

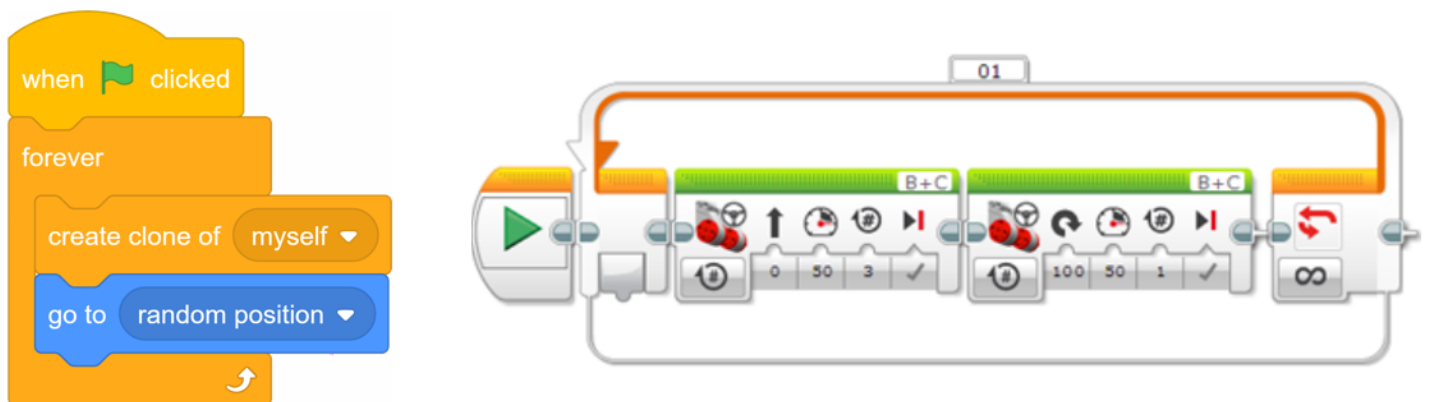
第二课 循环语句

循环语句的重要性

事实上，循环语句可以说是计算机程序中最重要语句之一。计算机的发明，就是因为人类需要去解决重复的工作。在第二次世界大战期间，盟军为了破译德军的加密通讯，即一种叫做Enigima的密码机，需要尝试非常多的密码组合，由此科学家图灵等就参与建造了最早的计算机的原型。想更详细地了解这部分历史，可以向ChatGPT询问下面地问题

- 1 计算机发明的原因之一就是破解德军的密码吗？

显然，有了计算机，很多重复的工作就可以被更方便地完成，在程序中，循环语句负责让程序重复运行一段功能一致的代码。



可能有的同学已经在Scratch或者机器人编程中，尝试过循环语句。在这门课程中，让我们跟着GPT一起，学习在Python中使用循环语句。

Python中的循环语句

我们把基础的问题抛给ChatGPT

- 1 什么是循环语句，请给我一些Python中循环语句的例子

在这里ChatGPT有时候会给我们输出两种不同的例子

- 1 循环语句用于重复执行一段代码，直到满足某个条件或达成某个目标。在Python中，主要有两种循环语

句：`for`循环和`while`循环。

```
2
3 ### 1. `for` 循环
4 `for` 循环用于遍历一个序列（例如列表、元组、字符串）或其他可迭代对象。
5
6 **示例 1**：遍历列表中的元素
7 ```python
8 fruits = ['apple', 'banana', 'cherry']
9 for fruit in fruits:
10     print(fruit)
11 ```
12
13 **输出**:
14 ```
15 apple
16 banana
17 cherry
18 ```
19
20 **示例 2**：使用 `range()` 函数生成一系列数字
21 ```python
22 for i in range(5):
23     print(i)
24 ```
25
26 **输出**:
27 ```
28 0
29 1
30 2
31 3
32 4
33 ```
34
35 ### 2. `while` 循环
36 ...
```

这里可能有学过图形化编程或者C语言的同学会发现Python的一些特点，Python的循环是一种更广义的语句，在学习的时候更强调

```
1 for element in iterable_set:
2     # 循环的程序体
```

