

# 第一课-钢琴键盘

## 实现一个钢琴键盘

在音乐课上，看见过各种各样的乐器。对于计算机来说，一个电脑也同时有键盘的输入和声音的输出。我们能不能编程来实现一个钢琴呢？在这一章，我们会试着以"键盘钢琴"作为例子，带大家体验完整的和ChatGPT一起编写python程序的流程。

### 第一次尝试

首先在ChatGPT中尝试我们的第一次编程，输入prompt

```
1 我希望实现一个pygame程序，
2
3 当按下1、2、3、4、5的时候
4
5 会发出do、re、mi、fa、so的声音，持续0.5秒
```

我们会得到这样的回复

```
1
2 要在Pygame中实现这个功能，你需要先安装Pygame库和一个小型的音频库，比如`pygame.mixer`。
  以下是一个简单的示例代码，它会在按下1、2、3、4、5键时播放对应的音符声音，持续0.5秒：
3 ```python
4 import pygame
5 import sys
6 # 初始化Pygame
7 pygame.init()
8 # 加载音符声音文件
9 # 确保你有以下文件: do.wav, re.wav, mi.wav, fa.wav, so.wav
10 # 这些文件应该放在你的项目目录中
11 sounds = {
12     pygame.K_1: pygame.mixer.Sound('do.wav'),
13     pygame.K_2: pygame.mixer.Sound('re.wav'),
14     pygame.K_3: pygame.mixer.Sound('mi.wav'),
15     pygame.K_4: pygame.mixer.Sound('fa.wav'),
16     pygame.K_5: pygame.mixer.Sound('so.wav')
17 }
18 # 设置音符持续时间
```

```

19 duration = 500 # 毫秒
20 # 创建一个窗口，大小为200x200
21 screen = pygame.display.set_mode((200, 200))
22 # 主循环
23 running = True
24 while running:
25     for event in pygame.event.get():
26         if event.type == pygame.QUIT:
27             running = False
28         elif event.type == pygame.KEYDOWN:
29             # 检查按键是否是我们关心的音符键
30             if event.key in sounds:
31                 # 播放对应的音符
32                 sounds[event.key].play(maxtime=duration)
33 # 退出Pygame
34 pygame.quit()
35 sys.exit()
36 ```
37 在运行这段代码之前，请确保你已经准备好了对应的音符声音文件（do.wav, re.wav, mi.wav,
    fa.wav, so.wav），并将它们放在你的项目目录中。这些声音文件可以通过在线资源获取，或者使用
    音乐软件自己制作。
38 当你运行这个程序并按下1、2、3、4、5键时，程序会播放对应的音符声音，持续0.5秒。注意，由于使用
    了`pygame.mixer.Sound.play(maxtime=duration)`，声音会在指定的时间后自动停止。
39

```

我们把这段程序保存到一个python文件中（查看资源包lesson1/piano\_v\_0.py）

## 碰到报错

如果同学们的python已经默认安装了pygame（使用anaconda或者一般标准安装下都是附带pygame的，不然的话就使用pip install pygame -i <https://mirrors.aliyun.com/pypi/simple/> 来进行安装）

那么就会出现这段报错

```

1 File "...\\Learn-Python-with-GPT\\pygame_codes\\2_piano_v_0.py", line 11, in
  <module>
2     pygame.K_1: pygame.mixer.Sound('do.wav'),
3 FileNotFoundError: No such file or directory.

```

这说明程序没有发现do.wav的声音。

这个时候我们有几种办法来处理这个报错

- 去网上寻找音色文件，凑齐do到so的wav音频文件

- 把报错信息复制给ChatGPT，与ChatGPT沟通解决办法
- 自己分析解决办法，修改需求

## 修改需求

作为本书较早给出的例子，为了简单起见，我们直接给出合理的解决办法。因为这个程序缺失的是do到so的音频，我们只需要在初始化的时候，建立这些音频就可以了。实际上python是有能力来新建这些音频的。所以我们把新的prompt修改为

```
1 我希望实现一个pygame程序，
2
3 当按下1、2、3、4、5的时候
4
5 会发出do、re、mi、fa、so的对应频率的声音，持续0.5秒
6
7 可以在初始化的时候先生成这些声音对应的wav文件
```

