Py to exe 转换代码： 在终端输入 pyinstaller -F --onefile \*\*\*\*.py 可以吧\*\*\*\*.py文件转成 \*\*\*\*.exe

对于Python开发用户来讲，PIP安装软件包是家常便饭。但国外的源下载速度实在太慢，浪费时间。而且经常出现下载后安装出错问题。所以把PIP安装源替换成国内镜像，可以大幅提升下载速度，还可以提高安装成功率。

国内源：

新版ubuntu要求使用https源，要注意。

清华：https://pypi.tuna.tsinghua.edu.cn/simple

阿里云：http://mirrors.aliyun.com/pypi/simple/

中国科技大学 https://pypi.mirrors.ustc.edu.cn/simple/

华中理工大学：http://pypi.hustunique.com/

山东理工大学：http://pypi.sdutlinux.org/

豆瓣：http://pypi.douban.com/simple/

临时使用：

可以在使用pip的时候加参数-i https://pypi.tuna.tsinghua.edu.cn/simple

例如：pip install -i https://pypi.tuna.tsinghua.edu.cn/simple pyspider，这样就会从清华这边的镜像去安装pyspider库。

今天的知识点：

print()

数据类型

变量

input()

运算符

>>> 是提示符，提示你在这里输入 Python 代码。

如果要退出命令行，可以使用快捷键 Ctrl + D，或者输入 exit()。

输入 1 + 1 试试，按下回车，可以看到 Python 自动帮你算出了结果。

print —— 打印

print('Hello World!') —— 打印 Hello World!

print() 就是你学到的第一个函数 —— 「打印函数」（函数可理解为方法），它可以帮你输出任何内容 —— 只要放进 括号 内。

那么'Hello World'为什么要用 引号 包起来呢？

用引号包起来的文字，被称为「字符串」，等于在告诉计算机：请把引号里的内容打印出来，不用管里面是什么。

观察下面的两行代码：

print(1 + 1)

copy

print('1 + 1')

copy

用前面学到的知识，猜一下这两行代码分别会打印什么。

然后在终端中验证一下：

图片描述

可以看到，前者打印出了 1+1 的结果；而加了引号后，原封不动地打印出了「 1 + 1 」。

⚠ ️注意：引号可以是单引号，也可以是双引号，但必须是英文符号，且前后一致。

试试打印一行诗句：

🔥 小练习： 请打印出 床前明月光，疑是地上霜

如果想打印多行文字怎么办呢？

可以使用三个引号，比如：

print('''

我去买几个橘子

你就站在此地不要走动

''')

copy

💡 int - 整数

即数学意义上的整数，可以是正整数或负整数，不带小数点。

💡 float - 浮点数

浮点数就是小数。所谓浮点，指的是小数的表示形式多变，3.14 可以表示为 0.314 x 10，小数点可以改变位置，故称为浮点数。

在计算机中，整数、浮点数字、字符串等所有的数据，都会被转换为二进制数值进行处理，处理完成后再转换为你想要的信息。转换的过程中可能就会出现精度的丢失，试试在 Python 命令行中输出 1.1 + 2.2 ，猜猜结果是多少？

图片描述

结果并不是 3.3，而是 3.3000000000000003。

💡 布尔值

布尔值只有 True 和 False 两种，注意大小写。布尔值通常用来比较和判断，举例如下：

图片描述

💡 空值 None

None 在 Python 中是一个特殊的值，表示“无”。注意它与数值 0 不同，0 是一个整数，而 无 表示 什么都没有。

我们再来学习一个很重要的概念 —— 「变量」。你可以将它理解为一个盒子 —— 把常用的数据放在里面，起一个好记的名字，下次用到时可以快速找到它。

比如说在游戏中，你会频繁用到 「血量（HP）、攻击力、防御力」等数据，这些数据需要用名字储存起来。

图片描述

定义这些变量的代码如下：

Level = 1

ATK = 5

HP = 100

copy

左边是「变量名」

右边是「数据」

而等号在Python里的意思是「赋值符」，代表把右边的数据赋值给左边。

图片描述

其实，早在小学的数学课上，你就开始使用 变量 的概念了。比如「鸡兔同笼」问题中，你可以“设鸡的数量为 x，兔子的数量为 y ”， x、y 就是变量，：

x = 鸡的数量

y = 兔子的数量

则：

x + y = 头数

2x + 4y = 脚数

变量中的数据，除了可以是整数，还可以是一段文字，比如：

word = 'Hello World!'