这样一个语句。在运行的时候，集合中的每一个元素都会被依次遍历一遍。我们之后会进一步用Excel表格的每一行，以及文件夹中的每一个文件，来强化大家对这个的认识。而对于数字循环来说，比如一个1加到100的程序，是这么实现的

```
1 total = 0
2 for i in range(1, 101):
3     total += i
4
5 print("1加到100的和是:", total)
```

其中range(1,101)表示从1到100(小于101)的正整数。

为循环编写合适的prompt

计算立方和

我们在小学的时候会学习一个加法公式。即正整数1到N的立方的和，等于1到N的和的平方

$$\sum 1^3 + 2^3 + \dots + n^3 = \left(\frac{n(n+1)}{2}\right)^2$$

我们想用计算机来验证整个公式的正确性，要怎么做呢？

```
1 n = 2023
2
3 请给我实现一段简单的Python程序，用for循环计算从1到n的立方的和
4
5 另外再计算1到n的和的平方
6
7 我想看看这两个结果是否相等
```

ChatGPT的输出是这样的

```
1 # Define the upper limit
2 n = 2023
3
4 # Calculate the sum of cubes from 1 to n
5 sum_of_cubes = 0
6 for i in range(1, n + 1):
7     sum_of_cubes += i ** 3
8
9 # Calculate the square of the sum from 1 to n
```

```

10 sum_of_numbers = sum(range(1, n + 1))
11 square_of_sum = sum_of_numbers ** 2
12
13 # Check if the two results are equal
14 print( sum_of_cubes, square_of_sum, sum_of_cubes == square_of_sum )

```

这段程序会最终输出 (4191339020176, 4191339020176, True)

可以看到这段程序中，有明显的for循环语句，for i in range(1, n + 1)，表示i会从1取到n（小于1+n）。然后会逐渐累积i的立方（i ** 3）。所以整个for循环语句实现了对等式左边的验算。

而sum_of_numbers实现了对等式右边的计算。这里 sum(range(1, n + 1)) 是个更为复杂的操作，这个操作实际上直接对range(1,n+1)中的所有正整数进行了求和。

在有的时候你也可能看到大模型这么计算立方和

```

1 # 计算从1到n的立方的和
2 sum_of_cubes = sum([i**3 for i in range(1, n+1)])

```

这实际上是一种更pythonic的代码风格，他的意思和我们前面使用带缩进的for语言也是一样的。不过语句上看起来更简洁。在很多时候，我们会看到GPT会输出pythonic的语句。我们并不要求同学们能够在编程中写出这样的语句。但是在生成出这样语句的时候，还是要能够尝试解读其对应的意思。

判断素数

让我们继续来一个简单一点的例子，我们想判断114517是不是一个素数，以及有哪些数可以整除这个数字，那我要怎么做呢？

```

1 实现一段简单的python程序，用循环判断
2
3 n = 114517是不是一个素数
4
5 从2到sqrt(n) 依次尝试这些数能不能整除n，把能整除的因子进行输出

```

这里可以看到ChatGPT可以生成对应的程序

```

1 import math
2
3 # 定义数字 n
4 n = 114517

```

```

5
6 # 假设 n 是素数
7 is_prime = True
8
9 # 从2到sqrt(n)检查所有可能的因子
10 for i in range(2, int(math.sqrt(n)) + 1):
11     if n % i == 0:
12         print(f"{i} 是 {n} 的因子")
13         is_prime = False
14
15 # 如果没有找到任何因子, n是素数
16 if is_prime:
17     print(f"{n} 是一个素数")
18 else:
19     print(f"{n} 不是一个素数")

```

for i in range(2, int(math.sqrt(n)) + 1) 表示i从2到math.sqrt(n)的取整进行循环。

然后if n % i == 0是一个条件语句，这个语句我们会在下一章进行强调讲解，这句的意思是判断n是否能被i整除，是的话会运行对应缩进里的程序段。

通常来说，传统的信息技术课堂要用一个学期来教会同学们实现“筛法求素数”的代码实现，但是在这里我们在最基础的一节课上，就让同学们体验了其中的关键步骤。

当然，在这节课上，我们还是要关注在prompt中，我们如何让程序最终生成了带循环的代码。我们发现在一个数学实验类的代码里，提示词很有可能是这么引导的：

- 对于A到B的所有整数，执行特定的操作
- 计算1到N的立方和（或者别的什么关联的变量）

这样的语句引导了循环在代码中的出现，并且这样的语句，和我们在数学学科用到的语言，是非常相似的。在后面两个例子中，我们会超出纯数学的循环，看看更高层的循环语句定义。

