Python 计算 1-6 章课后习题

19120198 孙天野

第一章 基础知识

1. 如何选择正确的 Python 版本?

1. 调查 python 版本与计算机系统的兼容情况
2. 调查所用模块与不同 python 版本的适配程度
3. 善用虚拟环境
4. 使用 Anaconda 集成开发环境，或是使用 DLE ， 自己配置所需的开发环境

2. 为什么说 Python 是基于值的内存管理模式?

python 的变量并不直接存储值，而是存储该值对应的地址或引用。所以当两个变量被赋予相同值时，这两个变量将会指向同一块内存地址。一个值在内存中只存储一份。

3. `/` 和 `//` 的区别

`/` 指浮点数除法，除法结果是浮点数；`//` 指整数除法，除法结果向下取整

4. 导入模块的对象有几种方式?

```
import module_name
import module_name as alias # 使用别名
from module_name import *
from module_name import module_class # 导入指定对象
```

5. 填空

pip 是目前比较常用的 python 扩展库管理工具

6. 解释 python 脚本程序的 `__name__` 变量及其作用

标识模块名，显示一个模块的某功能是被自己执行，还是被别的文件调用执行

7. 填空

运算符 `%` **可以**对浮点数进行求余数操作

8. 填空

一个数字 5 **是**合法的 python 表达式

9. 填空

在 Python 2.x 中，`input()` 函数接收到的数据类型由**界定符**确定，而在 Python 3.x 中该函数则认为接收到的用户输入数据一律为**字符串对象**

10. 编写程序，用户输入一个三位以上的整数，输出其百位以上的数字。例如用户输入 1234，则程序输出 12（提示：使用整除运算）

```
num = input('Input a number: ')
print(int(num) // 100)
```

第二章 python 序列

1. 为什么应尽量从列表的尾部进行元素的增加与删除操作?

从列表其他部分进行增删会涉及元素的移动，耗费时间

2. 填空

`range()` 函数在 Python 2.x 中返回一个**列表**，而 Python 3.x 的 `range()` 函数返回一个**迭代器**。

3. 编写程序，生成包含1000个0~100之间的随机整数，并统计每个元素的出现次数

```
import random

int_list = [random.randint(0, 100) for i in range(1000)]
int_dict = {}
for key in int_list:
    int_dict[key] = int_dict.get(key, 0)+1

for key, value in int_dict.items():
    print(str(key)+': '+str(value))
```

4. 填空

表达式 `[3] in [1,2,3,4]` 的值为 **False**

5. 编写程序，用户输入一个列表和 2 个整数作为下标，然后输出列表中介于 2 个下标之间的元素组成的子列表。例如用户输入[1,2,3,4,5,6] 和 2,5，程序输出 [3,4,5,6]

```
input_list = eval(input('Input a list: '))
a, b = input('input start and end: ').split(',')

print('The sub list is: ', end='')
print(input_list[int(a): int(b)+1])
```

6. 填空

列表对象的 `sort()` 方法用来对列表元素进行原地排序，该函数返回值为 **None**

7. 填空

列表对象的 `remove()` 方法删除首次出现的指定元素，如果列表中不存在要删除的元素，则抛出异常

8. 填空

假设列表对象 `aList` 的值为 `[3,4,5,6,7,9,11,13,15,17]`，那么切片 `aList[3:7]` 得到的值是 `[6,7,9,11]`

9. 设计一个字典，并编写程序，用户输人内容作为"键"，然后输出字典中对应的"值"，如果用户输入的"键"不存在，则输出"您输人的键不存在!

```
dict_a = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
while True:
    input_key = input('请输入键名: ')
    value = dict_a.get(input_key, '您输入的键不存在!')
    if value != '您输入的键不存在!':
        print('其值为: ', end='')
    print(value)
```

10. 编写程序，生成包含 20 个随机数的列表，然后将前 10 个元素升序排列，后 10 个元素降序排列，并输出结果

```
import random

a = [random.randint(0, 100) for i in range(10)]
b = [random.randint(0, 100) for i in range(10)]

a.sort()
b.sort(reverse=True)
a.extend(b)

print(a)
```

