第 1-6 章课后作业

2023 年 3 月 25 日

1 第 1-6 章课后作业

1.1 第一章

1. 简单说明如何选择正确的 Python 版本。

   选择 Python 版本时应根据自己的需求和必须的扩展库支持的版本决定

2. 为什么说 Python 采用的是基于值的内存管理模式?

   不同变量赋相同值时享用同一内存空间

[ ]:
```
a = 1
b = 1
print(id(a) == id(b))
```

```
True
```

3. 解释 Python 中的运算符 / 和 // 的区别

   / 为普通的浮点数除法，// 为整除

4. 在 Python 中导入模块中的对象有哪几种方式?

   `import 模块名 (as 别名)`

   `from 模块名 import 模块名/函数名/* (as 别名)`

5. Conda 和 PyPI 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。

   Python 的 `__name__` 变量储存当前运行的 Python 脚本或模块的名称，直接执行一段脚本的时候，这段脚本的 `__name__` 变量等于 `main`，当这段脚本被导入其他程序的时候，`__name__` 变量等于脚本本身的名字

7. 运算符可以对浮点数进行求与树操作。

8. 一个数字 5 是合法的 Python 表达式。

9. 在 Python 2.x 中，`input()` 函数接收到的数据类型由输入的内容，而在 Python 3.x 中概函数则认为接收到的用户输入数据一律为 `str`

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`，则程序输入 `12`（提示：使用整除运算）

[ ]:
```
# input 1234
print(int(input("请输入一个三位以上的整数: ")) // 100)
```

```
12
```

1.2 第二章

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?

