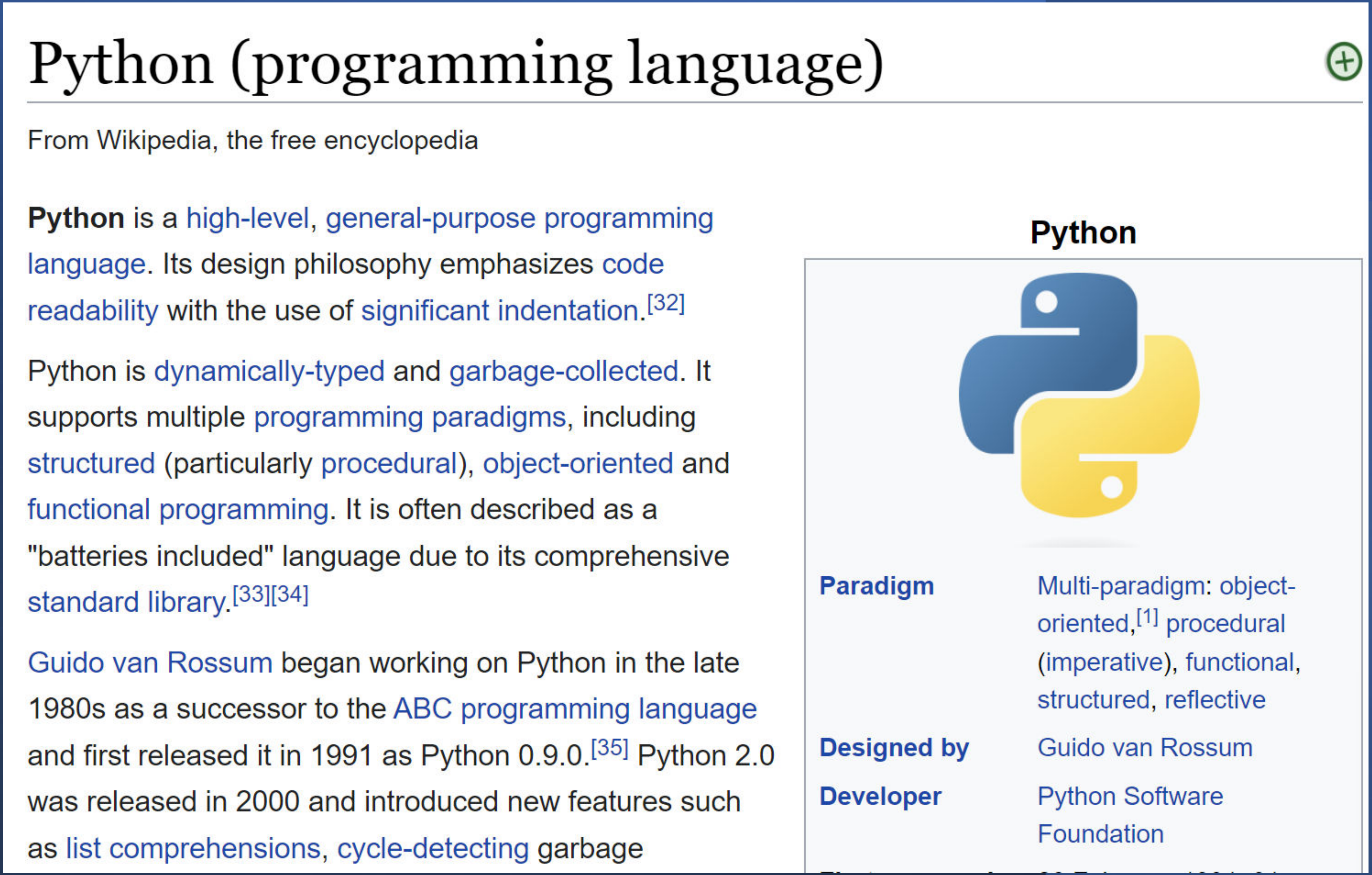
# Experiments

| 实验项目名称 | 实验地点 | 实验类型 | 实验时间<br>（周、节） |
|---|---|---|---|
| Axiom of Probability in Python | 致理楼<br>L3-101 | 验证 | 第1-4周 |
| Conditional Probability and Bayes Rule in Python | | | 第5-8周 |
| Bayes Classifier in Natural Language Processing | | 综合 | 第9-12周 |
| Application of Central Limit Theorem | | | 第13-16周 |

# Aim of Experiments

- Master the basic of Python.

- Employ probability and statistics tools to solve practical problems.



