关于本节中要掌握的内容

- List
- Plot

1 List

List由一组有序的元素组成，它是Python中非常通用的数据结构。它里面的元素可以是不同类型，可以将整数、字符串、浮点数、List放在同一个List中。List以左方括号开始，以右括号结束，每个元素使用逗号间隔。下面是一些示例：

```
list1 = [1,2,4,0,2,8]
list2 = [1,"234",5.06]
list3 = ["abc",True,"def"]
list4 = ["abc",["lalala","123abab"]]
list5 = []
```

- List可以是不同的类型
- List可以嵌套List
- 没有任何元素的list也是list

获取List中元素的值

因为List是有序的元素，所以我们可以使用它的索引（索引就是元素在List中的位置）来获取元素的值。但是需要注意的是索引是从0开始的，也就是List的第一个元素的索引是0。下面是一个使用示例：

```
list1 = [1,2,4,0,2,8]
print(f"List中的第1个元素是：{list1[0]},第3个元素是：{list1[2]}")
```

```
List中的第1个元素是：1,第3个元素是：4
```

- 下面的代码会输出什么？

```
list1 = [2,3,4]
s = 1+ list1[0] + list1[1] + list1[2] + list[3] + 5
print(f"1+2+3+4+5 = {s}")
```

是不是发现程序运行后生成了这样的错误：

```
IndexError: list index out of range
```

要注意：我们在使用List时，索引必须小于List的数目

- 当然，List的索引还有一个注意事项，必须使用整数类型，不能是字符串、浮点数等。例如：

```
l = [1,4,2,5]
print(l[2.3])
```

```
TypeError: list indices must be integers or slices, not float
```

- 思考一下下面的问题，对于一个嵌套的List，我们应该如何访问它的元素呢？

```
l = [1,"abc",[2,"def"]]
# 应该如何输出嵌套数组中的2和"def"呢？
```

对于嵌套的List元素，我们可以使用多重索引来访问它的元素，如下所示：

- 输出嵌套数组的元素

```
l = [1,"abc",[2,"def"]]
print(l[2][0],l[2][1])
```

- 输出

```
2 def
```

自己尝试输出下面嵌套数组的元素:

```
l1 = ["aaa","bbb"]
l2 = [1,2,"aaa",[],l1]
l3 = [2.3,True,l2,l1]
# 1.输出l3中的l2中l1中的第二个元素
# 2.输出l3中的l1中的第一个元素
```

- 输出嵌套数组的元素

```
print(l3[2][4][1])
print(l3[3][0])
```

- 输出

```
bbb
aaa
```

遍历List中的元素

前面我们访问了List中的单个元素，那么如何遍历List中的所有元素呢，这里需要借助到我们前面学到的for循环了：

```
l = ["aaa","bbb","ccc","ddd","eee","fff"]
for i in range(len(l)):
    print(f"{i}: {l[i]}")
```

> 这里我们使用了len函数获取到了List的元素个数，python中几乎所有数据结构的元素个数都可以使用len函数来获取哦

python中还有其他两种方式也可以遍历List中的元素

- 1.迭代器遍历

```
for it in l:
    print(f"{it}")
```

> 有没有发现，这里的迭代器示例代码没有使用i而使用it，一般的编程隐形规则中，i、j、k都意味着索引，it意味着数据结构中的元素

优点：几乎可以用于任何数据结构，使用起来非常简单
缺点：无法获取到当前循环的index

- 2.enumrate遍历

```
for idx,it in enumerate(l):
    print(f"{idx}: {it}")
```

优点：几乎可以用于任何数据结构，可以同时得到当前的循环索引和元素
缺点：使用起来稍微麻烦一些

- 我的建议是：不需要索引的遍历使用**方法1**，需要索引的遍历使用**方法2**

List遍历中的分支判断

一个数组中包含了学生的成绩，现在需要过滤掉低于60分的成绩，应该怎么做?