print(word)

copy

试一下上面的代码，看看会打印出什么结果。

变量的右边，甚至可以是另一个变量，如：

word2 = word

print(word2)

copy

上面这段代码的意思是把「word」中的数据，赋值给「word2」这一变量。

亲手操作一下试试：

图片描述

🔥 小练习：把 「实小楼」 存入名为的「name」的变量中，并打印出来。

💡 提示：「实小楼」是字符串，需要加引号

🦁️️：️“对了，还不知道你多大了，请用 Python 告诉我吧～”

输入下面的代码：

age = input('My age is :')

copy

执行代码后，弹出「My age is :」，在后面输入你的年龄，再按回车。

这时，你的年龄就被存入了「age」这个变量中，print 一下试试：

print(age)

copy

终端随之打印出了你的年龄。

图片描述

这就是我们今天学的第二个函数 —— input()，主要作用是让用户输入某个内容并接收它 —— 比如注册时需要填写账号密码，你就可以用 input() 搞定：

ID = input('请输入你的ID：')

copy

用户填写的信息就会被存入「ID」这个变量中。

（ ⚠注意️：如果input()函数执行后，你没有在后面输入内容，程序会一直卡住，等待你输入信息。）

任何一本编程书籍之中，字符串的内容总是很长。这是因为：处理字符串是计算机程序中最普遍的需求之一 ，人和计算机交互，所用的就是字符串而不是二进制数字。

用 引号 包裹起来的数据就是字符串。计算机无法识别人类的文字，所以引号的作用就是告诉计算机：我是字符串，不用管里面是什么内容。

包裹字符串的引号必须前后对应，成对出现。如果字符串中还有引号，要特别注意，如 He said "Today's weather is good."，直接写 print('He said "Today's weather is good."') 就会报错。因为：

图片描述

正确的写法是，在 Today 后面加入 转义符 \ ：

图片描述

亲手在终端中试一下：

图片描述

转义符 \ 还有一个常用形式是和 n 连起来用， \n 代表换行符，就是用 Enter （回车）键敲出来的东西）。

图片描述

💡 索引

字符串是一堆字符的集合，可以通过位置编号，来获得对应的字符。如 'hello\_shiyanlou'，编号 0 对应第一个字符 h，1 对应第二个字符 e，依次类推。这个位置的编号，我们称之为：索引。

这里你又学到了计算机世界的一大概念：

在计算机世界中，计数从 0 开始。

在计算机世界中，计数从 0 开始。

在计算机世界中，计数从 0 开始。

索引除了是正数，也可以是 负值，表示倒数第几个。比如最后一个字符的索引为 -1，倒数第二个字符的索引为 -2。

简单的示例：

>>> string = 'hello\_shiyanlou'

>>> string[0] # 获取第一个字符

'h'

>>> string[1] # 获取第二个字符

'e'

>>> string[2]

'l'

>>> string[-1] # 获取倒数第一个字符

'u'

