Python Programming

Lecture 1 Introduction

1.1 Course Intro

- This is a basic course for Python programming.
- "Life is short, you need Python."

  by Bruce Eckel, ANSI C++ Comitee member


Life is short, use Python
人生苦短
我用python
INTERN
parc
ZEUUX Project
zeuux.org
free software, free society
http://blog.csdn.net/liang19890820

- 什么是计算机“语言”？
- 什么是编程？为什么要编程？
- 为什么要数据分析？来源，精细化
- 有很多程序设计语言,比如C，Java，Basic等等
- 为什么要学Python？
  - 原因一：高级 （简单）
  - 原因二：普及，社区内容丰富
  - 原因三：强大，扩展性强

TIOBE编程语言社区发布了2022年2月编程语言排行榜

Feb 2022	Feb 2021	Change	Programming Language	Ratings	Change
1	3	^	Python	15.33%	+4.47%
2	1	v	C	14.08%	-2.26%
3	2	v	Java	12.13%	+0.84%
4	4		C++	8.01%	+1.13%
5	5		C#	5.37%	+0.93%
6	6		Visual Basic	5.23%	+0.90%
7	7		JavaScript	1.83%	-0.45%
8	8		PHP	1.79%	+0.04%
9	10	^	Assembly language	1.60%	-0.06%

