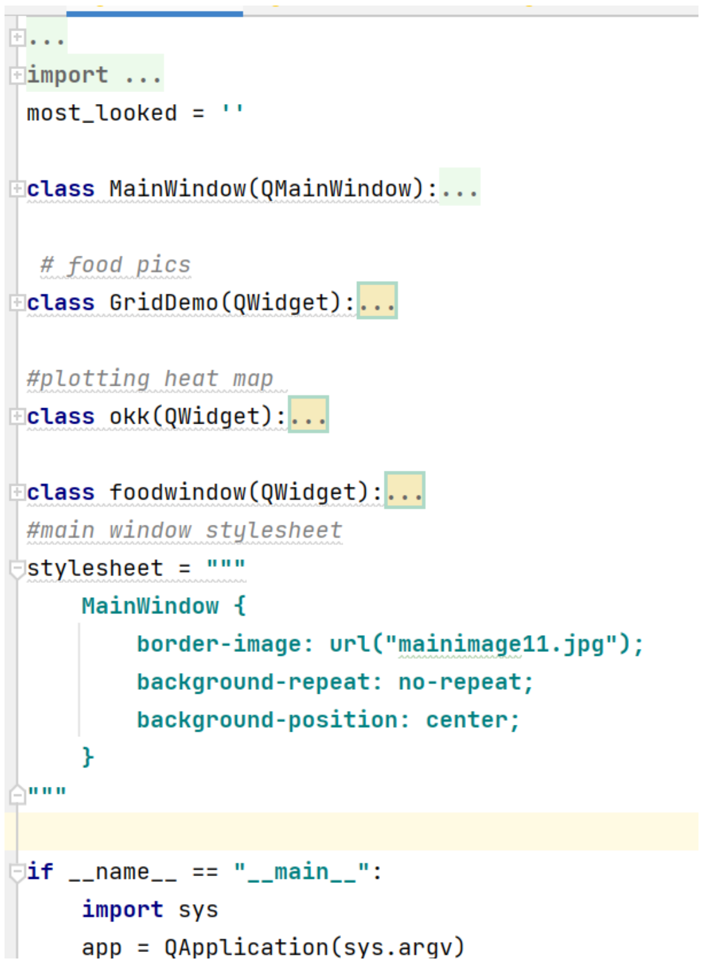
Food Recommender App (PYQT5)'s project code structure

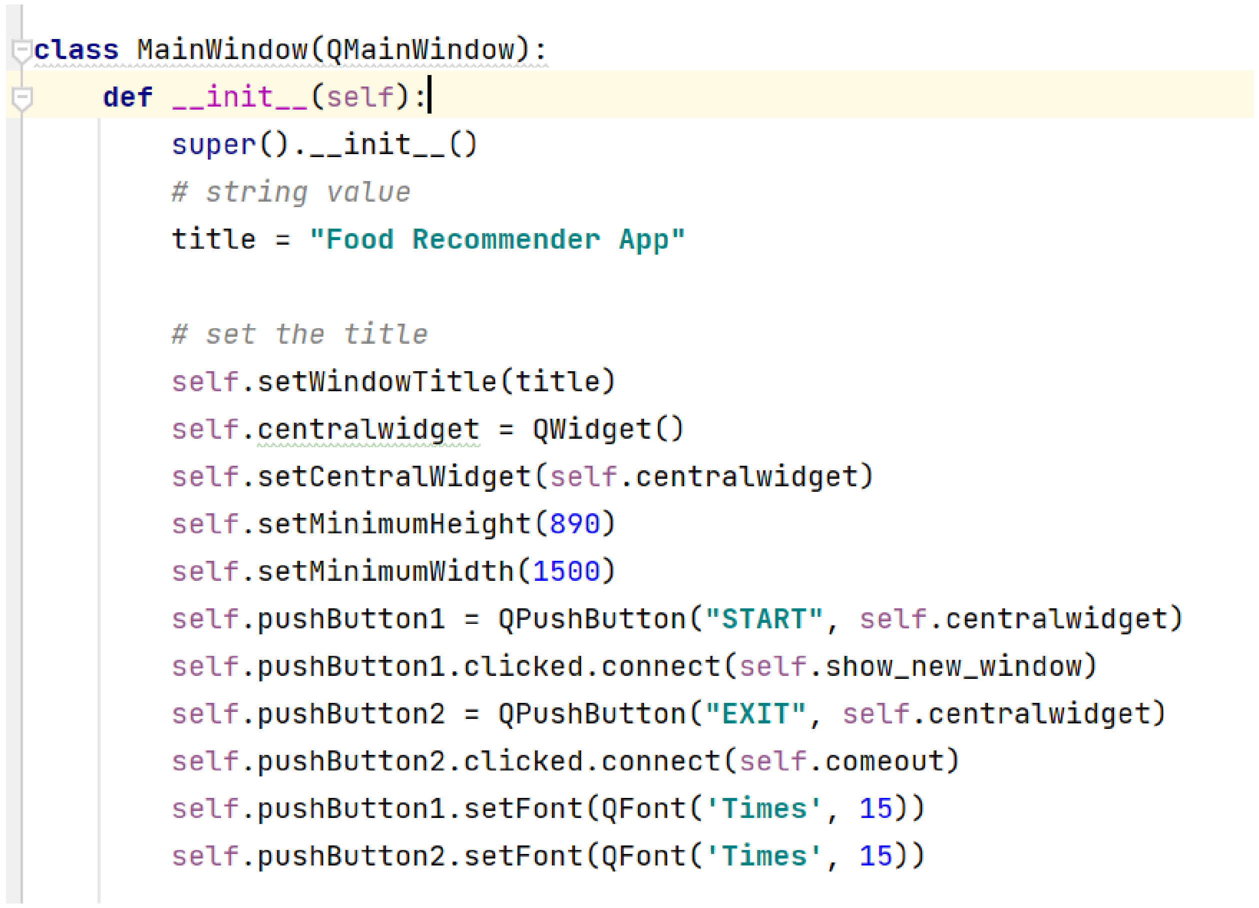
This is a replicated and improved version of the previous food recommender app that was built with the tkinter interface.



```
...
import ...
most_looked = ''

class MainWindow(QMainWindow):...

 # food pics
class GridDemo(QWidget):...

#plotting heat map
class okk(QWidget):...

class foodwindow(QWidget):...
#main window stylesheet
stylesheet = """
    MainWindow {
        border-image: url("mainimage11.jpg");
        background-repeat: no-repeat;
        background-position: center;
    }
"""


if __name__ == "__main__":
    import sys
    app = QApplication(sys.argv)
```