>>> string[-2] # 获取倒数第二个字符

'o'

copy

💡 format()

format() 是专门用来 格式化字符串 的函数，它最常用的功能就是「插入数据」和「数字格式化」。

1. 插入数据

先看下面的代码，相信现在的你已经能看懂是什么意思了。我们让用户输入姓名和年龄，并用 name 和 age 变量接收数据。

name = input('请输入姓名：')

age = input('请输入年龄')

copy

现在我想打印 “你叫xxx，今年x岁了” ，代码该怎么写？

有两种写法：

print('你叫'+name+'，今年' + age + '岁了')

copy

这种方法用 + 号串联打印的内容，虽然可行，但是比较麻烦，而且有局限性。

这时，你可以使用 format() 函数，实现插入效果，用法如下：

print('你叫{}，今年{}岁了'.format(name,age))

copy

{} 为占位符，我先把这个位置占住，具体数据在后面导入。

下图可以让你有更好的理解：

图片描述

2. 数字格式化

format() 的第二种常用功能，是格式化数字，比如我们要输出圆周率，但只保留两位小数，可以这么写：

print("{:.2f}".format(3.1415926))

copy

如果要保留三位，则改成 {:.3f}；不带小数，则改成 {:.0f}，依次类推。

format()功能还有很多，我们之后用到时在做学习。

注释，可理解为程序的注解、提示，用来帮助他人和自己阅读、理解、编写代码。python中，# 常被用作写单行注释，# 号后的内容会被忽略，不会被当作代码处理。

如：

# 注释可以写在开头：下面要打印“hello shiyanlou”

print("hello shiyanlou")

print("hello world") # 注释也可以写在后面：这里打印“hello world”

copy

注释 对程序员来说非常重要，甚至直接决定了你的寿命……写注释可以帮你理清思路，提高程序的可维护性；如果没有注释，你会连自己的代码都看不懂，而且极可能会被同事、上司打死……

另外，如果有一些代码暂时不需要被执行，可以先不要删掉它（因为后面可能还会使用），而是使用 # 将其注释掉。这样代码就暂时不会被执行了，之后再需要这段代码时，删掉 # 就好了。

运算，是计算机最基本的功能，我们先来学习一下运算符。

运算符用于执行程序的运算，我们这里先学习最简单的几种：「算术运算符」、「比较运算符」、「赋值运算符」、「逻辑运算符」。

不要被这些名词吓到，其实大多都是些你小学就学过的概念……

💡 算术运算符

即小学数学中的运算符，最常用的还是「加减乘除余」这五种：

运算符 名称 描述

+ 加 两个对象相加

- 减 得到负数或是一个数减去另一个数

\* 乘 两个数相乘或是返回一个被重复若干次的字符串

/ 除 x 除以 y

% 取模 返回除法的余数

\*\* 幂 返回 x 的 y 次幂

// 取整除 返回商的整数部分（向下取整）

🔥 练习一下：

>>> a = 12

>>> b = 3

>>> c = 5

>>> a + b

15

>>> a - b

9

>>> a \* b

36

>>> a / b

4.0

>>> a / c

2.4

>>> a % c

2

比较运算符 就是小学课本中的「大于、小于、等于……」，通常用于程序执行 循环和判断 中：

运算符 描述

== 等于：比较对象是否相等

!= 不等于：比较两个对象是否不相等

> 大于：返回 x 是否大于 y

< 小于：返回 x 是否小于 y

>= 大于等于：返回 x 是否大于等于 y

<= 小于等于：返回 x 是否小于等于 y

⚠️ 注意： 在 Python 和大多数高级语言中：一个等号代表赋值符，两个等号才代表数学意义上的“相等”。

🔥 练习一下：

>>> a = 1

>>> b = 2

>>> c = 3

>>> d = 2

>>> a == b

False

>>> a != b

True

>>> a > b

False

>>> a < b

True

>>> c >= d

True

>>> b <= c

True

copy

赋值运算符 主要是为了简写，如果看不懂可以先跳过，在以后的项目中会更容易理解。

运算符 描述 实例

= 赋值运算符 c = a + b ：将 a + b 的运算结果赋值为 c

+= 加法赋值运算符 c += a ：等效于 c = c + a

-= 减法赋值运算符 c -= a ：等效于 c = c - a

\*= 乘法赋值运算符 c \*= a ：等效于 c = c \* a

/= 除法赋值运算符 c /= a ：等效于 c = c / a

%= 取模赋值运算符 c %= a ：等效于 c = c % a

\*\*= 幂赋值运算符 c \*\*= a ：等效于 c = c \*\* a

//= 取整除赋值运算符 c //= a ：等效于 c = c // a

🔥 练习一下：

>>> a = 3

>>> b = 12

>>> c = 10

>>> c += a

>>> print(c)

