

在线实验，请到PC端体验

Python实现密码强度检测器

1.1 课程简介

本教程将带领大家编写一个简单的Python库——密码强度检测器。

2.1 接口设计

在我的上一个课程（Python3实现火车票查询工具 (<https://www.shiyanlou.com/courses/623>)）中，我们就已经采用了可以称之为面向API编程的方法——先设计API，再实现代码。这次我们继续贯彻这个思路。不同的是，上次的是一个命令行工具，而这次实现的是一次可以给其它程序员使用的一个简单的库。

首先为该模块起一个名字，考虑到模块功能的拓展，我们不要在名字中出现 password 字样，我们就叫它 check 吧。

```
>>> import check
>>> check.password('123')
terrible
>>> check.password('zxcvbnm')
simple
>>> check.password('fk3Ms0*!su')
strong
```

2.2 评测规则

说到密码强度，不用想也知道肯定是越长越复杂的密码强度越高，也就越安全。我们的任务就是把密码复杂度与强度量化，然后用计算机语言表达出来。

密码复杂度很好定义，长度，大写字母，小写字母，数字和特殊符号，密码包含的特征越多也就越，强度也就越高。我们设置个等级来评测密码强度，分别是：terrible, simple, medium, strong。

不同的应用可能对密码强度的要求一样，我们引入最小长度（min_length）和最小特征数（mim_types）变成可配置选项。这样我们就可以检测密码包含的特征，特征与密码与密码之间的关系可以简单定义为：

特征数	强度
小于最小长度	terrible
常用密码或有规则的密码	simple
没有达到最小特征数	midum
达到或超过最小特征数	strong

那么问题来了，如何定义常用密码呢？万能的Github上已经有人建立了常用密码表：

10000个最常用密码 (https://raw.githubusercontent.com/danielmiessler/SecLists/master/Passwords/10k_most_common.txt)

下载该文件，保存到 Code 目录下待用。

3.1 代码实现

如何检查特征呢？很容易想到的就是正则表达式：

动手实践是学习 IT 技术最有效的方式！

开始实验

```
NUMBER = re.compile(r'[0-9]')
LOWER_CASE = re.compile(r'[a-z]')
UPPER_CASE = re.compile(r'[A-Z]')
OTHERS = re.compile(r'^\0-9a-zA-Z]')
```

为了方便重用,我们先使用 `re` 模块的 `compile` 方法编译了这些特征。我们要检查密码是否包含一个特征调用 `re` 的 `search` 方法就可以了。

我们定义一个类,管理密码强度相关的东西:

```
class Strength:

    def __init__(self, valid, strength, message):
        self.valid = valid
        self.strength = strength
        self.message = message
```

我们定义了关于强度的三个属性,是否是有效的密码 (`valid`), 强度 (`strength`), 友好的提示信息 (`message`)。

检测密码是否是常用密码,我们只需检查密码是否在之前下载的10000个常用密码文件中:

```
def load_common_password():
    words = []
    with open('10k_most_common.txt', 'rb') as f:
        for word in f.readlines():
            words.append(word.strip())
    return words

COMMON_WORDS = load_common_password()
```

有规则的密码可能是在按照键盘顺序随便敲的一行,如 `adfgjkl`,又或者按照 `ASCII` 码顺序:

```
def is_regular(input):
    reverse = input[::-1]
    regular = ''.join(['qwertyuio', 'asdfghjkl', 'zxcvbnm'])
    return input in regular or reverse in regular

def is_by_step(input):
    delta = ord(input[1]) - ord(input[0])

    for i in range(2, len(input)):
        if ord(input[i]) - ord(input[i-1]) != delta:
            return False

    return True
```

接下来就是这个程序的核心了,检测密码的特殊,返回相应的强度:

```

class Password:

    TERRIBLE = 0
    SIMPLE = 1
    MEDIUM = 2
    STRONG = 3

    @staticmethod
    def is_regular(input):
        reverse = input[::-1]
        regular = ''.join(['qwertyuio', 'asdfghjkl', 'zxcvbnm'])
        return input in regular or reverse in regular

    @staticmethod
    def is_by_step(input):
        delta = ord(input[1]) - ord(input[0])

        for i in range(2, len(input)):
            if ord(input[i]) - ord(input[i-1]) != delta:
                return False

        return True

    @staticmethod
    def is_common(input):
        return input in COMMON_WORDS

    def __call__(self, input, min_length=6, min_types=3, level=STRONG):

        if len(input) < min_length:
            return Strength(False, 'terrible', '密码太短了')

        if self.is_regular(input) or self.is_by_step(input):
            return Strength(False, 'simple', '密码有规则')

        if self.is_common(input):
            return Strength(False, 'simple', '密码很常见')

        types = 0

        if NUMBER.search(input):
            types += 1

        if LOWER_CASE.search(input):
            types += 1

        if UPPER_CASE.search(input):
            types += 1

        if OTHERS.search(input):
            types += 1

        if types < 2:
            return Strength(level <= self.SIMPLE, 'simple', '密码太简单了')

        if types < min_types:
            return Strength(level <= self.MEDIUM, 'medium', '密码不错, 但还不够强')

        return Strength(True, 'strong', '完美的密码')

password = Password()

