命令参照：https://www.runoob.com/w3cnote/python-pip-install-usage.html

此外，由于很多包的安装服务器在国外，下载时可能会遭遇速度慢等问题，我们可以使用镜像源解决这一问题，国内出名的有清华源、阿里源等，方法见：https://blog.csdn.net/weixin_40240670/article/details/80616834。

3.Python的基本语法

3.1 Python的数据类型

参考“附件1-Python数据类型.ipynb”（在Jupyter Notebook中打开），对于尚未配置好Jupyter Notebook的同学，我们也提供了html版本。

3.2 Python的选择结构与循环结构

同上，参考附件2。