13

>>> b -= a

>>> b

9

>>> c \*= 5

>>> c

65

如果计算机只有计算功能，那它顶多是一个计算器；而我们今天使用的计算机之所以智能、可编程，最重要的原因是：它能够进行 逻辑处理。

逻辑运算符，即高中数学中的「 且、或、非 」，Python 中用 and、or、not 表示这三种逻辑。

运算符 逻辑表达式 结果

and（逻辑与） a and b 当 a、b 都为 True 时，返回 True；否则返回 False

or（逻辑或） a or b 当 a、b 任意一个为 True，就返回 True

not（逻辑非） not a 当 a 为 True 时，返回 False；反之亦然

🔥 练习一下：

⚠️ 补充：在逻辑运算中，数字 0 代表 False，1 代表 True。

>>> not 1

False

>>> not 0

True

>>> 1 and 1

1

>>> 1 and 0

0

>>> 1 or 0

1

！！！2020-3-12

在开发过程中经常被使用到的 Python 程序，是一个或多个的脚本文件，通常会以 .py 作为文件后缀。当完成脚本的编写后，可以用 Python 解释器来运行 Python 脚本。

在实验环境中，你可以使用 vim、sublime 或 gedit 来编写 Python 程序，这里以 gedit 举例：

💡 创建和打开文件

你可以通过两种方式，创建和打开 Python 文件：

1. 命令行方式创建

在 Xfce 终端中输入：

$ gedit test.py

copy

这行命令的作用是：用 gedit 打开 test.py 文件；如果该文件不存在，则 创建 它并打开。

执行命令后，我们看到 gedit 弹了出来，而且文件名是 test.py

图片描述

2. 在 gedit 中创建

还可以先打开 gedit 程序，再点击菜单栏的 文件-新建，即可创建一个新的空白文件。

💡 保存文件

在 test.py 文件中输入以下的代码内容：

print('hello shiyanlou')

copy

输入完成后，按下 ctrl + s 或点击 保存 按钮，即可保存。

（如果你是在 gedit 中新建的文件，保存时请输入文件名 test.py，并注意存放路径是否在 home 目录）

图片描述

💡 执行 Python 文件

保存文件后，先关闭 gedit ，然后打开 Xfce 终端，直接使用 Python3 解释器执行 test.py ：

$ python3 test.py

copy

可以看到 hello shiyanlou 输出到了屏幕上。

下面，我们来学习计算机系统中的一大核心知识——流程控制。

「 流程控制 」是一个程序的关键所在。简单的说，「流程控制」就是告诉计算机什么时候该干什么事。比如一个游戏中，你要让计算机清晰地知道 —— 什么情况下会掉血，按下什么键代表攻击，过多久能复活……

学会流程控制后，你就可以用程序做很多事，比如：

写一个扣税计算器：

图片描述

写一个抽卡游戏：

图片描述

接下来，我们先来学习流程控制中的条件判断。

其实我们每天都生活在条件判断中：早上如果闹钟⏰响了，执行起床/懒床程序；如果到了12点，执行吃饭程序；如果到了6点，执行下班或放学……

如果有一天，你的女朋友对你说，“下班买2个西瓜回来”，你会这样执行：

图片描述

换成 Python 代码，买西瓜的程序还是一样的，只不过把 如果、否则 换成了 if、else 。

图片描述

挺容易理解的吧？其实就是用代码告诉计算机：什么条件下该做什么。

很多编程语言都会使用 if 、else 作为流程控制的关键字。 if 语句最简单的构成是这样的：

if SomethingHappen:

DoThisCode

copy

⚠️ 注意第 1 行末尾的冒号 : 和第 2 行的 4 个空格缩进，原因会在后面详解。

除此之外，Python 还加入了 elif，即 else if，方便执行更复杂的判断。

图片描述

如：买 2 个西瓜，4 个苹果，5 根香蕉

代码如下（示例，非标准代码） ：

让我们写一个比大小的游戏，进一步熟悉条件判断的语法：

在终端中输入 gedit compare.py，然后在打开的 gedit 中输入：

a = int(input("Please enter a number: "))

## 让用户输入一个数字，用变量 a 接受

if a > 10: ##如果 a > 10，则执行内部的代码，反之跳过

print('a > 10')

## 请特别注意语句块内的空格缩进，请使用 4 个空格缩进。

## 请特别注意语句块内的空格缩进，请使用 4 个空格缩进。

## 请特别注意语句块内的空格缩进，请使用 4 个空格缩进。

elif a == 10:

print('a == 10')

else:

print('a < 10')

⚠️ 注意：以下五点，都很重要

if、elif、else 后要加冒号 :，告诉计算机这是一个判断语句，如满足条件，就执行语句块内部的代码。

Python 使用严格的缩进，来区分代码块的执行级别。比如 if 下面的代码要缩进四个空格，代表的是 if 内部的功能；else下面的代码同样要缩进四个空格，代表是else内部的功能。

推荐使用 4 个空格缩进；也可以在编辑器内，把 Tab 键改为四个空格，用 Tab 键缩进。

用 input() 接受到的用户输入，类型为字符串 string，但字符串不能和整数比较大小，所以要用 int() 函数将数据转换为 int 整数类型。

使用 int() 函数，可以把括号里的内容转换为整数，但要注意：只有数字或纯数字的字符串才能转换，如'a'、'abc'这样的内容就不行；float 类型的数字会只保留整数部分。

图片描述

保存文件，关闭 gedit 后，在终端中输入 python3 compare.py ，运行刚刚的程序。

图片描述

输入数字后，屏幕上打印出了比较结果，第一个条件控制程序成功！

💡 在一个判断中，还可以嵌套另一套判断，比如当男人喜欢上一个人时：

if 那个人是女人:

if 单身:

你可以追求她

else:

放弃

else：

成为基友

copy

💡 还可以使用 and 和 or 进行综合控制：

if 那个人是女人 and 单身: # A 和 B必须同时满足才能执行

你可以追求她

else：

成为基友

copy

又比如：你今天不想吃饭，除非吃火锅或者烧烤：

if 火锅 or 烧烤: # A 或 B 满足一个即可执行

吃饭

else：

不吃饭

copy

在程序开发过程中，当遇到一些代码暂时不写(等到后面写)，又不想程序在执行时报错的情况，可以使用 pass 关键字，程序执行遇到 pass 就会跳过这里的代码块继续执行后面的代码：

>>> a = 3

>>> if a<1:

... print("a<1")

... else:

... pass

...

>>> #程序没有报错

对人类来说，「循环」糟糕透了，它往往意味着重复和枯燥。但大多数人都在循环中工作和生活，比如收集用户信息、下载 500 张图片、上 12306 抢春运的火车票、给 20 个亲戚解释你不是修电脑的……想想就如噩梦一般，相信没有人想过这种流水线的生活。

但有了计算机，一切都大不一样了：它任劳任怨、不会抱怨，而且工作速度比人快成千上万倍。你只需要制订好一些规则，然后在旁边泡杯茶，看云展云舒即可。

Python 中包含两种循环方式，一种是 for 循环，一种是 while 循环。

for 循环 主要用于，对一个范围内的每个元素，进行指定操作。语法如下：

图片描述

比如在一个班级里，点名册就是一份列表，每个姓名都是一个元素。现在你是班长，每天上课前的点名太费劲了，你需要一个点名程序。

用for 循环，2 行代码即可实现该功能：

namelist = ['Sophia','Emma','Olivia','Ava','Mia','Isabella','Zoe','Lily','Emily','Madison','Jackson','Aiden','Liam','Lucas','Noah','Mason','Ethan','Caden','Logan','Jacob']