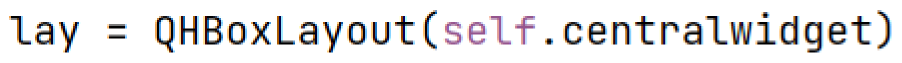
This is the overview of the code structure of the new food recommender app. It mainlys comprises 4 big sections. There are 4 classes , Mainwindow , Griddemo, okk and foodwindow and at the bottom , if __name__ is the part where we run the entire app.

Main page design



```python
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        # string value
        title = "Food Recommender App"

        # set the title
        self.setWindowTitle(title)
        self.centralwidget = QWidget()
        self.setCentralWidget(self.centralwidget)
        self.setMinimumHeight(890)
        self.setMinimumWidth(1500)
        self.pushButton1 = QPushButton("START", self.centralwidget)
        self.pushButton1.clicked.connect(self.show_new_window)
        self.pushButton2 = QPushButton("EXIT", self.centralwidget)
        self.pushButton2.clicked.connect(self.comeout)
        self.pushButton1.setFont(QFont('Times', 15))
        self.pushButton2.setFont(QFont('Times', 15))
```

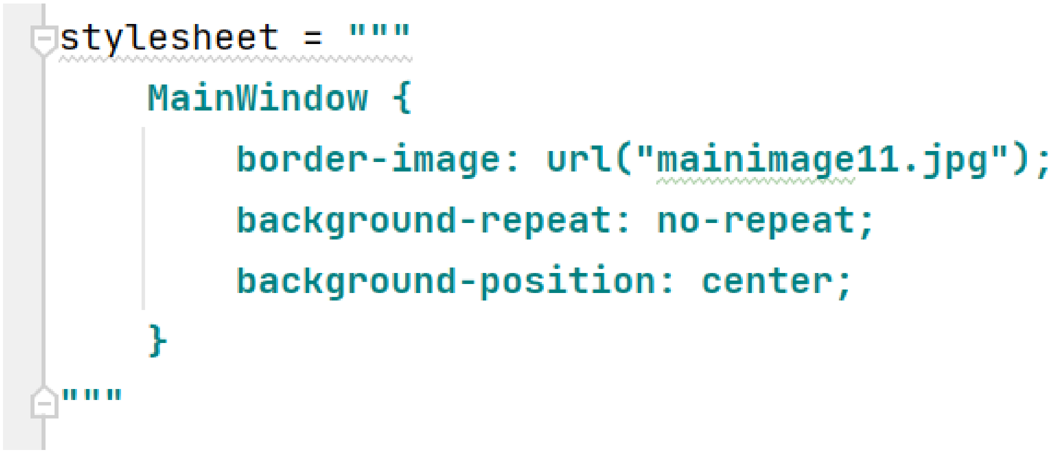
This is the section where the buttons of the main page and the design of the main page is at . Do note that since Mainwindow has QMainwindow as its parameter and Qwidget are set to be setCentralWidget , the adjustments here work differently than the other windows.



```python
lay = QHBoxLayout(self.centralwidget)
```

Also , when setting layout here , you've to put (self.centralwidget) inside the parameter of the layout.