# Python (programming language)

From Wikipedia, the free encyclopedia

**Python** is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.[32]

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.[33][34]

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0.[35] Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage

**Python**

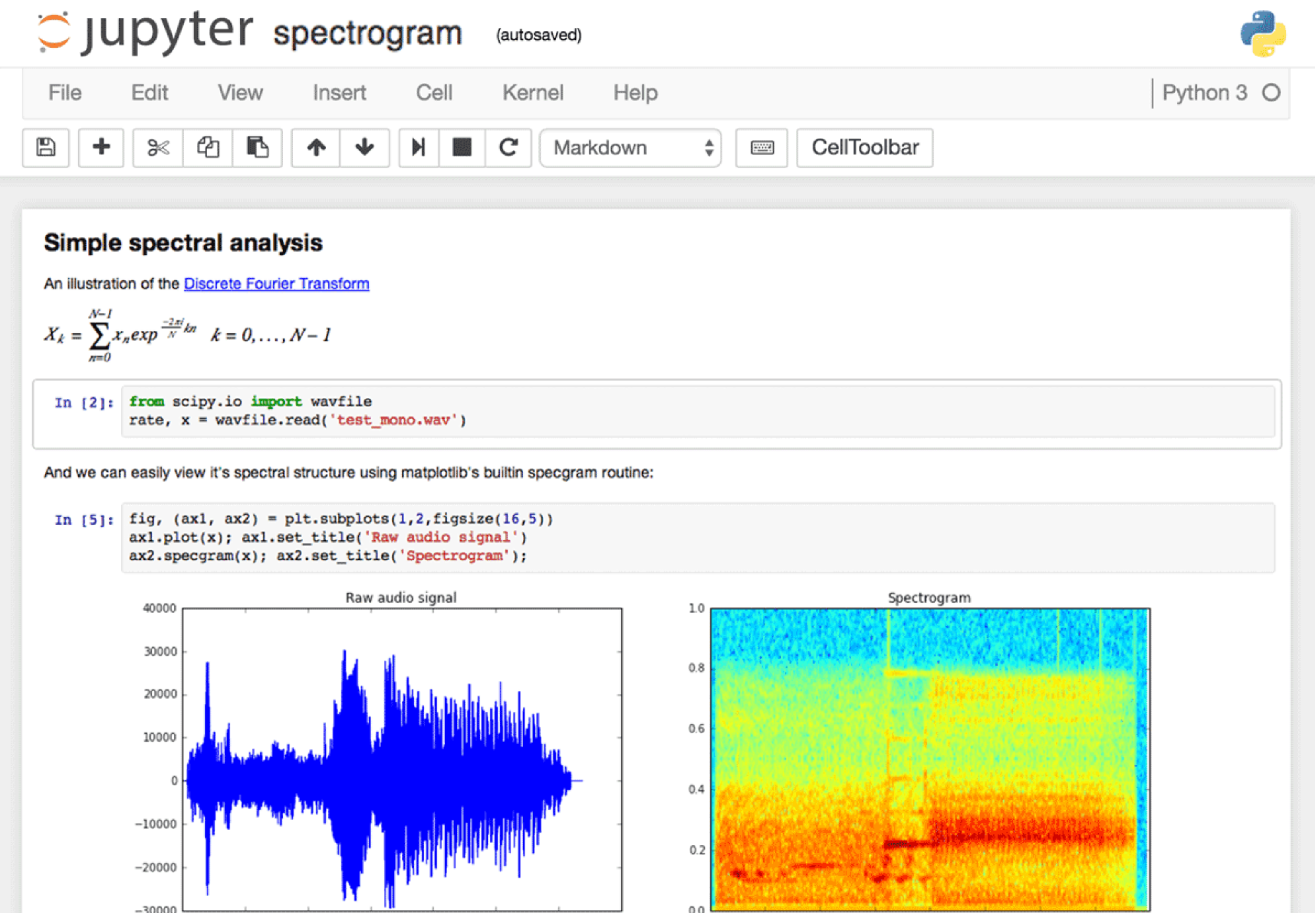
| | |
|---|---|
| **Paradigm** | Multi-paradigm: object-oriented,[1] procedural (imperative), functional, structured, reflective |
| **Designed by** | Guido van Rossum |
| **Developer** | Python Software Foundation |

# Why is Python so Popular?

- **Top reason: Easy to Learn and Use!**

Python language is one of the most accessible programming languages available because it has simplified syntax and is not complicated, which gives more emphasis on natural language. Due to its ease of learning and usage, python codes can be easily written and executed much faster than other programming languages.

