

实验四 Python字典和while循环

班级：21计科1

学号：B20210302123

姓名：何香仪

Github地址：<https://github.com/deliciousbeef/python>

CodeWars地址：<https://www.codewars.com/users/何屁屁>

实验目的

1. 学习Python字典
2. 学习Python用户输入和while循环

实验环境

1. Git
2. Python 3.10
3. VSCode
4. VSCode插件

实验内容和步骤

第一部分

Python列表操作

完成教材《Python编程从入门到实践》下列章节的练习：

- 第6章 字典
 - 第7章 用户输入和while循环
-

第二部分

在[Codewars网站](#)注册账号，完成下列Kata挑战：

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 !"。代码提交地址：<https://www.codewars.com/kata/5662b14e0a1fb8320a00005c>

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：

1	2	3
4	5	6
7	8	9
	0	

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! 代码提交地址：

<https://www.codewars.com/kata/5263c6999e0f40dee200059d>

第三题：RNA到蛋白质序列的翻译 (RNA to Protein Sequence Translation)

难度：6kyu

蛋白质是由DNA转录成RNA，然后转译成蛋白质的中心法则。RNA和DNA一样，是由糖骨架（在这种情况下是核糖）连接在一起的长链核酸。每个由三个碱基组成的片段被称为密码子。称为核糖体的分子机器将RNA密码子转译成氨基酸链，称为多肽链，然后将其折叠成蛋白质。

蛋白质序列可以像DNA和RNA一样很容易地可视化，作为大字符串。重要的是要注意，“停止”密码子不编码特定的氨基酸。它们的唯一功能是停止蛋白质的转译，因此它们不会被纳入多肽链中。“停止”密码子不应出现在最终的蛋白质序列中。为了节省您许多不必要（和乏味）的键入，已为您的氨基酸字典提供了键和值。

给定一个RNA字符串，创建一个将RNA转译为蛋白质序列的函数。注意：测试用例将始终生成有效的字符串。

```
protein ('UGCGAUGAAUGGGCUCGCUCC')
```

将返回CDEWARS

作为测试用例的一部分是一个真实世界的例子！最后一个示例测试用例对应着一种叫做绿色荧光蛋白的蛋白质，一旦被剪切到另一个生物体的基因组中，像GFP这样的蛋白质可以让生物学家可视化细胞过程！

Amino Acid Dictionary

```
# Your dictionary is provided as PROTEIN_DICT
PROTEIN_DICT = {
    # Phenylalanine
    'UUC': 'F', 'UUU': 'F',
    # Leucine
    'UUA': 'L', 'UUG': 'L', 'CUU': 'L', 'CUC': 'L', 'CUA': 'L', 'CUG': 'L',
    # Isoleucine
    'AUU': 'I', 'AUC': 'I', 'AUA': 'I',
    # Methionine
    'AUG': 'M',
    # Valine
    'GUU': 'V', 'GUC': 'V', 'GUA': 'V', 'GUG': 'V',
    # Serine
    'UCU': 'S', 'UCC': 'S', 'UCA': 'S', 'UCG': 'S', 'AGU': 'S', 'AGC': 'S',
    # Proline
    'CCU': 'P', 'CCC': 'P', 'CCA': 'P', 'CCG': 'P',
    # Threonine
    'ACU': 'T', 'ACC': 'T', 'ACA': 'T', 'ACG': 'T',
    # Alanine
    'GCU': 'A', 'GCC': 'A', 'GCA': 'A', 'GCG': 'A',
    # Tyrosine
    'UAU': 'Y', 'UAC': 'Y',
```

```
# Histidine
'CAU': 'H', 'CAC': 'H',
# Glutamine
'CAA': 'Q', 'CAG': 'Q',
# Asparagine
'AAU': 'N', 'AAC': 'N',
# Lysine
'AAA': 'K', 'AAG': 'K',
# Aspartic Acid
'GAU': 'D', 'GAC': 'D',
# Glutamic Acid
'GAA': 'E', 'GAG': 'E',
# Cystine
'UGU': 'C', 'UGC': 'C',
# Tryptophan
'UGG': 'W',
# Arginine
'CGU': 'R', 'CGC': 'R', 'CGA': 'R', 'CGG': 'R', 'AGA': 'R', 'AGG': 'R',
# Glycine
'GGU': 'G', 'GGC': 'G', 'GGA': 'G', 'GGG': 'G',
# Stop codon
'UAA': 'Stop', 'UGA': 'Stop', 'UAG': 'Stop'
}
```

