

# 第三课 点菜程序与条件语句

在这个章节中，我们会先学习编程中的另一个最重要的控制语句——条件语句。并且围绕一个餐厅点菜程序，进行编程。

## 基础概念

向ChatGPT提问以下问题

- 1 给我实现一个简易的Python的分支语句的句子，再向我解释分支语句的相关知识

ChatGPT的输出是

## Python 简易分支语句示例

```
1 x = 10
2
3 if x > 5:
4     print("x 大于 5")
5 elif x == 5:
6     print("x 等于 5")
7 else:
8     print("x 小于 5")
```

## 分支语句简介

分支语句（也称为条件语句）在编程中用于根据不同条件执行不同的代码块。在Python中，最常见的分支语句是 `if` 语句，它允许你根据一个或多个条件来控制程序的执行流。

### 1. `if` 语句

`if` 语句用于检查某个条件是否为真（`True`）。如果条件为真，那么代码块会被执行。

```
1 if 条件:
2     # 当条件为真时执行的代码块
```

## 2. elif 语句

elif 代表 "else if"，用于在前面的 if 语句条件不满足时，检查另一个条件。

```
1 if 条件1:
2     # 当条件1为真时执行的代码块
3 elif 条件2:
4     # 当条件2为真时执行的代码块
```

## 3. else 语句

else 语句用于处理所有不满足之前 if 和 elif 条件的情况。

```
1 if 条件1:
2     # 当条件1为真时执行的代码块
3 elif 条件2:
4     # 当条件2为真时执行的代码块
5 else:
6     # 当所有条件都不满足时执行的代码块
```

...

同学们可以进一步要求ChatGPT写出条件语句嵌套，以及条件语句和循环语句组合的例子。这里我们需要强调的是Python基于缩进的语言。所以所有的程序块都要在对应的条件的后一个缩进下。如果随意修改缩进很有可能会导致程序错误。

## 条件语句与状态变量

这里我们先用一个相对简单的例子，让大家感受一下在一个连续运行的程序中，使用状态变量和条件语句进行组合的效果

```
1 给我实现一个python的opencv摄像头demo，实时显示摄像头的画面
2
3 按下f键会使得画面左右翻转
```

如果你的机器没有摄像头的话，可以尝试下面这个需求

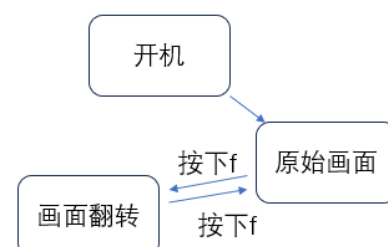
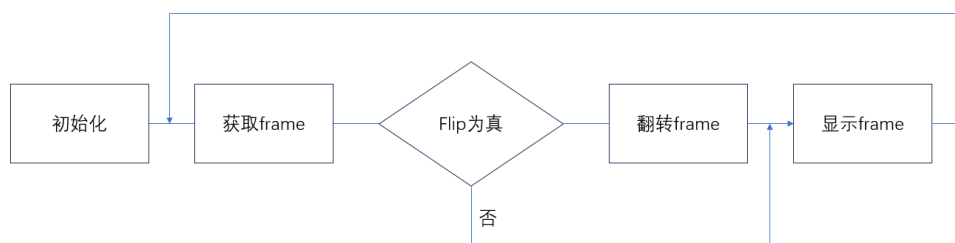
```
1 给我实现一个pygame程序，画面一开始是红色
2
```

3 按下空格的话，画面会变成蓝色，再按会变回红色

ChatGPT给的输出是这样的

```
1 import cv2
2
3 cap = cv2.VideoCapture(0) # 打开默认摄像头
4
5 if not cap.isOpened(): # 检查摄像头是否成功打开
6     print("无法打开摄像头")
7
8 flip = False # 初始化翻转标志
9
10 while True:
11     ret, frame = cap.read() # 读取摄像头帧
12
13     if not ret: # 检查帧是否读取成功
14         print("无法接收帧 (可能是流的结束)")
15         break
16
17     if flip: # 如果翻转标志为真，则左右翻转图像
18         frame = cv2.flip(frame, 1)
19
20     cv2.imshow('摄像头', frame) # 显示帧
21     key = cv2.waitKey(1) # 读取键盘输入
22
23     if key == ord('f'): # 按下 'f' 键来翻转图像
24         flip = not flip
25     elif key == ord('q'): # 按下 'q' 键来退出
26         break
27
28 cap.release() # 释放摄像头
29 cv2.destroyAllWindows() # 关闭所有窗口
30
```