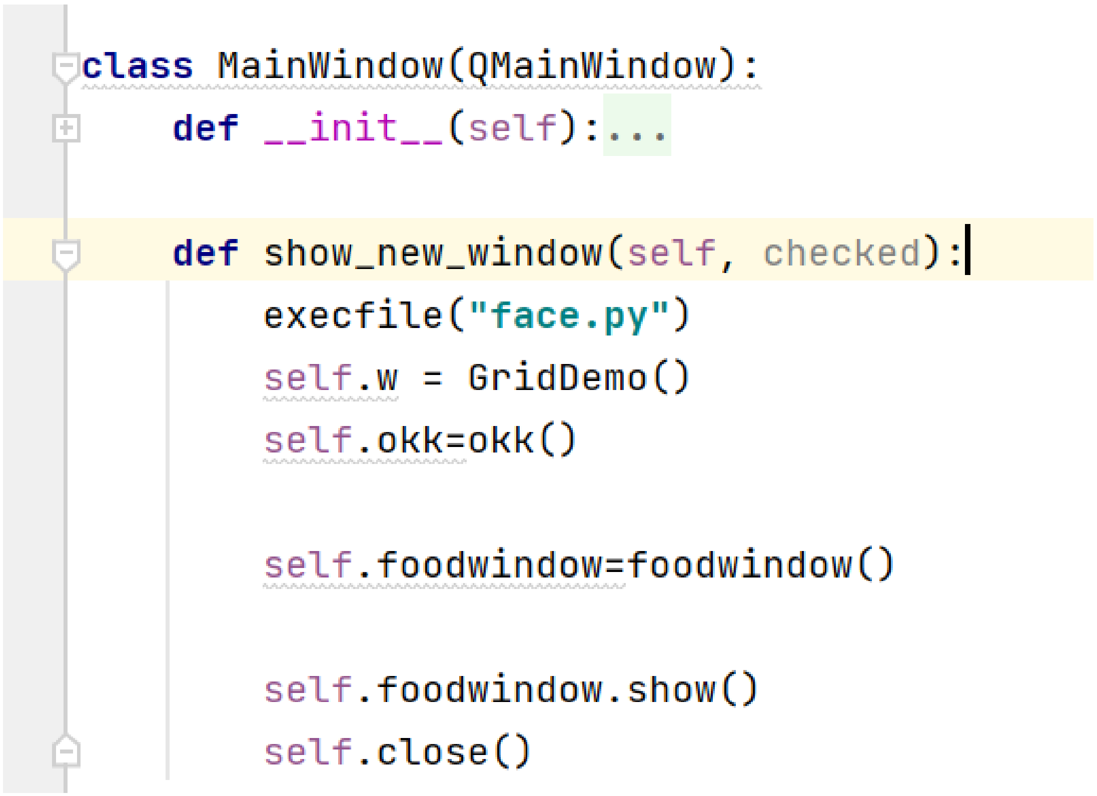
Main page picture



```
stylesheet = """
    MainWindow {
        border-image: url("mainimage11.jpg");
        background-repeat: no-repeat;
        background-position: center;
    }
"""
```

^This is where you can change the main page image

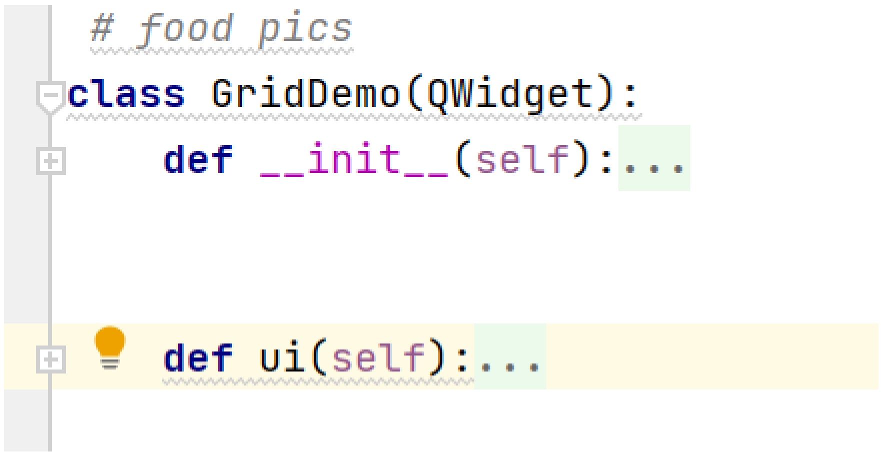
Switching around Windows pane in the app



```
class MainWindow(QMainWindow):
    def __init__(self):...

    def show_new_window(self, checked):
        execfile("face.py")
        self.w = GridDemo()
        self.okk=okk()

        self.foodwindow=foodwindow()

        self.foodwindow.show()
        self.close()
```

