第二次作业补充说明

提交格式说明

注意：

- 你最后提交的压缩包（如果不知道什么是压缩包就先去查一下）中有且只能有两个文件，且这两个文件的命名严格为**dsa1_1.py**和**dsa1_2.py**，不需要在后面加你的学号姓名什么的。
- 第一部分作业的python代码全放在同一个文件夹中，且**除了函数的定义之外最好不要有其他内容，提交前将你自己测试时写的内容去掉，注释除外**。我们只需要你提交你自己实现的若干个函数，我们会自动化测试你的这些函数是否满足要求。
- 你的函数名必须**严格按照作业要求中给出的名称进行命名**，函数参数类型也不能自己随便做假设，比如你不能自己假设所有的参数类型都是字符串。为了避免理解有问题，我们会给出每个函数的参数及返回值的类型要求。

第一部分类型要求及示例

1. `dsa1_reverse(s,n)`

   参数及返回值类型：`dsa1_reverse(s: str, n: int) -> str`

   注意：这样的写法是python3中的一个语法特性——函数注解，这个特性的功能和作用可以自己搜一下。主要是为了让看代码的人知道这个函数的参数和返回值应该是什么样的。上面这个示例的含义是，第一个参数 `s` 的类型应该是 `str`，第二个参数 `n` 的类型应该是 `int`，这个函数返回值类型应该是 `str`

   示例：

   ```
   dsa1_reverse("abcd", 1) # return val is "dabc"
   dsa1_reverse("mnbol", 2) # return val is "olmnb"
   ```

2. `dsa1_factorialSum(n)`

   参数及返回类型：`dsa1_factorialSum(n: int) -> int`

   示例：

   ```
   dsa1_factorialSum(1) # return val is 1
   dsa1_factorialSum(3) # return val is 9
   ```

3. `dsa1_generateDict(keys, values)`

   参数及返回类型：`dsa1_generateDict(keys: tuple, values: tuple) -> dict`

   示例：

   ```
   dsa1_generateDict((1, 2, 3), ("abc", "def", "ghi"))
   # return val is {1:"abc", 2:"def", 3:"ghi"}
   ```

4. `dsa1_en2num(s)`

   参数及返回类型：`dsa1_en2num(s: str) -> int`

   备注：忽略前导0（当然，返回的是int类型，也不存在前导0问题），详见示例

   示例：

```
dsa1_en2num("one-two-three-four-five") # return val is 12345
dsa1_en2num("zero-zero-one-two-three") # return val is 123
dsa1_en2num("zero-zero") # return val is 0
```

5. `dsa1_getUnion(s1,s2)`

   参数及返回类型：`dsa1_getUnion(s1: str, s2: str) -> set`

   示例：

   ```
   dsa1_getUnion("abc", "bcd") # return val is set(['a', 'b', 'c', 'd'])
   ```

   备注：上面示例中写的`set(['a', 'b', 'c', 'd'])`，是python中直接`print`一个`set`类型值后显示的格式，我们只是用这个格式来表示一个`set`，返回值就是一个`set`。

6. `dsa1_getDays(y,m)`

   参数及返回类型：`dsa1_getDays(y: int,m: int) -> int`

   示例：

   ```
   dsa1_getDays(2022, 3) # return val is 31
   ```

7. `dsa1_isNarcNum(n)`

   参数及返回类型：`dsa1_isNarcNum(n: int) -> bool`

   示例：

   ```
   dsa1_isNarcNum(153) # return val is True
   dsa1_isNarcNum(154) # return val is False
   ```

8. `dsa1_writeNarcNum(max)`

   参数及返回类型：`dsa1_writeNarcNum(max: int) -> None`
   备注：python中如果函数没有返回值，默认的返回值是`None`，这个题中不要求返回值，所以返回值这里写了个`None`

   示例：

   ```
   dsa1_writeNarcNum(999) # 调用后生成了一个名为narcissistic.txt的文件中
   ```

   备注：文件**务必不要用绝对路径**，直接用`open("narcissistic.txt", "w", encoding="utf-8")`就行了。

9. `dsa1_readNarcNum()`

   参数及返回类型：`dsa1_readNarcNum() -> list`

   示例：

   ```
   # 假设你narcissistic.txt文件中的内容如下：
   # 153
   # 370
   # 371
   dsa1_readNarcNum() # return val is [153, 370, 371]
   ```

备注：文件**务必不要用绝对路径**，直接用`open("narcissistic.txt", "r", encoding="utf-8")`就行了。

10. `dsa1_printTri(n)`

    参数及返回类型：`dsa1_printTri(n: int) -> None`

    示例：

```
dsa1_printTri(6)
```

    上面这个示例的输出结果如下，注意不要有多余的空行和空格：

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
```

OOP部分类型要求及示例

1. `People`类

   类型：`People`类中有两个属性`name`和`age`，`name`类型应为`str`，`age`类型应为`int`

   示例：

```
# 假设此时已经写好了你的People类
p = People("abc", 123)
p.getName() # return val is "abc"
p.getAge() # return val is 123
```

2. `Student`类

   类型：新加的属性`sno`的类型应该为`str`，注意，实际应用中，类似学号、身份证号这样的号码我们常用`str`来表示和存储，这里我们也这样做。

   示例：

```
# 假设此时已经写好了你的Student类
p = People("abc", 123, "2100023333")
p.getSno # return val is "2100023333"
```

3. `Xdict`类

   类型：`getKeys(self, value)`方法的参数`value`类型可以是多样的，只要是python中的字典支持的可以做`value`的类型就可以（实际上什么都可以做`value`）。为了简单起见，我们在测试时只会使用字符串和整数这样方便直接比较的值。你不需要考虑类似`getKeys({1:3, 3:4})`这样的操作（即使用字典作为参数）。

   示例：

```
# 假设此时已经写好了你的Xdict类
xd=Xdict({2:"a", 3:"a", 4:(2,3)})
xd.getKeys("a") # return val is [2,3]
```