统计Excel表格

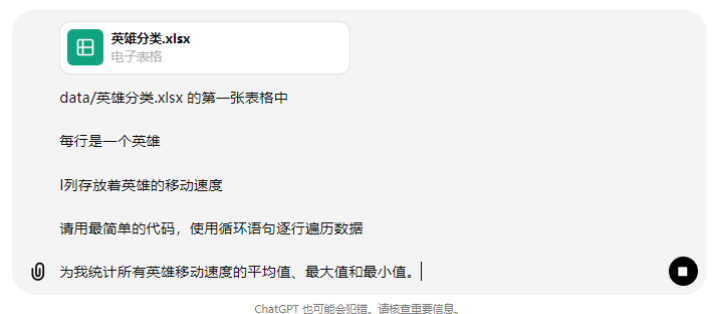
在"data/英雄分类.xlsx"中，存放着王者荣耀的一级英雄数据表格。

名字	位置	最大生命	最大法力	物理攻击	法术攻击	物理防御	法术防御	移速	攻击范围	生命回复	法力回复
鲁班大师	坦克	3287	420	166	0	127	50	380	近程	83	15
阿古朵	坦克	946	0	166	0	107	50	380	近程	50	0
廉颇	坦克	3558	420	163	0	132	50	370	近程	59	15
杨戬	战士	3339	420	166	0	98	50	380	近程	51	15
李信	战士	3418	0	176	0	106	50	370	近程	52	0
司空震	战士	3300	100	171	0	85	50	380	近程	50	0
关羽	战士	3322	10	170	0	101	50	380	近程	51	0
亚瑟	战士	3622	0	164	0	98	50	390	近程	55	0
钟无艳	战士	3150	430	164	0	100	50	380	近程	48	15
吕布	战士	3564	0	170	0	99	50	370	近程	54	0

我们可以尝试针对这里的英雄数据进行简单的统计

```
1 data/英雄分类.xlsx 的第一张表格中
2
3 每行是一个英雄
4
5 I列存放着英雄的移动速度
6
7 请用最简单的代码，使用循环语句逐行遍历数据
8
9 为我统计所有英雄移动速度的平均值、最大值和最小值。
```

这里由于很多ChatGPT以及国内的语言模型，都支持同步上传文件，我们强烈建议同学们在输入prompt的时候，同时上传文件。（这里展示了智谱清言和ChatGPT，输入prompt同时上传文件的场景）



这个时候ChatGPT会使用一个内部的虚拟环境，在生成代码后尝试执行。这里ChatGPT生成的代码如下

```
1 import openpyxl
2
3 # Load the workbook and select the first sheet
4 wb = openpyxl.load_workbook('/mnt/data/英雄分类.xlsx')
5 sheet = wb.worksheets[0]
6
7 # Initialize variables to store the sum, count, max and min values
8 sum_speed = 0
9 count = 0
10 max_speed = 0
11 min_speed = float('inf')
12
13 # Iterate through each row in column I
14 for row in sheet['I']:
15     # Make sure we're not reading a header or an empty cell
16     if isinstance(row.value, (int, float)):
17         speed = row.value
18         sum_speed += speed
19         count += 1
20         if speed > max_speed:
```

```

21         max_speed = speed
22         if speed < min_speed:
23             min_speed = speed
24
25     # Calculate the average speed
26     average_speed = sum_speed / count if count > 0 else 0
27
28     average_speed, max_speed, min_speed

```

我们注意到这里典型地使用了一个循环语句 `for row in sheet['I']`，表示对I列中地所有数据进行循环。当然，后面去获取具体地数值地时候比较复杂，使用了`row.value`来获取，这是因为`row`是一个比较复杂的对象，甚至有可能是空的数据。需要用额外的一句`isinstance(row.value, (int, float))`来判定是不是数值。如果是数值的话，后面的语句是比较好理解的。