```

这里我们使用了一个 Password 类打包了与密码检测相关的东西，并使用了一个 Python 的魔法方法 `__call__`，这样我们就可以像函数一样调用 Password 的实例 password 了。

完整代码

```

# coding: utf-8

"""
    check
    ~~~~~

    Check if your password safe.
"""

import re
import json

__all__ = ['password']

NUMBER = re.compile(r'[0-9]')
LOWER_CASE = re.compile(r'[a-z]')
UPPER_CASE = re.compile(r'[A-Z]')
OTHERS = re.compile(r'^[0-9a-zA-Z]')

def load_common_password():
    words = []
    with open('10k_most_common.txt', 'rb') as f:
        for word in f.readlines():
            words.append(word.strip())
    return words

COMMON_WORDS = load_common_password()

class Strength:

    def __init__(self, valid, strength, message):
        self.valid = valid
        self.strength = strength
        self.message = message

    def __repr__(self):
        return self.strength

    def __str__(self):
        return self.message

    def __bool__(self):
        return self.valid

class Password:

    TERRIBLE = 0
    SIMPLE = 1
    MEDIUM = 2
    STRONG = 3

    @staticmethod
    def is_regular(input):
        reverse = input[::-1]
        regular = ''.join(['qwertyuio', 'asdfghjkl', 'zxcvbnm'])
        return input in regular or reverse in regular

    @staticmethod
    def is_by_step(input):
        delta = ord(input[1]) - ord(input[0])

        for i in range(2, len(input)):
            if ord(input[i]) - ord(input[i-1]) != delta:
                return False

        return True

    @staticmethod
    def is_common(input):
        return input in COMMON_WORDS

    def __call__(self, input, min_length=6, min_types=3, level=STRONG):
        """
            动手实践是学习 IT 技术最有效的方式!
            开始实验
        """

```

```
if len(input) < min_length:
    return Strength(False, 'terrible', '密码太短了')

if self.is_regular(input) or self.is_by_step(input):
    return Strength(False, 'simple', '密码有规则')

if self.is_common(input):
    return Strength(False, 'simple', '密码很常见')

types = 0

if NUMBER.search(input):
    types += 1

if LOWER_CASE.search(input):
    types += 1

if UPPER_CASE.search(input):
    types += 1

if OTHERS.search(input):
    types += 1

if types < 2:
    return Strength(level <= self.SIMPLE, 'simple', '密码太简单了')

if types < min_types:
    return Strength(level <= self.MEDIUM, 'medium', '密码不错, 但还不够强')

return Strength(True, 'strong', '完美的密码')

password = Password()
```

打开 ipython, 导入我们编写的模块, 测试下我们之前定义的接口:



```
Terminal 终端 - ipython
文件(F) 编辑(E) 视图(V) 终端(T) 标签(A) 帮助(H)

In [1]: import check
In [2]: check.password('123')
Out[2]: terrible
In [3]: check.password('zxcvbnm')
Out[3]: simple
In [4]: check.password('fk3Ms0*lsu')
Out[4]: strong
```

It's worded!

3.2 测试

为什么要写测试? 很多时候你可能觉得你的程序没有测试也工作的好好的! 这很大程度上是因为你的程序功能点太少, 基本你可以通过手动测试它们。试想一下, 如果你的程序有一千多个功能点, 而很多功能之间又相互关联, 也就是说你修改了程序个一个地方, 很可能其它地方也跟着出错。如果你没有写测试, 你得手动把程序跑一遍, 多累啊!

动手实践是学习 IT 技术最有效的方式!