##点名册

for i in namelist:

print(i)

copy

图片描述

在环境中练习一下：

💡 提示： namelist 可以通过右侧工具栏中的「剪切板」粘贴进去；print() 前同样要注意 4 个空格的缩进。

图片描述

💡 range() 函数

如果需要处理一组数字列表，并且数字列表满足一定的规律，可以使用 Python 的内置函数 range()（范围）。

使用 range(x) 函数，可以生成一个从 0 到 x-1 的整数序列，比如我们想打印 0 到 9 的数，可以这样写：

for a in range(10):

print(a)

copy

此处输入图片的描述

还可以用 range(a,b) 取某个区间的数，比如要打印 1 到 10 ,你可以写：

for a in range(1,11):

print(a)

copy

⚠️ 注意： range(a,b) 包头不包尾，尾数要 + 1 。

图片描述

🔥 小练习：帮依萍写日记

图片描述

书桓走后，可怜的依萍每天只能对着日记，遥寄思念。现在，请你用 for 循环帮依萍写出“书桓走的第 n 天，想他”，一直写到第 10 天。

💡 提示：

range(a,b)

for ... in ...

print('书桓走的第 {} 天，想他'.format(i))

如果忘了 .format() 的用法，可以在上节实验文档中查阅一下

另外一种循环是 while 循环，它的语法和用途都和 for 循环 有些不同。

for 循环 适用于 已知循环次数 的循环，所以后面跟的是次数或区间，到达指定次数就停止。

但现实生活中，我们往往不知道要循环多少次，才能达成目标。比如“愚公移山”，愚公不知道门前有多少座山，但只要门前有山，就要盘它，这种情况我们要怎么写？

while 循环 可以帮你解决这个问题。while 后面跟的是一个条件，只要条件满足，这个循环就会一直进行下去。

具体语法如下：

图片描述

有了 while 循环 ，愚公可以这样移山：

while 山还在:

盘它！

copy

🔥 小练习： 下面，我们来用 while 循环，打印 1 到 10 的数字，亲手感受一下 while 循环有哪些不同:

a = 1

while a <= 10: ##同样注意冒号和语句块缩进

print(a)

a = a + 1

copy

图片描述

这个程序是这样运转的：

图片描述

一开始，a 的预设值为 1。

来到第二行的 while 循环，此时 a <= 10，满足条件，进入循环。

进入第三行，执行循环中的命令，打印 a 的值。

进入第四行，把 a 的值加 1 ，a 的值变成了 2，至此一轮循环结束。每轮循环结束后，程序会跳转到循环开头（即图中的第二行），再次判断是否满足循环的条件，如满足则再次执行循环。在这里，a = 2 ，2 <= 10，满足条件，则再次进入循环……

……

直到 a > 10 为止，循环条件不再满足，循环停止。

再总结一下两种循环的区别：

图片描述

💡break

在循环中，我们可以使用 break 和 continue 这两个关键字，来进一步控制流程。

break 表示停止当前循环，如：

for a in range(10):

if a == 5:

break

print(a)

copy

程序会打印 0 到 4 之间的数字。

💡continue

continue 表示跳过当前循环轮次，去执行下一轮循环。

比如这次我们想打印 1 到 10 的数，但不想打印 5 ，代码可以这样写：

a = 0

while a < 10:

a = a + 1

if a == 5:

continue

print(a)

copy

图片描述

至此，流程控制 就暂时介绍完了。流程控制 在编程中非常重要，请下来多多练习和总结，后面还有一个挑战等待你完成～

2020-3-14!!!

