



ساخت بازی 2048 با پایتون

Create 2048 game in python

Producer : Fatemeh Rahmani

Professor : Dr. Amir Seyed Danesh

Teaching Assistant : Mr. Ardalan Danesh

Faculty of Technology and Engineering - East of Guilan

University of Guilan

Project of Data Structures

1400-01

Contents Table of

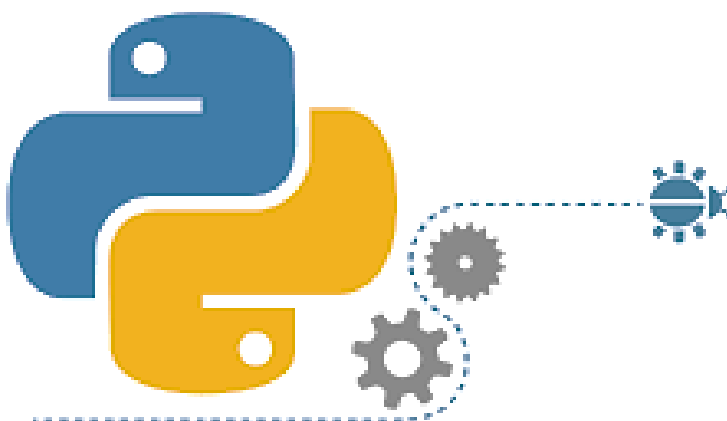
فهرست مطالب

Abstract.....	2	چکیده
Introduction & the project goal	3	مقدمه و هدف پروژه
Introduction to Python.....	4	آشنایی با Python
Libraries and usage their	7	Library ها و کاربردشان
Description of how the project works.....	9	شرح چگونگی عملکرد پروژه
Important points for using this program.....	31	نکات مهم برای بکار گیری این برنامه
Make exe file for game.....	32	تهیه فایل exe از بازی
The final code of game.....	34	کد نهایی بازی
Video link	49	لینک ویدئو ✓

چکیده

در این نوشتار قصد دارم به بررسی بازی 2048 در زبان پایتون که به عنوان پروژه پایانی درس ساختمان های داده برای این ترم در نظر گرفته شده است بپردازم .

گاهی اوقات نیازی نیست تا یک بازی حتما جلوه های بصری فوق العاده داشته باشد یا پر از گلوله و تفنگ باشد ؛ گاهی اوقات یک پازل و یک بازی معمایی خوب می تواند تا مدت ها سرگرم کننده باشد . 2048 یک پازل ساده و در عین حال جذاب است که هر چه بیشتر به انجام آن می پردازید بیشتر شیفته آن می شوید این بازی باعث بهبود تمرکز ، حافظه و هماهنگی بین دست و مغز نیز می شود. نمونه های بسیاری از این پازل برای تلفن های هوشمند عرضه شده است و ما این بار قصد داریم تا آن را خودمان پیاده سازی کنیم.




