我的 python 入门之路

- 写本篇文章的目的是为了让想学 python 的读者能在本文中获取一些经验，能在初学 python 的时候少走一点弯路。
- 本文主要分两部分进行讲解。第一部分主要介绍 python ，让读者对 python 有初步的了解；第二部分则主要讲述本人学习 python 经历，让想学 python 的入门者少走一点弯路。

目录

- 我的 python 入门之路
  - 一. python 介绍
    - 1. python 简介
    - 2. python 的应用领域
    - 3. python 的未来发展
  - 二. 个人学习 python 的经历
    - 1. 学习 python 的原因
    - 2. 学习 python 中遇到的问题
  - 三. 总结

一. python 介绍

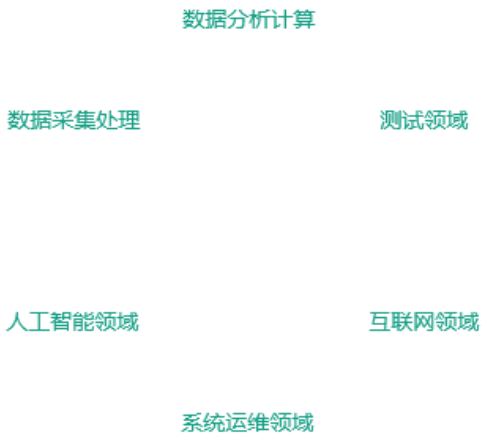
- 本节主要介绍 python 的有关内容。包括 python 简介，python 的应用，python 的未来发展。

1. python 简介

- Python 是一种跨平台的计算机程序设计语言。是一种面向对象的动态类型语言，最初被设计用于编写自动化脚本(shell)，随着版本的不断更新和语言新功能的添加，越来越多被用于独立的、大型项目的开发。

2. python 的应用领域

- Python是一种解释型脚本语言，可以应用于以下领域：



3. python 的未来发展

- 请查看文章： 未来5年，Python发展前景如何？哪个技术方向最吃香？

二. 个人学习 python 的经历

1. 学习 python 的原因

1). python 入门容易

- Python 是一门更接近自然语言的语言，或者说逻辑上更接近人的逻辑而不是机器的逻辑，这导致它虽然牺牲了一部分机器的效率但是却提高了人的编译效率。
- Python 代码十分的简洁，简洁到以前 C 里面要十几行代码完成的事情，到 Python 里面可能只需要几行。
- Python有十分多的内置函数。很多内容你都不需要自己去定义，直接调用函数就好了。
  参考：知乎 为什么很多人都说 Python 简单?