在标准的信息技术课程中，我们会用流程图，把这个过程给可视化出来。



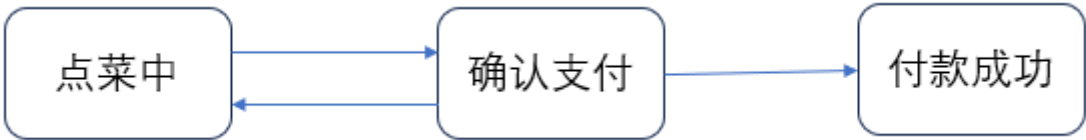
我们看到，在这个程序中，有一个很重要的变量就是flip。这个变量我喜欢称之为“状态变量”。你可以这么理解，如果这个程序运行到一半需要关掉，那什么样的信息可以完整描述当前程序运行的状态，使之顺利重启。对于当前应用来说，只需要记录flip是True还是False就可以了。所以flip就是完整的状态变量。

对于这种始终在运行的程序，我更喜欢使用状态变量的描述，即右边的图中的描述，在开机之后，系统就会展示原始画面，如果按下f，则会状态切换，如果再按下f，状态就会切换回去。在不同的状态下，执行的代码、策略是不一样的。我们在之前写足球机器人程序的时候，就经常使用这种描述，因为找球、持球、瞄准射门所需要使用的策略不同，所以机器是处在不同的状态的。对于不同的状态分别实现对应的代码模块，可以使得程序更清晰。

## 编写点菜程序

我们对一个简单的二状态的程序有了一个理解之后，我们来思考点菜程序的实现。

对于点菜程序来说，我们日常的很多购买程序，都是由确认支付的流程



所以我们需要专门定义一种状态就是确认支付。而机器的初始状态就是在点菜中。

我们先尝试编写下面的prompt来让ChatGPT写一下初步的程序

```
1 我希望实现一个简易的基于文字交互（input数字和print信息）的python点餐程序
2
3 仅仅使用循环和条件语句
4
5 有五道不同的菜和对应的价格
6
7 有一个状态变量记录当前机器的状态 点餐中，确认支付，完成支付
8
9 用户在确认支付时可以继续加菜或者确认支付
```

当然，这里需求其实没有去详细地描述状态相互之间是如何转换地，并且菜的名字和价格，也交给GPT去自由生成。

```
1 # 菜单及价格
2 menu = {
3     '1': ('宫保鸡丁', 18),
4     '2': ('麻婆豆腐', 15),
5     '3': ('清蒸鲈鱼', 28),
```

```
6     '4': ('糖醋排骨', 25),
7     '5': ('扬州炒饭', 12)
8 }
9
10 # 初始化订单和状态
11 order = {}
12 state = '点餐中'
13 total_price = 0
14
15 # 点餐程序
16 while True:
17     if state == '点餐中':
18         print("\n欢迎光临! 请选择菜品 (输入对应数字) : ")
19         for key, (dish, price) in menu.items():
20             print(f"{key}. {dish} - {price}元")
21         choice = input("请选择菜品编号 (输入'0'确认点餐) : ")
22
23         if choice == '0':
24             if not order: # 如果订单为空, 提示至少点一个菜
25                 print("请至少点一个菜! ")
26                 continue
27             state = '确认支付'
28         elif choice in menu:
29             dish, price = menu[choice]
30             if choice in order:
31                 order[choice] += 1
32             else:
33                 order[choice] = 1
34             total_price += price
35             print(f"已添加 {dish} 到订单中。")
36         else:
37             print("无效的输入, 请输入正确的菜品编号。")
38
39     elif state == '确认支付':
40         print("\n您的订单: ")
41         for key, quantity in order.items():
42             dish, price = menu[key]
43             print(f"{dish} x {quantity} - {price * quantity}元")
44         print(f"总计: {total_price}元")
45
46         confirm = input("是否确认支付? (输入'1'确认, 输入'2'继续加菜) : ")
47         if confirm == '1':
48             state = '完成支付'
49         elif confirm == '2':
50             state = '点餐中'
51         else:
52             print("无效的输入, 请输入'1'确认支付或'2'继续加菜。")
```

```
53
54     elif state == '完成支付':
55         print("\n支付成功! 感谢您的光临, 祝您用餐愉快!")
56         break
```

