Python 计算实验报告

组　　号	第 8 组
实验序号	1
学　　号	19120191
姓　　名	汪雨卿
日　　期	2020 年　4 月　11 日

一、 实验目的与要求

1. 熟悉Python 的开发调试环境；
2. 熟悉Python 外部库的调用；
3. 掌握Python 语言基本语法；
4. 熟悉Python 的数据结构.

二、 实验环境

1. 操作系统不限；
2. Python IDLE、PyCharm 等开发环境不限。

三、 实验内容

Part 1: 熟悉 Python 的开发环境

参照课本 1.1 节：使用 IDLE 和Python（Command Line）两种界面，用 P4 代码查看实验用机安装的 Python 版本。

参照课本 1.2 节（1）安装 Python 2.x 和Python3.x;（2）在IDLE 下使用命令行和创建脚本两种方式创建 Hello World。

参照课本 1.3 节，在 Python2.X 下，用 pip 命令安装扩展库 Swampy（解压附件中的压缩包）

熟悉PyCharm：

用桌面图标启动 PyCharm，浏览 PyCharm 工作环境，尝试修改 PyCharm 界面风格等，可参考 http://www.jetbrains.com/pycharm/documentation/。

用PyCharm 创建一个新工程，在工程中添加源文件，填写 Hellow World 代码，调试和运行。

熟悉Python 调试工具：

参照课本 8.6 和8.7 节，尝试代码调试功能。

参照课本1.5 节最末段（P23 页）尝试学习Python 的代码风格工具。

Part 2:熟悉 Python 基础编程

1．参照课本 1.4.2 节

A．验证Python 动态类型语言的特性；

B．用dir()查看Python 关键字；

2．把Python 作为计算器：

$$\underbrace{1515\cdots15}_{10个15}\times\underbrace{333\cdots33}_{20个3}$$

求下式计算结果，并且计算该结果各位数字之和：

$$\underbrace{2016\times2016\times2016\times\cdots\cdots\times2016}_{2016个2016}$$

3．判断下式计算结果十位数字的值

4．输入以下表达式并且查看结果

```
23+3、 23>3、 '23'+'3'、 23/3、 23//3、 23%3、 23**3
23+24.5、23+'3'、23+int('3')、'hello '+str("123")、int(23/3)、round(23/3,2)、round(23/3)、
0<23<100
```

5．文本对象： 声明字符串 s1 和s2，分别初始化为‘programming’和‘language’，观察以下表达式的计算结果

```
s1[1]、s1[:4]、s1[0]+s2[1:3]、s1.capitalize()+''+s2.upper()、s1.count('r')+s1.find('r')+
s1.rfind('r')、s3=s2.join('--')、s4='-'.join(s2)、L1=s4.split()、3*(s2[:2]+'')、"Python"
+s2.rjust(10)。
```

6．内置函数的使用

A．用dir()查看Python 内置对象，用 help（）分别查看任意 5 个关键字和 5 个内置对象的使用帮助，并且尝试使用；

B．比较函数 ord()和str()的差异；

7．模块导入

（1） 导入math 库：查看所有函数、比较对数运算函数、平方根计算、幂运算、比较整函数 ceil()和floor()等。

（2） 导入 random 库：生成 10 个[1，100]间的随机整数，计算最大值、最小值、和、平均（参照课本 P17）

Part 3：Python 应用编程

1．参照课本 1.9 节，编写例 1-例4。

附件 mypolygon.py:（1）运行和阅读代码；（2）理解代码功能。

四、 实验内容的设计与实现

说明：选择本实验中最有程序设计技巧或特色的并具有独立功能的源代码片断，并对其算法的技巧或特色进行必要的文字说明。每个程序都要有。

题目1：Python 代码理解 polygon.py:（1）运行和 阅读代码；（2）理解代码功能；（3）修改代码，练习调用文件中其他几个图形函数。

1. 在 python 中导入了 swampy 的库。通过 help()简单了解了 swampy 库中所包含的几个模块，本题中所使用的是 turtleworld 模块进行图像绘制。

2. 在写自定义的图像之前，对 turtleworld 模块中控制 turtle 运动的函数做了简单的了解。具体内容见下图呈现：（图片是来自自己做的 markdown 笔记）

4.polygon

TurtleWorld 模块

TurtleWorld中提供了几个函数来指挥乌龟的运动:

- fd : 前进 (forward)
  - fd(turtle,step(步长))
- bk : 后退 (backward)
- lt : 左转 (left turn)
  - lt(turtle,angle(转的角度，默认为90°))
- rt : 右转 (right turn)

每只乌龟都拿着一支笔，笔可以朝上或朝下:

- pu : 笔朝上 (pen up)
- pd : 笔朝下 (pen down)