我们在需求描述的最后一句，增加了一句"可以在初始化的时候先生成这些声音对应的wav文件"，看看这个时候程序发生了什么改变

```
1 import pygame
2 import math
3 import wave
4 import struct
5
6 # 初始化Pygame
7 pygame.init()
8
9 # 定义音符频率
10 note_freqs = {
11     'do': 261.63, # C4
12     're': 293.66, # D4
13     'mi': 329.63, # E4
14     'fa': 349.23, # F4
15     'so': 392.00 # G4
```

```
16 }
17
18 # 定义采样率
19 sampling_rate = 44100
20
21 # 生成一个音符的函数
22 def generate_note(freq, duration, filename):
23     # 创建WAV文件
24     obj = wave.open(filename, 'w')
25     obj.setnchannels(1) # 单声道
26     obj.setsampwidth(2) # 两个字节
27     obj.setframerate(sampling_rate)
28
29     # 生成正弦波
30     for i in range(int(sampling_rate * duration)):
31         value = int(32767.0 * math.sin(2 * math.pi * freq * i / sampling_rate))
32         data = struct.pack('<h', value)
33         obj.writeframesraw(data)
34
35     obj.close()
36
37 # 初始化时生成WAV文件
38 for note, freq in note_freqs.items():
39     generate_note(freq, 0.5, f'{note}.wav')
40
41 # 加载生成的WAV文件
42 note_sounds = {
43     pygame.K_1: pygame.mixer.Sound('do.wav'),
44     pygame.K_2: pygame.mixer.Sound('re.wav'),
45     pygame.K_3: pygame.mixer.Sound('mi.wav'),
46     pygame.K_4: pygame.mixer.Sound('fa.wav'),
47     pygame.K_5: pygame.mixer.Sound('so.wav')
48 }
49
50 # 设置音符持续时间
51 duration = 500 # 毫秒
52
53 # 创建一个窗口, 大小为200x200
54 screen = pygame.display.set_mode((200, 200))
55
56 # 主循环
57 running = True
58 while running:
59     for event in pygame.event.get():
60         if event.type == pygame.QUIT:
61             running = False
62         elif event.type == pygame.KEYDOWN:
```

```

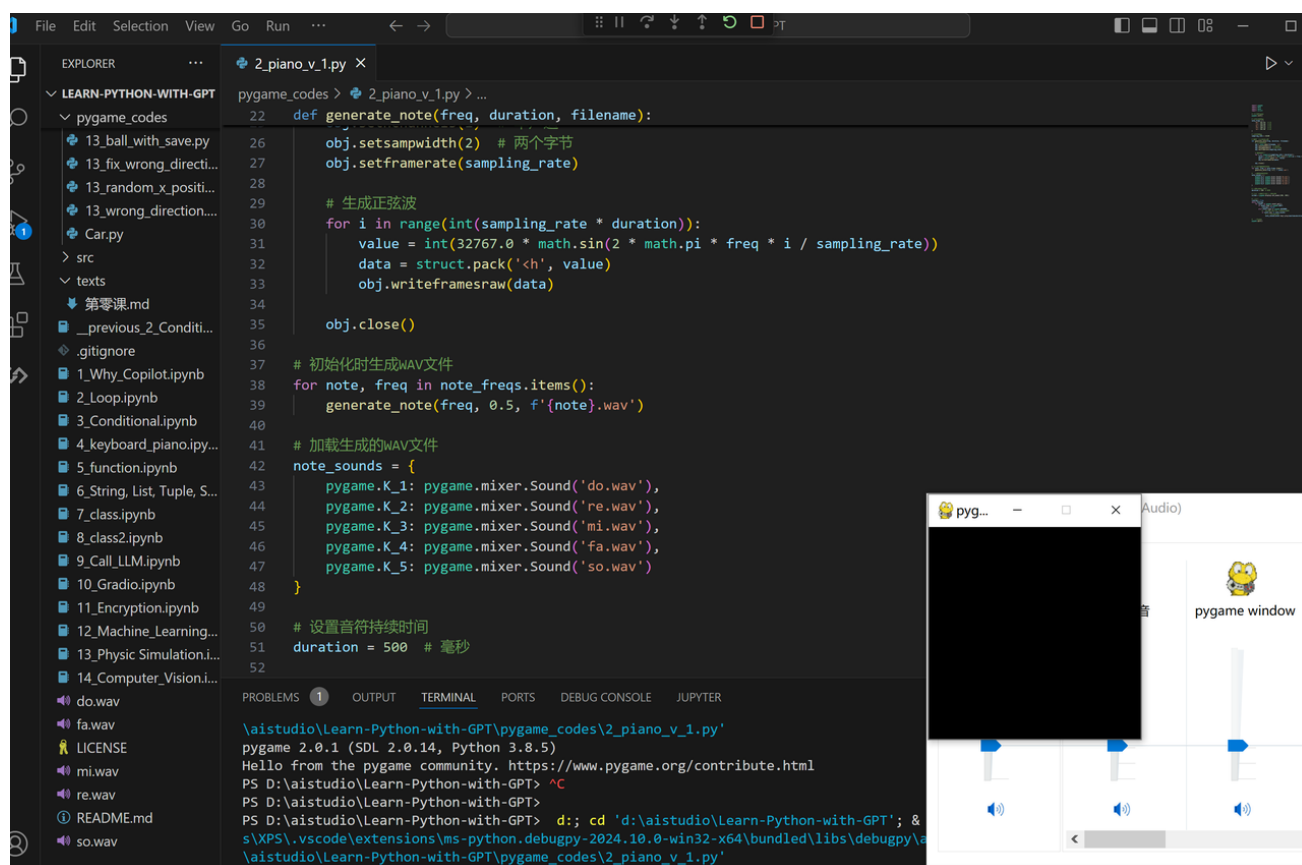
63         # 检查按键是否是我们关心的音符键
64         if event.key in note_sounds:
65             # 播放对应的音符
66             note_sounds[event.key].play(maxtime=duration)
67
68 # 退出Pygame
69 pygame.quit()