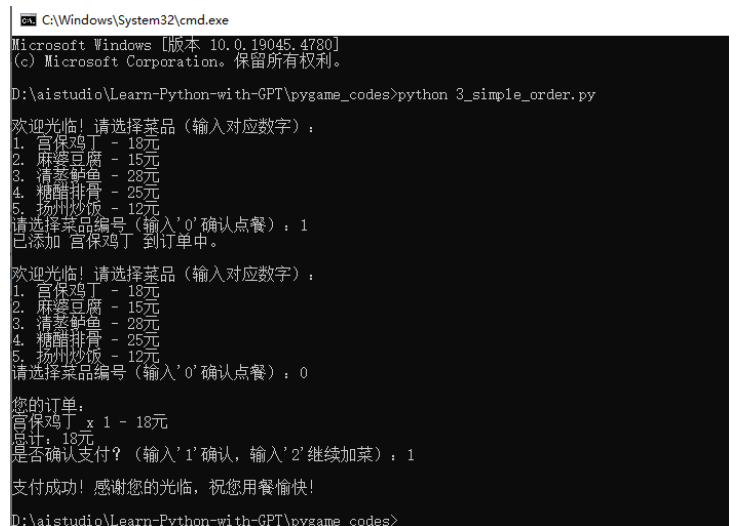
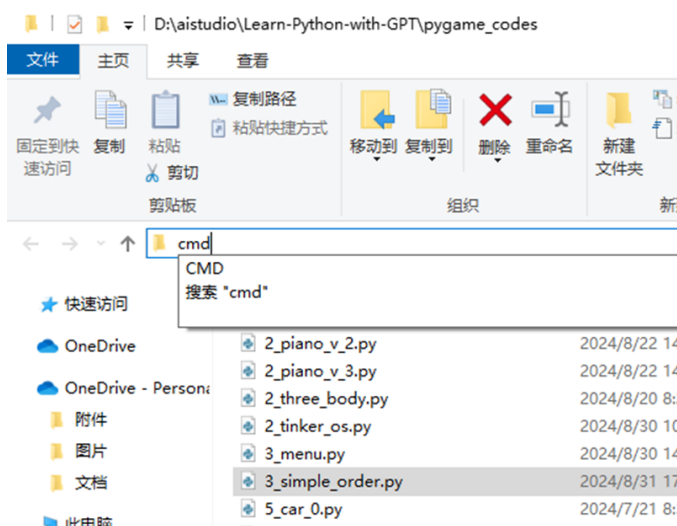
这里很明显整个程序在一个while循环中。并且初始化的时候会把程序的状态state赋值为"点餐中"。如果状态是点餐中,则会触发第一个条件语句(17行),进入第18-37行的代码段。并且其中在choice等于0的时候,会触发23行的条件,进入结算程序(状态改变发生在27行)。而state为其他值的时候,也由对应的代码段可以被阅读到。如果状态在确认支付的时候,用户选择1完成支付,那就会运行48行的完成支付,接着进入完成支付的状态最终退出程序。

这是一个典型的状态机形式的程序,即使这个代码很长,由于状态的存在,每一段代码都可以被清晰地阅读到。

## 在cmd里面运行程序

由于在VS Code里面运行地时候,Output默认在比较下面的位置,尽管也可以这么体验程序但是看起来有点狭窄。

我们这里额外讲一下怎么在cmd里面运行这个程序。



在windows的文件浏览器的目录栏里面输入cmd,打开命令提示符窗口。假设我们把程序保存在了3\_simple\_order.py,我们就运行python 3\_simple\_order.py,这样就能更为直接地体验这个程序了。

## 为点菜程序设计界面

在确保之前ChatGPT输出的程序可以正常运行后,我们开始考虑为刚才的程序设计一个界面。直接在连续的对话中输入

1 {在连续对话中保持之前的代码,或者粘贴代码}

2

3 刚才的程序可以正确运行，请帮我用tinker编写一个对应的界面。

在我们这套课程后面会经常出现这样的提示词，即和之前的编程有关联的改进提示词。这个时候同学们需要保证之前实现的，能够正常运行的代码，在我们和GPT的上一个聊天中。或者如果不在的话，你可以重新复制一遍代码。

```
51         else:
52             print("无效的输入，请输入'1'确认支付或'2'继续加菜。")
53
54     elif state == '完成支付':
55         print("\n支付成功！感谢您的光临，祝您用餐愉快！")
56         break
```