# About Jupyter Notebooks



## Simple spectral analysis

An illustration of the Discrete Fourier Transform

$$X_k = \sum_{n=0}^{N-1} x_n exp^{\frac{-2\pi i}{N}kn} \quad k = 0,\dots,N-1$$

```
In [2]: from scipy.io import wavfile
        rate, x = wavfile.read('test_mono.wav')
```

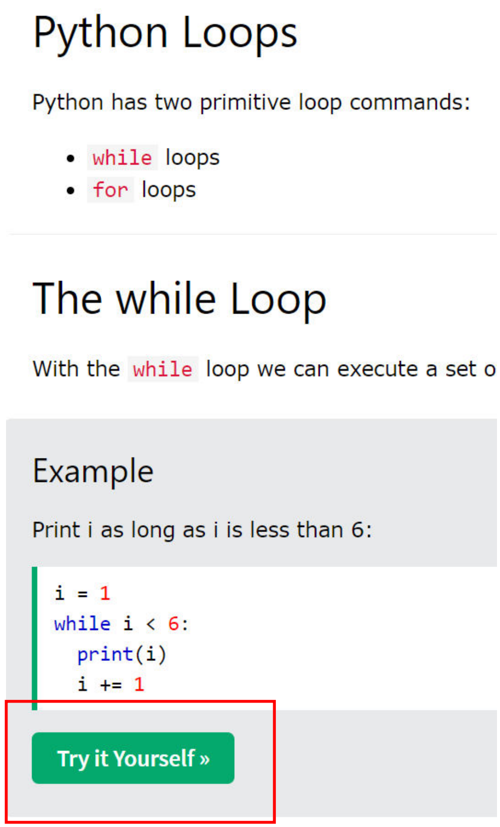
And we can easily view it's spectral structure using matplotlib's builtin specgram routine:

```
In [5]: fig, (ax1, ax2) = plt.subplots(1,2,figsize(16,5))
        ax1.plot(x); ax1.set_title('Raw audio signal')
        ax2.specgram(x); ax2.set_title('Spectrogram');
```

Easy to code & Easy to read

# How to learn Python?

- Python Tutorial



## Python Loops

Python has two primitive loop commands:

- `while` loops
- `for` loops

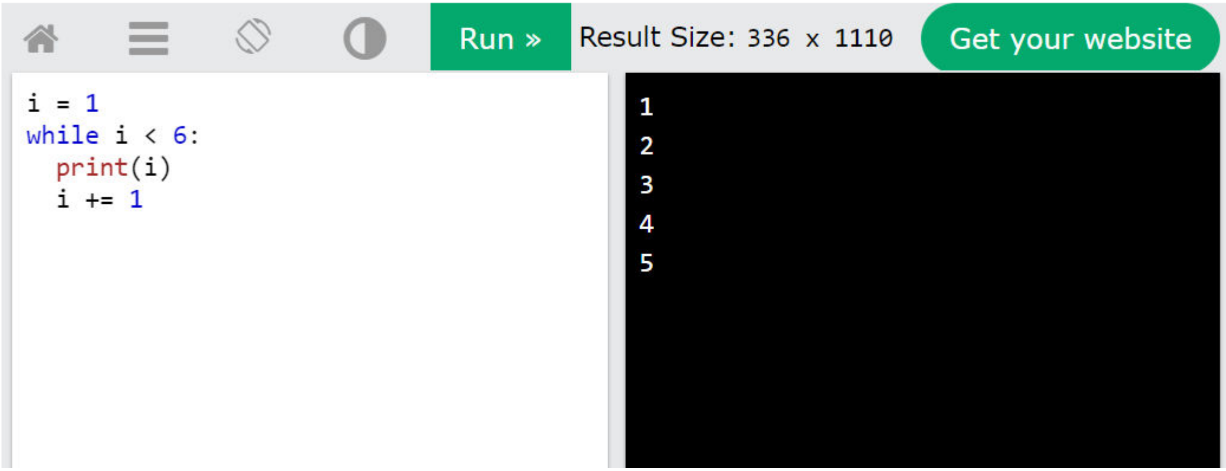
## The while Loop

With the `while` loop we can execute a set o

### Example

Print i as long as i is less than 6:

```
i = 1
while i < 6:
  print(i)
  i += 1
```



```
i = 1
while i < 6:
  print(i)
  i += 1
```

```
1
2
3
4
5
```

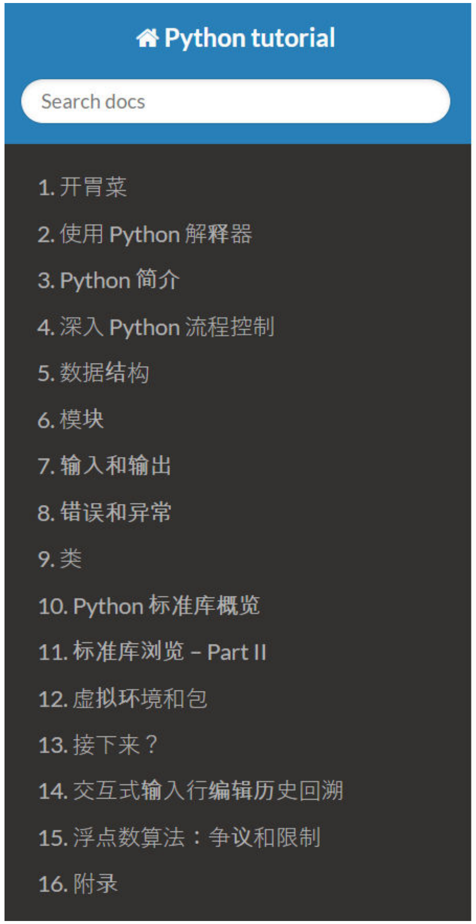
Try as you learn.