This is the section where the different windows are called and closed so that the app flows smoothly . It is advisable that you shouldn't change the flow here as it might affect the flow of the App.

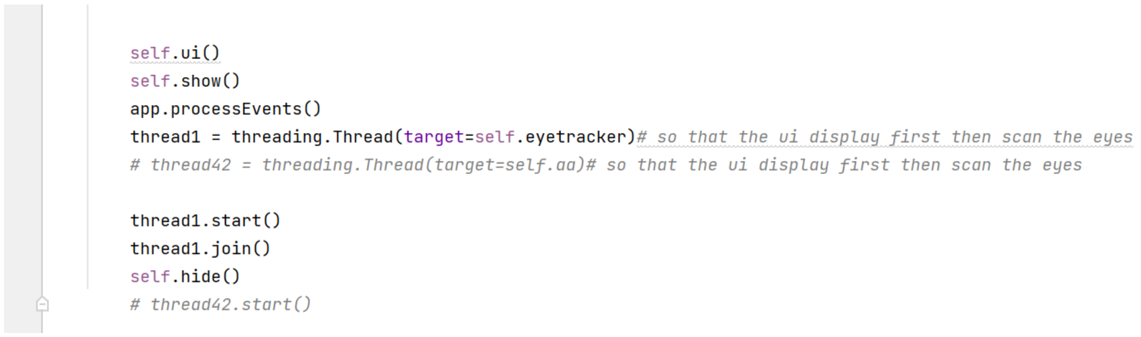
Menu Window



```
# food pics
class GridDemo(QWidget):
    def __init__(self):...


    def ui(self):...
```

This is the part where the different food menu will be randomized and show up on screen for the user to gaze at. Images can be changed in the ui function.

Sorting the order of letting image show first for the user to gaze then show heatmap

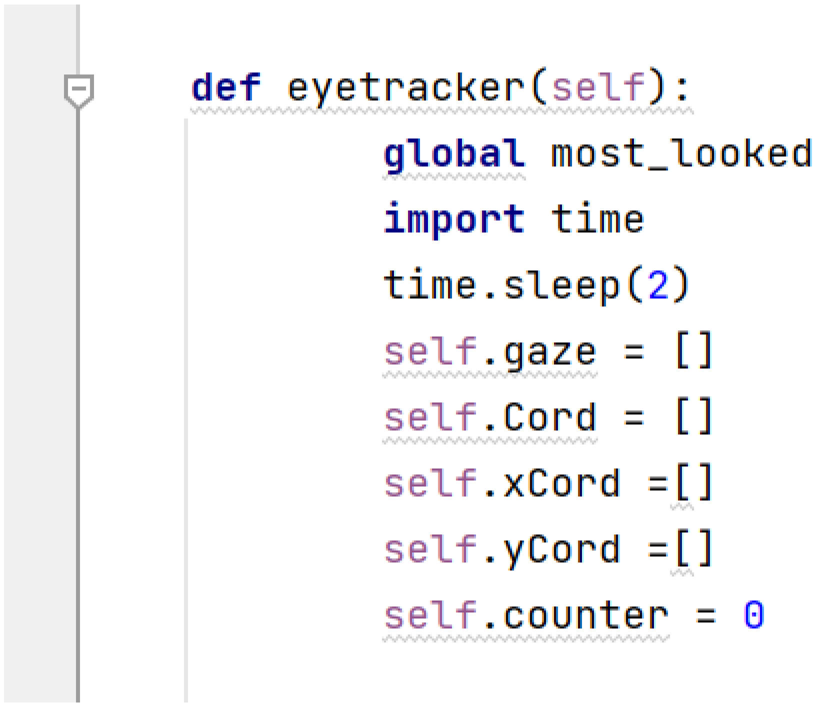


```
self.ui()
self.show()
app.processEvents()
thread1 = threading.Thread(target=self.eyetracker)# so that the ui display first then scan the eyes
# thread42 = threading.Thread(target=self.aa)# so that the ui display first then scan the eyes

thread1.start()
thread1.join()
self.hide()
# thread42.start()
```

^ This can be found in the grid demo classes, under the init function. These few lines will order which function should run first. It's advisable not to change anything here.