مقدمه و هدف ساخت بازی

بازی‌ها یا بهتر بگوییم پازل‌های بی‌شماری مانند سودوکو یا انواع و اقسام جدول وجود دارند که به عنوان سرگرمی اصلی بسیاری از انسان‌ها فارغ از هرگونه سن شناخته می‌شود. حالا فرض کنید بتوانید یکی از این پازل‌ها و بازی‌های فکری را به جای این که روی کاغذ و به وسیله خودکار انجام دهید آن را همیشه روی تلفن هوشمند خود داشته باشید و با لمس انگشتان دست به انجام آن بپردازید و یا حتی خودتان آن را بسازید! نمی‌دانم پیش از این تجربه بازی 2048 را داشته‌اید یا خیر ولی اگر این بازی را انجام نداده‌اید، حتماً برای یک بار هم که شده آن را امتحان کنید. این بازی برای اولین بار توسط توسعه دهنده وب ایتالیایی Gabriele Cirulli ساخته شد، 2048 یک پازل ساده و در عین حال جذاب است که هر چه بیشتر به انجام آن می‌پردازید بیشتر شیفته آن می‌شوید. نمونه‌های بسیاری از این پازل برای تلفن‌های هوشمند عرضه شده است اما یکی از موفق‌ترین و البته بهترین آن‌ها پازل ساخته استودیو Ketchapp است. شاید بسیاری از شما آیکون زرد رنگ این بازی را هر روز فشار می‌دهید تا کمی امتیاز خود در آن را بهبود ببخشید. کلیت پازل این است که باید در یک جدول 4 در 4 اعداد را به هم بچسبانید تا اعداد بزرگ‌تری درست کنید. همه این اعداد از 2 به توان 1 شروع می‌شوند و وقتی دو عدد 2 را با هم تلفیق می‌کنید عدد 4 ساخته می‌شود. نکته اینجا است که دیگر نمی‌توانید عدد 4 را با 2 تلفیق کنید و هر عدد با مشابه خود تلفیق می‌شود. شاید این نکته در ابتدای کار ساده به نظر برسد اما وقتی به اعداد بزرگ‌تری مانند 2048 می‌رسد آن‌جا است که چالش اصلی آغاز می‌شود. بسیاری از کاربران این بازی در شروع کار متدهای بسیاری را برای بازی انتخاب می‌کنند و نحوه‌های مختلفی را برای رسیدن به امتیاز بالاتر آزمایش می‌کنند اما آن چیزی که بیشتر از همه در کسب امتیاز بالا در این بازی موثر است فکر باز و صبر است. اگر این دو فاکتور را داشته باشید می‌توانید به مرور به امتیازهای بالا برسید. یکی از نکات مثبت این بازی سبک بودن آن است که برخلاف نمونه‌های مشابه خود، انجام این پازل را زجرآور نمی‌کند. مهم‌ترین شاخصه این بازی حس رقابتی آن است و این که کافی است تا در جمع دوستان خود امتیاز خود را بگویید و آن‌جا است که شیرینی این بازی دو چندان می‌شود. 2048 و پازل زیبا و اعتیادآورش را از دست ندهید. البته همیشه شارژر را هم به همراه داشته باشید زیرا وقتی در حال تلفیق اعداد با هم هستید متوجه گذر زمان نخواهید شد (۴)

-این پروژه فقط یک پروژه دانشجویی است و استفاده تجاری از آن مد نظر نمی‌باشد.

2048

در ادامه به بررسی و تحلیل پروژه بازی 2048 می‌پردازیم....



آشنایی با Python

امروزه تعداد زبان های برنامه نویسی بسیار زیاد است و هر کدام کاربردهای مختلفی دارند. هر کدام از این زبان ها مزایا و معایب خودشان را دارند. یکی از زبان های برنامه نویسی مطرح بین برنامه نویسان پایتون است که روز به روز به میزان محبوبیت آن اضافه می شود. از این زبان برنامه نویسی برای انجام کارهایی زیادی از جمله برنامه نویسی هوش مصنوعی، توسعه وب، ساخت اپلیکیشن های موبایل و دسکتاپ استفاده می شود.

پایتون یک زبان برنامه نویسی سطح بالا تفسیر شده برای برنامه نویسی عمومی است. این زبان دارای یک فلسفه طراحی است که بر خواندن کد، به خصوص با استفاده از فضای خالی مهم استوار است. **Python** دارای یک سیستم نوع پویا و مدیریت حافظه خودکار است و پارادایم های چندگانه برنامه نویسی را پشتیبانی می کند و مفسر پایتون برای بسیاری از سیستم عامل ها در دسترس است.

پایتون یک زبان اسکریپتی است که کدهای آن در پلتفرم های لینوکس، ویندوز، مکینتاش، سیستم عامل های موبایل و حتی پلی استیشن قابل اجراست و به دلیل قابلیت های فراوانی که دارد، به یکی از زبان های مورد علاقه ی برنامه نویسان وب تبدیل شده و شرکت های بزرگی مثل گوگل، یاهو، اینستاگرام، ناسا، یوتیوب و... در سطح بالایی در حال استفاده از آن هستند.

مزایای پایتون

- حضور ماژول های شخص ثالث (PyPI) Python
- شامل چندین ماژول شخص ثالث است که باعث می شود Python بتواند با بسیاری از زبان ها و سیستم عامل های دیگر ارتباط برقرار کند.
- کتابخانه های پشتیبانی گسترده
- پایتون کتابخانه استاندارد بزرگی را ارائه می دهد که شامل موضوعات مختلف مانند پروتکل اینترنت ، عملیات رشته ، ابزارها و سرویس های وب و رابط های سیستم عامل است. بسیاری از کارهای برنامه نویسی پر کاربرد قبلاً در کتابخانه استاندارد نگاشته شده اند که باعث می شود طول کد به طور قابل توجهی کاهش داده شود.