2). python 功能强大

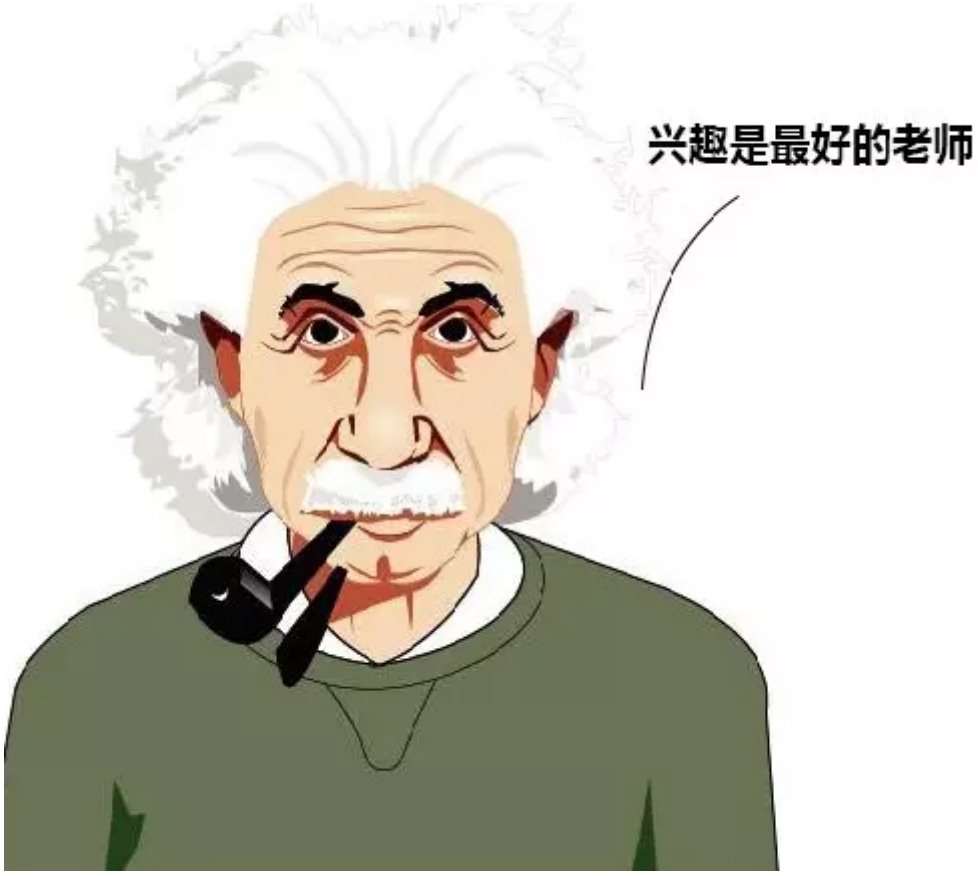
- python 现在的应用领域有：Web ， Internet开发，科学计算和统计，人工智能，教育，桌面界面开发，软件开发，后端开发。几乎我感兴趣的领域都有涉及。

3). 未来前景好

- 从上面提到的 python 的未来发展 中可以看出，学好 python 在以后会有非常好的发展。

4). 个人兴趣

- 学习一样东西最重要的一点是看你感不感兴趣。如果你自己都不感兴趣，那么学起来将会非常乏味。在学习 python 的时候，我对 python 中的 AI 应用以及 GUI 很感兴趣，所以我才愿意整天花时间去学习 python。



2. 学习 python 中遇到的问题

- 在初学 python 的时候，主要有两个问题需要解决。一是如何写出符合 python 规范的代码。二是如何组织 python 的目录结构。
- 下面我将从这两方面进行讲解。

1). 编码规范

- 学习一门语言，首先必须学会它的编码规范。只有你按照这个规范去做事，你写的代码才会得到别人的认可，才能融入到 python 这个开发团队中去。在这里，我要感谢我的 python 入门老师张总。这是在他一步步的指导下，我才能写出符合 python 规范的代码。python 的编程规范主要以 pep8 规范为主，如果大家想具体了解，可以去官网查看：pep8 规范
- 接下来，我就以我的代码演变过程为例，讲述 python 的一些编码规范

