SEARCH....

Search ... Search

SITEMAP

Python

Machine Learning

Pygame

Data Structures and Algorithms(Python)

Python Turtle

Games with Python

All Blogs On-Site

Python Compiler(Interpreter)

Online Java Editor

Online C++ Editor

Online C Editor

All Editors

Services(Freelancing)

RECENT POSTS

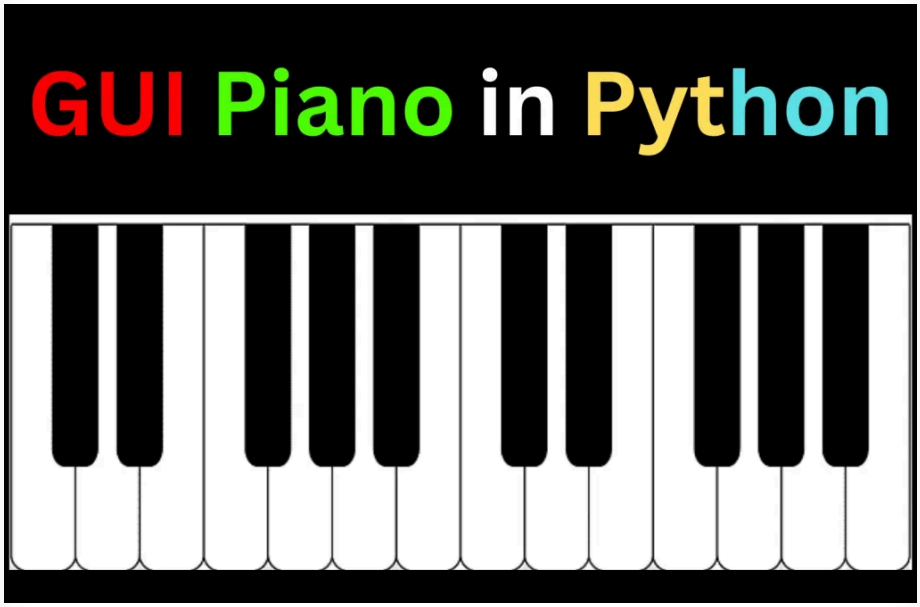
People are becoming AI Engineer with this free course in 2025: Here is how to join this…

Apply to Google's Student Training in Engineering Program (STEP) Intern, 2025

Self-Driving Car Saves Falling Pedestrian, Showcases Promise of

GUI Piano in Python

AYUSH PURAWR DECEMBER 12, 2022



Hello folks! Today we are back with one another article on **GUI Piano in Python**. We all have played with piano once in our life. Now today is the time to build one on ourselves. This piano project will have all the functionalities that our physical piano used to have. For the development of Piano in Python, we will use Python libraries, namely PyGame and pl. This will be a full-fledged working project wherein a user will get to learn a lot more about the PyGame library.

We will try to explain each and every code line in detail so that you can get a grip on concepts. Before moving on to the actual development of the Piano in Python, let us look at what we are going to build in detail and then go through the list of features we will add to this project on Piano in Python.

Basic idea

Here in this project on Piano in Python, we will build a fully functional advanced piano with all the keys that are available in the original piano. Our Piano's GUI will be based on a black-and-white theme. This piano will work based on various musical notes that will be used in the form of .wav file format and then we will link each key in the piano to the file in order to make the piano fully functional.

A note for all the readers of this project. Before moving ahead, let us talk about the directory structure of files for this project on GUI Piano in Python. Basically, the user will have to create the main folder and in that folder, there will be two **.py** files, namely: **main.py** and **piano_lists.py**. In the same folder, there will be a folder named **"assets"**, and in that folder, there will be another folder named **"notes"** which will hold all the different music files for this project on GUI Piano in Python. In the **"assets"** folder be a dedicated file for fonts with the name **Terserah. tff**

Autonomous Technology

Instant Karma: Employer Fires Tech Team with AI, Faces Backlash on LinkedIn While Seeking New Developers

LinkedIn's COO Reveals the AI Interview Question That Could Land You the Job in 2025

Features

- Make the piano key work using the keyboard as well as the mouse.
- A key should be highlighted when a green color mark presses it.
- A proper indication according to the keyboard should be mentioned on each key.
- No two keys must produce the same sound.