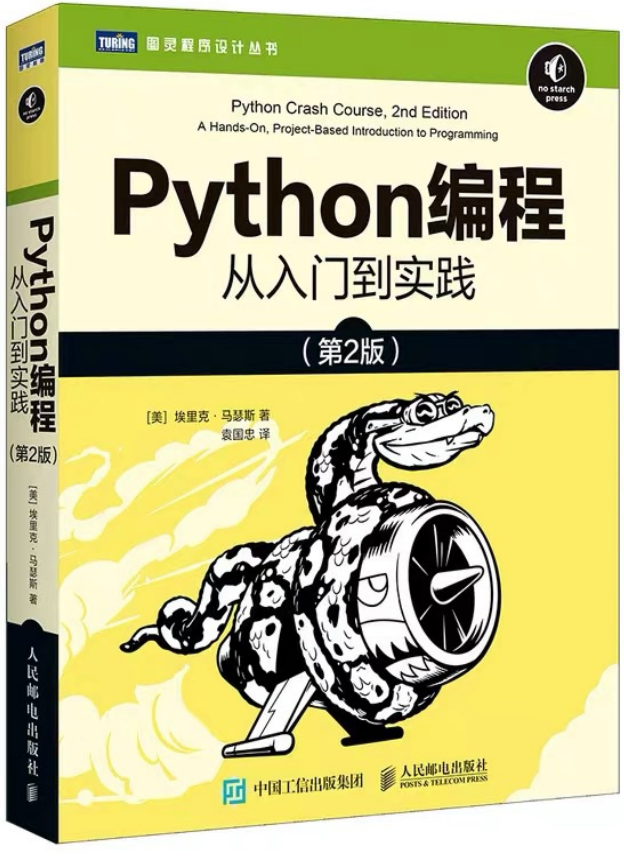
这门课主要涉及3个板块

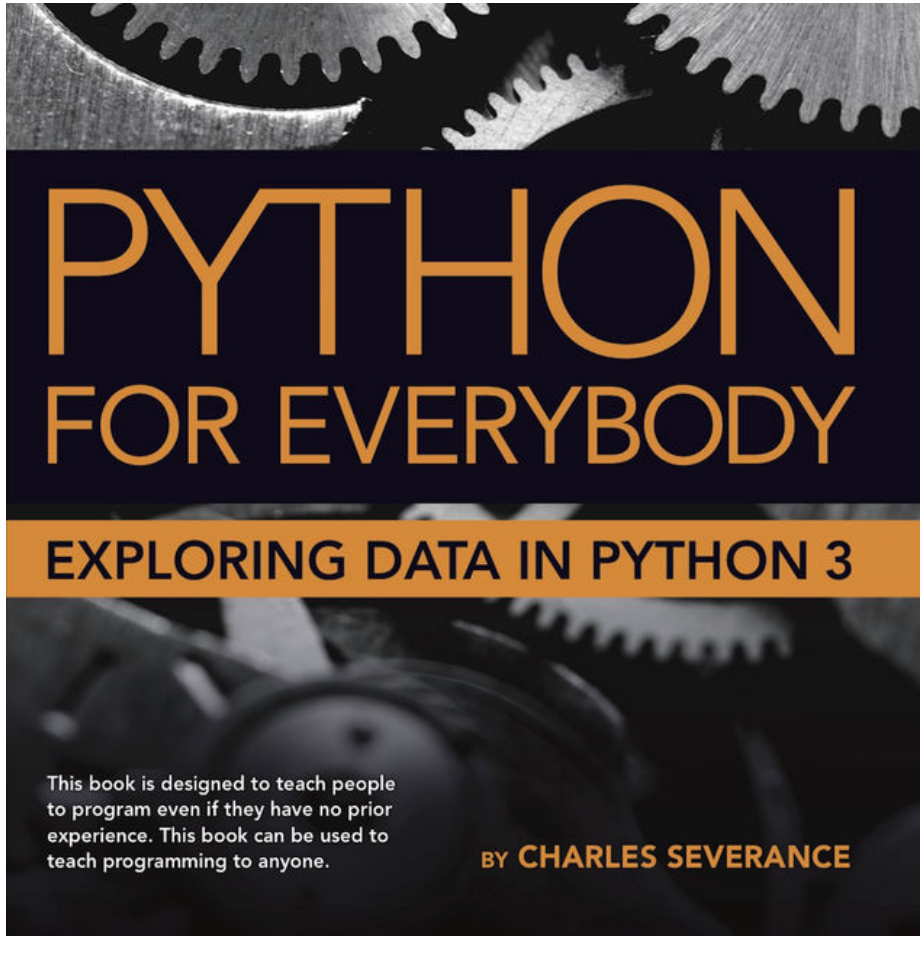
- 基本语法
- 数据分析
- 简单的算法

课程要求

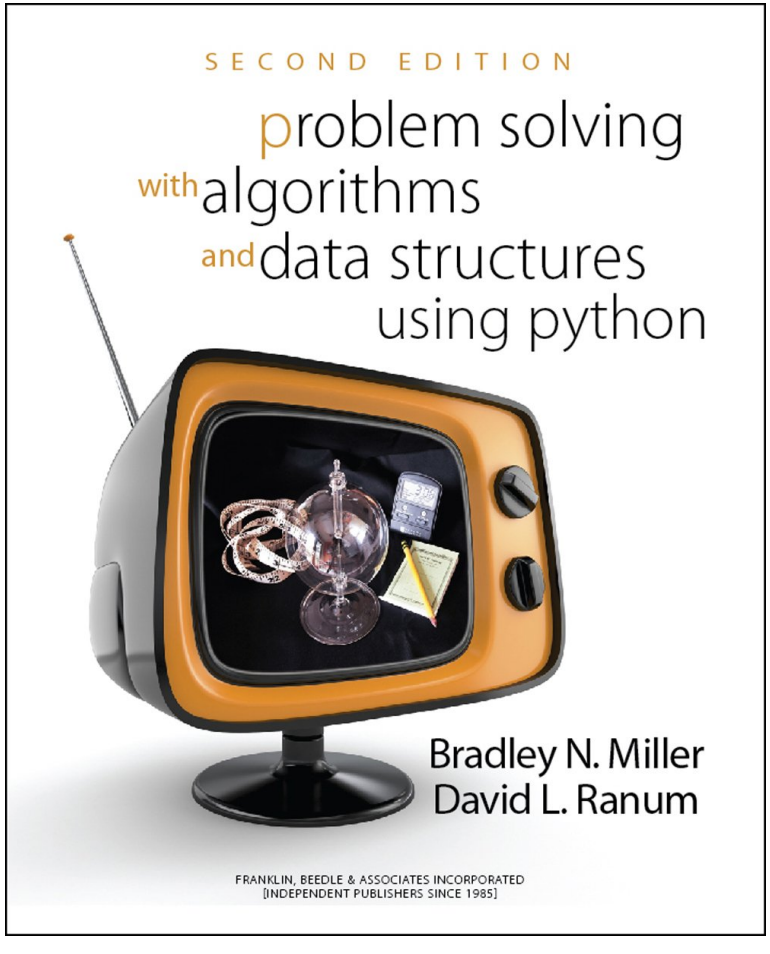
1. 一共三次作业（6%*3=18%)
2. 案例设计 （12%）
3. 期末考试 （60%）
4. 出勤率 (10%)
5. 答疑方式：每周二下午 15:30-16:30 or by appointment
6. 联系方式
    - 邮箱： wang.lu@mail.shufe.edu.cn
    - 办公室：武东路校区商学院楼412室
    - 请不要在企业微信里问代码方面的问题
    - 邮箱仅用于约面对面答疑的时间，请不要直接发送问题