开始实验

```
# coding: utf-8

""" Tests for check. """

import check
import unittest

class TestCheck(unittest.TestCase):

    def test_regular(self):
        rv = check.password('qwertyu')
        self.assertTrue(repr(rv) == 'simple')
        self.assertTrue('规则' in rv.message)

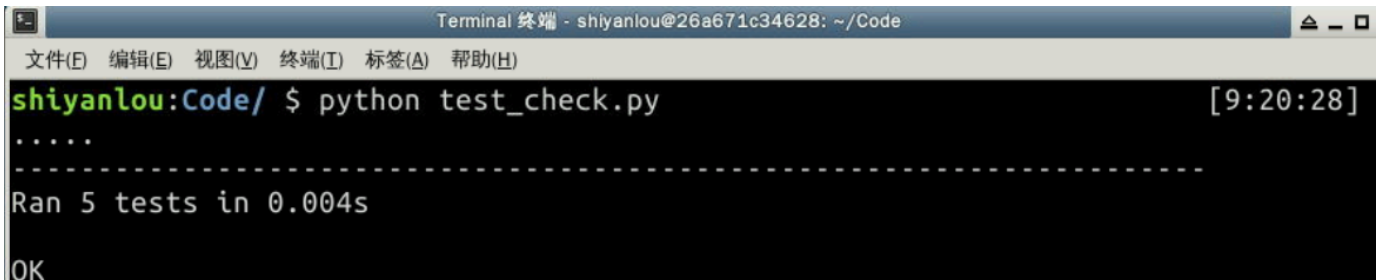
    def test_by_step(self):
        rv = check.password('abcdefg')
        self.assertTrue(repr(rv) == 'simple')
        self.assertTrue('规则' in rv.message)

    def test_common(self):
        rv = check.password('password')
        self.assertTrue(repr(rv) == 'simple')
        self.assertTrue('常见' in rv.message)

    def test_medium(self):
        rv = check.password('tdnwh01')
        self.assertTrue(repr(rv) == 'medium')
        self.assertTrue('不够强' in rv.message)

    def test_strong(self):
        rv = check.password('tdnWwh01.')
        self.assertTrue(repr(rv) == 'strong')
        self.assertTrue('完美' in rv.message)

if __name__ == '__main__':
    unittest.main()
```



```
Terminal 终端 - shiyanlou@26a671c34628: ~/Code
文件(F) 编辑(E) 视图(V) 终端(T) 标签(A) 帮助(H)
shiyanlou:Code/ $ python test_check.py [9:20:28]
.....
-----
Ran 5 tests in 0.004s
OK
```

4. 作业

1. check 的测试用例并不完善，试着再增加些测试用例
2. 为 check 添加一个检查邮箱是否有效的功能，并能获取邮箱类型，使用方法如下：

```
>>> import check
>>> e = check.email('foo@gmail.com')
>>> e
True
>>> e.type
'gmail'
>>> check.email('bar@123')
False
```

并添加测试用例。

课后作业参考链接：<https://www.shiyanlou.com/questions/6579> (<https://www.shiyanlou.com/questions/6579>)

课程教师

protream

动手实践是学习 IT 技术最有效的方式！

开始实验



共发布过3门课程

[查看老师的所有课程 > \(/teacher/634\)](/teacher/634)



动手做实验，轻松学IT



公司

(<http://weibo.com/shiyanlou2013>)

[关于我们 \(/aboutus\)](/aboutus)

[联系我们 \(/contact\)](/contact)

[加入我们 \(http://www.simplecloud.cn/jobs.html\)](http://www.simplecloud.cn/jobs.html)

[技术博客 \(https://blog.shiyanlou.com\)](https://blog.shiyanlou.com)

服务

[企业版 \(/saas\)](/saas)

[实战训练营 \(/bootcamp/\)](/bootcamp/)

[会员服务 \(/vip\)](/vip)

[实验报告 \(/courses/reports\)](/courses/reports)

[常见问题 \(/questions/?\)](/questions/)

<tag=%E5%B8%B8%E8%A7%81%E9%97%AE%E9%A2%98>

[隐私条款 \(/privacy\)](/privacy)

合作

[我要投稿 \(/contribute\)](/contribute)

[教师合作 \(/labs\)](/labs)

[高校合作 \(/edu/\)](/edu/)

[友情链接 \(/friends\)](/friends)

[开发者 \(/developer\)](/developer)

学习路径

[Python学习路径 \(/paths/python\)](/paths/python)

[Linux学习路径 \(/paths/linuxdev\)](/paths/linuxdev)

[大数据学习路径 \(/paths/bigdata\)](/paths/bigdata)

[Java学习路径 \(/paths/java\)](/paths/java)

[PHP学习路径 \(/paths/php\)](/paths/php)

[全部 \(/paths/\)](/paths/)

Copyright ©2013-2017 实验楼在线教育 | 蜀ICP备13019762号 (<http://www.miibeian.gov.cn/>)

动手实践是学习 IT 技术最有效的方式!

[开始实验](#)