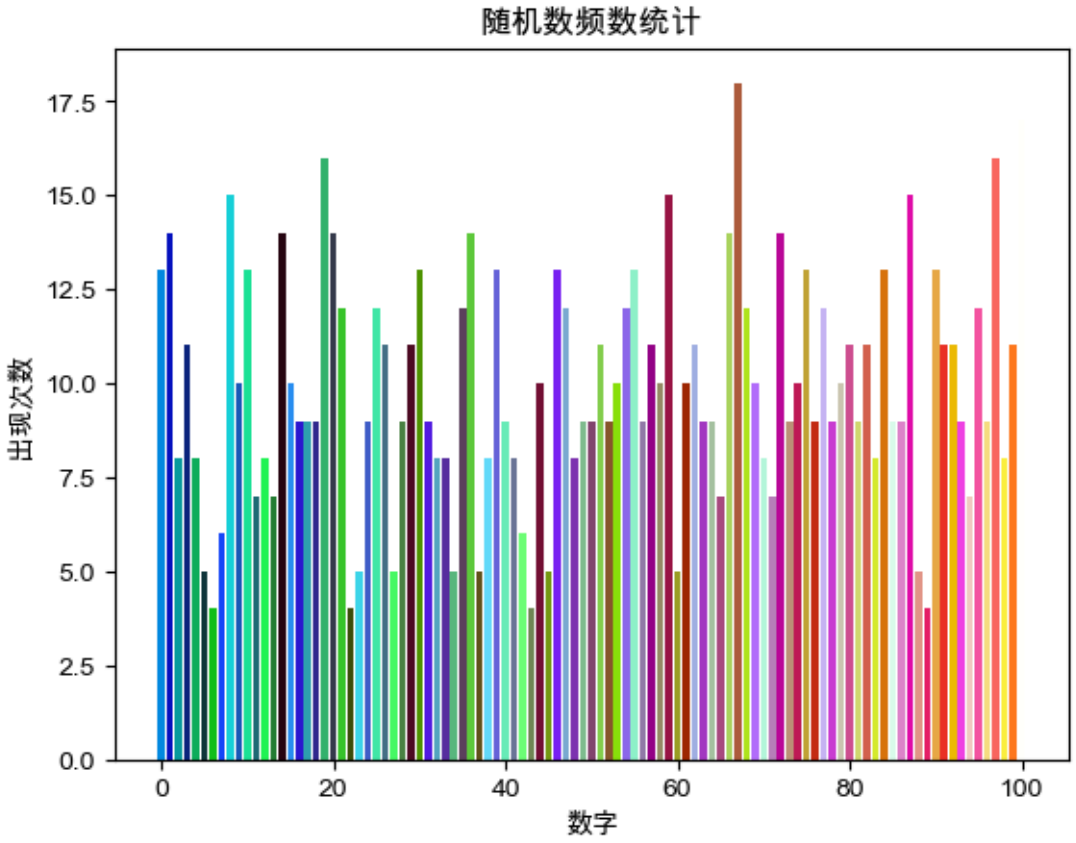
   因为 Python 中的列表底层是数组，因此尾部操作效率更高

2. `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
from matplotlib import pyplot as plt

nums = [random.randint(0, 100) for i in range(1000)]
x_data = [i for i in range(101)]
y_data = count = [0 for i in range(101)]

for i in nums:
    count[i] += 1

plt.rcParams['font.sans-serif'] = ['Arial Unicode MS']
plt.rcParams['axes.unicode_minus']=False

# 画图，plt.bar() 可以画柱状图
for i in range(len(x_data)):
    colors = str(hex(0xFFFFFF // 101 * (i + 1)))[2:]
    colors = "#" + ("0" if len(colors) < 6 else "") + colors
    plt.bar(x_data[i], y_data[i], color = colors)
# 设置图片名称
```

```
plt.title("随机数频数统计")
# 设置 x 轴标签名
plt.xlabel("数字")
# 设置 y 轴标签名
plt.ylabel("出现次数")
# 显示
plt.show()
```



4. 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字列表。例如用户输入 `[1, 2, 3, 4, 5, 6]` 和 `2, 5`，程序输出 `[3, 4, 5, 6]`。

[ ]:
```
# input: [1, 2, 3, 4, 5, 6] 2 5
nums = eval(input('list: '))
minimum = int(input('minimum: '))
```

```
maximum = int(input('maximum: '))
print(nums[minimum : maximum + 1])
```

```
[3, 4, 5, 6]
```

6. 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 NoneType

[ ]:
```
a = [1, 2, 3, 4, 5, 6]
print(type(a.sort()))
```

```
<class 'NoneType'>
```

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

8. 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 [6, 7, 9, 11]

9. 设计一个字典，并编写程序，用户输入内容作为“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“您输入的键不存在！”。

[ ]:
```
dictionary = {"a" : 1, "b" : 2, "c" : 3}
while True:
    key = input("请输入键: ")
    if key == "":
        break
    elif key in dictionary.keys():
        print("key: %s\nval: %d\n"%(key, dictionary[key]))
    else:
        print("key: %s\n您输入的键不存在！\n"%key)
```

```
key: a
val: 1

key: b
val: 2

key: c
val: 3

key: d
```

您输入的键不存在!

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。

[ ]:
```
from random import randint

nums = [randint(0, 100) for i in range(20)]
front = nums[:10]
front.sort()
back = nums[10:]
back.sort(reverse=True)

print(front)
print(back)
```

```
[9, 13, 16, 16, 33, 43, 50, 75, 86, 98]
[94, 82, 78, 70, 67, 49, 48, 31, 18, 1]
```

11. 在 Python 中，字典和集合都是用一对花括号作为界定符，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. 使用字典对象的 items() 方法可以返回字典的 “键-值对” 列表，使用字典对象的 keys() 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. 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化为字典，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. 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. 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. 可以使用 `del` 命令来删除元组中的部分数据。

1.3 第三章

1. 分析逻辑运算符 `or` 的短路求值特性。

   逻辑运算符 `or` 的短路求值特性：当 `or` 左边的表达式为 `True` 后，就不再计算右边的表达式，直接返回 `True`。

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。

[ ]:
```
for i in range(4):
    year = int(input("请输入年份 (四位整数): "))
    print("请输入年份 (四位整数): %d"%year)
    if year % 400 == 0 or year % 4 == 0 and year % 100 != 0:
        print("%s是闰年"%year)
    else:
        print("%s不是闰年"%year)
```

