这张图片是我们准备设计的界面的草图，是一种新的键盘排列用于我们上述的项目。具体内容如下：

中央是一个同心圆，其中小圆中有一个‘_____’的窗口用于呈现拼写的英文字符，如果字符串长度超出下划线就向左滚动

外圈的圆环被分成了四等分（上下左右四个方向），分组呈现了 26 各英文字母：

右边的象限为五个英语元音

上边象限为英文最常用的七个辅音

左边象限为依次顺序的另外七个辅音

下边象限为最不常用的七个辅音

具体选择单个字母的流程如下：

首先，将 cursor 划到相应的区域，cursor 经过的区域都会被点亮

如果 cursor 点击了四个象限中的一个，轮盘的布局会变换

以那一象限为中心，圆环被重新划分为两部分（两个半圆）

其中包含之前被选中象限的那个半圆又被七等分为小的扇形（瓦片形）

之前象限中包含的 5/7 个字母就被分配到扇形中（注意，如果是五个原因的话，七个扇形中最左和最右两格都是空的）

这时，cursor 划过时还是会点亮相应的扇形区域

如果 cursor 再次选中并点击就会将相应的字符存放到中央圆的窗口中

接下来注意到大圆外围被再次分成了四个区域（左上、左下、右上、右下）

左上的操作为数字/字母切换：如视图 6 中一样，字符 0-9 外加小数点占用了十一个扇形区域，被一次分配到除正下方三个以外的扇形区域，被选中的方法同字符相同。反之，如果当前界面为 NUM，则这个区域标为 CHR

右上的操作时为返回，当外环的四个想先之一被选中并进入半圆界面时，用户可以点击这个区域重新返回四象限界面。这时当发生误触时的改正方法

左下和右下分别是叉和勾，这两个符号对应的操作有 overloading:

当大圆的字符窗口没有存入任何东西的时候，直接选中这两个区域直接输出“Yes”和“No”

当用户开始拼写并且已存入未完成的字符串时，再选入叉表示删除上一个字符，同时字符窗口回划调整显示

当用户开始拼写并且已存入未完成的字符串时，分为两种情况：如果字符串最后一个字符为字母，则加一个空格。如果上一个字符为空格，则勾就代表确认键，这时整段窗口存入的字符串就会被输出到 text-to-speech 模组用于生成一个 MP3 文件

以上时目前我们的所有操作。通过搜索我们了解到使用 AAC 的患者可能对强光刺激不适，背景使用黑色。但是不同区域的边界使用深蓝色和淡蓝色来区分。同时被选定而点亮的部分如图示用黄色。勾和叉依惯例用绿色和红色，返回和 NUM 使用深蓝色。注意，所有颜色不要使饱和度过强，有适当的透明。

将这段依我们的需求改一改加入进键盘中

注意，目前的展示界面是在电脑上运行，而选择是通过鼠标完成。将来这个项目的输入操作会用眼动追踪技术，也就没有点击 click 这个操作，而是依靠停留在某一区域的时长判断的。我们到时会写一个转换模组将眼动的坐标转换为和 cursor 类似的指令。

为了将来更方便，在现在的程序中加入两种确认的方式：其一就是前面提到的，当鼠标单击 click 后选中，第二种是未来眼动时需要的，就是当鼠标滑动到一个位置后，那一区域的颜色会渐变，从纯黑到逐渐亮度提升直到和边界框架的颜色相同。这中间的判定时间先预设为 1.5 秒 （用颜色的亮度来表示进度），时间满了就自动确认当前区域的指令。

我们想先用 PySimpleGUI 来设计，因为其简介和易上手性。请依照上述的需求编写我们的程序，我接下来会在 IDE 中运行你写的代码并不断要求改进。

This image is a sketch of the interface we are going to design, a new keyboard arrangement for use in our above project

The details are as follows

In the centre is a concentric circle, where the smaller circle has a '_____' window that presents the spelled out English characters, scrolling to the left if the length of the string exceeds the underscore.

The outer circle is divided into quadrants (top, bottom, left, right, and four directions), grouping the 26 letters of the English alphabet:

The right quadrant shows the five English vowels

The upper quadrant shows the seven most common consonants in English.

The left quadrant shows the other seven consonants in descending order.

The bottom quadrant shows the seven least common consonants.

The procedure for selecting individual letters is as follows:

First, the cursor is drawn to the corresponding area, and the area where the cursor passes through will be lit.

If the cursor clicks on one of the four quadrants, the layout of the wheel will change.

Centred on that quadrant, the wheel is re-divided into two parts (two semicircles)

The half-circle containing the previously selected quadrant is subdivided into smaller sectors (tiles) in equal parts

The 5/7 letters contained in the previous quadrant are assigned to the sectors (note that if there are five reasons, the leftmost and rightmost squares of the seven sectors are empty).

At this point, the cursor will still light up the corresponding sector when it crosses over it.

If the cursor is selected again and clicked on it, the corresponding character is stored in the window in the centre circle.

Next, notice that the periphery of the large circle is again divided into four areas (upper left, lower left, upper right, lower right).

The upper-left operation is a numeric/alphabetic toggle: as in view 6, the characters 0-9 plus the decimal point take up eleven sectors, and are assigned to all but the three directly below them at once, and are selected in the same way as characters. Conversely, if the current interface is NUM, this area is labelled CHR.

The upper right operation is return, when the outer ring of the four want to first one is selected and enter the semicircle interface, the user can click on this area to return to the four quadrant interface. This is the corrective method in case of mis-touching.

The lower left and lower right are fork and tick respectively, and the operation corresponding to these two symbols has overloading:

When nothing has been loaded into the big circle character window, selecting these two areas directly outputs ‘Yes’ and ‘No’.

When the user starts to spell and has already deposited an unfinished string, then checking in the fork means deleting the previous character, while the character window swings back to adjust the display.

When the user starts to spell and has saved the unfinished string, there are two cases: if the last character of the string is a letter, then add a space. If the last character of the string is a space, then the tick represents the confirmation key, and then the entire string deposited in the window will be output to the text-to-speech module for generating an MP3 file.

That's all we've done so far. Through our search we have learnt that patients using AAC may be uncomfortable with bright light stimuli and the background is black. The background is black, but the borders of the different areas are differentiated using dark blue and light blue. The parts that are simultaneously selected and lit as shown are in yellow. Ticks and crosses are conventionally coloured green and red, and returns and NUM are coloured dark blue. Note that all colours should not be over saturated and have proper transparency.

Add this to the keyboard with the changes we need.

Note that the current presentation interface is running on a computer and the selection is done by mouse. In the future, this project will use eye-tracking for input operations, so there will be no clicks, but rather a judgement based on the length of time spent in a certain area. We will write a conversion module to convert the coordinates of the eye movement into a cursor-like command.

In order to make it more convenient in the future, we will add two kinds of confirmation methods in the current program: one of them is the one mentioned before, when the mouse clicks on the click, the second one is the one we need in the future, that is, when the mouse slides to a position, the colour of the area will be changed gradually, from pure black to the gradual increase of brightness until it is the same as the colour of the border frame. The decision time in between is preset to 1.5 seconds (with the brightness of the colours indicating the progress), and when the time is up, the command for the current region is automatically confirmed.

We would like to start with tkinter because of its native support by python and ease of use. Please write our programme according to the above requirements, I will then run the code you have written in the IDE and keep asking for improvements.