挑战：逢 7 就跳过

介绍

不知道你有没有玩过“逢 7 过”的游戏，游戏规则很简单：

几个人轮流报数，凡遇到 7 的倍数，或含 7 的数字就要跳过，否则就算失败。

今天我们就用 Python 来打印 1 到 100 之间，所有满足条件的数字。

图片描述

目标

请在 /home/shiyanlou 目录下创建 jump7.py 文件（请确保目录正确，否则无法通过挑战）

在文件中写入代码，打印 1 到 100 之间（包含 1 和 100），不是 7 的倍数、且不含 7 的数字，每行打印一个数字。

完成挑战后，请把文件复制到 /home/shiyanlou/Code 目录内，然后点击右侧工具栏的「 下载代码 」，把代码下载到本地， 我们在下个挑战中还会用到这段代码。

知识点

Python 开发环境

int 类型

变量

运算符

while循环

条件判断

提示：

⚠️ 前方高能，有剧透风险，喜欢挑战的同学请先独立思考，然后借着百度、谷歌来解出题目。在实际的开发工作中，大部分工作也是靠着 经验 + 思考 + 搜索引擎 来完成的，你可以提前熟悉这个感觉。

💡（一）

开发中，我们经常要先构思框架，再优化功能。

所以在这个挑战中，你可以先用循环，写出打印出 1 到 100 间的数字的程序，之后再往里面加条件判断。

💡（二）

7 的倍数，可以理解为除以 7 余数为 0 的数；

a % 7 == 0

copy

在 1 到 100 这个区间里：

个位上是 7 的数，即除以 10 余 7 的数；

a % 10 == 7

copy

十位上是 7 的数，即除以 10 取整数为 7 的数。

a // 10 == 7

copy

如果满足这些条件，就跳过；如不满足，则打印。可以用 elif 或 or 来加判定条件～

💡（三）

请保证文件名和存储路径都满足题目要求。

PS：还记得我们在第一节实验中，体验过的那个对战小游戏吗？其实你现在已经可以把那个游戏编写出来了！完成挑战后，不妨尝试一下～如果有什么问题，记得在群里和小伙伴们讨论一下。

你来了，快坐下，今天我们来讲讲编程界的两大神器： **Git** 和 **Github**。



**Git** 和 **Github** 都是程序员每天都要用到的东西 —— 前者是目前最先进的 **版本控制工具**，拥有最多的用户，且管理着地球上最庞大的代码仓库；而后者是全球最大  **代码托管平台、开源社区**。

如果你之前没接触过**「版本控制」**的概念，看到这里一定是一脸 xx 的，别急，看了这篇文章你一定能明白：

[**什么是版本控制**](http://blog.a0z.me/2014/05/21/GitBeginning/)

简单复述一下文章中的例子：

大四毕业生 **小张** 在写 **毕业论文**，他经常删删改改，有时还会后悔“昨天那个思路那么好，我怎么就给删了”……

有了多次教训后，他决定每次写之前都先复制一份，在复制的那份里修改，这么一来，文件夹里有了：

毕业论文\_初稿.doc

毕业论文\_修改1.doc

毕业论文\_修改2.doc

毕业论文\_修改3.doc

毕业论文\_完整版1.doc

毕业论文\_完整版2.doc

毕业论文\_完整版3.doc

毕业论文\_最终版1.doc

毕业论文\_最终版2.doc

毕业论文\_确定版1.doc

毕业论文\_确定版2.doc

……

小张想：“虽然很痛苦，但不至于丢掉以前的灵感了吧……等等，最终版和确定版哪个是昨天写的来着？？？”