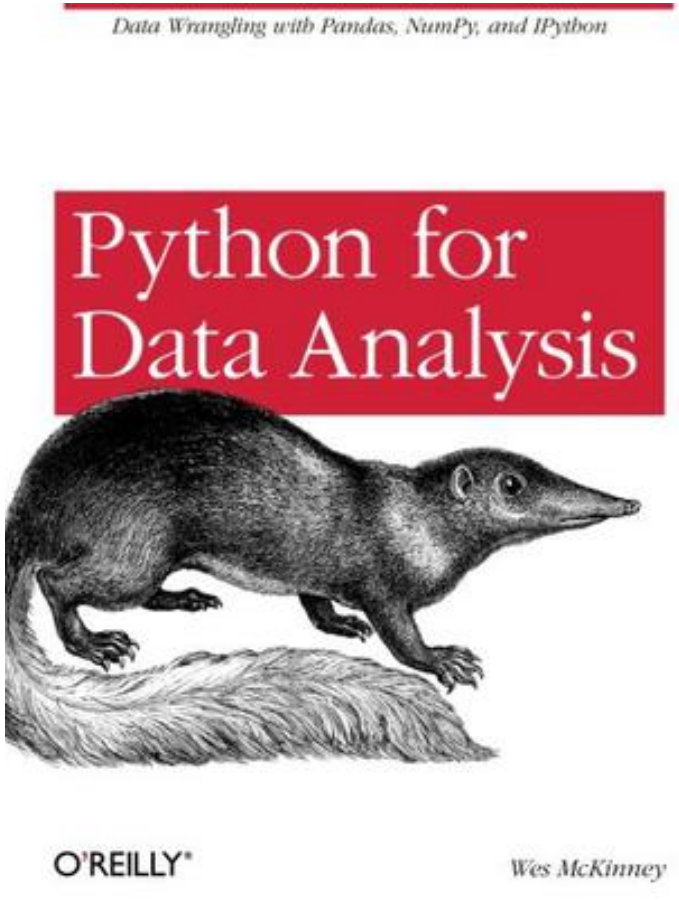
主要教材


TURING 图灵程序设计丛书
no starch press
Python Crash Course, 2nd Edition
A Hands-On, Project-Based Introduction to Programming
Python编程
从入门到实践
（第2版）
[美] 埃里克·马瑟斯 著
袁国忠 译
中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS


PYTHON
FOR EVERYBODY
EXPLORING DATA IN PYTHON 3
This book is designed to teach people to program even if they have no prior experience. This book can be used to teach programming to anyone.
BY CHARLES SEVERANCE

参考教材


SECOND EDITION
problem solving
with algorithms
and data structures
using python
Bradley N. Miller
David L. Ranum
FRANKLIN, BEEDLE & ASSOCIATES INCORPORATED
[INDEPENDENT PUBLISHERS SINCE 1985]


Data Wrangling with Pandas, NumPy, and IPython
Python for
Data Analysis
O'REILLY
Wes McKinney

- 课件从哪里获取?
  - PDF在上财教学网Blackboard系统上下载
  - 或者我的网站：wangwanglulu.com
- 作业如何提交?
  - 作业为编程题，以电子版的方式提交
  - 作业的时限为一周
  - 提交到上财教学网Blackboard系统
  - 下次布置作业时会讲解作业的格式
- 其他
  - 除课件外，我的网站上有与每次课相关的资料和课外阅读
  - 主要教材的代码下载与使用
  - 原始课件为reveal.js格式，如何使用?