```

可以看到，这次生成的程序，比我们第一次的程序要长了一些。多了一段generate\_note函数模组，这段函数会在程序第一次运行的时候，生成对应的音频文件。这样就能保证这个程序被更正常的运行。关于函数的具体意义，我们会在后续的章节进行更详细的介绍。

## 尝试运行

尝试将这个程序保存到独立的python文件中，或者打开资源包的lesson1/piano\_v\_1.py，尝试运行这段程序。



你可以发现pygame的窗口被顺利启动了，当你按下1, 2, 3, 4, 5的时候，程序会发出对应的声音。

## 分析我们的prompt

在这里我们可以回顾一下我们的prompt，prompt的第一句话

1 我希望实现一个pygame程序，

说明了我希望用什么样的库来实现这个程序。我们学习Python，就是看重Python能够快速调用大量现成的库，可以方便我们进行很多应用的快速开发。而对于很多应用程序来说，往往有多种方法来实现对应的功能，这句话是在明确我们希望ChatGPT用什么样对应的库来实现某个程序

后面的三句，是两段不同的需求。

- 1 当按下1、2、3、4、5的时候
- 2
- 3 会发出do、re、mi、fa、so的对应频率的声音，持续0.5秒

以及我们发现没有do.wav对应的音频后，补充的需求。

- 1 可以在初始化的时候先生成这些声音对应的wav文件

实际上这两个需求的顺序是可以先后交换的。这四句话的顺序也可以进行交换。甚至偶尔出现一些错别字的时候，ChatGPT仍然能写出可以运行的正确程序。

在本书的设计上，我们大多数的时间都会来编写这些“需求的prompt”。和传统编程课不同的是，1、编写这些prompt提示词，更接近人类的自然语言。2、对于刚入门编程的同学来说，极大程度避免了打错字和搞错缩进带来的麻烦。3、很多高级的功能都可以在更短的时间内去实现。

## 进一步优化钢琴程序

我们可以讨论一下，这个钢琴有什么可以改进的地方呢？

- 音阶太窄，在前面的例子里，我们使用了do、la、mi、fa、so来定义我们的声音。很多曲子需要更多的声音。并且我们希望我们的声音能够跨越两个八度，
- 不能两个键同时按下，同时按下会出现难听的混音
- 一个按键按下后必须等待0.5秒才能进行其他活动
- 音的持续长度不能控制
- 声音没有强弱
- 音色不好听