这里还有一个细节，如果我们使用上传excel的形式来写代码，虽然我们excel文件相对于我们的python运行环境可能是在'`data/英雄分类.xlsx`'，但是GPT有可能会把这个文件移动到'`/mnt/英雄分类.xlsx`'或者'`/mnt/data/英雄分类.xlsx`'。这是因为各个GPT的虚拟环境都有自己的目录组织结构。所以如果我们想把这段代码复制到本地进行运行，就需要修改这里的`mnt`对应的目录。

统计文件夹

在电脑用了一段时间之后，不同的文件夹下总会放了很多不同的东西。甚至有很深的文件夹层次不方便一个一个查看。比如我们不记得我们把照片和文档都放在哪个文件夹了，我们可以用下面的方法，去简单地统计文件夹中所有不同文件类型地数量

- 1 对于程序的运行目录"`."`"， 扫描整个目录，有多少不同类型的文件
- 2
- 3 统计每种类型的文件出现了多少次
- 4
- 5 用最简单的循环语句为我实现

ChatGPT给我们写了这样一段程序

```

1 import os
2
3 # 定义一个字典来存储文件类型和对应的计数
4 file_types_count = {}
5
6 # 使用os.walk来遍历目录
7 for root, dirs, files in os.walk("."):
8     for file in files:

```

```

9         # 获取文件扩展名
10        file_extension = os.path.splitext(file)[1]
11
12        # 如果字典中已经存在该扩展名, 计数加1, 否则初始化为1
13        if file_extension in file_types_count:
14            file_types_count[file_extension] += 1
15        else:
16            file_types_count[file_extension] = 1
17
18    # 输出每种文件类型及其出现次数
19    for file_type, count in file_types_count.items():
20        print(f"文件类型: {file_type}, 出现次数: {count}")

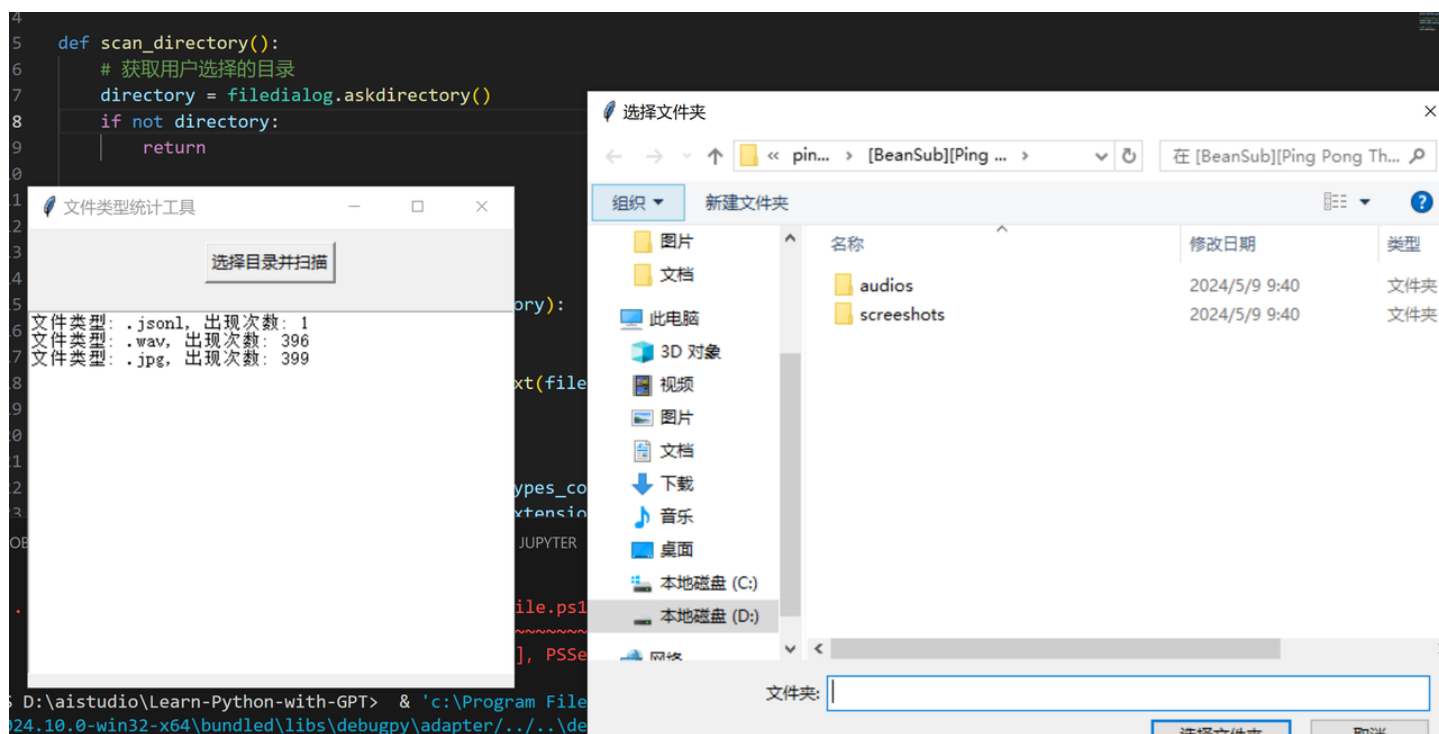
```