同时，他还要把论文发给学霸女友求帮忙，第二天他的文件夹里又有了：

毕业论文\_最终版3.doc

毕业论文\_女友版1.doc

毕业论文\_女友版2.doc

几星期的煎熬下来，文件夹里多了几十份文件，小张的论文也快成型了，是时候把自己和女朋友的内容合并起来了。

这时又发生了一件喜闻乐见的事：**U 盘中病毒了**，而电脑里只有 1 个月前的版本……

如何拯救生无可恋的小张？其实，如果小张一早知道用「版本控制」工具就好了，他的文件可以整整齐齐地排列，就像这样：

图片出处

“哎呀，早知道能这样，就不用手动控制那么多版本啦！”

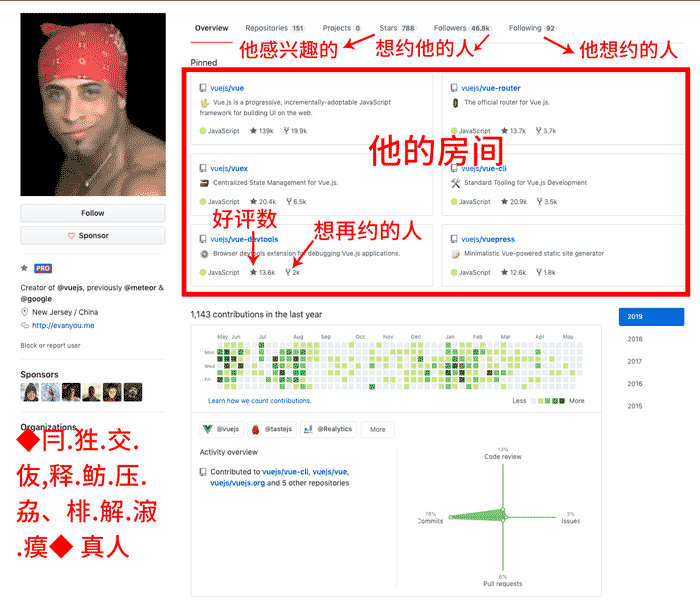
**但这还不够，如果能有一个支持「论文托管 + 论文版本控制」的网站就更好了。这样一来，小张不但能和女朋友合作编辑内容，还不用担心因电脑故障，导致之前论文版本的丢失。**

这时 —— **论文Hub** 出现了，它可以帮你托管论文，而且和版本控制工具无缝连接。

越来越多人发现了 **论文Hub** 的好处，相继把论文托管在论文 Hub 上 ，网站上的论文越来越多。一些优秀的作者还会把论文开源出来，让每个人都可以查阅、交流、学习……

慢慢的，**论文 Hub** 变成了全球最大的「交友社区」，并逐渐演化成了一种时尚 —— 找工作时，面试官会先问你有没有 **论文 hub** 的账号，有多少个赞、多少粉丝；而有优秀作品的人，会被大公司争抢录用……

这个 论文 hub，就是我们今天要学习的 **Github** ，只不过论文换成了程序代码。Github 大概长这样：



**在没有这两个工具时，编程可能是这样的：**

* 哪个同事修改了我的代码🔪我要杀了他
* 我把自己的代码改崩溃了🙂️我选择自杀
* 电脑崩溃、硬盘损坏、中毒，几万行代码找不到了😱

**但有了他们，一切都不一样了：**

* 同步代码到网络仓库，在家里写好代码上传，回到公司就可以继续写了，而且不怕丢失。
* 记录每次代码的修改，即使把程序写崩了，也能即使回溯到上一个版本。这在产品更新时也经常使用。
* 可以多人协作完成项目，每个人的提交都有清晰的记录。

在之后的学习中，你也会不断用到 **Git** 和 **Github**，把你完成的项目、学习记录，同步在 Github 的仓库中。这样做的结果是：你将有一份 **非常漂亮的 Github 主页**，能给你的简历加分很多。

接下来，我们将学习 Git 的基本操作，并注册 Github 账户，建立你的第一个代码仓库！

**知识点：**

* 版本控制
* Git 和 Github的历史
* 在 GitHub 创建仓库
* 添加修改到暂存区
* 提交代码
* 同步远程仓库