在这里很大程度上是因为我们的程序里面有一个time.sleep(0.5)。我们希望把这个特性去移除，这样我们其中几个问题就可以被解决。这个时候我们把我们的需求prompt修改为下面的文本：

```
1 我希望实现一个pygame程序，
2
3 当按下1、2、3、4、5、6、7、8、9、0的时候
4
5 会发出C4、D4、E4、F4、G4、A4、B4、C5、D5、E5的对应频率的声音
6
7 可以在初始化的时候先在piano_sounds目录生成这些声音对应的wav文件
8
9 当按键按下时候，会播放对应的声音
10
11 如果按键抬起，则声音会停止。
```

这里我们使用了A到G音名来定义声音。有的对音乐不了解的同学可能不了解这种定义的方法。我们可以向ChatGPT去询问这部分基础的知识。

这个有完整八度音阶的版本我们放在了资源包的lesson1/piano\_v\_2.py中。

## 向ChatGPT提问基础知识

在ChatGPT中输入这样的问题

```
1 do, re, mi这样的唱名和C, D, E这样的音名有什么不同？C4和C5又有什么区别？
```

ChatGPT的回答是

- 1 唱名和音名是音乐中两种不同的音高标识系统。唱名，... 有助于歌手记忆和准确演唱音乐作品中的音高。
- 2 音名，则是指具体的音符符号或音高的名...音名可以更准确地表示音高，而唱名则更侧重于音高的感受和演唱。
- 3 关于C4和C5的区别，C4通常指的是中央C音，位于钢琴键盘的中间位置，而C5则是比C4高一个八度的音。在音乐理论中，八度是指音高的一个范围，其中从一个音到它的高八度音包含了12个半音。因此，C5比C4的频率高出一倍，但它们属于同一个音符的不同八度。

在本书中，很多基础知识（当然主要是编程的基础知识）会使用这样的方式来呈现，即我们会组织一个问题，比如“什么是循环语句，给我一个Python编程中的例子。”这样的问题。并且把必要的回答直接呈现在书里。很多时候，同学们如果忘记了一些基础知识，或者想对一些新的内容进行一层一层的深入学习，都可以重新向ChatGPT进行提问。

## 为程序添加可视化

注意到我们现在的程序，是只有声音的部分的。那我们有没有办法去给程序增加合理的可视化呢？  
这里我们可以让ChatGPT进一步帮我们想办法

- 1 {粘贴之前的代码}
- 2
- 3 这段程序可以正常运行，可以帮助我为这段程序添加合理的可视化吗？我希望不同的按键按下时，会有不同的视觉效果。

这里"{粘贴之前的代码}"是指要把我们之前的代码完整地粘贴进去。当然，如果你在一些可以连续聊天使用的ChatGPT工具中的话，意味着之前生成的代码已经在我和ChatGPT沟通的上下文中了，那就不一定要重复粘贴了。

当然，这个prompt是有一些“冒险”的。因为他把创意设计的部分，交给了ChatGPT。在本书中，大多数的提示词生成的程序结果是“稳定”的，即同一个提示词，在多次生成，以及在不同厂家的ChatGPT进行使用的时候，往往都能得到合理的结果。但是这些需要ChatGPT去“设计一个方案”的提示词，往往每次运行会出现不同的结果。

在生成这段程序之前ChatGPT还会先描述下他打算怎么实现具体的方案

```
sounds[event.unicode].stop() # 停止播放

pygame.display.flip()
clock.tick(60)

pygame.quit()
这段程序可以正常运行，可以帮助我为这段程序添加合理的可视化吗？
我希望不同的按键按下时，会有不同的视觉效果。
```