Let us move on to the actual coding of our project.

Complete Code for Piano in Python

Before moving ahead, talking about the directory, we made the main folder and inside that, there are our 2 python scripts, and one folder named **"assets**", inside that folder we have a folder named **"notes**" and in that folder, we have stored all our music files. We also have a file for fonts in the **"notes"** folder.

NOTE: Please name all the py files as it is named below

main.py

```
#basic library imports for Piano in Python
import pygame
import piano_lists as pl
from pygame import mixer

#this will initialize the pygame library
pygame.init()
pygame.mixer.set_num_channels(50)

#this is the path to fonts that we will use
#other variables for the sound and window
font = pygame.font.Font('assets/Terserah.ttf', 48)
#the below is the declaration for the different size of fonts that we are going to use
medium_font = pygame.font.Font('assets/Terserah.ttf', 28)
small_font = pygame.font.Font('assets/Terserah.ttf', 16)
real_small_font = pygame.font.Font('assets/Terserah.ttf', 10)
fps = 60

#enables the creation of a fresh Clock object that may be used to monitor time. Additionally, the clo
#Every frame should include one call to this function. It will calculate the number of milliseconds sin
timer = pygame.time.Clock()
WIDTH = 52 * 35
HEIGHT = 400
screen = pygame.display.set_mode([WIDTH, HEIGHT])
white_sounds = []
black_sounds = []
active_whites = []
active_blacks = []
left_oct = 4
right_oct = 5

left_hand = pl.left_hand
right_hand = pl.right_hand
piano_notes = pl.piano_notes
white_notes = pl.white_notes
black_notes = pl.black_notes
black_labels = pl.black_labels

#for loop is for accessing notes from the assets folder for all the white key on piano
for i in range(len(white_notes)):
    white_sounds.append(mixer.Sound(f'assets\\notes\\{white_notes[i]}.wav'))

#this for loop will access all the music files from the assets folder for black notes
for i in range(len(black_notes)):
    black_sounds.append(mixer.Sound(f'assets\\notes\\{black_notes[i]}.wav'))
#this is to give a title to our pygame window for gui Piano in Python project
pygame.display.set_caption("Python Piano - CopyAssignment")

#this function will draw the piano keys on the window of Piano in Python
def draw_piano(whites, blacks):
    white_rects = []
    for i in range(52):
        #we made use of rect() function in order to draw the key of the piano for white keys
        rect = pygame.draw.rect(screen, 'white', [i * 35, HEIGHT - 300, 35, 300], 0, 2)
        white_rects.append(rect)
        #same goes for black keys on paino
        pygame.draw.rect(screen, 'black', [i * 35, HEIGHT - 300, 35, 300], 2, 2)
        key_label = small_font.render(white_notes[i], True, 'black')
        screen.blit(key_label, (i * 35 + 3, HEIGHT - 20))
    skip_count = 0
    last_skip = 2
```

```
    skip_track = 2
    black_rects = []
    for i in range(36):
        #this is to draw the small black rectangles on the larger keys in GUI Piano in Python
        rect = pygame.draw.rect(screen, 'black', [23 + (i * 35) + (skip_count * 35), HEIGHT - 300, 24, 2
        for q in range(len(blacks)):
            #this conditional will keep thrack of the green marker that we want to show up on each key
            #whenever a user pesses the key of Piano App in Python, a green marker should show up
            if blacks[q][0] == i:
                if blacks[q][1] > 0:
                    pygame.draw.rect(screen, 'green', [23 + (i * 35) + (skip_count * 35), HEIGHT - 300, 24,
                    blacks[q][1] -= 1

        #this variable will handle all the labels that the keys will have in our project
        key_label = real_small_font.render(black_labels[i], True, 'white')
        screen.blit(key_label, (25 + (i * 35) + (skip_count * 35), HEIGHT - 120))
        black_rects.append(rect)
        skip_track += 1