这里很明显会有两个循环。第一个循环是os库下面的一个函数，可以遍历路径下的所有子文件夹。第二个for是可以对每个子文件夹的所有文件进行循环。如果大家有兴趣的话，可以尝试把"."修改为"C:\Program Files"，看看程序目录里面都装了什么。

当然，每次手动设定目录太麻烦了。我们可以在连续对话中进一步输入

- 1 刚才的程序可以正常运行，但是每次手动修改目录太麻烦了，可以用tinker写一个可交互的界面吗？

我们看到这里ChatGPT顺利帮我们写了一个这样的界面。运行pygame_codes/2_tinker_os.py，可以看到对应的界面，选择文件夹之后，就可以进行扫描了。



当然，刚才那一句的prompt就属于一句“冒险”的提示词。他把一部分创意的工作交给了ChatGPT去处理。这个时候，每个语言模型以及每一次代码生成的结果，就有可能不尽相同了。

除了显示文件信息以外，其实Python还可以自动拷贝、移动、删除文件，甚至整理压缩包。但是这些操作对于初学编程的我们有一些过于危险了，所以在这里我们不进行深入的尝试了。

While循环

“人工智能”

多年前的时候流行过一个关于程序员的段子，说我们可以实现这么一个价值千万的“人工智能”

- 1 我希望实现一个python文字交互程序，用户每输入一句话，就输出对应的response
- 2
- 3 用户输入一句话的时候，如果末尾是 "吗?" 则替换成 "!"
- 4 如果末尾是"?" 则替换成"!" 。同时考虑中文和英文输入符号的可能性
- 5
- 6 请用最简单的python语句为我实现

ChatGPT输出对应的代码是这样的

```
1 while True:
2     user_input = input("请输入一句话: ")
3
4     # 去除前后的空白字符
5     user_input = user_input.strip()
6
7     # 检查末尾是否为 "吗?" 或 "?", 包括中文和英文符号
8     if user_input.endswith("吗?") or user_input.endswith("吗? "):
9         response = user_input[:-2] + "!"
10    elif user_input.endswith("?") or user_input.endswith("? "):
11        response = user_input[:-1] + "!"
12    else:
13        response = user_input
14
15    print("输出:", response)
```

对应的运行效果是这样的，

```
D:\aistudio\Learn-Python-with-GPT\pygame_codes>python 2_idiot_AI.py
请输入一句话：会说中文吗？
输出：会说中文！
请输入一句话：你是人工智能吗？
输出：你是人工智能！
请输入一句话：是吗？
输出：是！
请输入一句话：_
```

这个例子里面涉及到一个while循环的使用。因为对于这样一个程序，我们并不知道 输入对话 —— 进行回复 这个过程要重复多少次。所以这个时候我们要使用到while循环。

角谷猜想

同学们应该都听说过角谷猜想，即对于任意一个正整数，如果是偶数，我们就原地除以2；如果是奇数，我们就把他乘以3再加一。经过有限次后，总是能够进入4-2-1的循环

- 1 使用最简单的python代码，仅使用while循环语句
- 2
- 3 对于特定的正整数n = 2023，验证角谷猜想。
- 4 如果是偶数，则除以2，如果是奇数，则乘以3再加一
- 5
- 6 当变成1的时候退出，打印每一次的数字

