



首都师范大学

為學為師 求實求新

# 高级程序设计 ---Python与深度学习 1. 课程介绍

李冰

副研究员

交叉科学研究院



# 课程内容

- 课程介绍
- Python介绍
- Python 开发环境
  - Python安装与使用
  - Python包管理
  - Python IDE
- Python运行

# 课程介绍

- 课程编号：0701052024，xxjs014
- 学分：4分
- 总学时：54学时
- 授课老师：
  - 李冰，[bing.li@cnu.edu.cn](mailto:bing.li@cnu.edu.cn)
  - 计算机方向、深度学习模型轻量化设计

欢迎感兴趣的同学联系！

# 通过这么课，你将学到什么

- **Python语言基础**

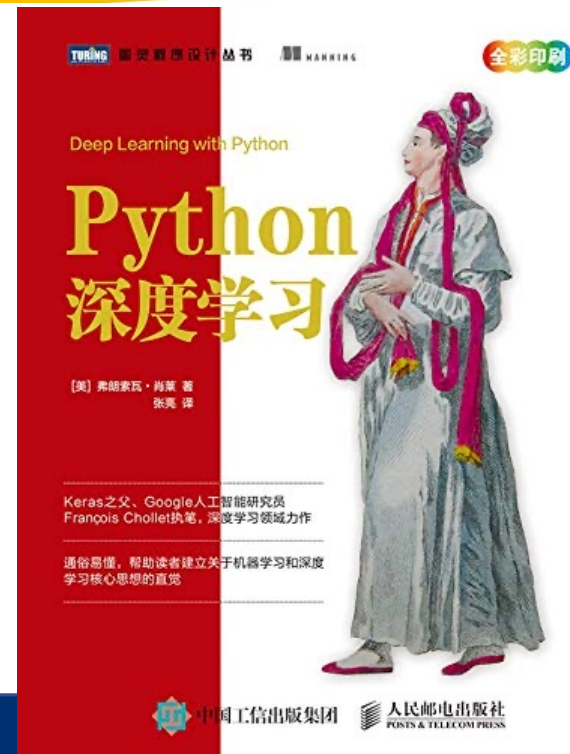
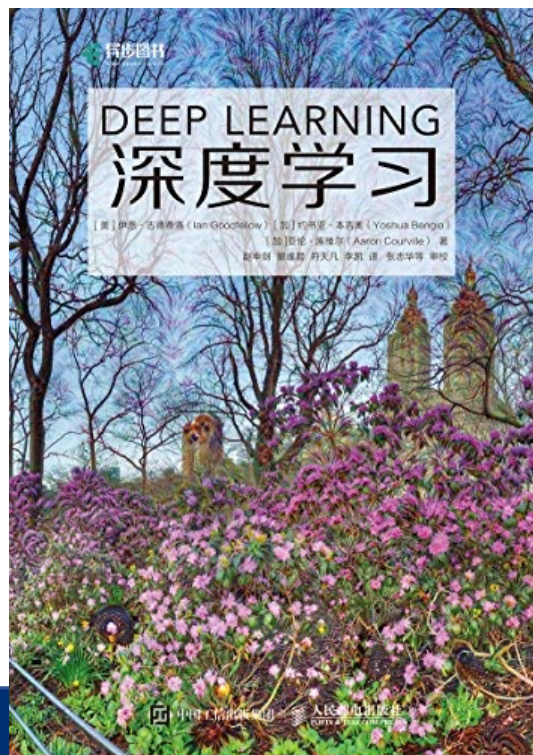
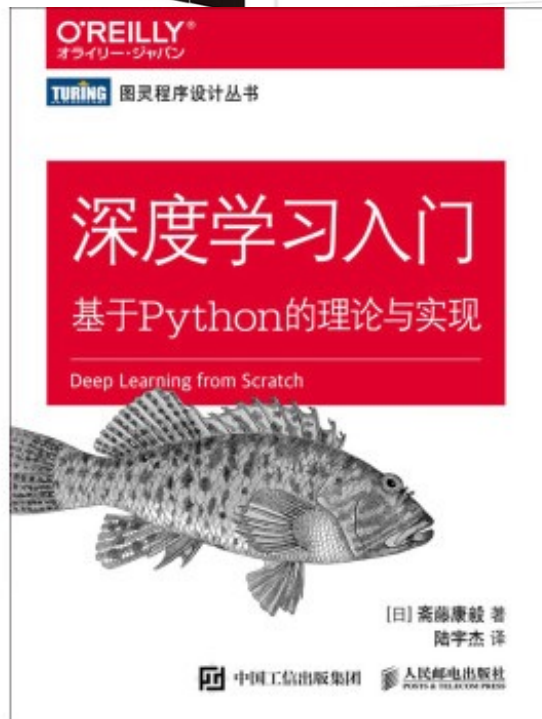
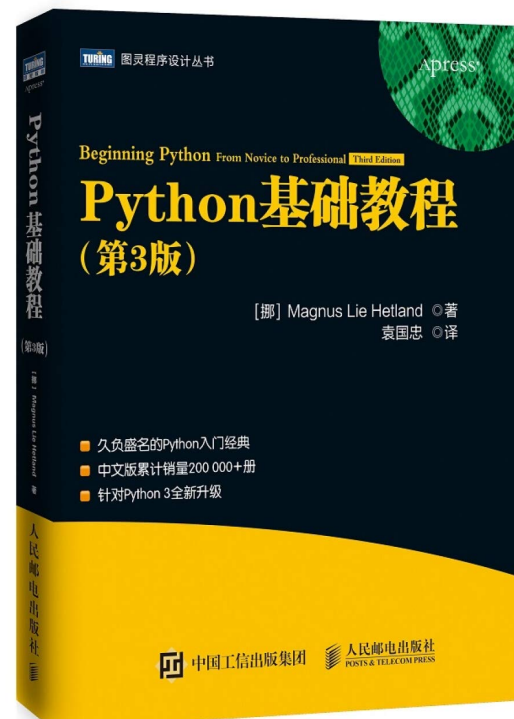
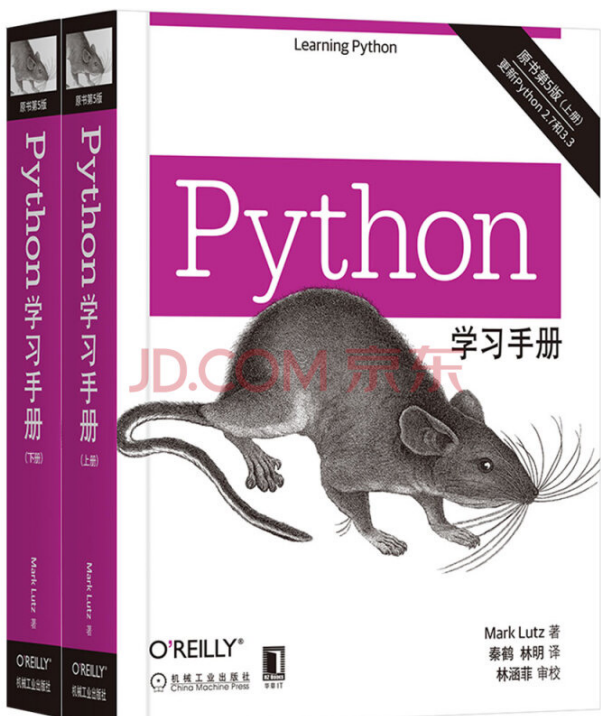
- 基础语法
- 数据类型
- 函数与模块
- 面向对象
- NumPy扩展库