A. 变量名规范

```
# 处理数字水印内部像素点
wmBack = imgWaterMark[:,:] > 0
imgWaterMark[wmBack] = 1
```

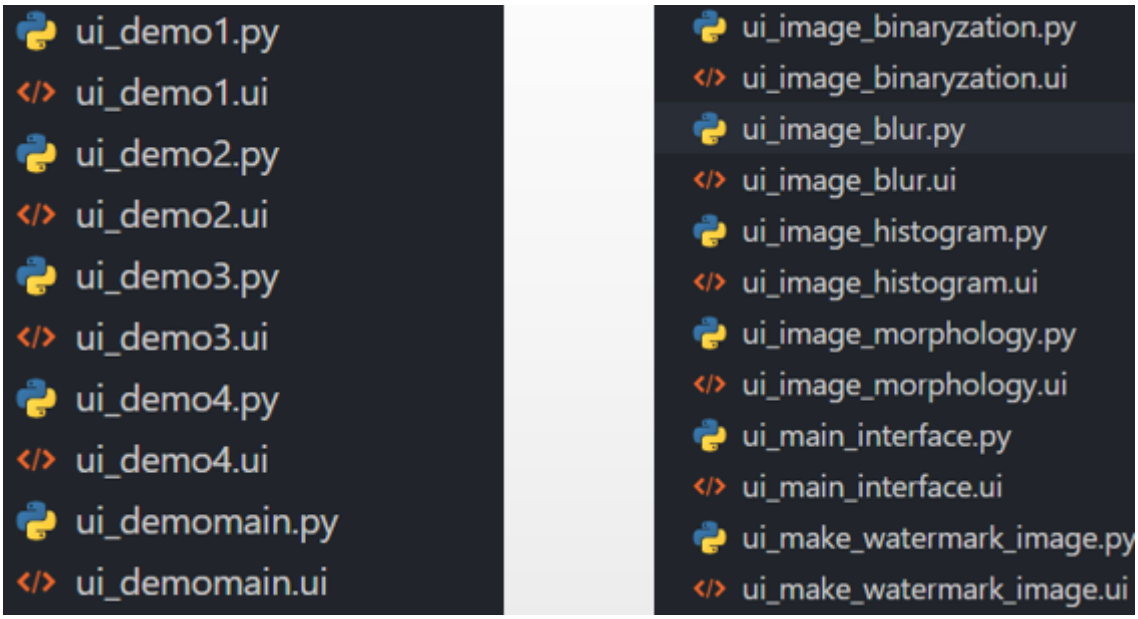
- 图中的变量名有两个地方的错误：
  1. 在 python 中变量名是小写字母、数字、下划线的组合。而图中使用的是小驼峰命名法。
  2. 在取变量名时，应该根据变量的作用去取名，而不是乱取。取名应该做到，当别人看到你的变量名时就能明白它的作用。

修改后的代码：

```
# 处理数字水印内部像素点
temp_watermark_image = watermark_image[:,:] > 0
watermark_image[temp_watermark_image] = 1
```

B. 模块名规范

- 在 python 中，模块是后缀名为 .py 的文件。



- 在 Python 中，模块名也是小写字母、数字、下划线的组合。在取模块名时，也要做到名如其意，不要像上图中的左图一样，随便取名。

C. docstring

- 简单来说，docstring 就是出现在模块、函数、类、方法里第一个语句的。
- 在写 python 程序时，一定要写 docstring，通过它别人能快速了解模块、函数、类、方法的使用，能节约大量的时间成本。
- 由于我的 docstring 写的也不是很好，这里就不贴出来了。大家可以查看这篇教程：Python Docstring 风格和写法学习 去了解如何写 docstring 。

D. 注释

- Python 中的注释有单行注释和多行注释。Python 中单行注释以 # 开头；多行注释用三个单引号 ''' 或者三个双引号 """ 将注释括起来。
- 注释注意事项
  - 注释不是越多越好。对于一目了然的代码，不需要添加注释。当然对于一个初学者来说，注释越多越好，在你写注释的时候，你就会更加深入的去理解这段代码的作用。
  - 对于复杂的操作，应该在操作开始前写上相应的注释。
  - 对于不是一目了然的代码，应该在代码之后添加注释。
  - 绝对不要描述代码。一般阅读代码的人都了解Python的语法，只是不知道代码要干什么。
- 最后附上一张我的注释演变图片

```python
# 图片文件信息初始化
self.__image = np.zeros((5), np.uint8)
self.__deal_img = np.zeros((5), np.uint8)
self.__key = []
self.__filename = ""
self.__filenamedir = ""
self.__load_img_h = 0
self.__load_img_w = 0
self.__load_img_shape = 0
```

```python
# 图片文件信息初始化
self.__preprocessing_img = np.zeros((5), np.uint8)    # 用来保存要处理的图片
self.__processing_img = np.zeros((5), np.uint8)       # 保存处理好的图片
self.__key = []                                       # 用来保存图像加密时，产生的密钥图片
self.__file_name = ""                                 # 用来保存处理图片的文件名
self.__file_name_dir = ""                             # 用来保存处理图片的绝对路径
self.__load_img_h = 0                                 # 用来保存图片的长
self.__load_img_w = 0                                 # 用来保存图片的宽
self.__load_img_shape = 0                             # 用来保存图片的维度
```