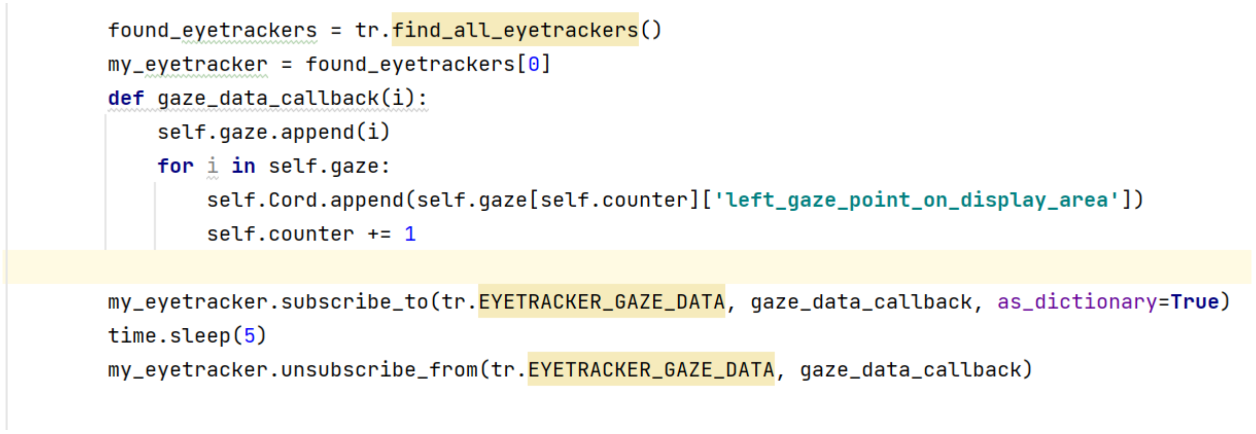
Eyetracker function

Eye Tracker codes fall under grid demo.



```
def eyetracker(self):
        global most_looked
        import time
        time.sleep(2)
        self.gaze = []
        self.Cord = []
        self.xCord =[]
        self.yCord =[]
        self.counter = 0
```

<u>Starting and off-ing the eye tracker device</u>

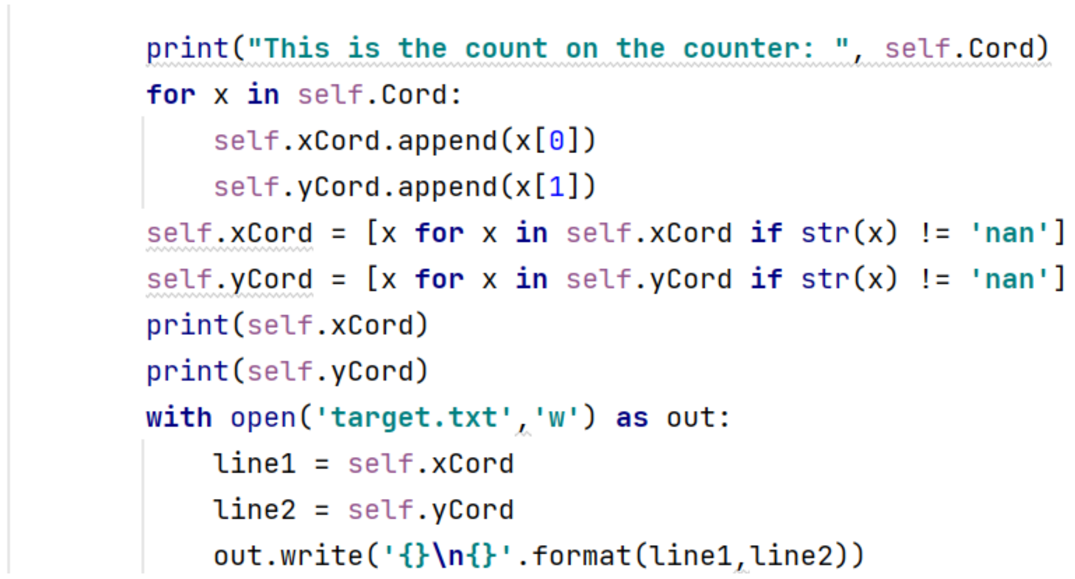


```
        found_eyetrackers = tr.find_all_eyetrackers()
        my_eyetracker = found_eyetrackers[0]
        def gaze_data_callback(i):
            self.gaze.append(i)
            for i in self.gaze:
                self.Cord.append(self.gaze[self.counter]['left_gaze_point_on_display_area'])
                self.counter += 1

        my_eyetracker.subscribe_to(tr.EYETRACKER_GAZE_DATA, gaze_data_callback, as_dictionary=True)
        time.sleep(5)
        my_eyetracker.unsubscribe_from(tr.EYETRACKER_GAZE_DATA, gaze_data_callback)
```

This part of the code controls when the eye tracker will turn on and how many seconds it will be on for .