- **深度学习的概念原理**

- 神经网络
- 反向传播算法
- 优化和正则化
- 经典深度神经网络模型
- 卷积神经网络
- 目标检测、语义分割、生成对抗网络等

# 考核方式及其他

- 考试
- 计分方法：
  - 平时成绩占30%，包括课堂表现和课后练习
  - 考试成绩占70%
- 教材及参考资料
  - Mark Lutz. Python学习手册. 北京：机械工业出版社, 2018.
  - Magnus Lie Hetland. Python基础教程. 北京：人民邮电出版社, 2018.
  - 斋藤康毅. 深度学习入门. 北京：人民邮电出版社, 2018.
  - Ian Goodfellow, Yoshua Bengio. 深度学习. 北京：人民邮电出版社, 2017.
  - Francois Chollet. Python深度学习. 北京：人民邮电出版社, 2018



# 课程目标

- 掌握Python程序设计基础
- 掌握深度学习的基本概念
- 培养动手实践能力

# 课程内容

- 课程介绍
- Python介绍
- Python 开发环境
  - Python安装与使用
  - Python包管理
  - Python IDE
- Python运行



# Python

- Python是一门编程语言，与C、C++，Python类似，但又不同。
  - 相同点
    - 面向对象语言：支持面向对象程序设计思想（封装、继承、多态）以及面向对象的编程风格。
  - 不同点
  - 开发效率极高：
    - 相比于众多其他的语言，使用 Python 编写时，程序包含的代码行更少。
  - Python 的语法简单、易编写
    - 相比其他语言，使用 Python 编写的代码更容易阅读、调试和扩展。
  - 初学者更易入门
    - 对初学者而言 Python 更容易入门，它支持广泛的应用程序开发，从文本处理到网络编程、Web开发、爬虫、科学计算以及**人工智能**。



# Python 优缺点

Python的优点



Python的缺点

易于学习：

Python广泛认同为最容易学的编程语言。



库：

大量令人惊讶的库和函数使得制作东西极为容易。



物联网：

Python也许会成为物联网最受欢迎的语言，如树莓派这样的新平台都基于Python。



速度：

作为解释型语言，比编译型语言慢很多。



移动端：

Python在移动计算方面很弱，很少有智能机应用由Python开发。



设计：

Python是动态型，它需要更多测试以及错误仅在运行时展示。

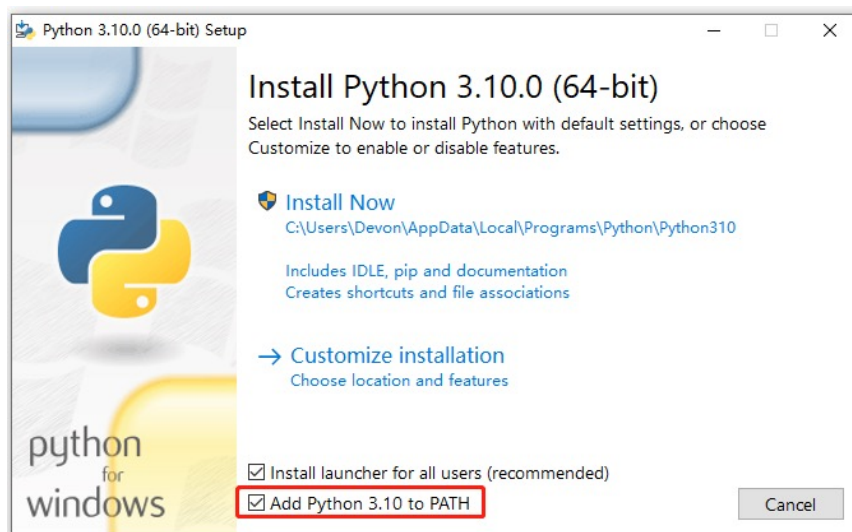


# Python 安装与使用

- 要开始学习Python编程，首先就得把Python安装到你的电脑里。
- 跨平台：
  - Python可以运行在Windows、Mac和各种Linux/Unix系统上。
- 可移植
  - 在Windows上写Python程序，放到Linux 上也是能够运行的 。
- Windows, Linux, MacOS下载安装
- 包管理工具安装