11. 填空

在 Python 中，字典和集合都是用一对**大括号**作为界定符，字典的每个元素有两部分组成，即**键和值**，其中**键**不允许重复

12. 填空

使用字典对象的 `items()` 方法可以返回字典的 “键-值对” 列表，使用字典对象 `keys()` 的方法可以返回字典的 “键” 列表，使用字典对象的 `values()` 方法可以返回字典的 “值” 列表。

13. 填空

假设有列表 `a=['name', 'age', 'sex']` 和 `b=['Dong',38,'Male']`，请使用一个语句将这两个列表的内容转换为字典，并且以列表 a 中的元素为 “键”，以列表 b 中的元素为 “值”，这个语句可以写为 `dict(zip(a, b))`

14. 填空

假设有一个列表 a，现要求从列表 a 中每 3 个元素取 1 个，并且将取到的元素组成新的列表 b，可以使用语句 `b = a[::3]`

15. 填空

使用列表推导式生成包含 10 个数字 5 的列表，语句可以写为 `[5 for i in range(10)]`

16. 填空

可以使用 del 命令来删除元组中的部分元素

第三章 选择与循环

1. 分析逻辑运算符 or 的短路求值特性

若第一个表达式为真，结果为真，不用判断第二个表达式的值

2. 编写程序，运行后用户输入 4 位整数作为年份，判断其是否为闰年。如果年份能被 400 整除，则为闰年；如果年份能被 4 整除但不能被 100 整除也为闰年

```
while True:
    year = int(input('Input a year: '))

    if year % 400 == 0 or (year % 4 == 0 and year % 100 != 0):
        print("It's a leap year")
    else:
        print("It's not a leap year")
```

3. 填空

Python 提供了两种基本的循环结构：`for` 和 `while`

4. 编写程序，生成一个包含 50 个随机整数的列表，然后删除其中所有奇数（提示：从后向前删）

```
import random

a = [random.randint(0, 100) for i in range(50)]

for i in range(len(a)-1, -1, -1):
    if a[i] % 2 == 1:
        del a[i]

print(a)
```

5. 编写程序，生成一个包含 20 个随机整数的列表，然后对其中偶数下标的元素进行降序排列，奇数下标的元素不变（提示：使用切片）

```
import random

a = [random.randint(0, 100) for i in range(20)]

a_even = a[::2]
a_even.sort(reverse=True)
a[::2] = a_even

print(a)
```

6. 编写程序，用户从键盘输入小于 1000 的整数，对其进行因式分解

```
num = int(input("Input a number less than 1000: "))
if num == 1 or num == 0:
    print('{} = {} × {}'.format(str(num), str(num), str(num)))
    exit()
print(str(num) + ' = ', end='')

num_list = []
while num > 1:
    for i in range(2, num + 1):
        if num % i == 0:
            num_list.append(str(i))
            num //= i
            break

for i in range(len(num_list)):
    print(num_list[i], end=' x ' if i != len(num_list)-1 else '\n')
```

7. 编写程序，至少使用两种不同的方法计算 100 以内所有奇数的和

```
# method 1
odd_sum = 0
for i in range(1, 100, 2):
    odd_sum += i
print('The odd sum is ' + str(odd_sum))
```

```
# method 2
odd_sum = 0
num = 100
while num > 0:
    if num % 2 == 1:
        odd_sum += num
    num -= 1

print('The odd sum is ' + str(odd_sum))
```

8. 编写程序，输出所有由 1、2、3、4 这四个数字组成的素数，并且在每个素数中每个数字只使用一次

```
num = [1, 2, 3, 4]
num_all = []
for i in range(0, len(num)):
    num[i], num[0] = num[0], num[i]
    for j in range(1, len(num)):
        num[j], num[1] = num[1], num[j]
        for k in range(2, len(num)):
            num[k], num[2] = num[2], num[k]
            num_all.append(int(str(
                num[0]) + str(num[1]) + str(num[2]) + str(num[3])))
            num[k], num[2] = num[2], num[k]
        num[j], num[1] = num[1], num[j]
    num[i], num[0] = num[0], num[i]

result = []
for number in num_all:
    for i in range(2, number):
        if not number % i:
            break
    else:
        result.append(number)

print("The result is: ", end="")
for number in result:
    print(number, end=" ")
```