笔朝上，不会绘制出轨迹（用于在到达指定位置前不绘制出轨迹）；笔朝下，会绘制出走过的轨迹。

画一个正方形

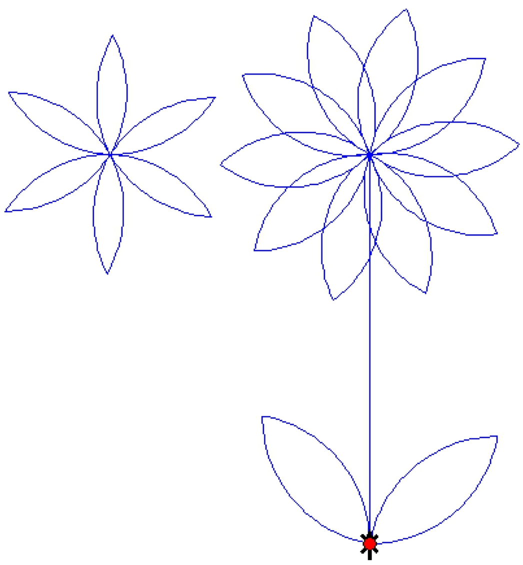
```python
#  写一个函数square ， 接受一个形参 t，用来表示一只乌龟,接受length,表示步长。
# 用来表示一只乌龟利用乌龟来画一个正方形。
#  写一个函数调用传入bob作为实参来调,用 square函数，并再运行一遍程序。
def square(t, length):
    t = Turtle()
    for i in range(4):
        fd(t, length)  # 参数2：步长
        lt(t)  # 参数2：angle 旋转角度，默认为90°
```

画一个n边形

```python
# 画一个n边形
def multi_square(t, length, n):
    # ''' 参数1：turtle, 参数2：步长, 参数3：边数
    t = Turtle()
    for i in range(n):
        fd(t, length)
        lt(t, 360 / n)
```

（更多基础图形的注释和使用详见源代码文件，此处省略。）

4. 利用掌握的函数制作：创意画：两朵花



关键代码部分： 亻 … 向移动。

```python
# 画一片花瓣
def petal(t, r, angle):
    # t: turtle;
    # r: radius;
    # angle: related to the thick of a petal
    for i in range(2):          # 画一个往返的弧线，构成一篇花瓣
        arc(t, r, angle)        # 调用arc依照角度画弧的函数
        lt(t, 180 - angle)      # 向左转向180 -angle = 返回


# 画一朵花
def flower(t, n, r, angle):
    """ t: turtle
    n: 花瓣数
    r: radius
    angle:旋转的角度
    """
    for i in range(n):
        petal(t, r, angle)
        lt(t, 360.0 / n)    # n 片花瓣旋转一周，偏转角度为360/n
```

```python
if __name__ == '__main__':
    # 创建世界和乌龟，并加速bob运动的速度
    world = TurtleWorld()
    bob = Turtle()
    bob.delay = 0.001           # 设置乌龟走过路程间隔的速度（提速）
    # 第一朵六片花
    flower(bob, 6, 96, 57)

    # 第二朵花
    move(bob, 200)              # 移动一段位置，避免和第一朵花重叠
    flower(bob, 10, 90, 80)     # 画第二朵花
    rt(bob, 90)                 # 转向，画花茎
    fd(bob, 300)
    lt(bob, 90)                 # 转向，画第一片叶子
    petal(bob, 100, 80)
    lt(bob, 90)                 # 转向，画第二片叶子
    petal(bob, 100, 80)

    wait_for_user()
```

题目 2. 输入输出：编写脚本文件，设计友好的用户输入输出提示，用户输入一个时间（24 小时制，包含时、分、秒）输出 1 秒后的时间。

1. 对于本题关键的处理部分分为两大块：第一个是异常处理；第二个是输入读取和输出格式。

2. 第二部分相对第一部分处理较为容易，故先描述。利用 split()读取三个由“：”分割的数字。将他们转换成 int 类型进行计算。输出时，若指定两位两位输出，若不足两位用 rjust(2, ‘0’)用 0 补全

