



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



- 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, VB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
- 关于AI编程（比如，Chatgpt, Copilot, Deepseek等）
 - 很好的编程辅助工具
 - 现阶段依然有一些局限（有幻觉）
 - 不建议在初学编程阶段过度依赖

TIOBE编程语言社区发布了2025年2月编程语言排行榜

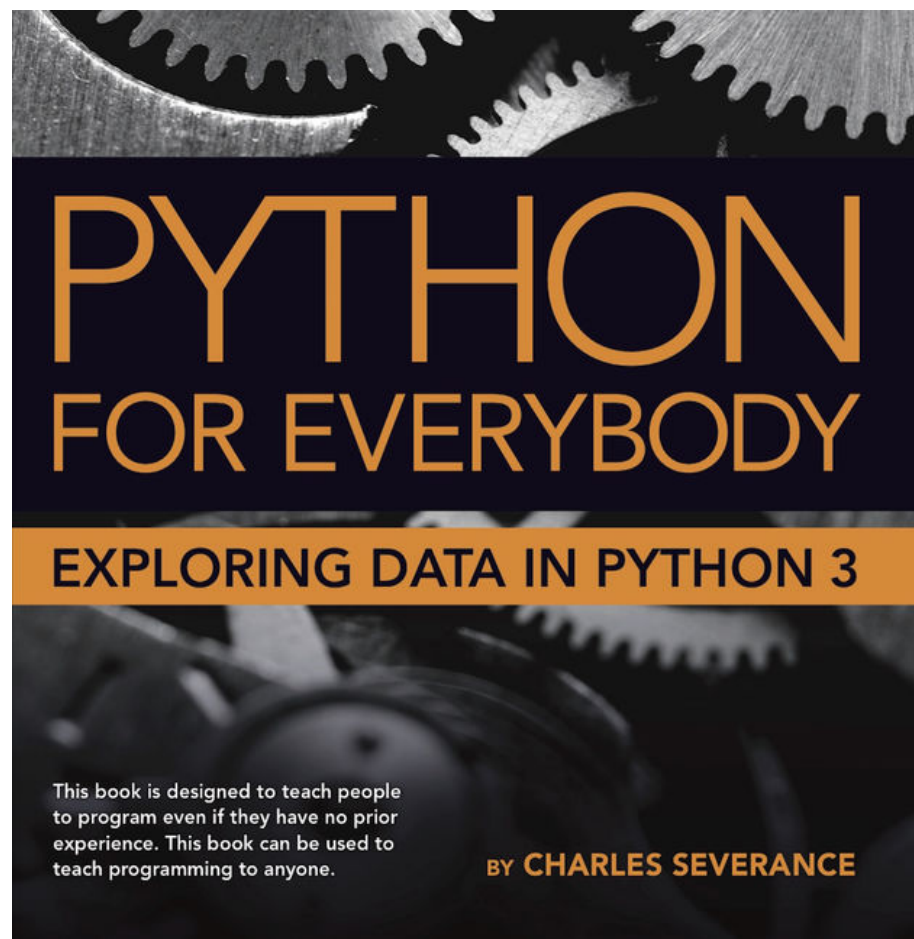
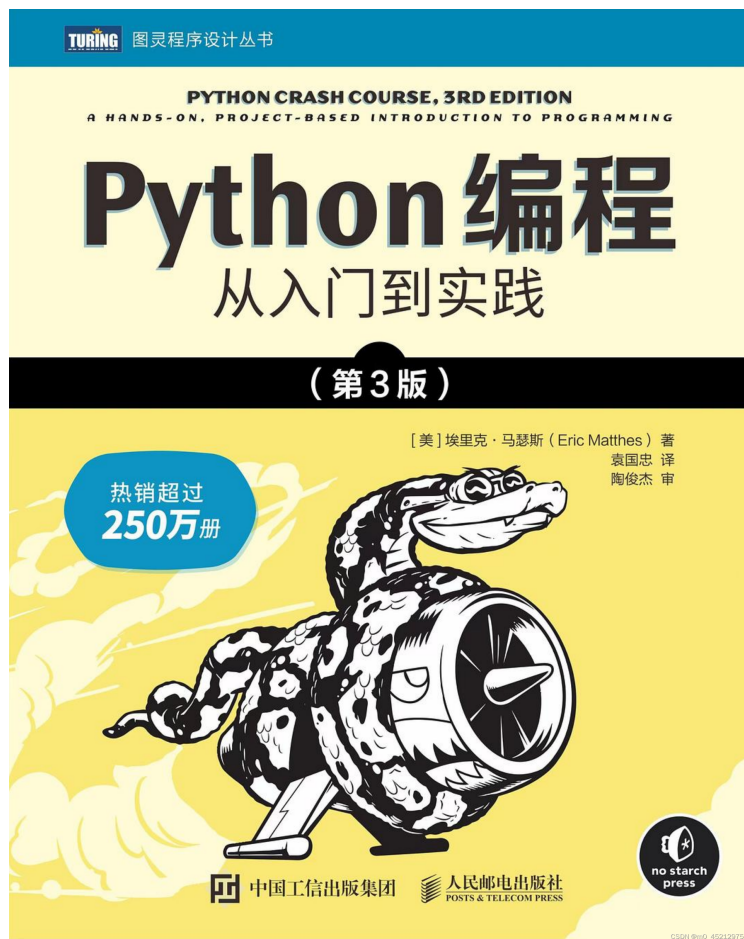
Feb 2025	Feb 2024	Change	Programming Language		Ratings	Change
1	1			Python	23.88%	+8.72%
2	3	▲		C++	11.37%	+0.84%
3	4	▲		Java	10.66%	+1.79%
4	2	▼		C	9.84%	-1.14%
5	5			C#	4.12%	-3.41%
6	6			JavaScript	3.78%	+0.61%
7	7			SQL	2.87%	+1.04%
8	8			Go	2.26%	+0.53%
9	12	▲		Delphi/Object Pascal	2.18%	+0.78%
10	9	▼		Visual Basic	2.04%	+0.52%