代码提交地址：<https://www.codewars.com/kata/555a03f259e2d1788c000077>

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。

代码提交地址：<https://www.codewars.com/kata/586ee462d0982081bf001f07/python>

第五题：莫尔斯码解码器 (**Decode the Morse code, advanced**)

难度：4kyu

在这个作业中，你需要为有线电报编写一个莫尔斯码解码器。有线电报通过一个有按键的双线路运行，当按下按键时，会连接线路，可以在远程站点上检测到。莫尔斯码将每个字符的传输编码为"点"（按下按键的短按）和"划"（按下按键的长按）的序列。

在传输莫尔斯码时，国际标准规定：

- "点" - 1个时间单位长。
- "划" - 3个时间单位长。
- 字符内点和划之间的暂停 - 1个时间单位长。
- 单词内字符之间的暂停 - 3个时间单位长。
- 单词间的暂停 - 7个时间单位长。

但是，该标准没有规定"时间单位"有多长。实际上，不同的操作员会以不同的速度进行传输。一个业余人士可能需要几秒钟才能传输一个字符，一位熟练的专业人士可以每分钟传输60个单词，而机器人发射器可能会快得多。

在这个作业中，我们假设消息的接收是由硬件自动执行的，硬件会定期检查线路，如果线路连接（远程站点的按钮按下），则记录为1，如果线路未连接（远程按钮弹起），则记录为0。消息完全接收后，它会以一个只包含0和1的字符串的形式传递给你进行解码。

例如，消息HEY JUDE，即..... - - - - - . - - - - - . - - - - - . - - - - - .可以如下接收：

```
1100110011001100000011000000111111001100111111001111110000000000000011001111110011
1111001111110000001100110011111100000011111100110011000000011
```

如您所见，根据标准，这个传输完全准确，硬件每个"点"采样了两次。

因此，你的任务是实现两个函数：

函数decodeBits(bits)，应该找出消息的传输速率，正确解码消息为点（.）、划（-）和空格（字符之间有一个空格，单词之间有三个空格），并将它们作为一个字符串返回。请注意，在消息的开头和结尾可能会出现一些额外的0，确保忽略它们。另外，如果你无法分辨特定的1序列是点还是划，请假设它是一个点。

函数decodeMorse(morseCode)，它将接收上一个函数的输出，并返回一个可读的字符串。

注意：出于编码目的，你必须使用ASCII字符.和-，而不是Unicode字符。

莫尔斯码表已经预加载给你了（请查看解决方案设置，以获取在你的语言中使用它的标识符）。

```
morseCodes(".-") #to access the morse translation of ".-"
```

下面是Morse码支持的完整字符列表：

A	.-
B
C
D	...
E	.
F
G	--.
H
I	..
J

K	---
L
M	--
N	..
O	---
P
Q	----
R	...-
S	...
T	-
U	..-
V	...-
W	---
X	----
Y	----
Z
0	-----
1
2
3
4
5
6
7
8
9
.
,
?
'
!
/
(.....
)
&
:
;
=
+
-
_
"
\$
@

代码提交地址：<https://www.codewars.com/kata/decode-the-morse-code-advanced>


第三部分

使用Mermaid绘制程序流程图

安装VSCode插件：

- Markdown Preview Mermaid Support
- Mermaid Markdown Syntax Highlighting

使用Markdown语法绘制你的程序绘制程序流程图（至少一个），Markdown代码如下：

程序流程图

显示效果如下：

```
flowchart LR
    A[Start] --> B{Is it?}
    B -->|Yes| C[OK]
    C --> D[Rethink]
    D --> B
    B ---->|No| E[End]
```

查看Mermaid流程图语法-->[点击这里](#)

使用Markdown编辑器（例如VScode）编写本次实验的实验报告，包括[实验过程与结果](#)、[实验考查](#)和[实验总结](#)，并将其导出为 **PDF格式** 来提交。