9. 编写程序，实现分段函数计算，如表 3-1 所示

表 3-1　分段函数计算

x	y
$x<0$	0
$0\leqslant x<5$	x
$5\leqslant x<10$	$3x-5$
$10\leqslant x<20$	$0.5x-2$
$20\leqslant x$	0

```
x = int(input('Input x: '))
print('y = ', end='')
if x < 0 or x >= 20:
    print(0)
elif 0 <= x < 5:
    print(x)
elif 5 <= x < 10:
    print(3 * x - 5)
elif 10 <= x < 20:
    print(0.5 * x - 2)
```

第四章 字符串与正则表达式

1. 假设有一段英文，其中有单独的字母 l 误写为 i，请编写程序进行纠正

```
import re

text = '''
We were both young when i first saw you
i closed my eyes and the flashback starts
i'm standing there
On a balcony in summer air
See the lights see the party the ball gowns
i see you make your way through the crowd
And say hello
Little did i know
That you were Romeo you were throwing pebbles
And my daddy said stay away from Juliet
And i was crying on the staircase
Begging you please don't go
And i said
Romeo take me somewhere we can be alone
i'll be waiting all there's left to do is run
You'll be the prince and i'll be the princess
It's a love story
Baby just say yes
'''

text = re.sub(r'[i]\s', 'I ', text)
text = re.sub(r'[i]\'', 'I\'', text)
print(text)
```

2. 假设有一段英文，其中有单词中间的字母 i 误写为 l，请编写程序进行纠正

```
import re


def replace_I2i(text):
    pattern = re.compile(r'\w[I]\w')
    return re.sub(pattern, 'i', text)
```

3. 有一段英文文本，其中有单词连续重复了 2 次，编写程序检查重复的单词并只保留一个。例如，文本内容为“This is is a desk.”，程序输出为“This is a desk.”

```
import re
```

```
def sub_repeated_words(text):
    pattern = re.compile(r'\b(\w+)(\s+\1){1,}\b')  # \1 是引用第一个括号的内容

    while True:
        repeated_words = re.search(pattern, text)
        if not repeated_words:
            break
        text = re.sub(repeated_words.group(), repeated_words.group(1), text)
    return text


text = 'This is is a a a desk'
print(sub_repeated_words(text))
```

4. 简单解释 Python 的字符串驻留机制

对多个字符串类对象进行赋值，内存中只会有一个副本，这多个字符串对象将共享此副本

5. 编写程序，用户输入一段英文，然后输出这段英文中所有长度为 3 个字母的单词

```
import re

def get_3letters_words(text, num: int = 3):
    pattern = re.compile(r'\b\w{3}\b')
    words = set(re.findall(pattern, text))
    for word in words:
        print(word, end=' ')
```

第五章 函数设计与使用

1. 运行 5.3.1 节最后的示例代码，查看结果并分析原因。

函数的默认值参数只会被处理一次。若再次调用函数，此时默认值不会被刷新，会在上一个结果上进行操作

2. 编写函数，判断一个整数是否为素数，并编写主程序调用该函数

```
def is_prime(num: int) -> bool:
    for i in range(2, num):
        if not num % i:
            return False
    return True


if __name__ == '__main__':
    while True:
        num = int(input('Input a number: '))
        if is_prime(num):
            print(str(num)+' is a prime')
        else:
            print(str(num)+' is not a prime')
```