2). python 目录结构

- 在进行 python 项目的时候，具有清晰的目录结构也是十分重要的事情。因为只有清晰的目录结构，才能方便以后项目内容的扩充和移植。
- 首先附上两张图。第一张是我初学 python 时的目录结构图，第二张是我学习 python 一段时间后的目录结构图。

  图一

.git	2019/10/30 10:14	文件夹	
.ipynb_checkpoints	2019/10/25 21:01	文件夹	
youtube_video_data	2019/10/23 23:27	文件夹	
.gitignore	2019/10/23 9:30	文本文档	2 KB
dogNames2.csv	2018/1/25 17:28	XLS 工作表	144 KB
img.jpg	2019/10/21 16:00	JPG 文件	24 KB
LICENSE	2019/10/23 9:30	文件	35 KB
Matplotlib.ipynb	2019/10/23 21:17	IPYNB 文件	686 KB
Numpy.ipynb	2019/10/24 19:42	IPYNB 文件	19 KB
Pandas.ipynb	2019/10/25 20:02	IPYNB 文件	13 KB
README.md	2019/10/23 9:30	MD 文件	1 KB
result.jpg	2019/10/22 20:02	JPG 文件	17 KB
t1.png	2019/10/23 10:19	PNG 文件	42 KB
test.jpg	2019/10/16 13:13	JPG 文件	80 KB
机器学习.ipynb	2019/10/25 21:40	IPYNB 文件	103 KB
图像数字化.ipynb	2019/10/22 23:52	IPYNB 文件	17 KB

  图二

.git	2019/12/23 19:43	文件夹	
dist	2019/12/17 17:10	文件夹	
docs	2019/12/16 19:12	文件夹	
src	2019/12/17 16:19	文件夹	
.gitignore	2019/12/18 18:41	文本文档	2 KB
LICENSE	2019/12/17 16:18	文件	2 KB
README.md	2019/12/23 19:43	MD 文件	3 KB
requirements.txt	2019/12/17 9:00	文本文档	5 KB

- 从两张图的对比中，我们可以得到：
  1. 第一张图结构混乱，几乎一个项目中的所有文件都在一个文件夹下
  2. 第二张图具有清晰的目录结构，人们可以通过目录名字就能想到目录下的存储的文件
- 通过图中的对比，我们就能初步感受得到 python 项目中目录结构的作用。这只是一个小型的项目，当项目做得越来越大时，目录结构将会越来越重要。
- 这里我推荐一篇教程：Python-如何设计结构清晰的目录结构，大家可以通过这篇教程来设计自己项目的目录结构。

3). 总结

- 这节主要讲述的是：我在学习 python 中遇到的两个方面的问题。这两个问题是大家在初学 python 都必须要解决的问题。只有解决这两个问题，大家才能写出优秀的 python 代码，好维护的 python 项目。

三. 总结

- 学好 python 并不是一蹴而就的事，需要我们一天天的积累，只有在时间的积累下，我们才能写出优秀的 python 代码。当然，在 python 入门的时候，要是有一个好的老师，我们可以少走很多弯路。在这里，我再次感谢我的 python 入门老师张总，正是他一步步的引导，我才能在较短的时间内，写出符合 python 规范的代码。
- 另外，在进行 python 项目的时候，书写代码只是其中的一部分，还有技术文档书写和版本控制，也是我们必须关心的事情。关于技术文档书写，目前我还处于摸索中。如果大家有心得，可以一起探讨。而关于版本控制，我也写了一篇教程，地址是：https://github.com/WanglinLi595/version-control，如果大家感兴趣的，可以抽时间去看一下。
- 最后感谢大家抽出自己宝贵的时间，来观看这篇教程。如果其中有错误的地方，欢迎大家指出。