要增加可视化效果以显示哪个音符正在播放，我们可以在屏幕上为每个音符创建一个区域，并在相应的按键被按下时改变这些区域的颜色。以下是如何修改你的代码来实现这个功能：

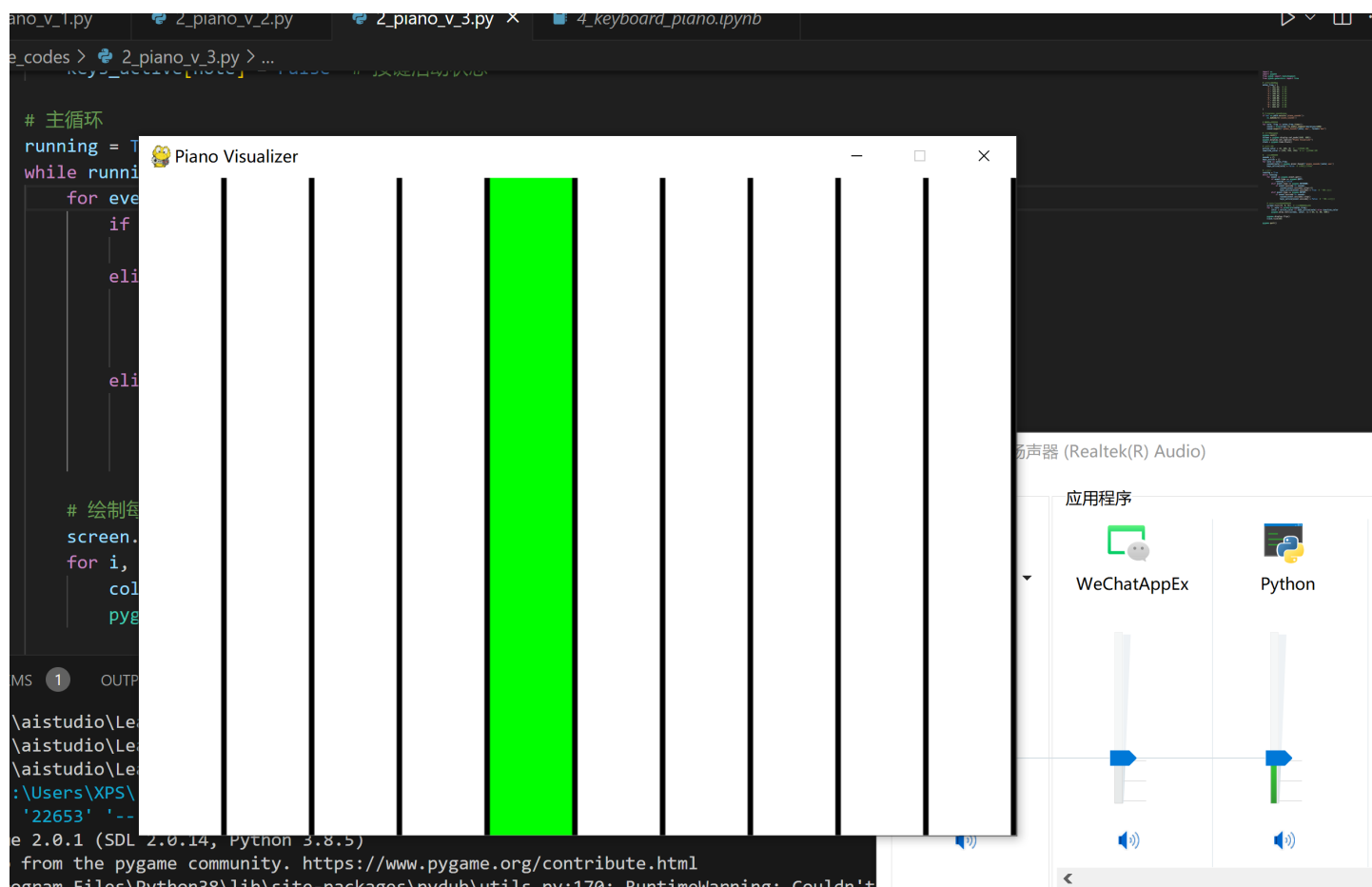
```
python 复制代码

import os
import pygame
from pydub import AudioSegment
from pydub.generators import Sine

# 定义音符频率
notes_freq = {
    '1': 261.63, # C4
```