- The line where there's subscribe_to , is the part where the eye tracker is called
- The line where the unsubscribe is found , tis the part where the eye tracker is off

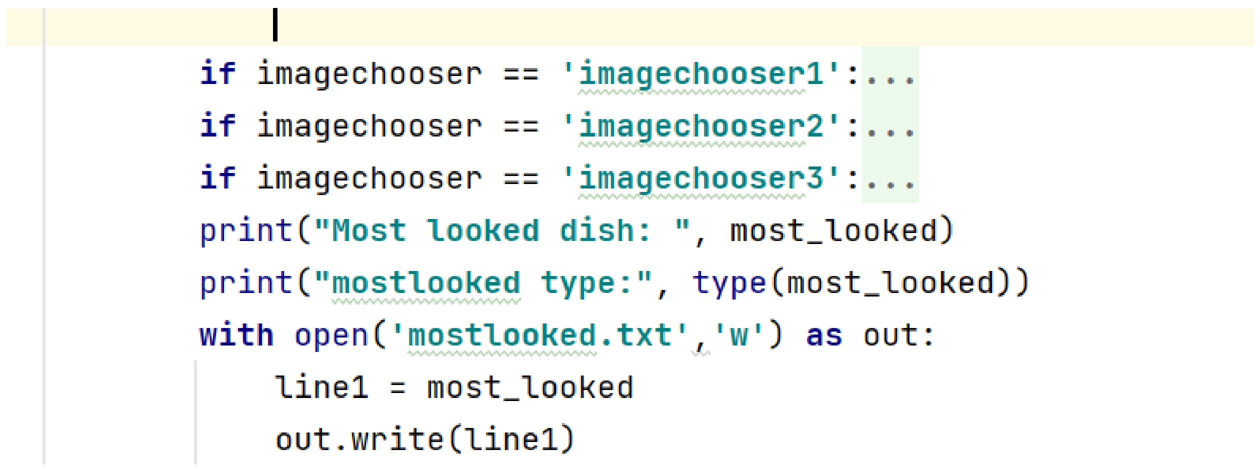
Sorting out the data collected from the eye tracker device



```
print("This is the count on the counter: ", self.Cord)
for x in self.Cord:
    self.xCord.append(x[0])
    self.yCord.append(x[1])
self.xCord = [x for x in self.xCord if str(x) != 'nan']
self.yCord = [x for x in self.yCord if str(x) != 'nan']
print(self.xCord)
print(self.yCord)
with open('target.txt','w') as out:
    line1 = self.xCord
    line2 = self.yCord
    out.write('{}\n{}'.format(line1,line2))
```

Under the eye tracker function , there's this part of the code where it sorts out the data collected from the eye tracker device and it's splitted to x and y coordinate and saved into a string.

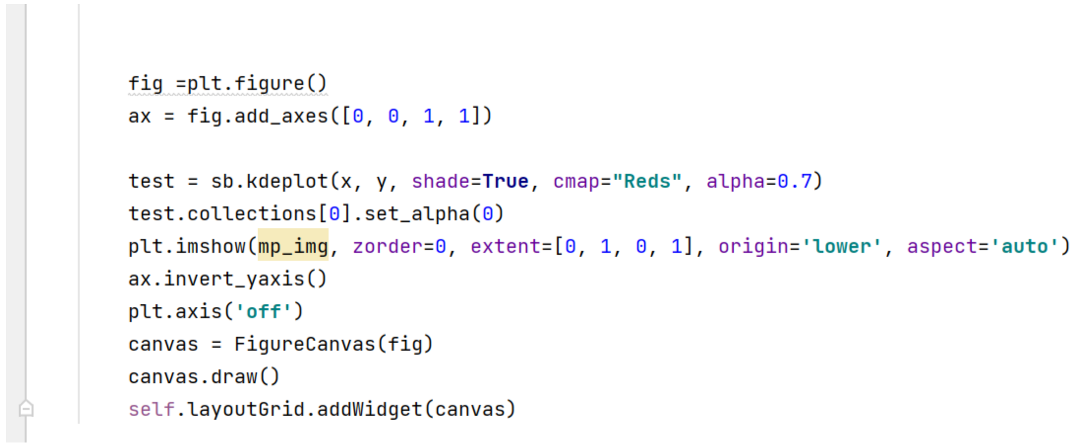
Setting the coordinates of each food item



```
if imagechooser == 'imagechooser1':...
if imagechooser == 'imagechooser2':...
if imagechooser == 'imagechooser3':...
print("Most looked dish: ", most_looked)
print("mostlooked type:", type(most_looked))
with open('mostlooked.txt','w') as out:
    line1 = most_looked
    out.write(line1)
```

^ This is the section where you change the coordinates of the food image so that in whatever range that the user looks at , it will be counted as that food .