# How to learn Python?

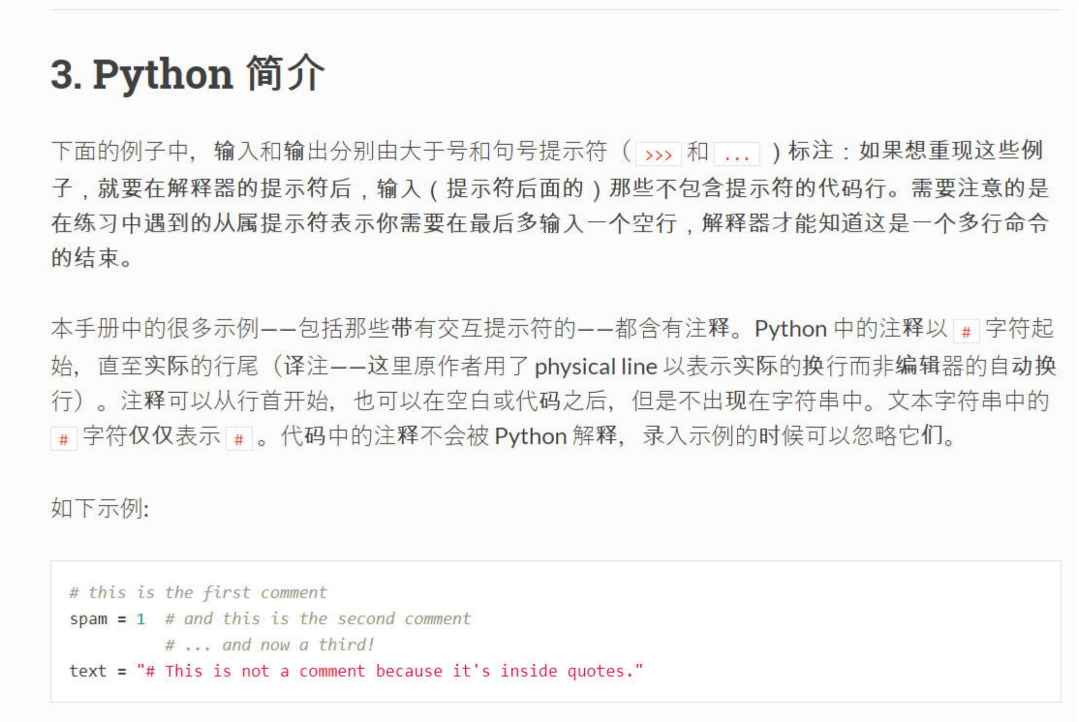
- 最详尽使用指南：超快上手Jupyter Notebook，https://zhuanlan.zhihu.com/p/32320214
- How to Use Jupyter Notebook in 2020: A Beginner’s Tutorial，https://www.dataquest.io/blog/jupyter-notebook-tutorial/
- Chapter 1.0 Getting Started in https://github.com/jckantor/CBE30338

# How to learn Python?

- **Python 入门指南 (in Chinese)**



1. 开胃菜
2. 使用 Python 解释器
3. Python 简介
4. 深入 Python 流程控制
5. 数据结构
6. 模块
7. 输入和输出
8. 错误和异常
9. 类
10. Python 标准库概览
11. 标准库浏览 – Part II
12. 虚拟环境和包
13. 接下来？
14. 交互式输入行编辑历史回溯
15. 浮点数算法：争议和限制
16. 附录



## 3. Python 简介

下面的例子中，输入和输出分别由大于号和句号提示符（ `>>>` 和 `...` ）标注：如果想重现这些例子，就要在解释器的提示符后，输入（提示符后面的）那些不包含提示符的代码行。需要注意的是在练习中遇到的从属提示符表示你需要在最后多输入一个空行，解释器才能知道这是一个多行命令的结束。

本手册中的很多示例——包括那些带有交互提示符的——都含有注释。Python 中的注释以 `#` 字符起始，直至实际的行尾（译注——这里原作者用了 physical line 以表示实际的换行而非编辑器的自动换行）。注释可以从行首开始，也可以在空白或代码之后，但是不出现在字符串中。文本字符串中的 `#` 字符仅仅表示 `#` 。代码中的注释不会被 Python 解释，录入示例的时候可以忽略它们。