实验过程与结果

请将实验过程与结果放在这里，包括：

- [第一部分 Python列表操作和if语句](#)
- [第二部分 Codewars Kata挑战](#)

第一题

```
def naughty_or_nice(data):
    cnt1 = cnt2 = 0 #cnt1记录淘气孩子的人数，cnt2记录乖孩子的人数
    for months, days in data.items():# 遍历顶层键值对
        for k, v in days.items():# 遍历内层键值对
            if v == "Naughty": cnt1 += 1
            else: cnt2 += 1
    if cnt1 > cnt2: return "Naughty!"
    return "Nice!"
```

```
graph TD;
    A[Start] --> B{遍历日期}
    B -->|Nice| C[cnt2++]
    B -->|Naughty| D[cnt1++]
    D --> E{Nice>Naughty?}
    C --> E
    E -->|yes| F[return Nice]
    E -->|no| G[return Naughty]
```

```
F---->H[end]
G---->H[end]
```

第二题

```
def get_pins(observed):
    # 创建一个字典，用于存储每个数字对应的相邻数字
    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': ['5', '7', '8', '9', '0'],
        '9': ['6', '8', '9']
    }

    # 如果观察到的PIN码只有一位，直接返回相邻数字
    if len(observed) == 1:
        return adjacent_digits[observed]

    # 递归生成PIN码变化
    sub_pins = get_pins(observed[1:])
    observed_digit = observed[0]

    # 将观察到的数字与后续数字的变化组合
    result = [digit + sub for digit in adjacent_digits[observed_digit]
               for sub in sub_pins]

    return result
```

第三题

```
def protein(rna):
    protein_sequence = ""
    i = 0

    while i < len(rna):
        codon = rna[i:i+3] # 从RNA中获取三个碱基的密码子
        # 使用字典查找对应的氨基酸
        amino_acid = PROTEIN_DICT[codon]
        if amino_acid == 'Stop':
            break # 如果遇到停止密码子，停止翻译
```



```
protein_sequence += amino_acid
i += 3 # 移动到下一个密码子

return protein_sequence
```

第四题

```
def fillable(stock, merch, n):
    # 检查库存字典中是否有商品
    if merch not in stock:
        return False
    # 检查库存是否足够
    if stock[merch] < n:
        return False
    else:
        stock[merch] -= n # 减少库存
        return True
stock = {"shirt": 10, "pants": 5, "hat": 20}
print(fillable(stock, "shirt", 3)) # True · 我们有足够的衬衫
print(fillable(stock, "hat", 15)) # False · 我们的帽子库
```

第五题

```
def decode_bits(bits):
    # 去除开头和结尾的0
    bits = bits.strip("0")

    # 计算最小单位的长度
    unit = 0
    for bit in bits:
        if bit != "0":
            unit += 1
        else:
            break

    count = 1
    for i in range(1, len(bits)):
        if bits[i] == bits[i-1]:
            count += 1
        else:
            # 如果当前的连续计数小于最小单位长度，则更新最小单位长度
            if count < unit:
                unit = count
                count = 1
            else:
                count = 1

    morse_code = ""
```

```

# 按照单词分割
words = bits.split("0" * 7 * unit)
for word in words:
    # 按照字符分割
    characters = word.split("0" * 3 * unit)
    for character in characters:
        # 按照最小单位长度分割
        signs = character.split("0" * unit)
        for sign in signs:
            if sign == "1" * 3 * unit:
                morse_code += "-"
            else:
                morse_code += "."
        morse_code += " "
    morse_code += " "

return morse_code

def decode_morse(morseCode):
    # 去除开头和结尾的空格
    morseCode.strip()

    result = ""
    characters = morseCode.split(" ")
    for character in characters:
        if character != "":
            result += MORSE_CODE[character]
        else:
            result += " "

    return ' '.join(result.split())

```

- [第三部分 使用Mermaid绘制程序流程图](#) 第一题

```

graph TD;
    A[Start] --> B{遍历日期}
    B -->|Nice| C[cnt2++]
    B -->|Naughty| D[cnt1++]
    D --> E{Nice>Naughty?}
    C --> E{Nice>Naughty?}
    E -->|yes| F[return Nice]
    E -->|no| G[return Naughty]
    F ----> H[end]
    G ----> H[end]