这里我们看看GPT自动生成的代码。

```
1 n = 2023
2 while n != 1:
3     print(n)
4     if n % 2 == 0:
5         n = n // 2
6     else:
7         n = n * 3 + 1
8 print(n) # 打印最后的1
```

这里有典型的while语句的用法，即

- 1 while 条件:
- 2 # 条件满足时执行的代码段

我们也可以要求GPT将这个代码重构为其他的形式

1 重构这段代码，展示break语句的使用

这个时候ChatGPT的输出为

```
1 n = 2023
2 while True: # 无限循环，使用break来退出
3     print(n)
4     if n == 1:
5         break # 当n为1时，退出循环
6     if n % 2 == 0:
7         n = n // 2
8     else:
9         n = n * 3 + 1
```

这里“重构(refactor)”是指保持代码的功能和运行结果不变，但是改变代码的组织结构。这里可以看到新的代码使用了while True这样的无限循环。和“人工智能”那个例子里面是一样的。那么我们怎么退出这样的无限循环呢？我们可以设置程序在满足特定的条件的时候，使用break语句来跳出循环。这样这个循环就不会无限进行下去了。

摄像头循环

一个典型的会出现while True和break语句的场景是摄像头循环。

1 给我写一个摄像头展示画面的python例子
2
3 并且画面是左右翻转的

对应的ChatGPT代码是

```
1 import cv2
2
3 # 初始化摄像头
4 cap = cv2.VideoCapture(0)
5
6 while True:
7     # 从摄像头读取一帧画面
8     ret, frame = cap.read()
```

```

9
10     if not ret:
11         print("无法获取摄像头数据")
12         break
13
14     # 左右翻转画面
15     flipped_frame = cv2.flip(frame, 1)
16
17     # 显示翻转后的画面
18     cv2.imshow('Flipped Camera', flipped_frame)
19
20     # 按 'q' 退出循环
21     if cv2.waitKey(1) & 0xFF == ord('q'):
22         break
23
24 # 释放摄像头资源
25 cap.release()
26 # 关闭所有OpenCV窗口
27 cv2.destroyAllWindows()

```

因为在opencv的程序里面，大家会约定俗成的设计成按"q"键退出，所以这里可以看到生成的代码会有一个显著的while True循环，以及按q退出的程序。当然，如果摄像头没有被成功启动，也是会自动退出循环的。

Pygame循环

在pygame里面也是会有典型的while循环，因为pygame的游戏是会持续进行的，这里给出一个简单的例子。

1 为我写一个pygame的例子，屏幕中间有一个红色的正方形，按w,s,a,d的时候正方形会向上下左右移动

由于篇幅的原因这里我们就不粘贴代码了。因为在后续的课程中，我们会经常编写一些pygame的例子。请同学们自行阅读生成的代码，体会循环在其中的作用。

总结

如果你使用ChatGPT，依靠需求提示词，来编写python程序。很多时候循环其实是“隐性”包含在需求中的。当然我们需要写的程序往往会包含持续运行、批量处理这样的需求，所以很多时候生成的代码中，总是会出现对应的循环语句。这个时候作为开发者，我们就需要读懂这些循环语句。

按照正常的编程学习的顺序，循环语句和条件语句总是会被先学习的。为了使得完全没有编程基础的同学也能够上手这本书中的内容，更好地理解ChatGPT生成的代码。我们在包括本章在内的少量章节

中，还是会偏重基础知识的学习。如果想把本章改为“项目制教学”的内容，也可以围绕文件夹统计，或者是excel统计，来设计对应的项目制的课程设计。

课后练习

- 在统计Excel表格的例子中，尝试统计英雄的物理防御和法术防御力的均值。
- 尝试在统计文件夹程序中，增加对文件大小的统计。
- 在循环的每一步都较慢的时候，我们经常会使用一种叫做tqdm的库来显示循环的进度。和ChatGPT询问这个库要怎么用，并实践一下。
- 在这节课的需求prompt中，我们往往会要求GPT使用“最简单的代码”，“使用循环语句”，如果删除这方面的要求，其他需求不变，看看生成的代码有什么不同。

提纲

- 基础的数值循环
- 更广义的循环
 - Excel的处理
 - 文件夹的处理
- while循环

 [第一课-钢琴键盘](#)