如下示例:

```
# this is the first comment
spam = 1  # and this is the second comment
          # ... and now a third!
text = "# This is not a comment because it's inside quotes."
```

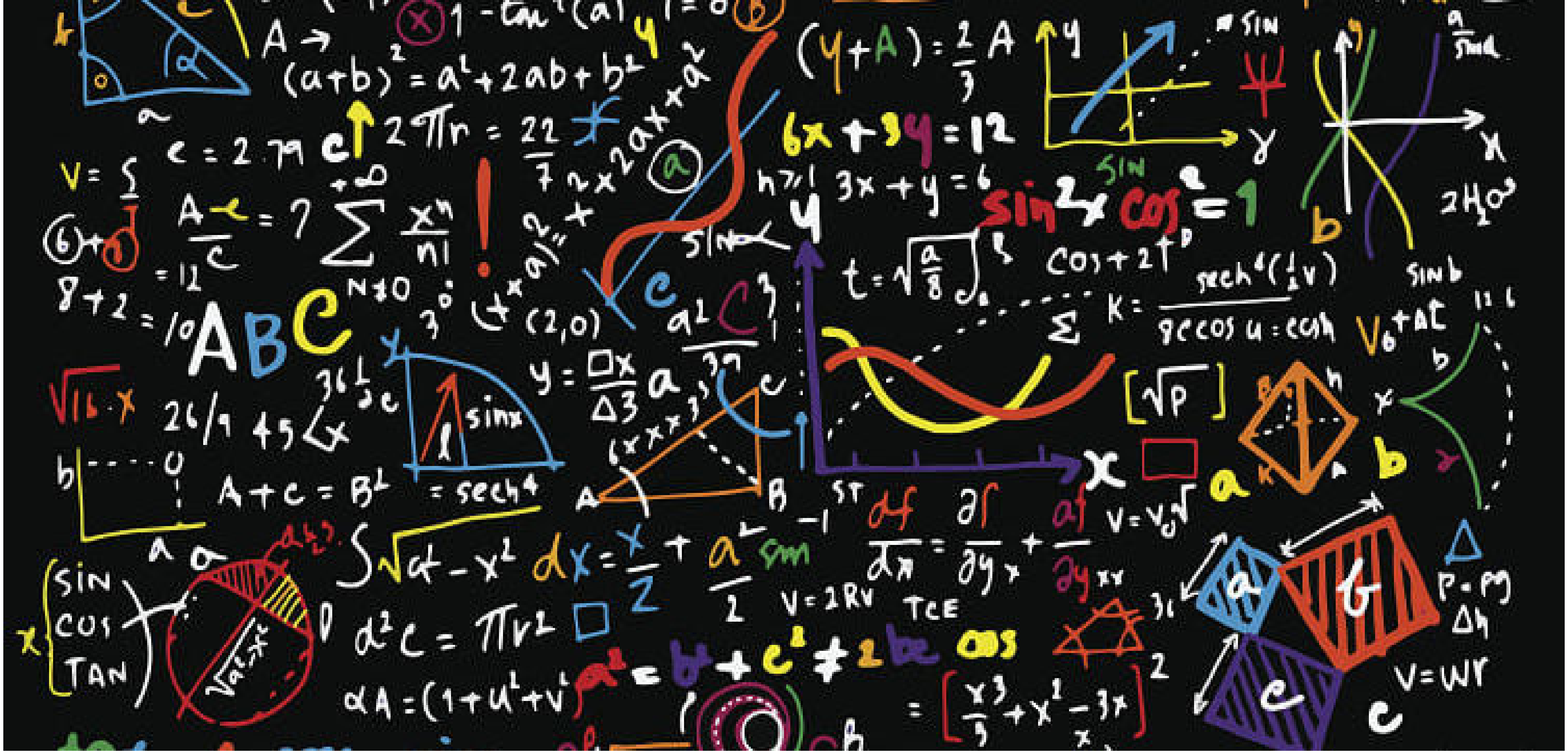
Comprehensive and detailed.