```

第四题

```
flowchart LR
```

```

A[开始] --> B{商品字符串是否在库存字典里面}
B -->|是| C{库存中商品数量是否大于等于客户所需商品数量}
C -->|是| D[返回 True]
C -->|否| E[返回 False]
B -->|否| F[返回 False]
D --> G[结束]
E --> G[结束]
F --> G[结束]

```

注意：不要使用截图，Markdown文档转换为Pdf格式后，截图可能会无法显示。

实验考查

请使用自己的语言并使用尽量简短代码示例回答下面的问题，这些问题将在实验检查时用于提问和答辩以及实际的操作。

1. 字典的键和值有什么区别？

- 键必须是唯一的，但值则不必。值可以取任何数据类型，但键必须是不可变的，如字符串，数字或元组。

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？

- 在Python中，当尝试读取字典中不存在的键时，会引发一个KeyError异常。如果希望在读取字典时使用一个默认值，而不是抛出异常，可以使用 dict.get() 方法。这个方法接受两个参数：键和默认值。如果键存在于字典中，get() 方法将返回键对应的值；如果键不存在，它将返回默认值。同样地，如果想在写入字典时使用默认值，而不是覆盖现有的值，可以使用 dict.setdefault() 方法。这个方法接受两个参数：键和默认值。如果键已经存在于字典中，setdefault() 方法将不进行任何操作。如果键不存在，它将把默认值添加到字典中。

3. Python中的while循环和for循环有什么区别？

- 适用场景不同：while循环适用于当你知道循环应该执行的大致次数，但这个次数不是确切的，或者你需要在满足特定条件时执行循环的情况。for循环适用于当你知道要循环处理的对象或数据的范围时，例如遍历一个列表、字典或range对象。
- 执行方式不同：while循环会一直执行，直到条件不再满足。如果不满足条件，循环将停止。for循环会执行指定的次数，每次执行完毕，计数器会自动加一，直到达到指定的次数。
- 代码可读性不同：在大多数情况下，for循环的代码更具有可读性，因为它明确地表明了正在遍历一个序列或集合。while循环的代码可能会更复杂，因为你需要自己管理循环的计数器或条件。
- 性能差异：在一些情况下，for循环可能会比while循环更高效，特别是在遍历大型数据集时，因为Python的内部实现为for循环提供了优化。
- 其他区别：while循环可以包含break语句来提前结束循环，而for循环不能。for循环可以包含continue语句来跳过当前循环的剩余部分并立即开始下一次循环，而while循环中的continue语句的行为可能不那么直观。for循环常常与Python的生成器或列表解析一起使用，以简化代码并提高效率。

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。

- `match`语句基于一种称为"模式 (Pattern) "的概念，用于对传入的表达式进行结构化比较。当模式匹配成功时，将执行相应的代码块。它主要对等幂的抽象语法树 (AST) 进行操作，也可以直接用于基本的Python数据类型，如列表、元组、字典等。
- 下面是`match`语句的基本使用方法：

`python value = ... # 需要进行模式匹配的值`

`match value:`

`case 1:`

`# 当value等于1时执行的代码`

`case 2:`

`# 当value等于2时执行的代码`

`case _:`

`# 当value不等于1或2时执行的代码` 在上述代码中，`case`后面跟着的值是模式，用于与`value`进行比较。如果模式匹配成功，则执行相应的代码块。如果所有的模式都不匹配，那么执行最后的`case:`代码块。如果没有模式能够匹配，且没有默认的`case:`代码块，那么将引发`MatchError`异常。

此外，你还可以使用一些特殊模式进行更复杂的匹配操作，例如模式、捕获模式、可选模式等。此外，Python 3.10还引入了`case`语句，它允许在模式匹配中进行更复杂的条件判断。

结构性模式匹配在处理复杂数据类型和多种可能情况时非常有用，它可以使代码更加简洁、易读和易于维护。虽然这个特性在Python 3.10中才正式引入，但在之前的版本中已经可以通过使用第三方库（如`patternmatch`）来使用类似的功能。

实验总结

总结一下这次实验你学习和使用到的知识，例如：编程工具的使用、数据结构、程序语言的语法、算法、编程技巧、编程思想。