Code Explanation:

Library or Module:

import sys	sys module helps to access some variables which has been maintained by the interpreter.
import time	time module offers various time-related functions.
import random	random module imports the random module, which contains a variety of things to do with random number generation.
import winsound	winsound module provides access to the basic sound-playing machinery provided by Windows platforms.
import threading	Threading in python is used to run multiple threads at the same time.
from PyQt5.QtWidgets import QApplication, QWidget, QLabel, QMainWindow, QPushButton, QMessageBox	PyQt5.QtWidgets is popular module in python GUI. From that module, some of the function has been used such as QApplication which contains main event loop, QWidget which is the base class of all user interface objects and QMainWindow which provide main application window and so on
from PyQt5.QtGui import QPainter, QColor, QImage, QPalette, QBrush,QFont, QIcon	The PyQt5.QtGui module deals with the graphical elements. It has some function or classes. For example, QPainter performs low-level painting on widgets, QPalette contains color groups for each widget state and so on.
from PyQt5.QtCore import Qt, QPoint, QBasicTimer, QSize	The QtCore module comprises the core non-GUI classes, including the event loop and Qt's signal and slot mechanism. It consists some classes such as QPoint, QBasicTimer, QSize.

Keypad Direction:

UP = Qt.Key_Up DOWN = Qt.Key_Down RIGHT = Qt.Key_Right LEFT = Qt.Key_Left	UP is variable which has been connected with keyboard upper arrow which increase the speed. Similarly, DOWN is connected with down arrow button which stops the movement board. RIGHT and LEFT helps to move the board right and left respectively.

Main Class:

class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
	Class has been declared as MainWindow. Then the constructor function has been declared.

# Setting the title and icon for main game window 	
self.setWindowTitle('Pong')
	The title of the window has been declared as Pong.

self.setWindowIcon(QIcon('pong.png'))	The pong icon in the top of the window has been set.

# Background image 	
oImage = QImage("test.png")
 	
# Set the window size e.g. height and width 	
sImage = oImage.scaled(QSize(500,500))
palette = QPalette()
palette.setBrush(10, QBrush(sImage))
self.setPalette(palette)
self.resize(500, 500)
	Background image has been set up and properly placed over the window using QPaletter.

# Setting the size, position, font and color of the Pushbutton 	
btn1 = QPushButton('Play Games', self)
btn1.move(400, 350)
btn1.resize(100,50)
btn1.setStyleSheet("background-color: gray")
btn1.setFont(QFont('SansSerif', 12))
	Button has been defined using QPushButton. The position, background-color and font size have also been declared.

btn1.clicked.connect(self.openSecond)	Button1 has been connected with the function called openSecond class.
btn2 = QPushButton('Close', self) btn2.move(400, 450) btn2.resize(100,50) btn2.setStyleSheet("background-color: gray") btn2.setFont(QFont('SansSerif', 12))	
btn2.clicked.connect(self.CloseApp)	Button2 has been set to close the window of the game.
btn3 = QPushButton('Help', self) btn3.move(400, 400) btn3.resize(100,50)	

`btn3.setStyleSheet("background-color: gray")` `btn3.setFont(QFont('SansSerif', 12))`	
`btn3.clicked.connect(self.openHelp)`	Button3 has been connected with openHelp function.

`def openSecond(self):`
`# Opening Second Page`
`self.SW = App()`
`self.SW.show()`
	openSecond function calls the main game contained by App class.

`def CloseApp(self):`
`# Closing the game`
`reply = QMessageBox.question(self,"Close Message","Are You Sure to Close Window", QMessageBox.Yes | QMessageBox.No, QMessageBox.No)`
`if reply == QMessageBox.Yes:`
`self.close()`
	Function CloseApp closes the main window by asking second time to the user. Here, QMessageBox has been used for that purpose.

`def openHelp(self):`
`# Opening help page`
`self.WW = Help()`
`self.WW.show()`
	Function openHelp calls another class named Help which contains instruction for the game.

Class Help:

`class Help(QMainWindow):`	The class Help has been declared as another main window.

`# Setting the window size e.g. height and width`
`def __init__(self):`
`super().__init__()`
`self.title = "Help"`
	Constructor has been declared which contains window title, icon, size.