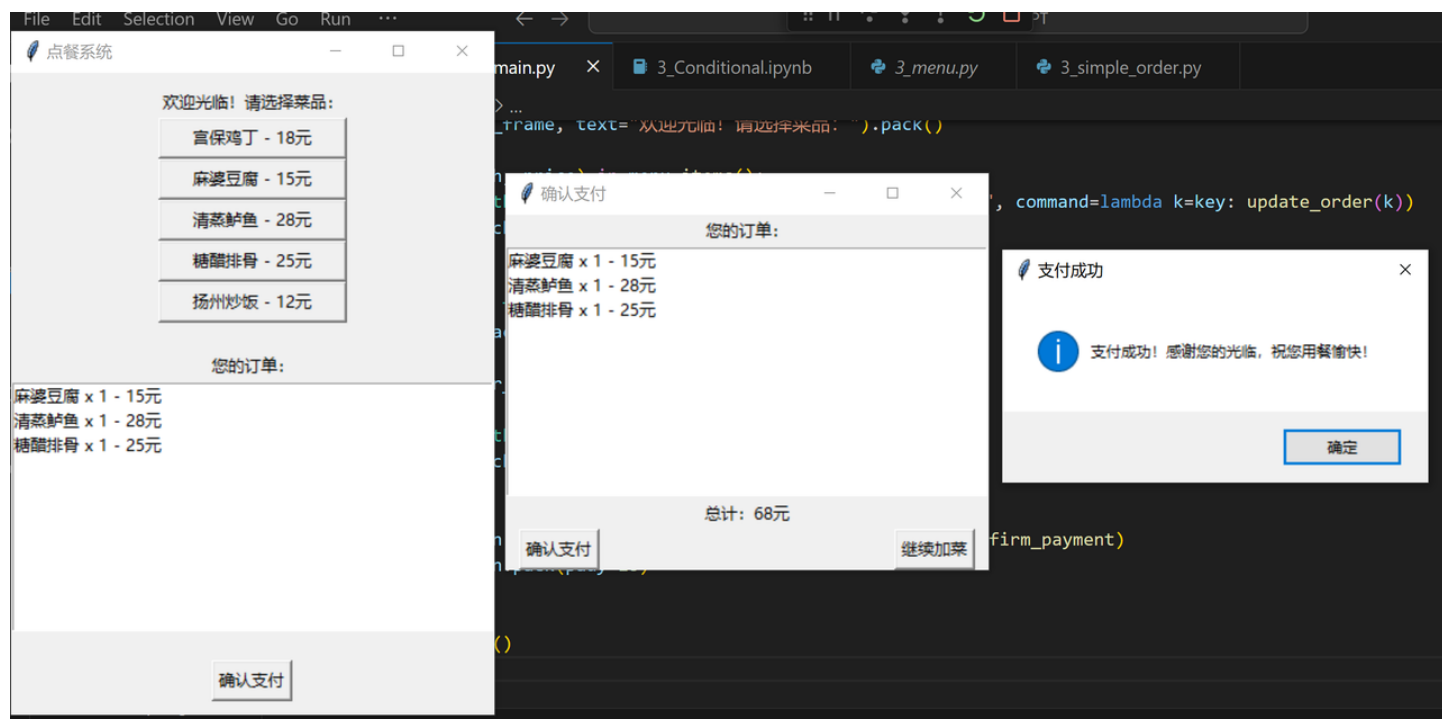
在这个程序中，用户可以通过输入数字来选择菜品，输入'0'来确认点餐并进入支付流程。在确认支付阶段，用户可以选择确认支付或者继续加菜。当用户确认支付后，程序会打印出订单详情并结束。

+ 新建对话

📁 刚才的程序可以正确运行，请帮我用tinker编写一个对应的界面。



实际的效果就是这样的。同学们可能会意识到，这个需求让GPT来做会是“冒险”的。因为这个需求在让GPT自己设计交互的流程和界面。



这个prompt我用不同的语言模型尝试了3-4次，基本都可以写出可以运行的代码，有一次出现了报错，当我把报错信息重新复制给GPT时，GPT第二次生成出的代码是可以顺利运行的。这个图给出了其中一次我觉得生成得较好的界面，对应的代码放在了pygame\_codes/3\_menu.py中。