# Important notes

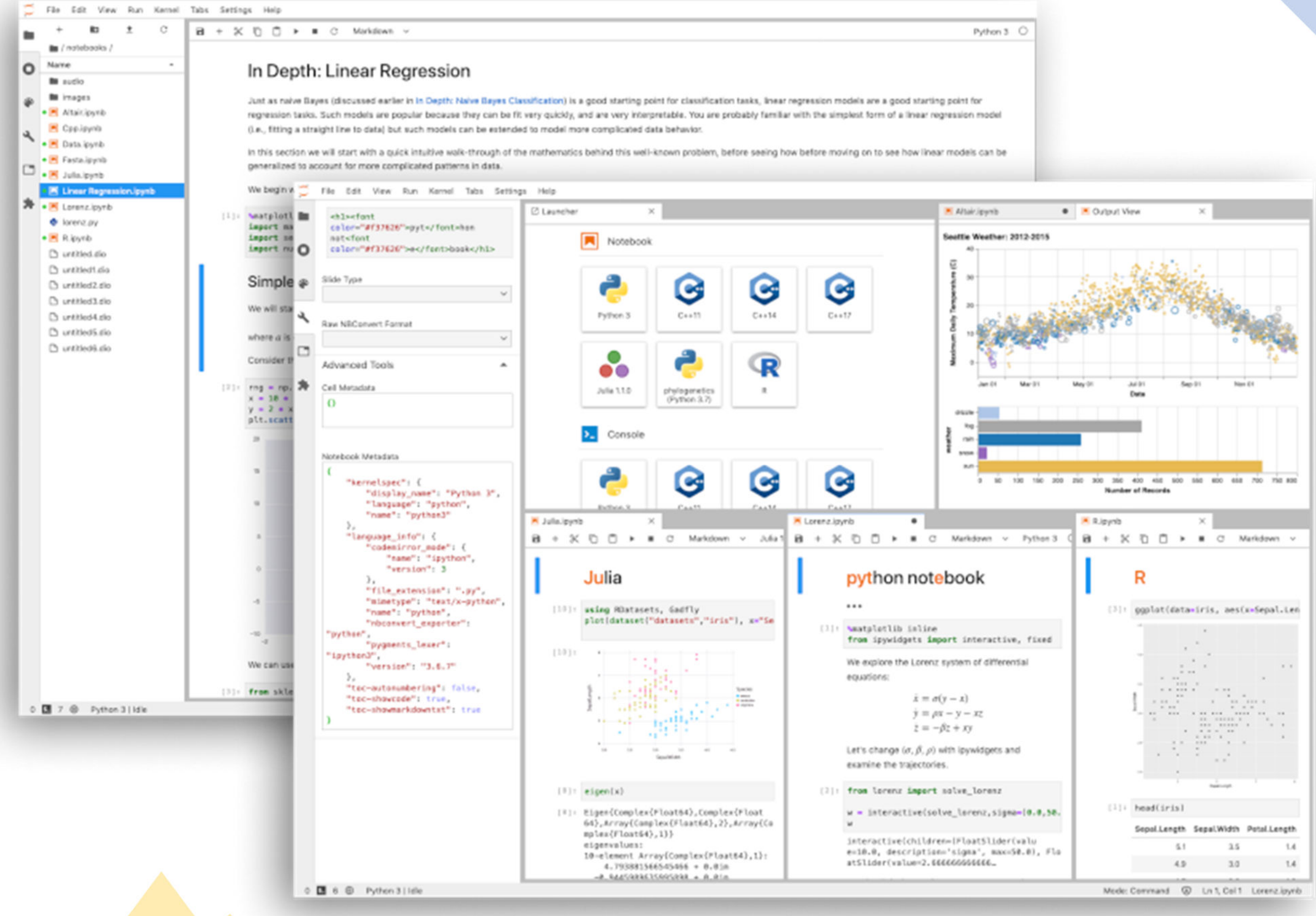
1. We not only obtain credits, but (more importantly) also learn through experiments.

2. Python, Jupyter notebook, and many more…

3. Not comfortable with the experiments? Do let me know.



# Experiments in Probability?



# In Depth: Linear Regression

Just as naive Bayes (discussed earlier in In Depth: Naive Bayes Classification) is a good starting point for classification tasks, linear regression models are a good starting point for regression tasks. Such models are popular because they can be fit very quickly, and are very interpretable. You are probably familiar with the simplest form of a linear regression model (i.e., fitting a straight line to data) but such models can be extended to model more complicated data behavior.

In this section we will start with a quick intuitive walk-through of the mathematics behind this well-known problem, before seeing how before moving on to see how linear models can be generalized to account for more complicated patterns in data.

We begin w

```
[1]: %matplotl
     import ma
     import se
     import nu
```

## Simple

We will sta

where a is

Consider th

```
[2]: rng = np.
     x = 10 *
     y = 2 * x
     plt.scatt
```

We can use

```
[3]: from skle
```

```
<h1><font
color="#f37626">pyt</font>hon
not<font
color="#f37626">e</font>book</h1>
```

```
{}
```

```
{
    "kernelspec": {
        "display_name": "Python 3",
        "language": "python",
        "name": "python3"
    },
    "language_info": {
        "codemirror_mode": {
            "name": "ipython",
            "version": 3
        },
        "file_extension": ".py",
        "mimetype": "text/x-python",
        "name": "python",
        "nbconvert_exporter":
"python",
        "pygments_lexer":
"ipython3",
        "version": "3.6.7"
    },
    "toc-autonumbering": false,
    "toc-showcode": true,
    "toc-showmarkdowntxt": true
}
```