3. 编写函数，接收一个字符串，分别统计大写字母、小写字母、数字、其他字符的个数，并以元组的形式返回结果

```python
def count_char(string: str) -> dict:
    capital = 0
    small = 0
    number = 0
    other = 0
    for char in string:
        if "A" <= char <= "Z":
            capital += 1
        elif "a" <= char <= "z":
            small += 1
        elif "0" <= char <= "9":
            number += 1
        else:
            other += 1
    return {'capital letter': capital, 'small letter': small,
            'number': number, 'other character': other}
```

4. 填空

在函数内部可以通过关键字 `global` 来定义全局变量

5. 填空

如果函数中没有 return 语句或者 return 语句不带任何返回值，那么该函数的返回值为 `None`

6. 调用带有默认值参数的函数时，不能为默认值参数传递任何值，必须使用函数定义时设置的默认值（判断对错）

错

7. 在 Python 程序中，局部变量会隐藏同名的全局变量吗?

会

8. lambda 表达式只能用来创建匿名函数，不能为这样的函数起名字（判断对错）

错

9. 编写函数，可以接收任意多个整数并输出其中的最大值和所有整数之和

```python
def print_max_sum(*num):
    print("The max number is "+str(max(num)))
    print("The sum is " + str(sum(num)))
```

10. 编写函数，模拟内置函数 `sum()`

```python
def sum(numbers: list) -> int:
    sum_number = 0
    for number in numbers:
        sum_number += number
    return sum_number
```

11. 填空

包含 `yield` 语句的函数可以用来创建生成器

12. 编写函数，模拟内置函数 `sorted()`

```python
def sorted(iterable: list, reverse=False) -> list:
```

```
    result = []
    iterable_copy = iterable[:]
    if not reverse:
        while iterable_copy:
            min_elem = min(iterable_copy)
            result.append(min_elem)
            iterable_copy.remove(min_elem)
    else:
        while iterable_copy:
            max_elem = max(iterable_copy)
            result.append(max_elem)
            iterable_copy.remove(max_elem)
    return result
```

第六章 面向对象程序设计

1. 继承 6.5 节例 6-2 中的 Person 类生成 Student 类，编写新的函数用来设置学生专业，然后生成该类对象并显示信息

```
class Student(Person):
    def __init__(self, name='', age=30, sex='unknown', major='Computer'):
        super(Student, self).__init__(name, age, sex)
        self.SetMajor(major)

    def SetMajor(self, major):
        if not isinstance(major, str):
            print("Major must be a string.")
            return
        self.__major = major

    def Show(self):
        super(Student, self).Show()
        print(self.__major)
```

2. 设计一个三维向量类，并实现向量的加法、减法以及向量与标量的乘法和除法运算

```
class Vecter:
    def __init__(self, x=0, y=0, z=0):
        self._x = x
        self._y = y
        self._z = z

    def __add__(self, other):
        result = Vecter()
        result._x = self._x + other._x
        result._y = self._y + other._y
        result._z = self._z + other._z
        return result

    def __sub__(self, other):
        result = Vecter()
        result._x = self._x - other._x
        result._y = self._y - other._y
        result._z = self._z - other._z
        return result
```

```
    def __mul__(self, number):
        result = Vecter()
        result._x = self._x * number
        result._y = self._y * number
        result._z = self._z * number
        return result

    def __div__(self, number):
        result = Vecter()
        result._x = self._x / number
        result._y = self._y / number
        result._z = self._z / number
        return result
```

3. 填空

面向对象程序设计的三要素分别为**封装**、**继承**和**多态**

4. 简单解释 Python 中以下划线开头的变量名特点

1. _xxx：保护成员，不能用 `from module import *` 来导入，只有类对象和子类对象能访问这些成员
2. __xxx__：系统定义的特殊成员
3. __xxx：私有成员，只有类对象自己能访问，子类对象也不能访问到这个成员

5. 填空

与运算符 “**” 对应的特殊方法名为 `__pow__()`，与运算符 “//” 对应的特殊方法名为 `__floordiv__()`

6. 填空

假设 a 为类 A 的对象且包含一个私有数据成员“__value”，那么在类的外部通过对象 a 直接将其私有数据成员 “__value” 的值设置为 3 的语句可以写作 `a._A__value = 3`