3. 本题的重点在于如何判断用户输入的时间格式是正确的。对于格式的错误，这里分为了四大类：

第一类：输入的时间长度超过“xx:xx:xx”的长度。如“12:12:12:12:12”。

第二类：输入的时间不是用“：”风格，或直接输入的是乱码。例如：“xx,xx,xx”或者“x:xx:xx”等等

第三轮：输入的时间格式是正确的，但是时间的范围有误。例如小时超过了 24 点，分钟和秒钟超过了 60 等。

第四类：输入的数据中有负数，导致 ValueError 的报错。

前三类异常判断，全是通过 if 语句进行判断。

第四类异常判断，通过 try…catch 的异常处理方式在 except 中捕捉 ValueError 并做相应的处理。

```
def time():
    while 1:
        try:
            x = input("请输入一个时间（24小时制，包含时、分、秒）")
            #  判断输入格式为xx:xx:xx:xx:...超过3个
            if len(x.split(':')) > 3:
                print('输入格式有误。（时间过多）')
                continue
            #  判断时间格式中间不是用“：”连接的情况
            if x[2] != ':' or x[5] != ':':
                print('输入格式有误。（分隔符错误）')
                continue
            # 判断时，分，秒超过24小时制的情况
            if x[0] == '2' and int(x[1]) > 3:
                print('输入格式有误。（小时超过24）')
                continue
            if int(x[3]) > 5:
                print('输入格式有误。（分钟超过60）')
                continue
            if int(x[6]) > 5:
                print('输入格式有误。（秒钟超过60）')
                continue
```

```
            h, m, s = x.split(':')

            s = int(s) + 1
            s = str(s)
            # 控制输出格式，  不足两位用“0”补齐
            h.rjust(2, '0')
            m.rjust(2, '0')
            s.rjust(2, '0')
            print(h + ':' + m + ':' + s)
            continue
        except ValueError:
            print("输入的值的类型有误。请重新输入...")
            continue
        return


if __name__ == '__main__':
    time()
```

题目 3. 反序对：如果一个单词是另一个单词的反向序列，则称这两个单词为“ 反向对”。编写代码输出 word.txt 中词汇表包含的反向对。

1. 对于本题，处理的第一步是对于文件读入的处理。通过学习了解到有两种打开方式：

一种直接用 f = open(‘url’,’ r’)的方式，利用 f.read()读入。也需要在结束周用 f.close（）关闭文件。

另一种，是用 with open(‘url’,’ r’) as f:的形式打开，它包含了关文件的操作，对文件的处理在 ‘:’ 控制域内执行。（第 3、4 题均使用第二种文件读入方式，之后不再赘述。）

2. 在处理反序对的时候，了解到可以通过切片中 a[::-1]的功能对 string 类型反向输出。因此，产生想法，可以通过两个相同的数据结构存放原单词的内容，以及反序单词的内容。

3. 最开始采用的通过数组匹配的方式，通过 list 的结构进行匹配。它虽然能够成功匹配数组中的元素，但是速度很满。因此，产生想法可以利用集合这种数据结构去存放单词信息。在改进后查询的速度确实相比 list 查询有所提升，但是速度依旧没有很快。最后，在参考组员使用集合求交集的方式查找。速度大幅度变快。

** 思考：set 查找快的原因是因为，set 和 dict 存储和查找的方式和 list 本身并不相同。

dict 的查询实现原理和查字典是一样的。假设字典包含了 1 万个汉字，我们要查某一个字，一个办法是把字典从第一页往后翻，直到找到我们想要的字为止，这种方法就是在 list 中查找元素的方法，list

越大，查找越慢。

第二种方法是先在字典的索引表里（比如部首表）查这个字对应的页码，然后直接翻到该页，找到这个字。无论找哪个字，这种查找速度都非常快，不会随着字典大小的增加而变慢。

dict 就是第二种实现方式，给定一个名字，比如'Michael'，dict 在内部就可以直接计算出 Michael 对应的存放成绩的“页码”，也就是 95 这个数字存放的内存地址，直接取出来，所以速度非常快。

而set只是隐藏了key的dict，因此推测也具有这样的性质，使得查找速度比list快。

```
with open("D:/Fish 鱼/SHU/Study/Semesters/07 SHU 2nd spring/Python/上机/Python计算实验一/words.txt", 'r') as f:
    lines = f.readlines()  # 读取全部内容
    content = set([x.strip() for x in lines])   # 把原单词存入content集合
    reverse = set([x[::-1] for x in content])   # 把反序的单词存入reverse集合


    def find():
        listd = list(content.intersection(reverse))      # intersection()求集合的交集
        listd.sort()
        for i in listd:
            print(i + " " + i[::-1])                # i[::-1]反向输出
        return listd
```

题目 4. 文本分析算法设计：

（1） 设计 Python 程序读入一个英文单词组成的文本文件，统计该文本文件中各个单词出现的次数。设计测试用例验证代码的正确性。

（2） 设计 Python 程序读入一个英文单词组成的文本文件，统计其中包含的某给定关键词列表中各个单词出现的频率。设计测试用例验证代码的正确性。

1. 本题中的两个其实相对而言是相似的。第二小题，相对于第一小题，只增加了指定单词的输入以及总单次数的求和。本题的关键在于如何精确的分割单词，并且统计每个单词的个数。