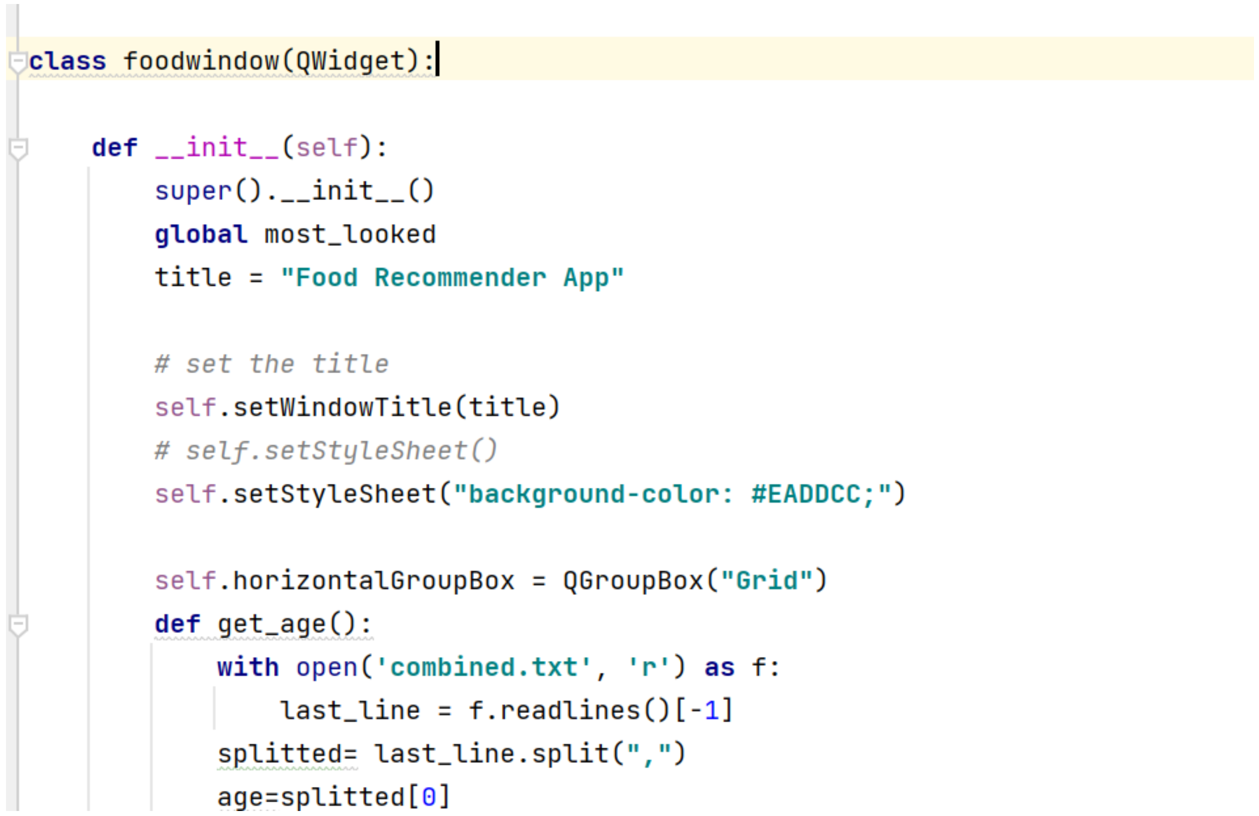
Plotting the heatmap



```
fig =plt.figure()
ax = fig.add_axes([0, 0, 1, 1])

test = sb.kdeplot(x, y, shade=True, cmap="Reds", alpha=0.7)
test.collections[0].set_alpha(0)
plt.imshow(mp_img, zorder=0, extent=[0, 1, 0, 1], origin='lower', aspect='auto')
ax.invert_yaxis()
plt.axis('off')
canvas = FigureCanvas(fig)
canvas.draw()
self.layoutGrid.addWidget(canvas)
```

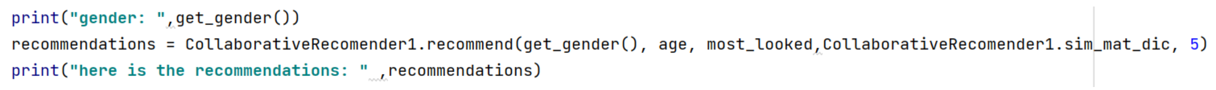
^ This is the function that uses the x and y coordinates to plot the heat map . Heat map here represents the evidence of where the user gazes at . This helps users to double check where they have gaze at. It is under okk class and it's in the ui2 function.

