

《Python Programming Project》

题目：密码生成器

专 业： 计算机科学与技术

学 号： 1805010207

学生姓名： 梁宇龙

项目日期： 2021 年 5 月 6 日

学术规范与诚信承诺书

本人梁宇龙已接受本课程指导教师的学术规范与诚信教育，愿意在课程考试、实验、撰写实验报告、课程报告及项目实施过程中，严格遵守《大连工业大学本科生学术道德规范（试行）》有关规定，恪守学术规范。

本人郑重承诺：

1. 不弄虚作假。不捏造、不伪造、不篡改引用资料或其他研究成果等；
2. 不抄袭、不剽窃、不作弊；
3. 不替人或请人代替写论文或设计；
4. 不复制或购买毕业设计、实验报告、课程报告、项目及软件代码等。
5. 按照规范引用他人的成果

如果有违反学术规范与诚信等行为，我愿意承担一切责任及由此带来的一切后果，并接受相关处理。

承诺人：梁宇龙

2021 年 5 月 3 日

目录

一、实验前准备.....	4
1、硬件环境.....	4
2、软件环境.....	4
3、python 环境内置准备.....	4
二、方案设计与论证.....	5
三、实现过程.....	5
1、构造大写字母串生成函数.....	5
2、构造特殊字符串生成函数.....	5
3、生成小写字母和数字字符串.....	6
4、编辑生成密码主函数.....	6
5、设置交互界面.....	6
四、实验结果.....	7
五、相关问题讨论.....	8
1、确定随机生成数的值.....	8
2、使各类型字符顺序错乱.....	8
3、如何生成不同类型的字符.....	9
六、总结.....	9
七、参考文献.....	9

一、实验前准备

1、硬件环境

主机：MacBook Pro (13-inch, 2020, Four Thunderbolt 3 ports)

处理器：2 GHz 四核 Intel Core i5

内存：16 GB 3733 MHz LPDDR4X

图形卡：Intel Iris Plus Graphics 1536 MB

2、软件环境

操作系统：macOS Big Sur 11.2.3

交互环境：内置终端 2.11 (440)

版本：Python 3.9.2

脚本编辑环境：Visual Studio Code 1.55.2

3、python 环境内置准备

修改提示符“>>>”为自定义内容

实现方案如下：

(1) 导入时间模块，并将标准时间存储为变量 localtime

```
import time

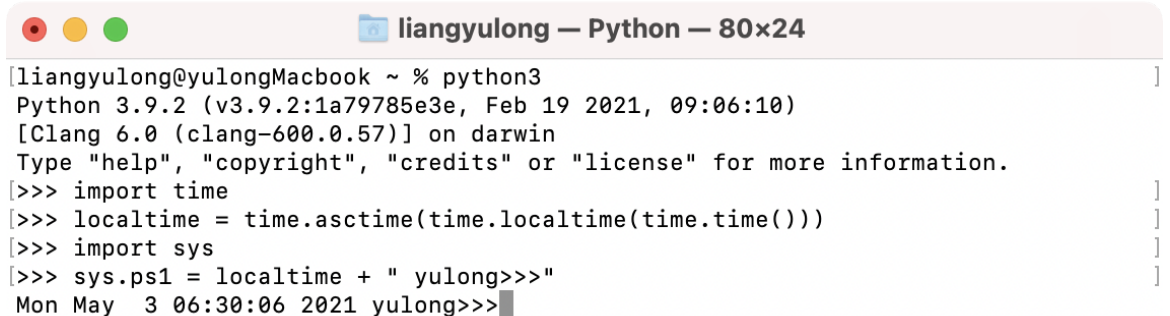
localtime = time.asctime(time.localtime(time.time()))
```

(2) 导入系统模块 sys，将 ps1 方法修改为 localtime 变量与名字组合字符串

```
import sys

sys.ps1 = localtime + " yulong>>>"
```

实现过程如下：



```
liangyulong — Python — 80x24
[liangyulong@yulongMacbook ~ % python3
Python 3.9.2 (v3.9.2:1a79785e3e, Feb 19 2021, 09:06:10)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> import time
[>>> localtime = time.asctime(time.localtime(time.time()))
[>>> import sys
[>>> sys.ps1 = localtime + " yulong>>>"
Mon May  3 06:30:06 2021 yulong>>>]
```

二、方案设计与论证

生成的密码包含大写字母、小写字母、特殊字符与数字组成，长度为 6~16。根据密码习惯，以小写字母和数字为主要，大写字母和特殊字符为辅助^[1]。使用 random 包在大写字母序列，特殊字符序列，小写字母和数字序列中抽取相应个数字符并打乱顺序即可完成密码生成。

三、实现过程

1、构造大写字母串生成函数

```
def get_upper():

    count = random.randint(1, 3) #表明生成的随机数只能在 1~3 之间

    return random.choices('ABCDEFGHIJKLMNOPQRSTUVWXYZ', k=count) #在大写字母序列中选出 k 个字符[2]
```

2、构造特殊字符串生成函数

```
def get_special_char():

    count = random.randint(1, 3)
```

```

        return random.choices('!@#$%^&*()_+~', k=count)    #在特殊字符串生成 k 个字符

```

3、生成小写字母和数字字符串

```

def get_lower(count): #参数 count 的值为字符总长度减去大写字母和特殊字符的长度

    string = 'abcdefghijklmnopqrstuvwxyz0123456789'

    return random.choices(string, k=count)

