Machine
Learning

Python科学计算生态

田宇伟

2019年2月

自我介绍

姓名 **田宇伟**

办公室 [redacted]

QQ [redacted]

Email [redacted]

课程介绍

Python科学计算生态

专业课程 / 必修课

3.5学分 / **64**学时

考试	作业	考勤
60%	30%	10%

参考书目

[1] **Python语言及其应用**，（美）Bill Lubanovic 著；丁嘉瑞、梁杰、禹常隆译，人民邮电出版社，2016，ISBN：978-7-115-40709-2

[2] **Python基础教程**，（挪）Magnus Lie Hetland 著；袁国忠译，人民邮电出版社，第**3**版，2018，ISBN：978-7-115-47488-9

[3] **Python科学计算**，张若愚，清华大学出版社，第**2**版，2016，ISBN：978-7-302-42658-5

同步课

Machine
Learning

2017级人工智能方向
Python科学计算生态

/confirm

确认加入课程

QQ群

姓名+年级+班级

申请加入

第一章

Python语言初识

TIOBE Index for February 2019

Feb 2019	Feb 2018	Change	Programming Language	Ratings	Change
1	1		Java	15.876%	+0.89%
2	2		C	12.424%	+0.57%
3	**4**	▲	**Python**	**7.574%**	**+2.41%**
4	3	▼	C++	7.444%	+1.72%
5	6	▲	Visual Basic .NET	7.095%	+3.02%
6	8	▲	JavaScript	2.848%	-0.32%
7	5	▼	C#	2.846%	-1.61%
8	7	▼	PHP	2.271%	-1.15%
9	11	▲	SQL	1.900%	-0.46%
10	20	⇈	Objective-C	1.447%	+0.32%

Programming language Python is the language of 2018 in the TIOBE index(most increase in ratings in one year).

Source: www.tiobe.com

什么是Python

n. 巨蟒；大蟒

计算机编程语言

名字由来

Python语言的创始人，**吉多·范罗苏姆（Guido van Rossum）**是自七十年代风靡全球的英国六人喜剧团体巨蟒剧团（Monty Python）的忠实粉丝，因此其给自己新创造的计算机语言起名Python。

为什么选择Python

- 简单
- 免费、开源
- 高级语言
- 可移植性
- 交互性
- 解释性
- 面向对象
- 可扩展性
- 可嵌入性
- 丰富的库

Python缺点

- 性能相对不是特别好，运行效率差，速度慢
- Python的多线程由于其解析器GIL（Global Interpreter Lock，全局解释器锁）存在无法并行运行
- 代码缩进决定代码的逻辑关系

为什么是Python3

“Python 2.x is legacy, Python 3.x is the present and future of the language.”[note]

“Python 2.7 will reach the end of its life on January 1st, 2020.”

note: from https://wiki.python.org/moin/Python2orPython3

Python开发环境（操作系统）

Ubuntu **16.04/18.04** LTS

- Linux 发行版中被用作个人桌面最多的系统，很多公司选择使用Ubuntu Server作为生产环境的操作系统

- 发布于2016年4月 / 2018年4月，**LTS是Long Term Support**（长期支持）的缩写，官方会提供长达 5 年的技术支持(包括常规更新/Bug 修复/安全升级)。

为什么选择Linux?

- Python在Linux下运行速度会比Windows的快，特别是对于数据分析任务

- 在Linux下搭建Python环境相对容易一些，很多Linux发行版自带了Python程序

- 在Linux下更容易解决第三方库的依赖问题

安装Python3

Ubuntu 16.04 LTS **Python 3.5.2**
Ubuntu 18.04 LTS **Python 3.6.7**

检查python3版本：

```
$ python3 --version
Python 3.6.7
```

运行Python

启动Python，在其 **REPL**[注] 中直接输入相应的命令

```
$ python3
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

可以立刻查看命令运行结果，方便进行一些快速实验验证

注: **REPL** —— **Read-Eval-Print-Loop** 读取-执行-输出-循环

运行Python

- 将完整的代码写成 **.py** 脚本，比如 `hello.py` ，然后通过 `python hello.py` 执行

```
# hello.py '#' 开头的行会被注释掉
print("Hello World!")
```

Python包管理

安装第三方包的方法：

1. 通过Python社区开发的 **pip** 等工具
2. 使用系统本身自带的包管理器（ **yum** ， **apt-get** 等）
3. 通过源码安装（ `python setup.py install` ）

第三方包主要分布在 **The Python Package Index**（ https://pypi.python.org/pypi ）官方仓库（简称 **PYPI** ）以及**Github**、**Gitlab**、**Bitbucket**等代码托管服务上。

Python包管理

pip 是一个用来安装和管理Python包的工具

已经内置到Python 2.7.9和Python 3.4及其以上的版本里

常用命令

`install`	安装包
`uninstall`	卸载已安装的包
`list`	列出已安装的包
`show`	显示已安装包的信息
`search`	在PyPi上搜索包

IPython

为什么不用Python自带交互式解释器

- 不能在退出时保存历史记录以备未来查询
- 不支持Tab自动补全
- 不能快速获得模块/函数/类的信息，如参数、文档、原始代码等
- 不方便在交互环境下执行Shell命令

安装IPython: `$ pip3 install ipython --user`

```
$ ipython --version
7.3.0
```

IPython

- **获得对象信息：**输入你想查看的对象，然后加上一个或者两个？，就能获得多种对象信息。比如“**exit?**”
- **Magic函数：**IPython有很多Magic函数，分为两种类型。一种是**Line magics**，单行函数，需要使用 **%** 开头；另一种是**Cell Magics**，多行函数或者希望执行其他语言的代码，需要使用 **%%** 开头。
- **调用系统Shell命令。**只需在命令前加**！**即可。
- **Tab自动补全。**
- **历史记录。**

Jupyter Notebook

一个基于Web的、可以使用**IPython**绝大多数功能的记事本应用。它允许将交互式的代码、图片在Web上展示出来，支持不同语言的语法高亮、缩进、Tab自动补全、在线编辑和保存等功能，广泛用于如下场景：

- 项目说明文档
- PPT演示，尤其是交互的数据分析相关的PPT演示
- 个人技术笔记
- 技术博客

Jupyter Notebook

安装

```
$ pip3 install jupyter
```

使用

```
$ jupyter notebook
```

Python开发工具

编辑器

- Visual Studio Code（开源，免费）
- Sublime Text
- Vim（开源，免费）
- Emacs（开源，免费）

集成开发环境（IDE）

- PyCharm（有免费版）

Python之禅

```
In [1]: import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
… …
```

《Python之禅》 Tim Peters

优美胜于丑陋
明了胜于隐晦
简洁胜于复杂
复杂胜于混乱
扁平胜于嵌套
宽松胜于紧凑
可读性很重要
即便是特例，也不可违背这些规则
虽然现实往往不那么完美
但是不应该放过任何异常
除非你确定需要如此
如果存在多种可能，不要猜测
肯定有一种——通常也是唯一一种——最佳的解决方案
虽然这并不容易，因为你不是Python之父
动手比不动手要好
但不假思索就动手还不如不做
如果你的方案很难懂，那肯定不是一个好方案
如果你的方案很好懂，那肯定是一个好方案
命名空间非常有用，我们应当多加利用