第一次作业讲评

李一飞

2023.02.27

0.1 浮点数存在误差

（参见IEEE 754浮点数标准）

```
In [2]: 11 // 1.1
Out[2]: 9.0

In [3]: 12 // 1.2
Out[3]: 10.0

In [4]: 13 // 1.3
Out[4]: 9.0

In [5]: 14 // 1.4
Out[5]: 10.0

In [6]: 15 // 1.5
Out[6]: 10.0

In [7]: 16 // 1.6
Out[7]: 9.0

In [8]: 17 // 1.7
Out[8]: 10.0
```

```
In [10]: print(f"{1.7:.30f}")
1.699999999999999955591079014994

In [11]: print(f"{1.6:.30f}")
1.600000000000000088817841970013

In [12]: 17 // 1.7
Out[12]: 10.0

In [13]: 16 // 1.6
Out[13]: 9.0
```

```
>>> print(f"{1.1:.20f}")
1.10000000000000008882
```

0.2 无限递归

- n小于0时，永远不会触发递归终止条件，导致无限递归错误

```python
def rec_sum(n):
    if n == 0:
        return 0
    else:
        return n + rec_sum(n-1)

rec_sum(-4)  # 如果输入参数为-4，观察运行结果如何？ #10 报错
```

3.7s

```
RecursionError: maximum recursion depth exceeded in comparison
```

0.3 python生成器表达式的使用

```
1  s = '[2,6,1,3,7,2,7]'
2  nums = s[1:-1].split(",")
3  result = [num for num in nums if nums.count(num) == 1] #筛选只出现过一次的元素
4
5  print(result)
```

```
['6', '1', '3']
```

0.4 递归调用展平字典

- 函数内部也可以定义函数

```python
src = {'a':{'b':1,'c':2},'d':{'e':3,'f':{'g':4}}}

def flat_dict(dic):
    res={}
    def flat(prefix, dic):
        for key, val in dic.items():
            if type(val)==dict:
                flat(prefix+'.'+key,val)
            else:
                res[prefix+'.'+key]=val
    for key,val in dic.items():
        if type(val)==dict:
            flat(key,val)
        else:
            res[key]=val
    return res

print(flat_dict(src))
```

```
{'a.b': 1, 'a.c': 2, 'd.e': 3, 'd.f.g': 4}
```

0.5 for循环中的循环变量

- for循环的循环变量由生成器控制

```
1  num = 210
2  result = []
3
4  for factor in range(2, int(num ** 0.5) + 1):#开方分解因子
5      print("factor above:", factor)
6      while num % factor == 0:
7          result.append(chr(factor + 48))
8          num = num // factor
9      factor += 1
10     print("factor below:", factor)
11
12 print("*".join(result))
```

[6] ✓ 0.0s

```
factor above: 2
factor below: 3
factor above: 3
factor below: 4
factor above: 4
factor below: 5
```

0.5* for循环中的循环变量

- 与C/C++的不同点
  - Python会先实例化对应的生成器，再进入循环
  - C/C++中没有“生成器”的概念

```python
s = "abcde"
for i in range(len(s)):
    s+=str(i)
print(s)
```

✓ 0.0s

```
abcde01234
```

Python

```cpp
string s = “abcde”;
for (int i = 0; i < strlen(s); ++i){
     s += to_string(i);
}
// This loop will not end
```

C/C++

1.1 可迭代对象相乘

```
1  def product(container_iterable):
2      a=list(container_iterable)
3      result=1
4      for num in a:
5          result=result*num
6      return result
```

- 如果container_iterable为空？

```
def product(container_iterable):
    result = None
    for v in container_iterable:
        if result is None:
            result = v
        else:
            result *= v
    return result
```

1.2 字符串大小写转换

- 做法1: 使用ord()和chr()函数
  - ord(a): 得到字符a的ASCII码
  - chr(x): 得到ASCII码为x的对应字符

  # “写的python代码太C/C++了”

- 做法2: 使用upper()和lower()函数
  - upper(x): 将字符串x中所有字母变为大写
  - 注意，不能直接在字符串上操作（因为字符串对象不可变）

- 做法3: 直接调用库函数

```python
def swap_case(s):
    ## 请补充完整代码
    result=''#记录转换后字符串
    for a in s:
        if 65 <= ord(a) <= 90:
            a = chr(ord(a) + 32)
        elif 97 <= ord(a) <= 122:
            a = chr(ord(a) - 32)
        else:
            pass
        result+=a
    return result
```

```python
def swap_case(s):
    res = ""
    for c in s:
        if c.isupper():
            res = res + c.lower()
        elif c.islower():
            res = res + c.upper()
        else:
            res = res + c
    return res
```

```python
def swap_case(s):
    ## 请补充完整代码
    return s.swapcase()
```