# Julia

```
[10]: using RDatasets, Gadfly
      plot(dataset("datasets","iris"), x="Se
```

```
[8]: eigen(x)
```

```
[8]: Eigen{Complex{Float64},Complex{Float
64},Array{Complex{Float64},2},Array{Co
mplex{Float64},1}}
eigenvalues:
10-element Array{Complex{Float64},1}:
    4.793881566545466 + 0.0im
   -0.9445998635995098 + 0.0im
```

# python notebook

```
[1]: %matplotlib inline
     from ipywidgets import interactive, fixed
```

We explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

Let's change $(\sigma, \beta, \rho)$ with ipywidgets and examine the trajectories.

```
[2]: from lorenz import solve_lorenz

     w = interactive(solve_lorenz,sigma=(0.0,50.
     w
```

```
interactive(children=(FloatSlider(valu
e=10.0, description='sigma', max=50.0), Flo
atSlider(value=2.6666666666666…
```

# R

```
[3]: ggplot(data=iris, aes(x=Sepal.Len
```

```
[1]: head(iris)
```

| Sepal.Length | Sepal.Width | Petal.Length |
|---|---|---|
| 5.1 | 3.5 | 1.4 |
| 4.9 | 3.0 | 1.4 |

# How to get FULL credit?

- **Run** all Jupyter notebook without error.
- **Finish** some detailed experimental requirement.
- **Extent** the experiment based on your own ideas.
- **Write** the experimental report carefully.

# Submission of Report

- **Submit**:
  - Jupyter Notebook, Experiment X_Homework.ipynb;
  - Experimental Report Experiment X_Report.docx.

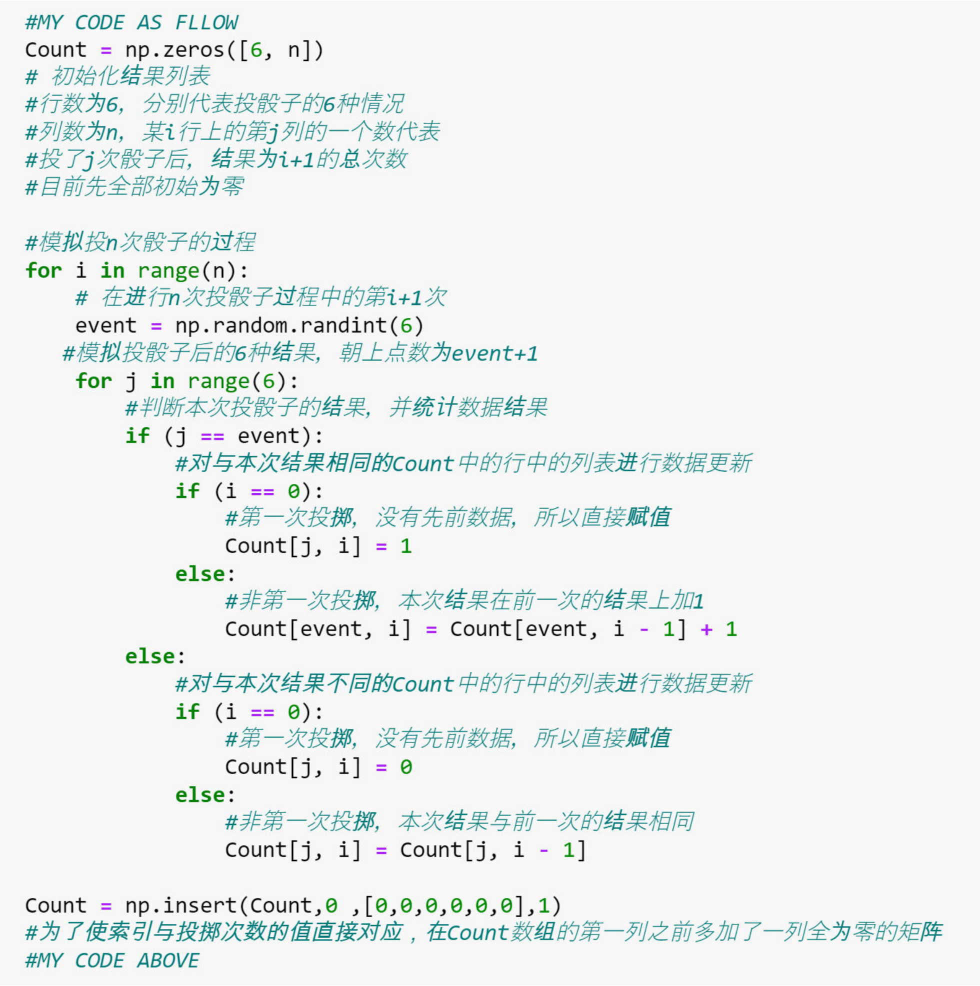
- **Title** of your email: Name-Student Number-Experiment X