课程要求

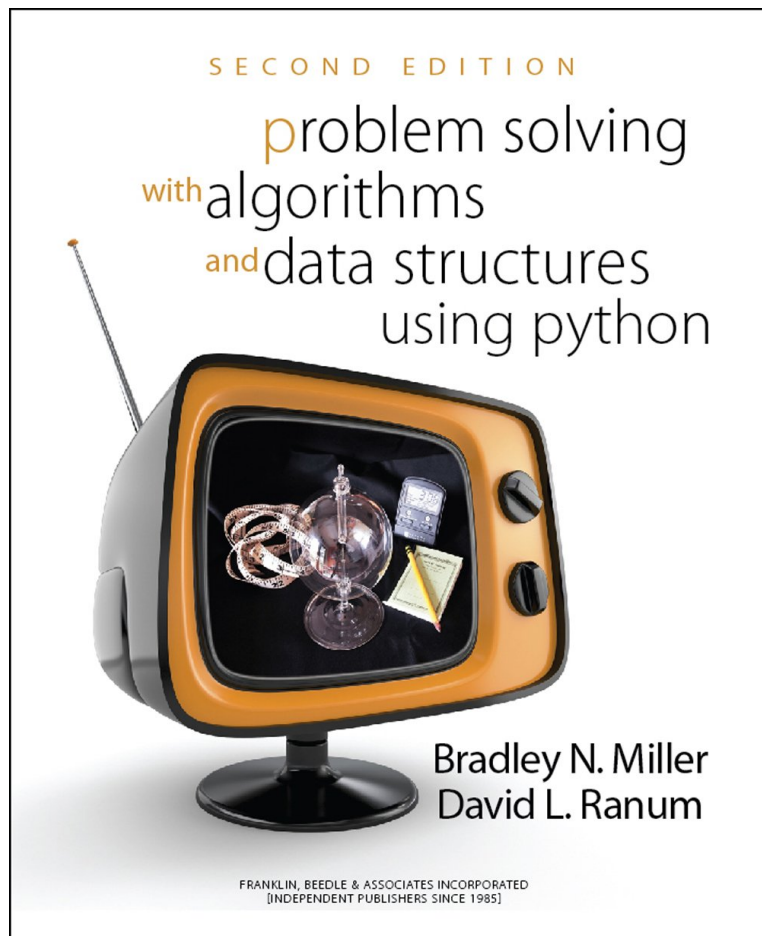
1. 一共三次作业 ($6\% \times 3 = 18\%$)
2. 案例设计 (12%)
3. 期末考试 (60%)
4. 出勤率 (10%)
5. 答疑方式：每周二下午 15:30-16:30 或者邮件约时间
6. 联系方式
 - 邮箱：wang.lu@mail.shufe.edu.cn
 - 办公室：武东路校区商学院楼412室
 - 请不要在企业微信里问代码方面的问题
 - 邮箱仅用于约面对面答疑的时间，请不要直接发送问题