1.3 分支、条件表达式

- 熟悉if、elif、else关键字的使用

本题中没有规定score的上限/下限

```
def grade(score):
    ## 请补充完整代码
    if score >= 85:
        return 'A'
    elif 75 <= score <=84:
        return 'B'
    elif 60 <= score <=74:
        return 'C'
    else:
        return 'F'
```

1.4 List / Dict / Tuple类型基本操作

- 添加、删除元素等

```
course = ['编译', '毛概', '操统']
elective = {'A':['音数'], 'C':['三宝'], 'E':['西音','西美']}

course.append('Python')  # list添加元素
del course[0]    # list删除元素

elective['A'] = ['音数', '三宝']
del elective['C']  # dict删除元素
elective['E'] = ['西音', '西美', '音数']
elective['D'] = ['西美']  # dict赋值
```

2.1 循环语句、表达式

计算数列前10项和

$$a_n = 2 * a_{n-1} + \frac{1}{a_{n-1}}, a_1 = 1$$

```python
a = 1  #  a1 = 1
s = 0  # 加和的结果

# 以下为待补全部分
for i in range(1,11):
    s+=a
    a=a*2+1/a
print(s)

# 验证结果
assert s > 1645 and s < 1646
```

[9] ✓ 0.0s

```
1645.5462828566815
```

2.2 循环表达式与列表

找出列表中只出现过一次的数字，并将这些数字按原次序保存进另一个列表unique_nums中
例如： 输入：2,6,1,3,7,2,7 输出：6,1,3

```
[12]
```

```python
s = '2,6,1,3,7,2,7'
nums = s.split(',')
unique_nums = []

def make_unique(s):
    nums = [int(x) for x in s.split(',')]
    unique_nums = []

    # 以下为待补全部分
    unique_nums = [num for num in nums if nums.count(num) == 1]
    return unique_nums
print(make_unique(s))

# 验证结果
assert(make_unique('2,6,1,3,7,2,7')==[6,1,3])
assert(make_unique('1,2,3,5,8,1,1,8,6,7,8,0')==[2,3,5,6,7,0])
```

```
[6, 1, 3]
```

2.3 将以 ':' 和 '|' 为分隔符的字符串处理成 python 字典

例如： 输入：'k:1|k1:2|k2:3|k3:4'　　输出：{'k': '1', 'k1': '2', 'k2': '3', 'k3': '4'}

```
s = 'k:1|k1:2|k2:3|k3:4'
result = dict()

# 以下为待补全部分
ss = s.split('|')  # ss: ['k:1', 'k1:2', 'k2:3', 'k3:4']
for x in ss:
    y = x.split(':')  # y: ['k', '1']
    y1 = y[0]  # y1: k
    y2 = y[1]  # y2: 1
    result[y1] = y2 # result: {'k': '1', 'k1': '2', 'k2': '3', 'k3': '4'}
print(result)
# 验证结果
assert 'k1' in result
assert result["k1"] == '2'
```

```
{'k': '1', 'k1': '2', 'k2': '3', 'k3': '4'}
```

2.3 将以 ':' 和 '|' 为分隔符的字符串处理成 python 字典

例如： 输入：'k:1|k1:2|k2:3|k3:4'   输出：{'k': '1', 'k1': '2', 'k2': '3', 'k3': '4'}

```
s = 'k:1|k1:2|k2:3|k3:4'
result = dict()

# 以下为待补全部分
```

```
# 介绍一种比较简洁的写法
result = dict([data.split(':') for data in s.split('|')])
# 刘肖2000013055
```

```python
# 判断一个数x是否为素数
def is_prime(x):
    pass
    if x == 1:
        return False
    for factor in range(2, int(x ** 0.5) + 1):#开方分解因子
        if x % factor == 0:
            return False
    return True
```

```python
# 利用filter机制生成1-200的素数列表
prime_list = []
list1=list(range(1,201))
# filter(is_prime, list1): 以is_prime作为条件，对list1中的元素进行过滤，
# 得到一个新的可迭代对象
prime_list = list(filter(is_prime, list1))
print(prime_list)
```

2.5 词频统计

- 1. 处理标点:

```
for ch in ',.':
    s.replace(ch, '')
```

也可用正则表达式来split字符串

- 2. 处理成小写：

```
s=s.lower()
```

或者全部变成大写，只要对大小写不敏感即可

- 3. 统计词频：

```
for word in s:
    word_freq[word] = word_freq.get(word, 0) + 1
```

总结

- 平时需要多写多练，学习并熟悉python的各种基本语法
  - 这也是每节课都留作业供大家练习的原因（狗头
- 一些（可能是信科的）同学写出很偏C/C++风格的代码
  - python和C/C++还是有很多不同之处，例如变量作用域/生存期，循环的实现方式等
  - 要注意比较二者的区别，避免写python时受到C/C++惯性思维的影响
- 在写代码的同时要注意代码风格的培养，养成好的习惯
  - Google python代码规范