```
请输入年份 (四位整数): 2000
2000 是闰年
请输入年份 (四位整数): 1900
1900 不是闰年
请输入年份 (四位整数): 2020
2020 是闰年
请输入年份 (四位整数): 2023
2023 不是闰年
```

3. 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）。

[ ]:
```
from random import randint

nums = [randint(0, 100) for i in range(50)]

for i in range(len(nums) - 1, -1, -1):
    if nums[i] % 2:
        del nums[i]
print(nums)
```

```
[18, 22, 28, 2, 84, 72, 58, 42, 36, 14, 68, 100, 34, 30, 34, 70, 40, 96, 96, 86,
30, 90, 18, 88, 20, 0, 72]
```

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）。

[ ]:
```python
from random import randint

nums = [randint(1, 100) for i in range(20)]
nums_o = nums[::2]
nums_o.sort(reverse=True)

for i in range(10):
    nums[i * 2] = nums_o[i]
print(nums)
```

```
[86, 41, 75, 49, 65, 17, 60, 35, 59, 9, 49, 10, 31, 92, 31, 66, 21, 86, 7, 45]
```

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。例如，$10 = 2 \times 5$，$60 = 2 \times 2 \times 3 \times 5$。

[ ]:
```python
def IsPrime(num:'int > 0'):
    if num <= 1:
        return False
    if num == 2 or num == 3:
        return True
    if num % 2 == 0:
        return False
    for i in range(3, int(num ** 0.5) + 1, 2):
        if num % i == 0:
            return False
    return True

def Factorization(num:'int > 0'):
    temp = num
    factors = []
    min_factor = 2
    if IsPrime(num) or num == 1:
        factors.append(str(num))
        temp /= num
    while min_factor <= temp:
        if temp % min_factor == 0:
            factors.append(str(min_factor))
            temp /= min_factor
```

```
        else:
            min_factor += 1 if min_factor == 2 else 2
    print("%d = %s"%(num, ' * '.join(factors)))


num = int(input("请输入小于 1000 的整数以因式分解: "))
print("请输入小于 1000 的整数以因式分解: %d"%num)
Factorization(num)
```

```
请输入小于 1000 的整数以因式分解: 60
60 = 2 * 2 * 3 * 5
```

7. 编写程序，至少使用两种不同方法计算 100 以内所有奇数的和。

[ ]:
```
ans1 = sum([i for i in range(1, 100, 2)])
ans2 = int((1 + 99) * 50 / 2)
print("%d %d"%(ans1, ans2))
```

```
2500 2500
```

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

[ ]:
```
import itertools
from pprint import pprint

def IsPrime(num:'int > 0'):
    if num <= 1:
        return False
    if num == 2 or num == 3:
        return True
    if num % 2 == 0:
        return False
    for i in range(3, int(num ** 0.5) + 1, 2):
        if num % i == 0:
            return False
    return True

# 利用 itertools 库中的 permutations 函数，给定一个排列，输出他的全排列
```

```
def allPermutation(src:"list"):
    permutation = []
    for i in range(len(src)):
        permutation.append(src[i])
    # itertools.permutations 返回的只是一个对象，需要将其转化成 list
    # 每一种排列情况以元组类型存储
    all_permutation = list(itertools.permutations(permutation))
    return all_permutation

nums = [1, 2, 3, 4]
permutation = allPermutation([str(i) for i in nums])
pnums = [int(''.join(i)) for i in permutation]
for num in pnums:
    if IsPrime(num):
        print(num)
```