        if last_skip == 2 and skip_track == 3:
            last_skip = 3
            skip_track = 0
            skip_count += 1
        elif last_skip == 3 and skip_track == 2:
            last_skip = 2
            skip_track = 0
            skip_count += 1
    #this will move the green block from white spaces to another white spaces
    for i in range(len(whites)):
        if whites[i][1] > 0:
            j = whites[i][0]
            pygame.draw.rect(screen, 'green', [j * 35, HEIGHT - 100, 35, 100], 2, 2)
            whites[i][1] -= 1

    return white_rects, black_rects, whites, blacks


    for i in range(36):
        #this is to draw the small black rectangles on the larger keys in Piano GUI in Python
        rect = pygame.draw.rect(screen, 'black', [23 + (i * 35) + (skip_count * 35), HEIGHT - 300, 24, 2
        for q in range(len(blacks)):
            #this conditional will keep thrack of the green marker that we want to show up on each key
            #whenever a user pesses the key of Piano GUI in Python, a green marker should show up
            if blacks[q][0] == i:
                if blacks[q][1] > 0:
                    pygame.draw.rect(screen, 'green', [23 + (i * 35) + (skip_count * 35), HEIGHT - 300, 24,
                    blacks[q][1] -= 1

        #this variable will handle all the labels that the keys will have in our project
        key_label = real_small_font.render(black_labels[i], True, 'white')
        screen.blit(key_label, (25 + (i * 35) + (skip_count * 35), HEIGHT - 120))
        black_rects.append(rect)
        skip_track += 1
        if last_skip == 2 and skip_track == 3:
            last_skip = 3
            skip_track = 0
            skip_count += 1
        elif last_skip == 3 and skip_track == 2:
            last_skip = 2
            skip_track = 0
            skip_count += 1
    #this will move the green block from white spaces to another white spaces
    for i in range(len(whites)):
        if whites[i][1] > 0:
            j = whites[i][0]
            pygame.draw.rect(screen, 'green', [j * 35, HEIGHT - 100, 35, 100], 2, 2)
            whites[i][1] -= 1

    return white_rects, black_rects, whites, blacks


def draw_hands(rightOct, leftOct, rightHand, leftHand):
# left hand side keys are handles by the below section of code
    pygame.draw.rect(screen, 'dark gray', [(leftOct * 245) - 175, HEIGHT - 60, 245, 30], 0, 4)
    pygame.draw.rect(screen, 'black', [(leftOct * 245) - 175, HEIGHT - 60, 245, 30], 4, 4)
    text = small_font.render(leftHand[0], True, 'white')
    screen.blit(text, ((leftOct * 245) - 165, HEIGHT - 55))
    text = small_font.render(leftHand[2], True, 'white')
    screen.blit(text, ((leftOct * 245) - 130, HEIGHT - 55))
    text = small_font.render(leftHand[4], True, 'white')
    screen.blit(text, ((leftOct * 245) - 95, HEIGHT - 55))
    text = small_font.render(leftHand[5], True, 'white')
    screen.blit(text, ((leftOct * 245) - 60, HEIGHT - 55))
    text = small_font.render(leftHand[7], True, 'white')
    screen.blit(text, ((leftOct * 245) - 25, HEIGHT - 55))
    text = small_font.render(leftHand[9], True, 'white')
    screen.blit(text, ((leftOct * 245) + 10, HEIGHT - 55))
    text = small_font.render(leftHand[11], True, 'white')
    screen.blit(text, ((leftOct * 245) + 45, HEIGHT - 55))
    text = small_font.render(leftHand[1], True, 'black')
    screen.blit(text, ((leftOct * 245) - 148, HEIGHT - 55))
    text = small_font.render(leftHand[3], True, 'black')
    screen.blit(text, ((leftOct * 245) - 113, HEIGHT - 55))
    text = small_font.render(leftHand[6], True, 'black')
    screen.blit(text, ((leftOct * 245) - 43, HEIGHT - 55))
    text = small_font.render(leftHand[8], True, 'black')
    screen.blit(text, ((leftOct * 245) - 8, HEIGHT - 55))
    text = small_font.render(leftHand[10], True, 'black')
    screen.blit(text, ((leftOct * 245) + 27, HEIGHT - 55))