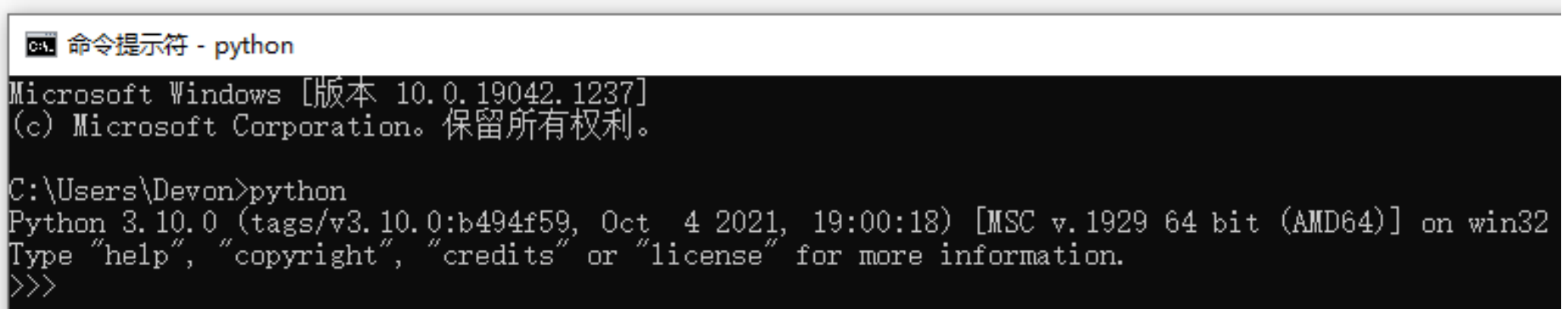
# Windows

- 从 [Python 官方网站](#) 下载 Python 安装包，下载最新的稳定版本是 Python 3.10.0，下载 [Windows X86-64 executable installer](#) 并双击安装。
- 为了可以在命令提示符窗口直接运行并且方便第三方软件找到 Python 路径，**勾选安装界面的 “Add Python 3.10 to Path” 选项**
- Python 官网: <https://www.python.org/downloads/windows/>



# Windows

- 完成 Python 的安装后，首先确认一下 Python 的版本。
- 打开 Windows 命令提示符窗口，输入 `python --version` 命令，
- 该命令会输出已经安装的 Python 的版本信息。也可以输入 `Python` 命令，启动 Python 自带的解释器环境。



```
命令提示符 - python
Microsoft Windows [版本 10.0.19042.1237]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Devon>python
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

# Linux

- Linux发行版本众多
  - Ubuntu、Linux Mint
  - Red Hat Enterprise Linux、CentOS等。
- 大多数系统已经默认集成 Python 环境，可能版本稍有不同。例如 [Ubuntu 20.04.3 LTS](#) 默认集成了 Python 3.8.10
- 通过终端窗口输入 "python" 命令来查看本地是否已经安装Python以及Python的安装版本。

# MacOS

- macOS 系统自带有 Python2.x 版本的环境，也可以在链接 <https://www.python.org/downloads/mac-osx/> 下载最新版本pkg安装包进行安装。
- 通过终端窗口输入 "python" 命令来查看本地是否已经安装Python以及Python的安装版本。

# Python包管理器

- 包管理器
  - 包管理器与包的开发、安装、更新密切相关。
  - 很多时候，Python系统的开发依赖各种包，手动管理非常困难。
  - 包之间相互依赖
    - 比如安装A需要先安装B，而安装B需要安装C和D。
- Pip: 最常用的包管理器
- conda: 功能丰富强调、提供环境管理



# Pip

- pip 是最重要的 Python 包管理工具，提供对Python 包的查找、下载、安装、卸载的功能。
  - 可以通过 pip 的命令下载、安装和卸载 PyPi (Python Package Index) 中的 Python 应用包，还可以轻松地将这种网络安装的方式加入到自己开发的 Python 应用中。
    - 注意: *Python 2.7.9 + 或 Python 3.4+ 以上版本都自带 pip 工具。*

