实验四 Python字典和while循环

班级： 21计科3班

学号： B20200202217

姓名： 黄澍

Github地址：https://github.com/Hassium1/python_course

CodeWars地址：https://www.codewars.com/users/Hassium1

实验过程与结果

第一部分

第6章 字典

练习 6.1：人　使用一个字典来存储一个人的信息，包括名、姓、年龄和居住的城市。该字典应包含键 first_name、last_name、age 和 city。
将存储在该字典中的每项信息都打印出来。

```
Person={'first_name':'john','last_name':'smith','age':30,'city':'new york'}
print(Person['first_name'])
print(Person['last_name'])
print(Person['age'])
print(Person['city'])
```

结果：
john
smith
30
new york

练习 6.2：喜欢的数 1　使用一个字典来存储一些人喜欢的数。请想出 5 个人的名字，并将这些名字用作字典中的键。再想出每个人喜欢的一个数，并将这些数作为值存储在字典中。打印每个人的名字和喜

欢的数。为了让这个程序更有趣，通过询问朋友确保数据是真实的。

```
Person_number={'john':13,'tom':7,'jerry':9,'lucy':77,'lily':15}
print('john',Person_number['john'])
print('tom',Person_number['tom'])
print('jerry',Person_number['jerry'])
print('lucy',Person_number['lucy'])
print('lily',Person_number['lily'])
```

结果：
john 13
tom 7
jerry 9
lucy 77
lily 15

练习 6.5：河流　创建一个字典，在其中存储三条河流及其流经的国家。例如，一个键值对可能是 'nile': 'egypt'。
● 使用循环为每条河流打印一条消息，如下所示。
The Nile runs through Egypt.
● 使用循环将该字典中每条河流的名字打印出来。
● 使用循环将该字典包含的每个国家的名字打印出来。

```
River={'nile':'egypt','yangtze':'china','mississippi':'usa'}
for i in River:
    print('The',i,'runs through',Rive[i])
```

结果：
The nile runs through egypt
The yangtze runs through china
The mississippi runs through usa

```
River={'nile':'egypt','yangtze':'china','mississippi':'usa'}
for river in River.keys():
    print(river)
```

结果：
nile

yangtze
mississippi

```
River={'nile':'egypt','yangtze':'china','mississippi':'usa'}
for name in River.values():
    print(name)
```

结果:
egypt
china
usa

练习 6.11：城市　创建一个名为 cities 的字典，将三个城市名用作键。对于每座城市，都创建一个字典，并在其中包含该城市所属的国家、人口约数以及一个有关该城市的事实。表示每座城市的字典都应包含 country、population 和 fact 等键。将每座城市的名字以及相关信息都打印出来。

```
cities={'beijing':{'country':'china','population':1000,'fact':'beijing is a capital of china'},
        'shanghai':{'country':'china','population':1000,'fact':'shanghai is a capital of china'},
        'tokyo':{'country':'japan','population':1000,'fact':'tokyo is a capital of japan'}}
for cityname,info in cities.items():
    print(f"cityname:{cityname}")
    print(f"country:{info['country']}")
    print(f"population:{info['population']}")
    print(f"fact:{info['fact']}")
```

结果:

cityname:beijing

country:china

population:1000

fact:beijing is a capital of china

cityname:shanghai

country:china

population:1000

fact:shanghai is a capital of china

cityname:tokyo

country:japan

population:1000

fact:tokyo is a capital of japan

第二部分

第一题：淘气还是乖孩子（Naughty or Nice）

难度： 7kyu

圣诞老人要来镇上了，他需要你帮助找出谁是淘气的或善良的。你将会得到一整年的JSON数据，按照这个格式：

```
{
    January: {
        '1': 'Naughty','2': 'Naughty', ..., '31': 'Nice'
    },
    February: {
        '1': 'Nice','2': 'Naughty', ..., '28': 'Nice'
    },
    ...
    December: {
        '1': 'Nice','2': 'Nice', ..., '31': 'Naughty'
    }
}
```