`self.setWindowIcon(QIcon('help.png'))`
`self.left = 435`
`self.top = 115`
`self.width = 500`
`self.height = 500`
`self.widget()`	

def widget(self):
 	
#Setting the background image
self.setWindowTitle(self.title)
self.setGeometry(self.left, self.top, self.width, self.height)
self.setFont(QFont('SansSerif', 12))
 	
oImage = QImage("test.png")
sImage = oImage.scaled(QSize(500,500))
palette = QPalette()
palette.setBrush(10, QBrush(sImage))
self.setPalette(palette)
self.resize(500, 500)
 	
#Instruction
label1 = QLabel("For Moving Left, press LEFT KEY", self)
label1.resize(400, 30)
 	
label2 = QLabel("For Moving Right, press RIGHT KEY", self)
label2.resize(400, 80)
 	
label3 = QLabel("For Stoping the paddle, press DOWN KEY", self)
label3.resize(400, 130)
 	
label4 = QLabel("For Slow Motion, press UP KEY", self)
label4.resize(400, 180)
 	
label5 = QLabel("For Going Back to Main Window, press ESCAPE", self)
label5.resize(400, 230)
 	
label6 = QLabel("For Pausing, press SPACE or P", self)
label6.resize(400, 280)
 	
self.show()
	Background image has been set with the help of QPalette.
	
	Level contains informations to play the game.
	
	The show function appears the window.

Class APP:

class App(QWidget):	The class App has been declared.
# Setting the User Interface, layout, scorce def __init__(self): super().__init__() self.padding = 500 self.highScore = 0 self.time = QBasicTimer() self.UI() self.start()	The constructor function setting the shape of the window, high score, time. It also declaring UI (User Interface) and start function.
def UI(self):	

# Setting the icon, background image, score 	
self.setWindowTitle('Pong')
self.setFixedSize(self.wh, self.wh)
self.setWindowIcon(QIcon('pong.png'))
 	
oImage = QImage("test.png")
sImage = oImage.scaled(QSize(500,500))
palette = QPalette()
palette.setBrush(10, QBrush(sImage))
self.setPalette(palette)
	Function User Interface has been defined.
	
	Window title, icon, window size and background image has been attached.

self.scoreLabel = QLabel('Score: 000', self) self.scoreLabel.move(390,0) self.scoreLabel.setFont(QFont('SansSerif', 12)) self.highScoreLabel = QLabel('Highscore: 000', self) self.highScoreLabel.move(390,15) self.highScoreLabel.setFont(QFont('SansSerif', 12)) self.show()	Score and High score have been fixed to certain location and set the value 000.
def paintEvent(self, e):	

# Setting paint color using painter 	
painter = QPainter()
	Function paintEvent has been defined.

painter.begin(self)
self.paintBoard(painter)
self.paintBall(painter)
painter.end()
 	
def keyPressEvent(self, e):
 	
# Setting the key 	
pressed = e.key()
 	
if pressed in (Qt.Key_P,
Qt.Key_Space):
self.pause()
 	
if pressed == Qt.Key_Escape:
self.pause()
self.close()
 	
elif pressed in (DOWN, RIGHT, LEFT):
self.direction = pressed
 	
elif pressed == UP:
self.speedingBall()
 	
def start(self):
 	
# Starting the game after 20 	
microsecond
self.time.start(20, self)
 	
mid = self.padding/2
self.score = 0
self.speed = 0
 	
# Setting the ball in the middle 	
self.ball = QPoint(mid, mid)
 	
# Setting the ball direction 	
x = random.choice([x for x in range(-
2,3) if x])
self.dirOfBall = QPoint(x, 3)
 	
self.pos = mid - 50
self.direction = DOWN
self.paused = False
 	
self.repaint()
 	
def pause(self):
 	
# Pausing the game 	
if self.paused:
	QPainter function has been called and passed it to the paintBall and paintBoard function.
	
	keyPressEvent function has been defined.
	
	Pauss button has been set by P or Space button of the keyboard.
	
	Escape button has been set to quit the game window.
	
	Down, Left and Right button depicts the stop mode, left mode or right mode of the board.
	
	Up button remarks speed mode of the ball.
	
	Start function has been defined.
	
	Starting time has been set to 20 microseconds.
	
	Score, speed has been initialized with zero.
	
	Ball will start moving from the middle position.
	