1.2 Python Intro

- Python的简史，特色和安装

Python历史

Python的创始人为吉多·范罗苏姆（Guido van Rossum）。1989年的圣诞节期间，吉多·范罗苏姆为了在阿姆斯特丹打发时间，决心开发一个新的脚本解释程序，作为ABC语言的一种继承。之所以选中Python作为程序的名字，是因为他是BBC电视剧——蒙提·派森的飞行马戏团（Monty Python's Flying Circus）的爱好者。

Python的特色

Python的设计哲学是“优雅”，“明确”，“简单”。Python开发者的哲学是“用一种方法，最好是只有一种方法来做一件事”。

——出自Wikipedia的Python

废话不多说，上手试一下

Python的安装

- 请使用Python 3.x (很多教材是2.x版本)
- Anaconda，两种模式
- 实际演示：第一个Python程序

```python
print("Hello, world!")
```

- 第二个Python程序

```python
import antigravity
```

- 第三个Python程序

```python
import this
```

程序设计的一些准则

怎么把Python学好?

1. 初学者要勤写注释（什么是注释？语音输入）
2. 遇到困难，勤查资料寻找办法解决问题（CSDN, stack overflow）
3. 搞清楚需求，解决问题最重要
4. 尽量写容易看明白的代码，尽量写少的代码
5. 不要复制粘贴，一个字一个字的敲
6. 不用太在意速度，不要高估自己写的代码

最后请各位记住一点。。。

- 编程终究只是“术”，是解决问题的工具
- 这是你必须掌握的技巧，不是你追求的终极目标
- 你的思考和创造力是最重要的

还有一点很遗憾。。。

即使学会了这门课，你还是不会“修电脑”。

1.3 Values

Three types of values

- Integer (整数)
  - e.g., $1, 2, 15, -10$
- Float (浮点数)
  - numbers with decimal point, e.g., 1.2, -3.8.
  - for large or small numbers,
  - $5.91 \times 10^{10} = 5.91e10$, $0.00233 = 2.33e-3$.
- String (字符串)
  - e.g., 'happy'

Python can tell you what type a value has.

>>> means what you input in the IPython console. The first column below is the line number.

```
1 >>> type(4)
2 int
3
4 >>> type('Hello, World!')
5 str
6
7 >>> type(3.2)
8 float
```

Note: If numbers are in quotation marks, they are strings.

```
1 >>> type('17')
2 str
3
4 >>> type('3.2')
5 str
```

Escape string

```
>>> print('happy')
happy
```

If we want to print the single quotation marks?

```
>>> print("'happy'")
'happy'
```

If we want to print the double quotation marks?

```
>>> print('\"happy\"')
"happy"
```

If we want to print the slash?

```
>>> print('\\happy\\')
\happy\
```

If we want to print the double slashes?

```
>>> print(r'\\happy\\')
\\happy\\
```

Line break

```
>>> print('I\'m learning\nPython.')
I'm learning
Python.
```

Tab

```
>>> print('I\'m learning\tPython.')
I'm learning    Python.
```

```
>>> print('\\\n\\')
>>> print('\\\t\\')
>>> print(r'\\\t\\')
```

1.4 Variables

- A variable is a name that refers to a value.
- An assignment statement creates new variables and gives them values.

```
>>> message = 'something'
>>> n = 17
>>> pi = 3.1415926535897931
```

- To display the value of a variable, you can use a print statement.

```
>>> print(n)
17
>>> print(pi)
3.141592653589793
```