你的函数应该返回 "Naughty!"或 "Nice!"，这取决于在某一年发生的总次数（以较大者为准）。如果两者相等，则返回 "Nice！"。

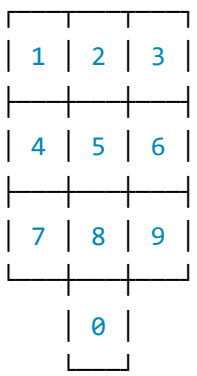
```python
def naughty_or_nice(data):
    naughty_count = 0
    nice_count = 0

    for month in data:
        for day in data[month]:
            if data[month][day] == 'Naughty':
                naughty_count += 1
            elif data[month][day] == 'Nice':
                nice_count += 1

    if naughty_count > nice_count:
        return "Naughty!"
    elif naughty_count < nice_count:
        return "Nice!"
    else:
        return "Nice!"
```

第二题： 观察到的PIN（The observed PIN）

难度：4kyu

好了，侦探，我们的一个同事成功地观察到了我们的目标人物，抢劫犯罗比。我们跟踪他到了一个秘密仓库，我们认为在那里可以找到所有被盗的东西。这个仓库的门被一个电子密码锁所保护。不幸的是，我们的间谍不确定他看到的密码，当罗比进入它时。

键盘的布局如下：



他注意到密码1357，但他也说，他看到的每个数字都有可能是另一个相邻的数字（水平或垂直，但不是对角线）。例如，代替1的也可能是2或4。而不是5，也可能是2、4、6或8。

他还提到，他知道这种锁。你可以无限制地输入错误的密码，但它们最终不会锁定系统或发出警报。这就是为什么我们可以尝试所有可能的（*）变化。

*可能的意义是：观察到的PIN码本身和考虑到相邻数字的所有变化。

你能帮助我们找到所有这些变化吗？如果有一个函数，能够返回一个列表，其中包含一个长度为1到8位的观察到的PIN的所有变化，那就更好了。我们可以把这个函数命名为getPINs（在python中为get_pins，在C#中为GetPINs）。

但请注意，所有的PINs，包括观察到的PINs和结果，都必须是字符串，因为有可能会有领先的 "0"。我们已经为你准备了一些测试案例。
侦探，我们就靠你了!

```
def get_pins(observed):
    adjacent_digits = {
        '0': ['0', '8'],
        '1': ['1', '2', '4'],
        '2': ['1', '2', '3', '5'],
        '3': ['2', '3', '6'],
        '4': ['1', '4', '5', '7'],
        '5': ['2', '4', '5', '6', '8'],
        '6': ['3', '5', '6', '9'],
        '7': ['4', '7', '8'],
        '8': ['0', '5', '7', '8', '9'],
        '9': ['6', '8', '9']
    }

    # 初始化结果列表
    result = []

    # 对于每个观察到的数字
    for digit in observed:
        # 获取与该数字相邻的所有可能数字
        adjacent = adjacent_digits[digit]
        # 如果结果列表为空，则直接将相邻数字添加到结果中
        if not result:
            result.extend(adjacent)
        else:
            # 否则，将结果中的每个PIN码与相邻数字组合生成新的PIN码
            new_result = []
            for pin in result:
                new_result.extend([pin + adj for adj in adjacent])
            result = new_result

    return result
```

第四题： 填写订单（Thinkful - Dictionary drills: Order filler）

难度：8kyu

您正在经营一家在线业务，您的一天中很大一部分时间都在处理订单。随着您的销量增加，这项工作占用了更多的时间，不幸的是最近您遇到了一个情况，您接受了一个订单，但无法履行。

您决定写一个名为 `fillable()` 的函数，它接受三个参数：一个表示您库存的字典 `stock` ，一个表示客户想要购买的商品的字符串 `merch` ，以及一个表示他们想购买的商品数量的整数n。如果您有足够的商品