最终，ChatGPT生成的程序合理实现了他自己预期的方案。





我们看代码上，

```
1 import os
2 import pygame
3 from pydub import AudioSegment
4 from pydub.generators import Sine
5
6 # 之前生成音符的代码
7 # ....详情见 lesson1/piano_v_3.py...
8
9 # 初始化pygame
10 pygame.init()
11 screen = pygame.display.set_mode((640, 480))
12 pygame.display.set_caption("Piano Visualizer")
13 clock = pygame.time.Clock()
14
15 # 定义颜色
16 active_color = (0, 255, 0) # 激活时的颜色
17 inactive_color = (255, 255, 255) # 非激活时的颜色
18
19 # 加载音频文件
20 sounds = {}
21 keys_active = {}
22 for note in notes_freq:
```

```

23     sounds[note] = pygame.mixer.Sound(f'piano_sounds/{note}.wav')
24     keys_active[note] = False # 按键活动状态
25
26 # 主循环
27 running = True
28 while running:
29     for event in pygame.event.get():
30         if event.type == pygame.QUIT:
31             running = False
32         elif event.type == pygame.KEYDOWN:
33             if event.unicode in sounds:
34                 sounds[event.unicode].play(-1)
35                 keys_active[event.unicode] = True # 标记为活动
36         elif event.type == pygame.KEYUP:
37             if event.unicode in sounds:
38                 sounds[event.unicode].stop()
39                 keys_active[event.unicode] = False # 标记为非活动
40
41     # 绘制每个音符的可视化
42     screen.fill((0, 0, 0)) # 用黑色填充屏幕
43     for i, note in enumerate(notes_freq):
44         color = active_color if keys_active[note] else inactive_color
45         pygame.draw.rect(screen, color, (i * 64, 0, 60, 480))
46
47     pygame.display.flip()
48     clock.tick(60)
49
50 pygame.quit()
51

```

他实际上建立了一个key\_active的列表（关于列表数据结构我们也会在后面的课程说明），在每个键按下或者抬起的时候，就会更改列表中对应该元素的状态，再根据这个状态，来修改对应方块的颜色。

我在不同的ChatGPT中试验了这个需求提示词，发现大多数情况下，语言模型都会用矩形和颜色的形式来渲染不同的音符。当然，如果同学们想实现更复杂的可视化效果，比如去可视化不同的波形，或者是像音游一样出现矩形瀑布流一样的效果，就需要花更多的时间去调整需求的描述。在这里我们留给同学们自己去实验了。

## 小结

我们其实希望在本课程较早期的部分，就给同学们展示完整的一个ChatGPT辅助编程的流程。并且实现一个具有互动形式的程序。键盘钢琴就是一个这样的例子。在这里我们其实做了很多尝试，包括在

第一次尝试中，遇到文件缺失而报错，从而重新分析需求，调整了需求提示词，再次生成了程序。以及进一步要求ChatGPT修改程序，增加可视化等等。这在我们后面的课程中，会反复出现。

## 课后练习

### 修改音色（电子）

在这里我们生成音乐的时候使用了`math.sin`函数来生成正弦波形式的音频。

- 与ChatGPT讨论，为什么使用`sin`形式的函数来生成音频。这样的音频和正常的器乐声音有什么区别？

然后尝试这个改进

- 实际上在电子音发展的早期，有很多以前的音乐会使用三角波、方波等波形，尝试把wav替换为这些波形进行播放

### 修改音色（自然）

查询不同乐器的音色（谐频分布），尝试让程序演奏接近某种乐器的电子音

### 方便和弦

电子琴能够方便地按出和弦伴奏（比如C E G构成C大和弦）。一般来说，四个左右的和弦就可以合理地为首歌进行伴唱。能否让程序实现简单地和弦，如果能在课堂上展示键盘弹唱，就更有意思了。

### \* 简易音游

简单实现一首歌的谱子，并以瀑布流的形式展示音符，要求用户在对应时间，按下对应的按键，再根据按键的命中与否，判断一首歌的评分。实现这样一个简易的音游程序。

[目 第一课-钢琴键盘](#)

[目 第二课 循环语句](#)

[目 第三课 点菜程序与条件语句](#)

[目 第四课 在程序中调用大模型](#)

[目 第五课 吃豆子](#)

[目 第六课 文字搜索和RAG](#)

[目 第七课 背单词软件](#)