```
1423
2143
2341
4231
```

9. 编写程序，实现分段函数计算，如表 3-11111 所示。

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

[ ]:
```
def func(x):
    if x < 0:
        return 0
    if 0 <= x and x < 5:
        return x
    if 5 <= x and x < 10:
        return 3 * x - 5
```

```
    if 10 <= x and x < 20:
        return 0.5 * x - 2
    else:
        return 0

print(func(-1))
print(func(2))
print(func(8))
print(func(15))
print(func(25))
```

```
0
2
19
5.5
0
```

1.4 第四章

1. 假设有一段英文，其中有单独的字母 I 误写为 i，请编写程序进行纠正。

[ ]:
```
import re
# input: "i believe i can fly."
s = input("请输入一段英文: ")
print(f"请输入一段英文: {s}")
print(re.sub(r"\bi\b", "I", s))
```

```
请输入一段英文: i believe i can fly.
I believe I can fly.
```

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

[ ]:
```
import re
# input: "I belIeve I can fly."
s = input("请输入一段英文: ")
print(f"请输入一段英文: {s}")
print(re.sub(r"\BI\B", "i", s))
```

```
请输入一段英文: I belIeve I can fly.
I believe I can fly.
```

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值保留一个。例如，文本内容为 “This is is a desk.”，程序输出为 “This is is a desk.”

[ ]:
```
import re
from pprint import pprint
# input: "This is is a desk desk."
s = input("请输入一段英文: ")
print(f"请输入一段英文: {s}")
pattern = re.compile(r"\b(\w+)(\W+\1)+\b")
print(pattern.sub(r"\g<1>", s))# 网上找到的\g，不清楚具体实现的原理
```

```
请输入一段英文: This is is a desk desk.
This is a desk.
```

4. 简单解释 Python 的字符串驻留机制

   只保存一个相同且不可变的字符串, 不同的值存储在字符串的停留池中。python 的停留机制只保留一份相同字符串的副本。在后续创建相同的字符串时, 不会开辟新的空间, 而是将字符串的地址赋予新创建的变量。需要注意的是，长字符串不遵守驻留机制。

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
# input: "I believe I can fly."
s = input("请输入一段英文: ")
ans = re.findall(r"\b[A-Za-z]{3}\b", s)
print(f"str: {s}")
print(f"ans: {' '.join(ans)}")
```

```
str: I believe I can fly.
ans: can fly
```

1.5 第五章

1. 运行 5.3.1 节最后的事例代码，查看结果并分析原因。

[ ]:
```
def old_demo(newitem, old_list = []):
    old_list.append(newitem)
    return old_list
print(old_demo('5', [1, 2, 3, 4]))
print(old_demo('aaa', ['a', 'b']))
```

```
print(old_demo('a'))
print(old_demo('b'))
```

```
[1, 2, 3, 4, '5']
['a', 'b', 'aaa']
['a']
['a', 'b']
```

[ ]:
```
def new_demo(newitem, old_list = None):
    if old_list == None:
        old_list = []
    old_list.append(newitem)
    return old_list
print(new_demo('5', [1, 2, 3, 4]))
print(new_demo('aaa', ['a', 'b']))
print(new_demo('a'))
print(new_demo('b'))
```

```
[1, 2, 3, 4, '5']
['a', 'b', 'aaa']
['a']
['b']
```

1. 原因:

   多次调用两数并且不为默认值参数传递值时, 默认值参数只在第一次调用时进行解释，因此在首次调用 old_demo: `print(old_demo('a'))` 时，`old_list` 被初始化为 []，其地址固定，后续对于 `old_list` 的修改都基于该地址，后续调用的过程中不会再次解释 `old_list`，因此在 `print(old_demo('b'))` 时，`old_list = ['a']`，因此输出结果为 `['a', 'b']`

   而在 `new_demo` 中，`old_list` 被初始化为 `None`，在函数中被重新解释为空列表，因此当再次调用时，虽然 `old_list` 没有被重新解释，但其值仍指向 `None`。