```

```
# right hand side keys are handles by the below section of code
    pygame.draw.rect(screen, 'dark gray', [(rightOct * 245) - 175, HEIGHT - 60, 245, 30], 0, 4)
    pygame.draw.rect(screen, 'black', [(rightOct * 245) - 175, HEIGHT - 60, 245, 30], 4, 4)
    text = small_font.render(rightHand[0], True, 'white')
    screen.blit(text, ((rightOct * 245) - 165, HEIGHT - 55))
    text = small_font.render(rightHand[2], True, 'white')
    screen.blit(text, ((rightOct * 245) - 130, HEIGHT - 55))
    text = small_font.render(rightHand[4], True, 'white')
    screen.blit(text, ((rightOct * 245) - 95, HEIGHT - 55))
    text = small_font.render(rightHand[5], True, 'white')
    screen.blit(text, ((rightOct * 245) - 60, HEIGHT - 55))
    text = small_font.render(rightHand[7], True, 'white')
    screen.blit(text, ((rightOct * 245) - 25, HEIGHT - 55))
    text = small_font.render(rightHand[9], True, 'white')
    screen.blit(text, ((rightOct * 245) + 10, HEIGHT - 55))
    text = small_font.render(rightHand[11], True, 'white')
    screen.blit(text, ((rightOct * 245) + 45, HEIGHT - 55))
    text = small_font.render(rightHand[1], True, 'black')
    screen.blit(text, ((rightOct * 245) - 148, HEIGHT - 55))
    text = small_font.render(rightHand[3], True, 'black')
    screen.blit(text, ((rightOct * 245) - 113, HEIGHT - 55))
    text = small_font.render(rightHand[6], True, 'black')
    screen.blit(text, ((rightOct * 245) - 43, HEIGHT - 55))
    text = small_font.render(rightHand[8], True, 'black')
    screen.blit(text, ((rightOct * 245) - 8, HEIGHT - 55))
    text = small_font.render(rightHand[10], True, 'black')
    screen.blit(text, ((rightOct * 245) + 27, HEIGHT - 55))

#this will draw the upper section of Piano GUI In Python
def draw_title_bar():
    instruction_text = medium_font.render('Up/Down Arrows Change Left Hand', True, 'black')
    screen.blit(instruction_text, (WIDTH - 500, 10))
    instruction_text2 = medium_font.render('Left/Right Arrows Change Right Hand', True, 'black')
    screen.blit(instruction_text2, (WIDTH - 500, 50))
    title_text = font.render('CopyAssignment Paino!', True, 'white')
    screen.blit(title_text, (298, 18))
    title_text = font.render('CopyAssignment Paino!', True, 'black')
    screen.blit(title_text, (300, 20))


run = True
#while loop for all the keys
while run:
    left_dict = {'Z': f'C{left_oct}',
                 'S': f'C#{left_oct}',
                 'X': f'D{left_oct}',
                 'D': f'D#{left_oct}',
                 'C': f'E{left_oct}',
                 'V': f'F{left_oct}',
                 'G': f'F#{left_oct}',
                 'B': f'G{left_oct}',
                 'H': f'G#{left_oct}',
                 'N': f'A{left_oct}',
                 'J': f'A#{left_oct}',
                 'M': f'B{left_oct}'}

    right_dict = {'R': f'C{right_oct}',
                  '5': f'C#{right_oct}',
                  'T': f'D{right_oct}',
                  '6': f'D#{right_oct}',
                  'Y': f'E{right_oct}',
                  'U': f'F{right_oct}',
                  '8': f'F#{right_oct}',
                  'I': f'G{right_oct}',
                  '9': f'G#{right_oct}',
                  'O': f'A{right_oct}',
                  '0': f'A#{right_oct}',
                  'P': f'B{right_oct}'}
    timer.tick(fps)
    screen.fill('gray')
    white_keys, black_keys, active_whites, active_blacks = draw_piano(active_whites, active_blacks
    draw_hands(right_oct, left_oct, right_hand, left_hand)
    draw_title_bar()
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
        if event.type == pygame.MOUSEBUTTONDOWN:
            black_key = False
            for i in range(len(black_keys)):
                if black_keys[i].collidepoint(event.pos):
                    black_sounds[i].play(0, 1000)
                    black_key = True
                    active_blacks.append([i, 30])
            for i in range(len(white_keys)):
                if white_keys[i].collidepoint(event.pos) and not black_key:
                    white_sounds[i].play(0, 3000)