```

4、编辑生成密码主函数

```

def generate_password():

    length = random.random() * 100
    length = int(length % 10 + 6 ) #设置密码长度，使其保持在 6~16 之间，可用
    random.randint(6, 16)代替

    lst = [] #初始化密码串
    upper_lst = get_upper() #调用大写字母串生成函数
    special_char = get_special_char() #调用特殊字符生成函数
    lst.extend(upper_lst)
    lst.extend(special_char) #将大写字母和特殊字符加入密码串中

    surplus_count = length - len(lst) #求出小写字母和数字串长度（密码剩余长度）
    lower_lst = get_lower(surplus_count) #获取小写字母和数字串
    lst.extend(lower_lst) #将获取的串加入密码串中

    random.shuffle(lst) #将生成串顺序打乱
    return ''.join(lst) #将所有字符输出在一行

```

5、设置交互界面 ^[3]

```

if __name__ == '__main__':

    a = int(input("请输入要生成的密码个数: "))

    for i in range(0,a):

```

```
print(generate_password())
```

四、实验结果

```
liangyulong — Python — 80x24
0
[Mon May 3 06:30:06 2021 yulong>>>def get_upper():
[...     count = random.randint(1,3)
[...     return random.choices('ABCDEFGHIJKLMNOPQRSTUVWXYZ',k=count)
[...
Mon May 3 06:30:06 2021 yulong>>>

liangyulong — Python — 80x24
0
[Mon May 3 06:30:06 2021 yulong>>>def get_special_char():
[...     count = random.randint(1,3)
[...     return random.choices('!@#$%^&*()_~',k=count)
[...
Mon May 3 06:30:06 2021 yulong>>>

liangyulong — Python — 80x24
0
[Mon May 3 06:30:06 2021 yulong>>>def get_lower(count):
[...     string = 'abcdefghijklmnopqrstuvwxyz0123456789'
[...     return random.choices(string,k=count)
[...
Mon May 3 06:30:06 2021 yulong>>>

liangyulong — Python — 80x24
0
[Mon May 3 06:30:06 2021 yulong>>>def generate_password():
[...     length = random.random() * 100
[...     length = int(length % 10 + 6)
[...     lst = []
[...     upper_lst = get_upper()
[...     special_char = get_special_char()
[...     lst.extend(upper_lst)
[...     lst.extend(special_char)
[...     surplus_count = length - len(lst)
[...     lower_lst = get_lower(surplus_count)
[...     lst.extend(lower_lst)
[...     random.shuffle(lst)
[...     return ''.join(lst)
[...
Mon May 3 06:30:06 2021 yulong>>>
```

(各函数在 python 环境下导入)

```
machine learning — -zsh — 80x24
Last login: Mon May  3 09:25:46 on ttys000
[liangyulong@yulongMacbook ~ % cd machine\ learning
[liangyulong@yulongMacbook machine learning % date
2021年 5月 3日 星期一 09时 27分 54秒 CST
[liangyulong@yulongMacbook machine learning % python3 pro.py
请输入要生成的密码个数：6
g%fanPYowYa
1eU8~y
N^+q(1W
4zVR^c2g
$R2saT~a
q$M@vZt~S
[liangyulong@yulongMacbook machine learning % python3 pro.py
请输入要生成的密码个数：3
5&utoV^5wZ
s(Wo)^ie
bE(J@%W
liangyulong@yulongMacbook machine learning %
```

(在 macOS terminal 环境下执行脚本)

五、相关问题讨论

1、确定随机生成数的值

在本项目中，共有两处需要每次产生随机个数的字符，一处为生成密码的长度，另一处为密码中大写字符/特殊字符长度，本次项目采用两种不同的解决方案。

在生成密码长度实现中，由于 random 产生的随机数在 0~1 之间，因此先将其扩大 10 倍后取整，再加上基础值 6 位即可构成长度在 6~16 位之间的密码。

在生成大写字符/特殊字符函数中，则使用 random 包中自带的 randint(a, b) 方法，a、b 表示取值范围的上限和下限，使用该方法可生成一个在 a~b 之间的整数。

2、使各类型字符顺序错乱

在 random 包中包含一个 shuffle 方法，可以实现将列表中内容顺序打乱，使用该方法可完成字符顺序错乱的实现。^[4]

3、如何生成不同类型的字符

在不同类型字符生成中,比较简单的方法是划分不同区域的 ASCII 码。但在本次实验中,random 包中自带的 `choices(string,k)` 方法可以更好的实现抓取功能,其中 `string` 表示要抓取的字符库, `k` 表示抓取的个数,使用该方法可以将字符串中抓取 `k` 个字符,完成抓取功能。

六、总结

本次项目设计主要体会 random 包的使用和相应方法的实现过程。在学习 random 包中相应方法之前,使用 C 语言中相关方法也可以实现,但 random 包中好多方法可以直接获得返回值,在小型程序中虽然不能体现其简便性,但若大规模项目中,将很大程度提升编码能力。

在翻阅资料的过程中,我发现了 python 中好多包可以直接将曾经需要自己编写的函数直接调用,希望接下来可以学习更多的知识运用到自己的程序设计当中。

七、参考文献

[1]Zhou Tao,Chen Libin,Guo Jing,Zhang Mengmeng,Zhang Yanrui,Cao Shanbo,Lou Feng,Wang Haijun. MSIFinder: a python package for detecting MSI status using random forest classifier[J]. BMC Bioinformatics,2021,22(1):.

[2]王照. Python 语言编程特点及应用[J]. 电脑编程技巧与维护,2021,(03):19-20+44.

[3]李杨婧. Python 语言中循环结构教学模式的探讨[J]. 福建电脑,2021,37(02):164-165.

[4]Du Lianming,Liu Qin,Fan Zhenxin,Tang Jie,Zhang Xiuyue,Price Megan,Yue Bisong,Zhao Kelei. Pyfastx: a robust Python package for fast random access to sequences from plain and gzipped FASTA/Q files.[J]. Briefings in bioinformatics,2020,,:.