```
$ pip --version      # Python2.x 版本命令
```

```
$ pip3 --version     # Python3.x 版本命令
```

```
$ pip install <package_name> # 安装包命令
```

```
$ pip install numpy # 安装numpy包
```

```
$ pip list # 使用 list 命令列出已经安装的 Python 包
```

```
$ pip search <search_string> #搜索安装包
```

```
$ pip uninstall <installed_package_name> #卸载一个Python包
```

# Conda

- conda 是 Anaconda 下用于包管理和环境管理的命令行工具。
- 安装方法有很多种，推荐使用 [Anaconda Distribution](#) 版。
- 一次性完成安装 NumPy、Matplotlib 等数据分析常用库、以及 Jupyter Notebook 等集成开发工具。

## Linux 中安装

```
$ sudo chmod +x Anaconda3-2021.11-Linux-x86_64.sh  
$ sudo ./Anaconda3-2021.11-Linux-x86_64.sh
```

或者

```
$ bash Anaconda3-2021.11-Linux-x86_64.sh
```

# Conda—环境管理

```
# 基于 python3.6 创建一个名为 test_py3 的环境
```

```
$ conda create --name test_py3 python=3.6
```

```
# 基于 python2.7 创建一个名为 test_py2 的环境
```

```
$ conda create --name test_py2 python=2.7
```

```
# 激活 test_py2 环境
```

```
$ conda activate test_py2
```

```
$ python
```

```
Python 2.7.18 |Anaconda, Inc.| (default, Apr 23 2020, 22:42:48)
```

```
[GCC 7.3.0] on linux2
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>>
```

```
# 切换到test_py3
```

```
$ conda activate test_py3
```

```
$ python
```

```
Python 3.6.12 |Anaconda, Inc.| (default, Sep 8 2020, 23:10:56)
```

```
[GCC 7.3.0] on linux
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>>
```

```
# 切换到默认Python环境
```

```
$ conda activate --stack
```

# Python 运行

- 执行 Python 程序的方式有多种。
- **Python Console**
  - 在系统环境下输入 “python” 后即可运行 Python Console。
  - 示范较小的例子时常常用到
  - Cpython 使用 “>>>” 作为提示符
  - IPython 使用 “In [序号]:” 作为提示符。

```
[libin@Bing-Pro ~]$python
Python 3.7.3 (default, Mar 27 2019, 16:54:48)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

```
Python 3.7.4 (default, Aug 9 2019, 12:36:10)
[Clang 4.0.1 (tags/RELEASE_401/final)] on darwin
```

```
In[2]:
```

# Python 运行

```
In [1]: print('hello world')
```

```
hello world
```

```
In [4]: 'spam!' * 8
```

```
Out[4]: 'spam!spam!spam!spam!spam!spam!spam!spam!'
```

在 Python 的标准库 `os` 和 `sys` 模块中的函数，分别是显示当前所工作的目录名称和系统平台名称以及版本号：

```
In [5]: import os
print(os.name)
print(os.getcwd())
```

```
posix
/Users/bing/OfflineDocument/Teaching/python-deeplearning/python-deep-learning-master
```

```
In [6]: import sys
print(sys.platform)
print(sys.version)
```

```
darwin
3.7.3 (default, Mar 27 2019, 16:54:48)
[Clang 4.0.1 (tags/RELEASE_401/final)]
```

# Python 运行

- 脚本文件
  - 将代码写入脚本文件中，用一条命令运行脚本文件。

```
$ python script1.py
```

- 注意：要指定文件后缀名为 .py
- 需要使用 `print` 语句来看程序文件的输出

# Python 集成开发环境IDE

- 很多文本编辑器都对 Python 有不同程度的支持，并且加上专门为 Python 设计的编辑器插件也会有很高的可用性。
  - Emacs, Vim, Sublime Text, Notepad++, UltraEdit等。
- 这里介绍常用的几种IDE。
  - Jupyter Notebook, Visual Code Studio, 和Pycharm。