- **Email** your experimental materials to:

  2210433032@email.szu.edu.cn

- **Deadline** of Experiment 1: **30-Sep-2022**

# How to write your code?



```
#MY CODE AS FLLOW
Count = np.zeros([6, n])
# 初始化结果列表
#行数为6, 分别代表投骰子的6种情况
#列数为n, 某i行上的第j列的一个数代表
#投了j次骰子后, 结果为i+1的总次数
#目前先全部初始为零

#模拟投n次骰子的过程
for i in range(n):
    # 在进行n次投骰子过程中的第i+1次
    event = np.random.randint(6)
   #模拟投骰子后的6种结果, 朝上点数为event+1
    for j in range(6):
        #判断本次投骰子的结果, 并统计数据结果
        if (j == event):
            #对与本次结果相同的Count中的行中的列表进行数据更新
            if (i == 0):
                #第一次投掷, 没有先前数据, 所以直接赋值
                Count[j, i] = 1
            else:
                #非第一次投掷, 本次结果在前一次的结果上加1
                Count[event, i] = Count[event, i - 1] + 1
        else:
            #对与本次结果不同的Count中的行中的列表进行数据更新
            if (i == 0):
                #第一次投掷, 没有先前数据, 所以直接赋值
                Count[j, i] = 0
            else:
                #非第一次投掷, 本次结果与前一次的结果相同
                Count[j, i] = Count[j, i - 1]

Count = np.insert(Count,0 ,[0,0,0,0,0,0],1)
#为了使索引与投掷次数的值直接对应，在Count数组的第一列之前多加了一列全为零的矩阵
#MY CODE ABOVE
```

Code sample from last year.

- Good practice in programming.

1. Code indentation
2. Meaningful naming
3. Comments that add context

- 6 codings best practices for beginner programmers

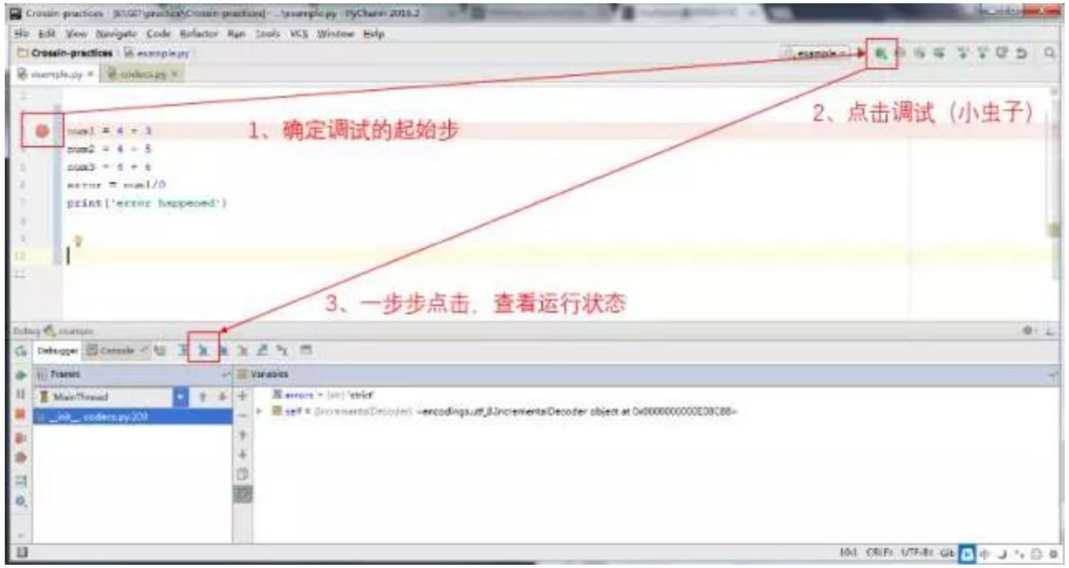
# How to debug?

- Win10下安装Anaconda+CUDA+TensorFlow+PyTorch+PyCharm [https://zhuanlan.zhihu.com/p/59031211](https://zhuanlan.zhihu.com/p/59031211)

- PyCharm 快速上手指南 [https://zhuanlan.zhihu.com/p/26066151](https://zhuanlan.zhihu.com/p/26066151)

**4、调试**

强大的 **Pycharm** 为我们提供了方便易用的断点调试功能，步骤如下图所示：



```
num1 = 4 + 3
num2 = 4 + 5
num3 = 5 + 6
error = num1/0
print('error happened')
```

1、确定调试的起始步

2、点击调试（小虫子）

3、一步步点击，查看运行状态

简单介绍一下调试栏的几个重要的按钮作用：