                    active_whites.append([i, 30])
        if event.type == pygame.TEXTINPUT:
            if event.text.upper() in left_dict:
                if left_dict[event.text.upper()][1] == '#':
                    index = black_labels.index(left_dict[event.text.upper()])
                    black_sounds[index].play(0, 1000)
                    active_blacks.append([index, 30])
                else:
                    index = white_notes.index(left_dict[event.text.upper()])
                    white_sounds[index].play(0, 1000)
                    active_whites.append([index, 30])
            if event.text.upper() in right_dict:
                if right_dict[event.text.upper()][1] == '#':
```

```
                index = black_labels.index(right_dict[event.text.upper()])
                black_sounds[index].play(0, 1000)
                active_blacks.append([index, 30])
            else:
                index = white_notes.index(right_dict[event.text.upper()])
                white_sounds[index].play(0, 1000)
                active_whites.append([index, 30])

    #this for loop is to handle the mouse click events in Piano in Python
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
        if event.type == pygame.MOUSEBUTTONDOWN:
            black_key = False
            for i in range(len(black_keys)):
                #The PyGame Rect class has documentation on the point-collision. In essence, you prov
                # If a point is located inside the boundaries of the rectangle, the function Rect. collidepoir
                if black_keys[i].collidepoint(event.pos):
                    black_sounds[i].play(0, 1000)
                    black_key = True
                    active_blacks.append([i, 30])
            for i in range(len(white_keys)):
                if white_keys[i].collidepoint(event.pos) and not black_key:
                    white_sounds[i].play(0, 3000)
                    active_whites.append([i, 30])
        if event.type == pygame.TEXTINPUT:
            if event.text.upper() in left_dict:
                if left_dict[event.text.upper()][1] == '#':
                    #The Python index() function aids in locating a certain element's or item's position insic
                    #It produces the list's supplied element's lowest possible index.
                    #A ValueError is returned if the requested item doesn't exist in the list.
                    index = black_labels.index(left_dict[event.text.upper()])
                    black_sounds[index].play(0, 1000)
                    active_blacks.append([index, 30])
                else:
                    index = white_notes.index(left_dict[event.text.upper()])
                    white_sounds[index].play(0, 1000)
                    active_whites.append([index, 30])
            if event.text.upper() in right_dict:
                if right_dict[event.text.upper()][1] == '#':
                    index = black_labels.index(right_dict[event.text.upper()])
                    black_sounds[index].play(0, 1000)
                    active_blacks.append([index, 30])
                else:
                    index = white_notes.index(right_dict[event.text.upper()])
                    white_sounds[index].play(0, 1000)
                    active_whites.append([index, 30])

        #this conditional block is to handle the arrow keys event for the working of Piano in Python
        #we used conditionals to keep track of the arrow keys press event
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_RIGHT:
                if right_oct < 8:
                    right_oct += 1
            if event.key == pygame.K_LEFT:
                if right_oct > 0:
                    right_oct -= 1
            if event.key == pygame.K_UP:
                if left_oct < 8:
                    left_oct += 1
            if event.key == pygame.K_DOWN:
                if left_oct > 0:
                    left_oct -= 1

    pygame.display.flip()
#this will quite the  window of the pygame
pygame.quit()
```