2. 编写函数，判断一个整数是否为素数，饼编写主程序调用该函数。

[ ]:
```
def IsPrime(num:'int > 0'):
    if num <= 1:
        return False
    if num == 2 or num == 3:
        return True
```

```
    if num % 2 == 0:
        return False
    for i in range(3, int(num ** 0.5) + 1, 2):
        if num % i == 0:
            return False
    return True

if __name__ == '__main__':
    print(IsPrime(97))
```

```
True
```

3. 编写函数，接受一个字符串，分别统计大写字母、小写字母、数字、其他字符的个数，并以元组的方式返回结果。

[ ]:
```
def CharCount(string: "str"):
    upper = 0
    lower = 0
    digit = 0
    for char in string:
        if 'A' <= char and char <= 'Z':
            upper += 1
        elif 'a' <= char and char <= 'z':
            lower += 1
        elif '0' <= char and char <= '9':
            digit += 1
    return (upper, lower, digit, len(string) - upper - lower - digit)

if __name__ == "__main__":
    print(CharCount("4 you, A thousand times over."))
```

```
(1, 20, 1, 7)
```

4. 在函数内部可以通过关键字 `global` 来定义全局变量。
5. 如果函数中国呢没有 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（错）

7. 在 Python 程序中，局部变量会隐藏同名的全局变量吗？请编写代码进行验证。（会）

[ ]:
```
itest = 1

def fntest():
    itest = 2
    print(f"itest in fntest(): {itest}")

print(f"itest: {itest}")
fntest()
print(f"itest: {itest}")
```

```
itest: 1
itest in fntest(): 2
itest: 1
```

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（错）

9. 编写函数，可以接受任意多个整数并输出其中的最大值和所有整数之和。

[ ]:
```
def MaxandSum(*nums):
    return (max(nums), sum(nums))

print(MaxandSum(1, 2, 3, 4))
```

```
(4, 10)
```

10. 编写函数，模拟内置函数 sum()。

[ ]:
```
def my_sum(*nums):
    ans = 0
    if len(nums) >= 1:
        if type(nums[0]) in (list, tuple):
            for num in nums[0]:
                ans += num
        else:
            for num in nums:
                ans += num
    return ans
    # print(type(nums[0]))
print(my_sum(1, 2))
```

```
print(my_sum([1, 2]))
print(my_sum([]))
```

```
3
3
0
```

11. 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()。

[ ]:
```
from random import randint

def my_sort(data, reverse=True):
    if len(data) >= 2:
        mid = data[len(data)//2]
        left, right = [], []
        data.remove(mid)
        for num in data:
            if (num >= mid) == reverse:
                right.append(num)
            else:
                left.append(num)
        return my_sort(left, reverse) + [mid] + my_sort(right, reverse)
    else:
        return data

array = [randint(1, 1000) for i in range(100)]
print(my_sort(array))
print(my_sort(array, False))
```

```
[15, 24, 33, 36, 37, 39, 74, 88, 92, 95, 106, 134, 135, 160, 162, 163, 183, 196,
209, 239, 240, 247, 248, 251, 261, 267, 269, 270, 276, 301, 318, 335, 336, 336,
341, 342, 345, 396, 400, 412, 421, 424, 425, 460, 460, 463, 466, 498, 512, 524,
530, 548, 556, 561, 564, 567, 585, 586, 586, 603, 615, 615, 624, 627, 632, 633,
638, 651, 654, 658, 682, 699, 716, 719, 734, 737, 744, 746, 756, 775, 781, 781,
789, 806, 813, 821, 832, 845, 847, 865, 869, 874, 927, 945, 946, 958, 970, 970,
976, 994]
[994, 976, 970, 970, 958, 946, 945, 927, 874, 869, 865, 847, 845, 832, 821, 813,
```

```
806, 789, 781, 781, 775, 756, 746, 744, 737, 734, 719, 716, 699, 682, 658, 654,
651, 638, 633, 632, 627, 624, 615, 615, 603, 586, 586, 585, 567, 564, 561, 556,
548, 530, 524, 512, 498, 466, 463, 460, 460, 425, 424, 421, 412, 396, 345, 342,
341, 336, 336, 335, 318, 301, 276, 270, 269, 267, 261, 251, 248, 247, 240, 239,
209, 196, 183, 163, 162, 160, 135, 134, 106, 95, 92, 88, 74, 39, 37, 36, 33, 24,
15]
```