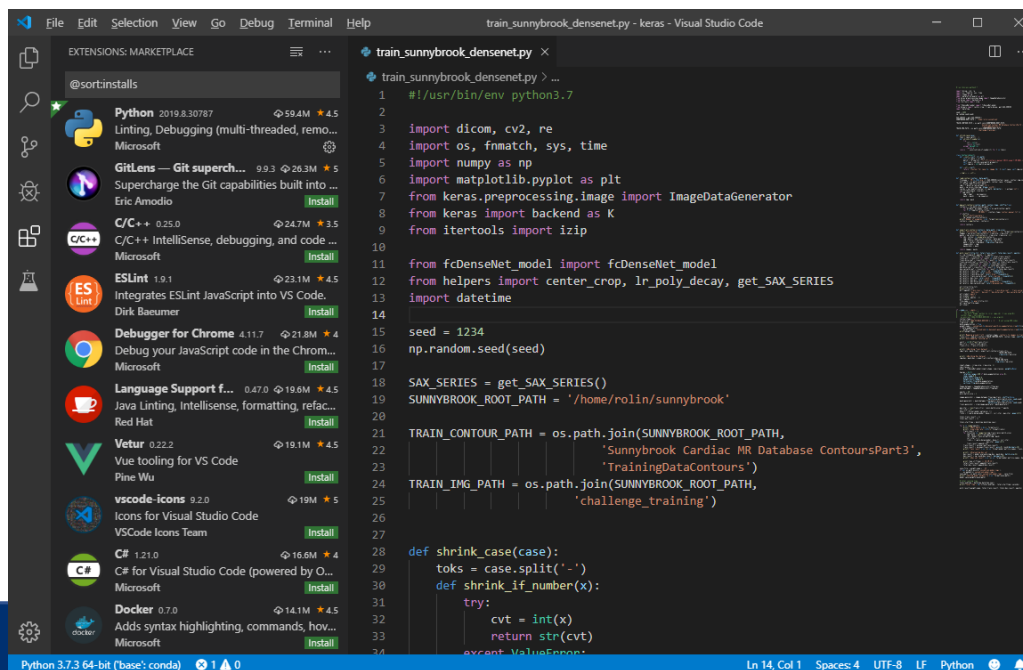
# Jupyter Notebook

- Jupyter Notebook 是一款开源的 Web 应用程序，可以在网页页面中直接编写代码和运行代码，代码的运行结果也会直接在代码块下显示。
- 如在编程过程中需要编写说明文档，可在同一个页面中直接编写，便于作及时的说明和解释。
- 它可以将冗长的实验代码拆分为可独立执行的短代码，这使得开发具有交互性，而且如果后面的代码出现问题，你也不必重新运行前面的所有代码。
- 支持语法高亮、缩进、代码补全等功能，支持Markdown语法。