piano_lists.py

```
#this below is the list for left hand keys
left_hand = ['Z', 'S', 'X', 'D', 'C', 'V', 'G', 'B', 'H', 'N', 'J', 'M']
#list for right hand keys
right_hand = ['R', '5', 'T', '6', 'Y', 'U', '8', 'I', '9', 'O', '0', 'P']

#various notes of the piano is stored in the form of list, that will be used in main.py file
piano_notes = ['A0', 'A0#', 'B0', 'C1', 'C1#', 'D1', 'D1#', 'E1', 'F1', 'F1#', 'G1', 'G1#',
               'A1', 'A1#', 'B1', 'C2', 'C2#', 'D2', 'D2#', 'E2', 'F2', 'F2#', 'G2', 'G2#',
               'A2', 'A2#', 'B2', 'C3', 'C3#', 'D3', 'D3#', 'E3', 'F3', 'F3#', 'G3', 'G3#',
               'A3', 'A3#', 'B3', 'C4', 'C4#', 'D4', 'D4#', 'E4', 'F4', 'F4#', 'G4', 'G4#',
               'A4', 'A4#', 'B4', 'C5', 'C5#', 'D5', 'D5#', 'E5', 'F5', 'F5#', 'G5', 'G5#',
               'A5', 'A5#', 'B5', 'C6', 'C6#', 'D6', 'D6#', 'E6', 'F6', 'F6#', 'G6', 'G6#',
               'A6', 'A6#', 'B6', 'C7', 'C7#', 'D7', 'D7#', 'E7', 'F7', 'F7#', 'G7', 'G7#',
               'A7', 'A7#', 'B7', 'C8']

#this list is for the white keys
white  notes = ['A0', 'B0', 'C1', 'D1', 'E1', 'F1', 'G1',
               'A1', 'B1', 'C2', 'D2', 'E2', 'F2', 'G2',
               'A2', 'B2', 'C3', 'D3', 'E3', 'F3', 'G3',
               'A3', 'B3', 'C4', 'D4', 'E4', 'F4', 'G4',
               'A4', 'B4', 'C5', 'D5', 'E5', 'F5', 'G5',
               'A5', 'B5', 'C6', 'D6', 'E6', 'F6', 'G6',
               'A6', 'B6', 'C7', 'D7', 'E7', 'F7', 'G7',
               'A7', 'B7', 'C8']
#for black keys on Piano GUI in Python
```

```
black_notes = ['Bb0', 'Db1', 'Eb1', 'Gb1', 'Ab1',
               'Bb1', 'Db2', 'Eb2', 'Gb2', 'Ab2',
               'Bb2', 'Db3', 'Eb3', 'Gb3', 'Ab3',
               'Bb3', 'Db4', 'Eb4', 'Gb4', 'Ab4',
               'Bb4', 'Db5', 'Eb5', 'Gb5', 'Ab5',
               'Bb5', 'Db6', 'Eb6', 'Gb6', 'Ab6',
               'Bb6', 'Db7', 'Eb7', 'Gb7', 'Ab7',
               'Bb7']

#this list will handle all the names on each key of Piano GUI in Python
#the rednering of this list is done in main.py file
#this will be done using the for loop in our main file
black_labels = ['A#0', 'C#1', 'D#1', 'F#1', 'G#1',
                'A#1', 'C#2', 'D#2', 'F#2', 'G#2',
                'A#2', 'C#3', 'D#3', 'F#3', 'G#3',
                'A#3', 'C#4', 'D#4', 'F#4', 'G#4',
                'A#4', 'C#5', 'D#5', 'F#5', 'G#5',
                'A#5', 'C#6', 'D#6', 'F#6', 'G#6',
                'A#6', 'C#7', 'D#7', 'F#7', 'G#7',
```