- The type of a variable is the type of the value it refers to.

```
>>> type(message)
str
>>> type(n)
int
>>> type(pi)
float
```

- Note that = means "assignment", not equation.

```
>>> x = 15
x = x + 10
```

- Python is a Dynamically Typed Language. What is Dynamically Typed Language?

```
>>> apple = 123
# apple is an integer
>>> print(apple)
123
```

```
>>> apple = 'ABC'
# Now apple turns to be a string
>>> print(apple)
ABC
```

- We can assign the value of a variable to another variable.

```
>>> a = 'ABC'
>>> b = a
>>> print(b)
ABC
```

Variable name

- Variable names can be arbitrarily long. They are case sensitive.
- You cannot start with a number.
- It is common to use underscore character _.
- Spaces are not allowed in variable names, but underscores can be used to separate words in variable names.
- Illegal name

```
>>> 76trombones = 'big parade'
SyntaxError: invalid syntax

>>> more@ = 1000000
SyntaxError: invalid syntax

>>> class = 'Advanced Theory'
SyntaxError: invalid syntax
```

Keywords

- You cannot use the following keyword as variable names since they have been reserved by Python to recognize the structure of the program.

```
and del from None True
as elif global nonlocal try
assert else if not while
break except import or with
class False in pass yield
continue finally is raise
def for lambda return
```

You would be better to use recognizable names. Check the following names.

```
>>> a = 35.0
>>> b = 12.50
>>> c = a * b
>>> print(c)
```

```
>>> hours = 35.0
>>> rate = 12.50
>>> pay = hours * rate
>>> print(pay)
```

```
>>> x1q3z9ahd = 35.0
>>> x1q3z9afd = 12.50
>>> x1q3p9afd = x1q3z9ahd * x1q3z9afd
>>> print(x1q3p9afd)
```

Which ones are better?

1.5 Operators, input and output

Operators

- $+, -, *, /, **$(exponentiation), %, //
- Order of operations

```
>>> minute=50
>>> minute/20
2.5 # float
```

```
>>> minute=50
>>> minute//20
2 # floored integer
```

```
>>> minute=50
>>> minute%20
10 # remainder. useful!
```

```
>>> student_number=2017003425
>>> student_number//1000000
2017
>>> student_number%10000
3425
```

- String operations

```
>>> first=10
>>> second=15
>>> print(first+second)
25

>>> first='100'
>>> second='150'
>>> print(first + second)
100150
```

Input and Output

- input() and print()

```
>>> x = input()
Some silly stuff

>>> print(x)
Some silly stuff
```

- You can pass a string to input to be displayed to the user before pausing for input
- Note: By using input(), the type of what the user inputs is always string!

```
>>> name = input('What is your name?\n')
What is your name?
Chuck # This is what you input

>>> print(name)
Chuck # The type is string.
```

- If you expect the user to type an integer, you can try to convert the return value to integer using the int() function

```
>>> prompt = 'What is the speed?\n'
>>> speed = input(prompt)
What is the speed?
17

>>> int(speed)
# or float(speed)
17

>>> int(speed) + 5
22
```

- It may cause error, if the value cannot be converted.

```
>>> speed = input(prompt)
What is the speed?
I do not know. #This is what you input.

>>> int(speed)
ValueError: invalid literal for int() with base 10:
```

- To write or display long program, Editor will be used. Green box shows the output of a sequence of code. Red box shows the error message. Basically, the Editor is the same with IPython console.

```
age = 23
message = "Happy " + age + "rd Birthday!"
print(message)
```

```
Traceback (most recent call last):
File "birthday.py", line 2, in <module>
message = "Happy " + age + "rd Birthday!"
TypeError: Can't convert 'int' object to str implicitly
```

```
age = 23
message = "Happy " + str(age) + "rd Birthday!"
print(message)
```

```
Happy 23rd Birthday!
```

Summary

- Course intro
- Python intro
- Tips for programming
- Values, variables, operators, input and output
- Reading: Python for everybody Chapter 2