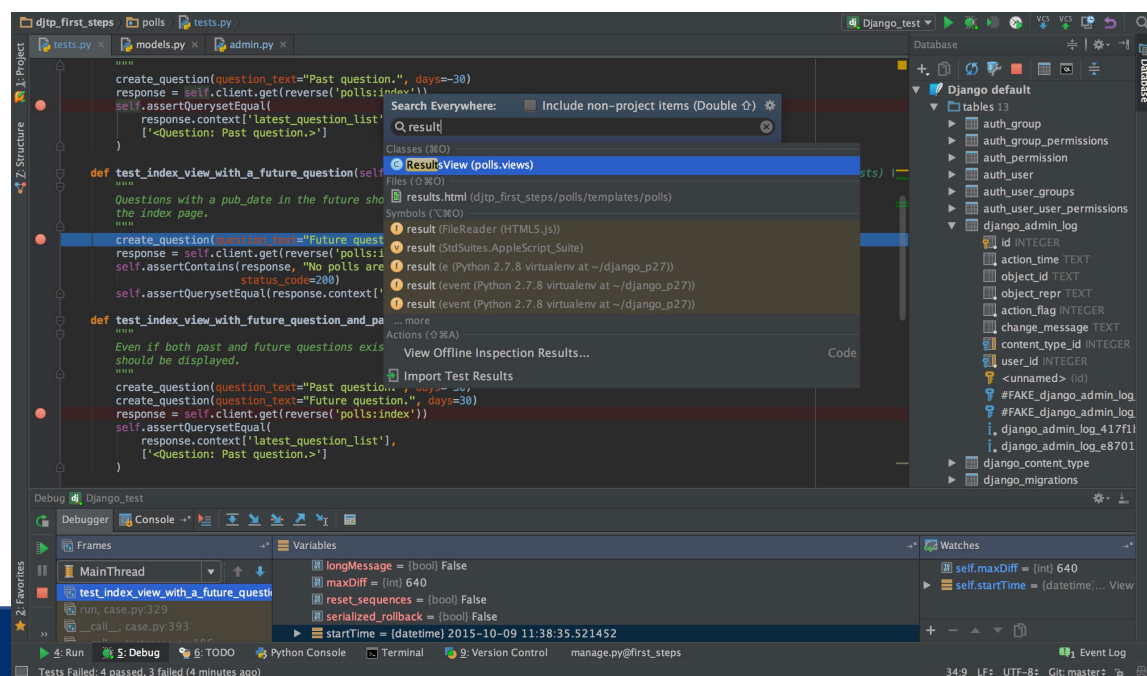
# Visio Studio Code

- Visual Studio Code (简称VS Code) 是一个由 Microsoft 开发，支持 Windows 、 Linux 和 macOS 且开源的代码编辑器。
- 它支持多种编程语言，支持测试，并内置了Git 版本控制功能，同时也具有开发环境功能，例如代码补全、代码片段和代码重构等。
- 该编辑器支持用户个性化配置，例如改变主题颜色、键盘快捷方式等各种属性和参数，同时还在编辑器中内置了扩展程序管理的功能。



# Pycharm

- PyCharm 是由 JetBrains 开发的一款 Python IDE，支持 Windows、Linux和 macOS 系统。
- PyCharm 功能包括调试、语法高亮、Project管理、代码跳转、智能提示、自动完成、单元测试、版本控制等。
- PyCharm 分为免费的社区版 (Community)、面向企业开发者的付费专业版 (Professional)、和面向教育者的教育版本。



# Python如何运行程序

- 字节码编译

- 先将源代码（文件中的语句）编译成所谓字节码。
- 编译是一个简单的翻译步骤，而且字节码是源代码底层的、与平台无关的表现形式。
- 字节码可以提高执行速度：比起文本文件中的原始代码语句，字节码的运行速度要快得多。

- Python虚拟机（PVM）

- 字节码发送到通常称为 Python 虚拟机（Python Virtual Machine, PVM）上来执行。
- PVM 就是迭代运行字节码指令的一个大循环，是实际运行脚本的组件。从技术上来讲，它才是 “Python 解释器” 的最后一步。

# Python的版本

- 陈旧的 Python 2 和较新的 Python 3两个不同的版本
  - 尽管只是统一语言的不同版本，但 Python 3 几乎无法运行为 Python 2 版本编写的代码。从 `print` 函数到运算符再到标准库，Python 3 都相对 Python 2 发生了很大变化。
- 最后一个Python 2.x 版本是发布于2010年的 Python 2.7
- 2020年4月起官方已经不再对 Python 2.x 进行支持和维护。
- 很多遗留的老系统依旧运行在 Python2 的环境中，目前最终的版本是2020年4月发布的 Python 2.7.18。
- Python 3.0 发布于2008年，目前最新的版本是2021年10月发布的 Python 3.10.0，并且从 3.9 版本起不再支持 Windows 7 系统。

# 小结

- 课程介绍
- Python开发环境准备