这门课主要涉及3个板块：基本语法，数据分析，基础算法

主要教材



参考教材



- 课件从哪里获取？
 - PDF在上财教学网Canvas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Canvas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 (小的版本之间差异不大)
- [Anaconda](#), 两种模式
- 实际演示: 第一个Python程序

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

- 第二个Python程序

```
1 import antigravity
```

- 第三个Python程序

```
1 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？AI辅助）
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

1.3 Values (値)

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 (转义符)

```
1 >>> print('happy')
2 happy
```

If we want to print the single quotation marks?

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

If we want to print the double quotation marks?

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

If we want to print the slash?

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

If we want to print the double slashes?

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

Line break

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

Tab

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

```
1 >>> print('\\\\n\\\\')
2 >>> print('\\\\t\\\\')
3 >>> print(r'\\\\t\\\\')
```

1.4 Variables (变量)

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

```
1 >>> message = 'something'  
2 >>> n = 17  
3 >>> pi = 3.1415926535897931
```

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

```
1 >>> print(n)  
2 17  
3 >>> print(pi)  
4 3.141592653589793
```

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

```
1 >>> type(message)  
2 str  
3 >>> type(n)  
4 int  
5 >>> type(pi)  
6 float
```

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

```
1 >>> x = 15
2 >>> x = x + 10
```

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

```
1 >>> apple = 123
2 # apple is an integer
3
4 >>> print(apple)
5 123
```

```
1 >>> apple = 'ABC'
2 # Now apple turns to be a string
3
4 >>> print(apple)
5 ABC
```

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

```
1 >>> a = 'ABC'
2 >>> b = a
3 >>> print(b)
4 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

```
1 >>> 76trombones = 'big parade'
2 SyntaxError: invalid syntax
3
4 >>> more@ = 1000000
5 SyntaxError: invalid syntax
6
7 >>> class = 'Advanced Theory'
8 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.

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

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

```
1 >>> a = 35.0
2 >>> b = 12.50
3 >>> c = a * b
4 >>> print(c)
```

```
1 >>> hours = 35.0
2 >>> rate = 12.50
3 >>> pay = hours * rate
4 >>> print(pay)
```

```
1 >>> x1q3z9ahd = 35.0
2 >>> x1q3z9afd = 12.50
3 >>> x1q3p9afd = x1q3z9ahd * x1q3z9afd
4 >>> print(x1q3p9afd)
```

Which ones are better?

1.5 Operators, input and output (运算符, 输入和输出)

Operators

- $+$, $-$, $*$, $/$, $**$ (exponentiation), $\%$, $//$

- Order of operations

```
1 >>> minute=50
2 >>> minute/20
3 2.5 # float
```

```
1 >>> minute=50
2 >>> minute//20
3 2 # floored integer
```

```
1 >>> minute=50
2 >>> minute%20
3 10 # remainder. useful!
```

```
1 >>> student_number=2017003425
2 >>> student_number//1000000
3 2017
4 >>> student_number%10000
5 3425
```

- String operations

```
1 >>> first=10
2 >>> second=15
3 >>> print(first+second)
4 25
5
6 >>> first='100'
7 >>> second='150'
8 >>> print(first + second)
9 100150
```

Input and Output

- input() and print()

```
1 >>> x = input()
2 Some silly stuff
3
4 >>> print(x)
5 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!**

```
1 >>> name = input('What is your name?\n')
2 What is your name?
3 Chuck # This is what you input
4
5 >>> print(name)
6 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

```
1 >>> prompt = 'What is the speed?\n'
2 >>> speed = input(prompt)
3 What is the speed?
4 17
5
6 >>> int(speed)
7 # or float(speed)
8 17
9
10 >>> int(speed) + 5
11 22
```

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

```
1 >>> speed = input(prompt)
2 What is the speed?
3 I do not know. #This is what you input.
4
5 >>> int(speed)
6 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.

```
1 >>> age = 23
2 >>> message = "Happy " + age + "rd Birthday!"
```

```
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
```

```
1 >>> age = 23
2 >>> message = "Happy " + str(age) + "rd Birthday!"
3 >>> 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