1.6 第六章

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。

[ ]:
```python
class Person:
    def __init__(self, name='', age=20, sex='man'):
        self.setName(name)
        self.setAge(age)
        self.setSex(sex)

    def setName(self, name: str):
        if not isinstance(name, str):
            print('name must be string.')
            return
        self.__name = name

    def setAge(self, age):
        if not isinstance(age, int):
            print('age must be integer.')
            return
        self.__age = age

    def setSex(self, sex: str):
        if sex != 'man' and sex != 'woman':
            print('sex must be "man" or "woman".')
            return
        self.__sex = sex

    def show(self):
        print('Name:', self.__name)
```

```
        print('Age:', self.__age)
        print('Sex:', self.__sex)

class Student(Person):
    def __init__(self, name = "", age = 20, sex = "man", profession = ""):
        super().__init__(name, age, sex)
        self.setProfession(profession)

    def setProfession(self, profession):
        if not isinstance(profession, str):
            print("profession must be a string")
            return
        self.__profession = profession

    def show(self):
        super().show()
        print('Profession:', self.__profession)

student = Student(name = "Milin", age = 20, sex = "man", profession = "Computer␣
 ↪Science")
student.show()
```

```
Name: Milin
Age: 20
Sex: man
Profession: Computer Science
```

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 Vector3D:
    def __init__(self, x, y, z):
        self.setX(x)
        self.setY(y)
        self.setZ(z)

    def setX(self, x):
        if not isinstance(x, int) and not isinstance(x, float):
```

```
            print('x must be number.')
            return
        self.__x = x

    def setY(self, y):
        if not isinstance(y, int) and not isinstance(y, float):
            print('y must be number.')
            return
        self.__y = y

    def setZ(self, z):
        if not isinstance(z, int) and not isinstance(z, float):
            print('z must be number.')
            return
        self.__z = z

    def __add__(self, other):
        return Vector3D(self.__x + other.__x, self.__y + other.__y, self.__z +␣
↪other.__z)

    def __sub__(self, other):
        return Vector3D(self.__x - other.__x, self.__y - other.__y, self.__z -␣
↪other.__z)

    def __mul__(self, other):
        return Vector3D(self.__x * other, self.__y * other, self.__z * other)

    def __truediv__(self, other):
        return Vector3D(self.__x / other, self.__y / other, self.__z / other)

    def __floordiv__(self, other):
        return Vector3D(self.__x // other, self.__y // other, self.__z // other)

    def __str__(self):
        return f'({self.__x}, {self.__y}, {self.__z})'
```

```
x = Vector3D(1, 2, 3)
y = Vector3D(4, 5, 6)

print(x + y)
print(x - y)
print(x * 3)
print(x / 2)
```

```
(5, 7, 9)
(-3, -3, -3)
(3, 6, 9)
(0.5, 1.0, 1.5)
```

3. 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. 特殊方法：变量名前后都为双下划线，用于特殊用途，如 `__init__` 构造函数，`__del__` 析构函数，`__add__` 加法运算符等

    2. 私有成员：双下划线前缀开头的变量为私有成员，解释器将重写属性名称为 `_(classname)__(membername)`

    3. 保护变量：单下划线前缀开头的变量为保护变量，不能由 `from module import *` 导入，只有类对象和子类对象能够访问这些变量

5. 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. 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` 直接将私有数据成员“`__value`”的值设置为 3 的语句可以写作 `a._A__value = 3`