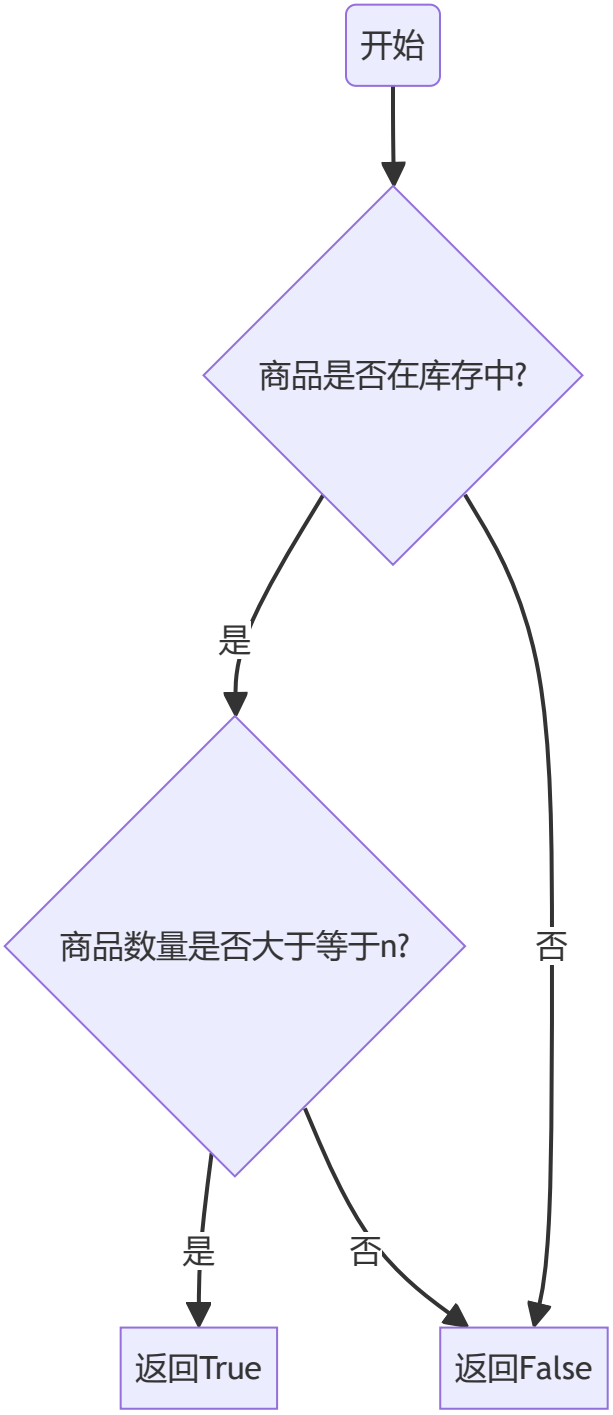
库存来完成销售，则函数应返回 `True` ， 否则应返回 `False` 。

有效的数据将始终被传入，并且n将始终大于等于1。

```
def fillable(stock, merch, n):
    # Your code goes here.
    if merch in stock:
        # 如果商品数量大于或等于n，则返回True
        if stock[merch] >= n:
            return True
        # 否则，返回False
        else:
            return False
    # 如果商品不在库存字典中，返回False
    else:
        return False
```

第三部分

第四题： 填写订单（Thinkful - Dictionary drills: Order filler）



实验考查

1. 字典的键和值有什么区别?

答：键（key）是字典中用于索引和唯一标识值的部分。每个键必须是唯一的，且只能包含不可变的数据类型（如字符串、数字或元组）作为键。
值（value）是与键相关联的数据。值可以是任意类型的对象，包括字符串、数字、列表、元组、字典等。

2.在读取和写入字典时，需要使用默认值可以使用什么方法?

答：在读取和写入字典时，可以使用dict.get()方法来获取字典中的值并设置默认值。该方法接受一个键作为参数，并返回与该键关联的值。如果字典中不存在该键，则返回指定的默认值（如果提供了默认值），否则返回None

3.Python中的while循环和for循环有什么区别?

答：while循环是根据条件表达式的布尔值来重复执行一段代码，直到条件为False为止。它用于在不确定循环次数的情况下执行代码块。
for循环是用于遍历可迭代对象（如列表、元组、字符串等）中的元素。它会依次取出可迭代对象中的每个元素，并执行相应的代码块。

4.阅读PEP 636 – Structural Pattern Matching: Tutorial, 总结Python 3.10中新出现的match语句的使用方法。

答：PEP 636 – Structural Pattern Matching: Tutorial是一个关于Python 3.10中新增的匹配语句（match statement）的教程。该语句是一种用于模式匹配的方式，类似于其他编程语言中的switch语句，可以根据不同的模式执行相应的代码块。
根据PEP 636中给出的教程，Python 3.10中的匹配语句使用match关键字，并且可以与case子句结合使用。

实验总结

本次实验主要学习了字典的常用操作，包括创建字典、访问字典元素、修改字典元素、删除字典元素、字典的遍历等。还学习了match语句，该语句是一种用于模式匹配的方式，类似于其他编程语言中的switch语句，可以根据不同的模式执行相应的代码块。以及while循环和for循环的区别。