这里直观考虑有两种方法，一是将低于60分的成绩从数组中去除掉，二是将满足60以上的成绩放到另外一个数组中，然后覆盖掉

- 去除掉低于60分的成绩

```
scores = [52,99,100,42,70,80]
i = 0
while True:
    if i >= len(scores):
        break
    if scores[i] <60:
        scores.pop(i)
    else:
        i += 1
print(scores)
```

- 输出

```
[99, 100, 70, 80]
```

- 借助中间数组

```
scores = [52,99,100,42,70,80]
scores = [i for i in scores if i >= 60]
# temp = []
# for it in scores:
#     if it >= 60:
#         temp.append(it)
# scores = temp
print(scores)
```

- 输出

```
[99, 100, 70, 80]
```

list的切片

当我们想要获取list中的某一部分时，就需要使用到list的切片功能了。

```
l1 = ["a","b","c","d","e"]
l2 = l1[1:4]
print(l2)
l3 = l1[:5]
print(l3)
l4 = l1[2:]
print(l4)
```

```
['b', 'c', 'd']
['a', 'b', 'c', 'd', 'e']
['c', 'd', 'e']
```

- 切片包含了切的开始、结束、冒号，要注意区分索引和切片
- 切片的结束是不包含的

- 尝试获取下面的list切片并输出

```
['c', 'd', 'e']
['a', 'b', 'c', 'd']
['b', 'c', 'd', 'e']
```

list的添加和修改

现在我们有一个list a，我们想让它变成list b

```
a = ["a","b","c"]
# b = ["b","b","c","e","f"]
a[0] = "b"
a.append("e")
a.append("f")
print(a)
```

- 只需要通过索引就可以修改list的元素，要注意不要越界
- 通过append()函数可以在list后面添加元素

- 尝试一下将下面的list a，变成list b

```
a = ["a","b","c"]
# b = ["c","c","c","c","a"]
```

list的拼接

列表也可以被连接和复制。我们使用"+"拼接两个列表，得到一个新李彪；我们使用"*"复制一份列表。如下所示：

```
l = [1,2,3,4,5]
l1 = l + l + [6,7,8]
l2 = l*3
print(l1)
print(l2)
```

- 输出

```
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 6, 7, 8]
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

实际应用

1.将下面的list变成字符串，倒数第二个字符串前面加上--

```
# 原始list
l = ["a","b","c","d"]
# 应输出
"a,b,--c,d"
# 尝试改变下面的list
l1 = ["a","b","c","d"]
l2 = ['apples', 'bananas', 'cats']
```

2.尝试将以下以下图形保存到list中，并输出

```
*
**
***
****
*****
```

```
    *
   ***
  *****
 *******
*********
```

3.计算并输出在l1中的l2的元素

```
l1 = [0,1,2,3,4,5,6,7,8,9]
l2 = [3,4,8]
```

pyplot

matplotlib是一个python 2D绘图库，它里面包含了pyplot这个模块。

- 安装matplotlib

```
pip install matplotlib
```

- 简单绘图

```
# 导入matplotlib的pyploy模块，并设置别名为plt
import matplotlib.pyplot as plt

# 绘图，第一个参数为横坐标，第二个参数为纵坐标
plt.plot([1, 2, 3, 4], [1, 3, 5, 7])
# 显示绘图窗口
plt.show()
```

更复杂的一些设置

```
import matplotlib.pyplot as plt

# plt.plot([1, 2, 3, 4], [1, 3, 5, 7])
# 设置显示的大小
plt.figure(figsize=(6, 3))
# 设置标题
plt.title("title")
# blue orbicular
# red  -
plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'ro',label = 'tab 1')
# 图列显示的位置
plt.legend(loc='best')
# 设置x轴范围0-6, y轴0-20
plt.axis((0, 6, 0, 20))
plt.show()
```

倍投法则（带图例）

```
import random
import matplotlib.pyplot as plt
# 游戏开始，掷骰子，玩家选择大或小
# 玩家选择正确，则从1开始，输了翻倍，赢了继续从1开始

balance = 1000000
bets = 100
balance -= bets
ypoints = []
xpoints = []
for i in range(100000):
    # balance -= 10
    ypoints.append(balance)
    # win
    if random.random()>=0.5:
        balance += (bets*2)
        bets = 100
    else:
        bets *= 2
        balance -= bets
        if balance<0:
            ypoints.append(balance)
            print('game over')
            break
for i in range(len(ypoints)):
    xpoints.append(i)
plt.plot(xpoints, ypoints)
plt.show()
```

自己尝试

- 现在有n个人，初始都有m点能量，现在进行游戏，每轮每个人都可能会随机从某个人身上抢到1点能量；也可能抢不到。能量可以为负数，进行x轮，输出图像