# 其他场景下条件语句的应用

## 实时人脸检测

有的同学可能像李鲁鲁老师一样是，小的时候会接触更多的机器人编程场景。在机器人编程的场景中，有一类很常见的条件语句就是对于传感器信息的判断。在这里我们也可以用摄像头来实现一个实时检测的场景

- 1 为我写一个用opencv对摄像头进行实时人脸检测的例子
- 2
- 3 先将检测到的人脸框进行可视化
- 4
- 5 如果检测到人脸，则把画面渲染成半透明的红色（保留r通道，其他通道除以4）
- 6 如果没有人脸，则把画面渲染成半透明的绿色（保留g通道，其他通道除以4）

对应生成出的代码

```
1 import cv2
2 face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
   'haarcascade_frontalface_default.xml')
3 cap = cv2.VideoCapture(0)
4 while True:
5     ret, frame = cap.read()
6     if not ret:
7         print("无法捕获视频流");
8         break
9
10    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
11    faces = face_cascade.detectMultiScale(gray, 1.1, 4)
12
13    if len(faces) > 0:
14        for (x, y, w, h) in faces: cv2.rectangle(frame, (x, y), (x+w, y+h),
   (0, 0, 255), 2)
15        frame[:, :, 1] //= 4; frame[:, :, 0] //= 4
16    else:
17        frame[:, :, 0] //= 4; frame[:, :, 2] //= 4
18
19    cv2.imshow('Face Detection', frame)
20    if cv2.waitKey(1) & 0xFF == ord('q'):
21        break
22 cap.release()
23 cv2.destroyAllWindows()
```



可以看到这里`len(faces) > 0` 就是是否检测出人脸的判定条件。

## pygame循环中的条件语句应用

我们在第一课和第二课已经实现过一些pygame的程序，比如在第一课最终的程序 `pygame_codes/2_piano_v_3.py`，或者第二课的正方形控制程序 `pygame_codes/1_example_game.py` 中，我们都可以看到典型的条件语句

键盘钢琴中的部分代码:

```
1 while running:
2     for event in pygame.event.get():
3         if event.type == pygame.QUIT:
4             running = False
5         elif event.type == pygame.KEYDOWN:
6             if event.unicode in sounds:
7                 sounds[event.unicode].play(-1)
8                 keys_active[event.unicode] = True # 标记为活动
9         elif event.type == pygame.KEYUP:
10            if event.unicode in sounds:
11                sounds[event.unicode].stop()
12                keys_active[event.unicode] = False # 标记为非活动
```

上下左右游戏中的部分代码

```
1 while running:
2     keys = pygame.key.get_pressed() # 获取按键状态
3     if keys[pygame.K_w]:
4         square_y -= 10
5     if keys[pygame.K_s]:
6         square_y += 10
7     # ...
```

这里`pygame.event.get()`会获取游戏运行时的很多事件，对于这些事件，包括了鼠标和键盘的一些动作。在钢琴键盘中，我们可以通过分析哪些键按下(`pygame.KEYDOWN`)和抬起(`pygame.KEYUP`)，再获得键的对应的字母/数字。就可以实现钢琴的控制。而在方块方向控制的游戏中，则使用了更方便的指令`pygame.key.get_pressed()`来获得按下的键的情况，然后依次判断对应的按键比如w和s有没有被按下，有的话则执行对应的操作。

同学们都可以看到在这样的代码中，while循环和条件语句的组合应用。

# 总结

在需求的prompt中，条件语句往往会更明显，以"如果...就..."的形式出现在需求中。在我们的课程中，我们实际上给出了一些相当于传统编程教学来说更难例子。我们希望向学生强调，在编程的时候同学们可以更清楚地认识到条件语句，实际上是要有一些变量来参与的。指出这些变量是什么，有时甚至比设计条件表达式要更为重要。比如在点菜程序中，就比较明显的给出了一个状态机的实现，并且可以从中看到程序在不同状态之间的跳转。而在更实时的程序中，条件和事件的判断也起着非常重要的作用。

## 课后练习

- 在上一章角谷猜想验证的例子中，查看条件语句的应用。
- 对于点菜程序，如果用户不想点了，增加取消或者清空的功能；另外为了防止用户误操作，增加一次取消确认的环节
- 我们课上给的交互式点餐的程序，在一次点餐后就会关闭，可以尝试将这个点餐程序修改为点完餐之后清空已点菜单，方便下一个同学进行点餐。
- \*可以和GPT进行讨论，尝试在界面中插入不同菜品的图像。