Recommendation Pane



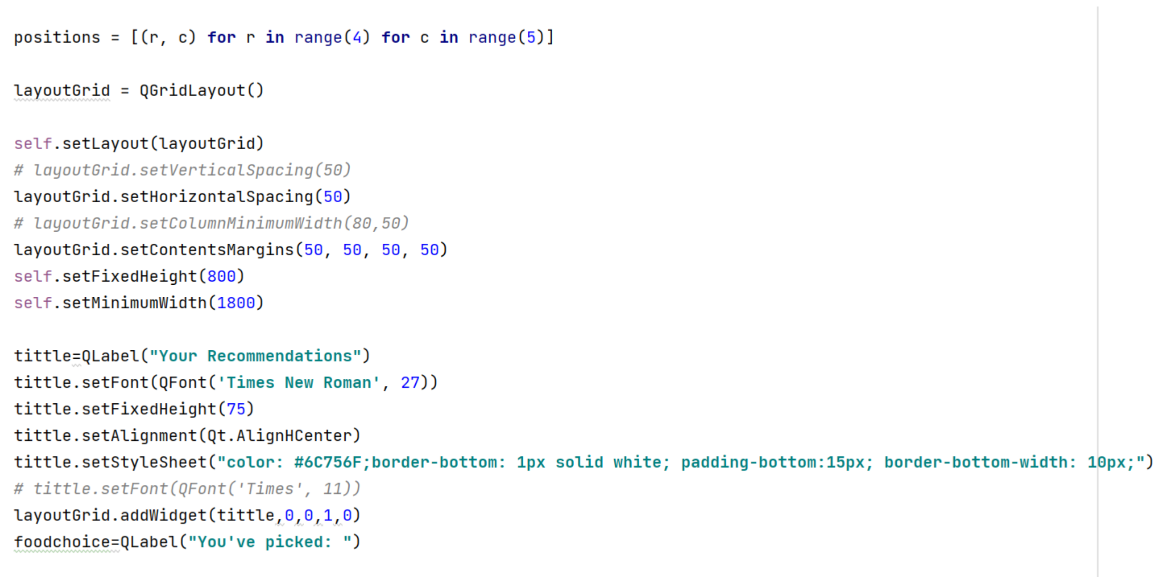
```
class foodwindow(QWidget):

    def __init__(self):
        super().__init__()
        global most_looked
        title = "Food Recommender App"

        # set the title
        self.setWindowTitle(title)
        # self.setStyleSheet()
        self.setStyleSheet("background-color: #EADDCC;")

        self.horizontalGroupBox = QGroupBox("Grid")
        def get_age():
            with open('combined.txt', 'r') as f:
                last_line = f.readlines()[-1]
            splitted= last_line.split(",")
            age=splitted[0]
```

^This is the section where the design of the recommendation pane and the calling of the model is at.



```
print("gender: ",get_gender())
recommendations = CollaborativeRecomender1.recommend(get_gender(), age, most_looked,CollaborativeRecomender1.sim_mat_dic, 5)
print("here is the recommendations: " ,recommendations)
```

^ Model is called here and recommendations will be given back .



```python
positions = [(r, c) for r in range(4) for c in range(5)]

layoutGrid = QGridLayout()

self.setLayout(layoutGrid)
# layoutGrid.setVerticalSpacing(50)
layoutGrid.setHorizontalSpacing(50)
# layoutGrid.setColumnMinimumWidth(80,50)
layoutGrid.setContentsMargins(50, 50, 50, 50)
self.setFixedHeight(800)
self.setMinimumWidth(1800)

tittle=QLabel("Your Recommendations")
tittle.setFont(QFont('Times New Roman', 27))
tittle.setFixedHeight(75)
tittle.setAlignment(Qt.AlignHCenter)
tittle.setStyleSheet("color: #6C756F;border-bottom: 1px solid white; padding-bottom:15px; border-bottom-width: 10px;")
# tittle.setFont(QFont('Times', 11))
layoutGrid.addWidget(tittle,0,0,1,0)
foodchoice=QLabel("You've picked: ")
```

^ Here is where the food pics will appear as a grid form , with clickable images.

END