2. 对于单词分割的部分，在操作的时候进行了不同尝试。

a. 第一种思想：把文件中所有的字符变成小写字符；识别特殊的符号，把符号全部用空格代替，然后利用 split(“ ”)去做单词的分割。

```
#     txt = txt.lower()  # 把所有单词都转换成小写
#     for ch in ",.!、!@#$%^';\n":  # 将所有除单词以外的符号换成空格
#         txt.replace(ch, '')
#     return txt
#
#
# # 分析提取单词 txtArr是列表
# txtArr = getTxt().split(" ")
#
```

但是，发现对于文本格式并不严格，例如“，”后面没有空格等情况不能做很好的单词分割。

b. 第二种思想：把文件中所有的字符变成小写字符；引入 re 库，通过正则匹配，匹配每一个单词。

```
#     用正则匹配 \b表示英文字母开始和结束 \w+表示多个英文字符,返回list类型数据
list_word = re.findall(r'\b\w+\b', content)
```

效果较好，故采用。

3. 单词数统计：

最开始的想法是利用计数器 count，后来发现利用 list set 等本身性质操作更为简洁。故有了优化的计数方案。

利用集合存储单词，去掉重复的单词。返回一个字典，key 是单词，value 是出现的次数。其中，value 利用 list 的 count 函数

```
    # 去掉重复单词, 利用集合去除list中重复的单词
    set_word = set(list_word)

    # 返回每个再集合中的单词，再文中出现的次数 是一个字典
    return {word: list_word.count(word) for word in set_word}
```

4. 对于字数统计的展示：

通过 sorted()函数对单词列表排序。

其中利用 lambda 函数（可以不指明函数名的函数），对单词依照次数由大到小排序。

同时利用 format()指定输出格式。（第一小题完成）

```
dict_word_times = get_dict_word_times()

# 把单词按照次数由多到少排序
list_sorted_words = sorted(dict_word_times, key=lambda w: dict_word_times[w], reverse=True)
for word in list_sorted_words:
    print("{} -- {} times".format(word, dict_word_times[word]))
```

5. 对于频率的计算

通过 len(list_word)获取最开始读入的单词总数，之后的匹配也只需通过 set 的查找即可。

最后通过计算返回频率。

```
dict_word = {word: list_word.count(word) for word in set_word}
count1 = dict_word[str(x)]

print("总单词数为: " + str(len(list_word)) + " list_word" + x + " 有 " + str(count1) + " 个 ")
return count1 / len(list_word)
```

五、 测试用例

There are moments in life when you miss someone so much that you just want to pick them from your dreams and hug them for real!

Dream what you want to dream; go where you want to go; be what you want to be, because you have only one life and one chance to do

all the things you want to do.

May you have enough happiness to make you sweet,enough trials to make you strong, enough sorrow to keep you human,enough hope to

make you happy? Always put yourself in others' shoes. If you feel that it hurts you, it probably hurts the other person, too.

The happiest of people don't necessarily have the best of everything; they just make the most of everything that comes along

their way. Happiness lies for those who cry, those who hurt, those who have searched, and those who have tried, for only they can

appreciate the importance of people who have touched their lives.Love begins with a smile, grows with a kiss and ends with a

tear. The brightest future will always be based on a forgotten past, you can't go on well in lifeuntil you let go of your past

failures and heartaches.

When you were born, you were crying and everyone around you was smiling. Live your life so that when you die, you're the one who

is smiling and everyone around you is crying.

Please send this message to those people who mean something to you, to those who have touched your life in one way or another, to

those who make you smile when you really need it, to those that make you see the brighter side of things when you are really

down, to those who you want to let them know that you appreciate their friendship. And if you don't, don't worry, nothing bad

will happen to you, you will just miss out on the opportunity to brighten someone's day with this message.

六、 收获与体会

本次上机的内容，part1~part3 帮助我很快的了解并且熟悉的 python 的编译环境，以及常用的一些函数的功能。以至于在完成四道上机题目的过程中可以很快速的进行代码处理，并且深入到更为关键的部分，而不会为一些不必要的调试所烦恼。对比以往学习 C++和 java 的过程中，往往会出现先学习语法，

并在写代码的过程中再去熟悉使用的编译环境。从而导致有些调试快捷键的使用，以及程序调试往往在很后面才了解，降低了 debug 的效率。

而对于完成上机题的过程中，小组合作和自己研究 python 的使用，帮助我快速的将学到的知识投入使用。同时，也在优化程序的过程中学习到了更多的内容。让我体验到了钻研优化代码的快乐。这四道题，第一次完成其实花费的时间，并没有想象中的那么久。但是，在准备验收和验收之后对于第一版的代码的优化，反而花费了较多的时间。比如第二题的时间输入，最开始并没有完善到很好的异常处理；第三题的查找速率的提升，也借助了队友的邦之；最后一题的字典查询，也是反复尝试打磨才选定了最后的匹配方式。

总结来说，这次上机让我学习到了很多，无论是 python 相关的知识，还是入门一种新的语言的学习方式。