Reference Material

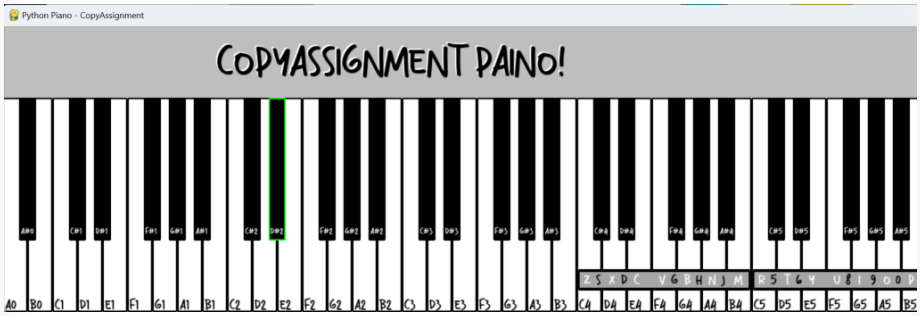
Here is the link to all the music files and fonts file

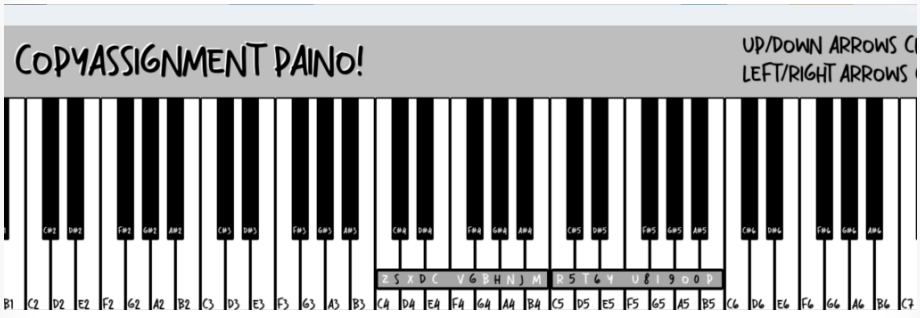
Link: Click here for reference material

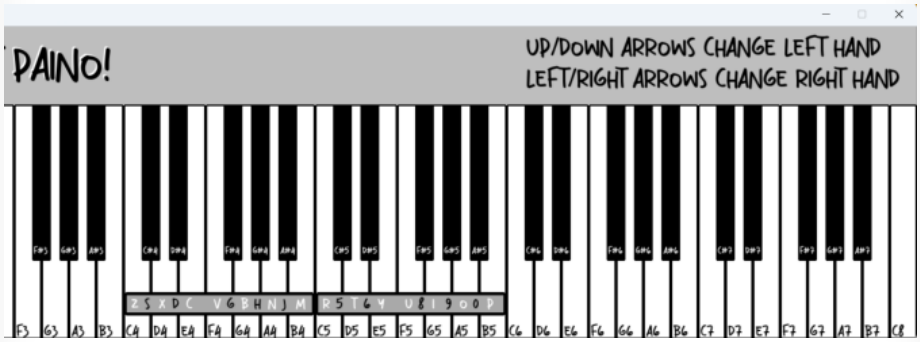
NOTE: Now if you want to add your own music notes and name them accordingly till the letter **Cc0 to Cc7, Dd0 to Dd7, Ee0 to Ee7, Ff0 to Ff7, Gg0 to Gg7,** and **C0 to C7, D0 to D7, E0 to E7, F0 to F7, G0 to G7**. Just a note that for any music file that you use, please name it as per the above format.

Output for Piano in Python

Image output:







Video output:

00:00 00:31

Summary

Here is the end of our article on **GUI Piano in Python**. All in all, we utilized a variety of libraries and learned about a variety of functions. We really think that this lesson will serve as an excellent learning resource for you and that it will help you strengthen your CV. We made use of the pygame library that helped us to take input from the user which is in the form of keyboard key press events. Apart from that, we made use of the pl library that helped us in the making of this Piano App in Python. We heartily thank you for visiting our website.

Best 100+ Python Projects with source code

Also Read:

Create your own ChatGPT with Python	SQLite | CRUD Operations in Python	Event Management System Project in Python
Ticket Booking and Management in Python	Hostel Management System Project in Python	Sales Management System Project in Python
Bank Management System Project in C++	Python Download File from URL | 4 Methods	Python Programming Examples | Fundamental Programs in Python
Spell Checker in Python	Portfolio Management System in Python	Stickman Game in Python
Contact Book project in Python	Loan Management System Project in Python	Cab Booking System in Python
Brick Breaker Game in Python	Tank game in Python	GUI Piano in Python
Ludo Game in Python	Rock Paper Scissors Game in Python	Snake and Ladder Game in Python
Puzzle Game in Python	Medical Store Management System Project in Python	Creating Dino Game in Python
Tic Tac Toe Game in Python	Test Typing Speed using Python App	Scientific Calculator in Python
GUI To-Do List App in Python Tkinter	Scientific Calculator in Python using Tkinter	GUI Chat Application in Python Tkinter

Author: Ayush Purawr

← Password and Notes Manager in Java

Chat App with Node.js and Socket.io →