	Selecting ball direction randomly.
	
	position has been initialized to 450.
	Direction has been initialized to down.

self.paused = False
self.time.start(20, self)
else:
self.paused = True
self.time.stop()
 	
def latestScore(self):
 	
# Updating the score 	
self.scoreLabel.setText('Score: {}'.format(str(self.score).zfill(3)))
 	
if self.score > self.highScore:
self.highScore = self.score
 	
self.highScoreLabel.setText('Highscore: {}'.format(str(self.highScore).zfill(3)))
def paintBoard(self, painter):
 	
# Painting the board 	
painter.setBrush(QColor(0, 0, 0))
painter.drawRect(self.pos, self.wh-60, 100, 10)
 	
def paintBall(self, painter):
 	
# Painting the ball 	
painter.setBrush(QColor(51, 255, 51))
painter.drawEllipse(self.ball, 6, 6)
 	
def kill(self):
 	
# After unsuccessful attempt, the game starts again 	
self.time.stop()
self.score = 0
self.latestScore()
time.sleep(0.5)
self.start()
 	
def speedingBall(self):
 	
# Increasing the speed 	
if self.speed:
self.speed = 0
self.dirOfBall *= 0.5
return
self.dirOfBall *= 2
self.speed = 1
 	
def hitWall(self):
	Repaint function has been called to make the game dynamic.
	Function pauss function pausing the game by comparing the Boolean values.
	
	Function latestScore has been defined which update the score and high score.
	
	Changing the score board according to every successful hit by board.
	
	It is comparing the score with high score to decide the current score is high or not.
	
	Function paintBoard is painting and shaping the board.
	
	Function paintBall is painting and shaping the board.
	
	Function kill is stopping the game, making the score to zero and updating the high score. It makes the game stop for 0.5 millisecond and then starts again.
	
	Function speedingBall introduce slow or speed motion of the ball.

`        # Hitting the wall and change the direction`
`direction = False`
 	
`        if self.ball.y() - 6 <= 0:`
`            direction = UP`
`        elif self.ball.y() + 6 >= self.padding:`
`            direction = DOWN`
`        elif self.ball.x() + 6 >= self.padding:`
`            direction = RIGHT`
`        elif self.ball.x() - 6 <= 0:`
`            direction = LEFT`
 	
`        # Making sound when hit any side of the walls`
`        if direction != False:`
`            soundThread = threading.Thread(target=self.playSound, args = ('cross.wav',))`
`            soundThread.start()`
 	
`        return direction`
 	
`    def hitBoard(self):`
 	
`        # After touching the board and the ball will go up`
`if self.padding-50 >= self.ball.y() + 6 >= self.padding-60:`
`            if self.pos <= self.ball.x() <= self.pos + 100:`
`                self.score += 1`
`                self.latestScore()`
 	
`                soundThread = threading.Thread(target=self.playSound, args = ('circle.wav',))`
`                soundThread.start()`
`                return True`
`        return False`
 	
`    def playSound(self, nameOfFile):`
 	
`        # Playing the sound`
`        winsound.PlaySound(nameOfFile, winsound.SND_FILENAME)`
 	
`    def timerEvent(self, e):`
 	
`        # Changing the direction of the board`
`        if self.direction == RIGHT and self.pos < self.wh-100:`
	Function hitWall changing the direction of the ball after getting hit by the wall.
	
	When the ball hit the top, it sends the info directions as UP
	
	It will sound every time it touches the wall.
	
	Function hitBoard defined whether the ball has been touched the board or not.
	
	if the ball is in this range, the score will be updated.
	
	Every time it hits the board, a sound will be heard.
	
	Function playSound is thread to make sure the sound file doesn’t interrupt the movement of the ball.
	
	Function timerEvent has been defined.

`            self.pos += 6`
`        elif self.direction == LEFT and self.pos > 0:`
`            self.pos -= 6`
	The direction of the board can be controlled using the left and right arrow button.

`        hit = self.hitWall()`
 	
`        # Increasing the speed`
`        speed = self.speed + 1`	

`        if hit:`
 	
`            # Inrease the speed of the ball`
`            if hit == UP:`
 	
`self.balldir.setY(random.randint(2, 4)*speed)`
`            elif hit == LEFT:`
 	
`self.balldir.setX(random.randint(2, 4)*speed)`
`            elif hit == RIGHT:`
 	
`self.balldir.setX(random.randint(2, 4)*-1*speed)`
`            elif hit == DOWN:`
`                self.kill()`
 	
`        elif self.hitBoard():`
 	
`self.balldir.setY(random.randint(2, 4)*-1*speed)`
	Speed of the ball has been increased after hitting wall.
	
	If the ball hit the wall, it will change the direction randomly.

`        self.ball += self.balldir`	If the ball hit the board, the ball will go up.

`        self.repaint()`
 	
`if __name__ == "__main__":`
`     def run_app():`
`        app = QApplication(sys.argv)`
`        ex = MainWindow()`
`        ex.show()`
`        app.exec()`
`     run_app()`
	repaint function will provide dynamic mode of the game.
	
	The main file which executes the whole code.