- منبع باز
زبان پایتون تحت مجوز OSI تأیید شده است که استفاده و توزیع آن را آزاد می کند ، از جمله برای اهداف تجاری.
علاوه بر این ، توسعه آن توسط جامعه ای انجام می شود که از طریق میزبانی کنفرانس ها، برای کد آن همکاری می کنند و ماژول های بی شماری را برای توسعه آن فراهم می کنند.
- یادگیری سریع و آسان:
پایگاه گسترده کاربران و توسعه دهندگان فعال باعث شده است تا یک بانک منابع اینترنتی غنی برای ترغیب توسعه و ادامه پذیرش زبان ایجاد شود.
- ساختار داده های کاربر پسند :
پایتون دارای ساختار داخلی داده ها و فرهنگ نامه ها است که می تواند برای ساخت سریع داده های زمان اجرا سریع استفاده شود.
- بهره وری و سرعت
پایتون دارای طراحی شی گرا تمیز است ، قابلیت های کنترل پیشرفته یک فرایند را فراهم می کند ، و توانایی های ادغام و پردازش متن دارد ، که همه اینها به افزایش سرعت و بهره وری آن کمک می کند. پایتون گزینه ای مناسب برای ساخت برنامه های پیچیده دارای چند پروتکل تحت شبکه محسوب می شود.





معایب پایتون

- سرعت:
پایتون کندتر از C یا ++C است. پایتون یک زبان سطح بالا است، برخلاف C یا ++C به سخت افزار نزدیک نیست.
- توسعه موبایل:
پایتون یک زبان خیلی خوب برای توسعه موبایل نیست. این یک زبان ضعیف برای محاسبات موبایل است. به همین دلیل است که برنامه های اندکی در تلفن های همراه مانند Carbonnelle در آن ساخته شده اند.
- مصرف حافظه:
پایتون برای کارهای فشرده حافظه گزینه مناسبی نیست. به دلیل انعطاف پذیری انواع داده ها، مصرف میزان حافظه پایتون نیز زیاد است.
- دسترسی به پایگاه داده:
پایتون با دسترسی به بانک اطلاعات محدودیت هایی دارد. در مقایسه با فن آوری های رایج مانند JDBC و ODBC، لایه دسترسی به پایگاه داده Python کمی توسعه نیافته و بدوی است.
- خطاهای زمان اجرا:
برنامه نویسان پایتون در زمینه طراحی زبان چندین موضوع را ذکر کردند. از آنجا که این زبان به صورت پویا تایپ می شود، به آزمایش بیشتری نیاز دارد و دارای خطاهایی است که فقط در زمان اجرا نشان می دهد.



Library ها و کاربردها

در نوشتن این برنامه از Library هایی با کارایی های متفاوت استفاده شده که در اینجا به بررسی آنها می پردازیم:

Library	کاربرد ماژول ها
tkinter	<p>Tkinter ماژول داخلی پایتون است که برای ایجاد برنامه های GUI استفاده می شود. کار با Tkinter بسیار ساده است. این ماژول، جزئی از کتابخانه ی استاندارد پایتون است و نیازی به نصب جداگانه ندارد زیرا به همراه خود پایتون نصب می شود. از این رو، Tkinter یکی از پرکاربردترین ماژول ها برای ایجاد برنامه های GUI در پایتون است و من در این پروژه کلیت اجرای بازی را با آن پیاده سازی کرده ام.</p> <p>* رابط کاربری گرافیکی (Graphical User Interface) که به اختصار با عنوان GUI شناخته می شود، نوعی رابط کاربری است که به کاربران امکان می دهد از طریق شاخص های تصویری و با استفاده از مواردی همچون آیکون ها، منوها، پنجره ها و ... با کامپیوتر ارتباط برقرار کنند.</p> <p>GUI برخلاف رابط خط فرمان (Command Line Interface) است که کاربران از طریق صفحه کلید و تایپ دستورات، با کامپیوتر ارتباط برقرار می کنند.</p>
pygame	<p>Pygame ماژولی از ابزارهای زبان پایتون است که برای توسعه بازی ویدئویی به کار می روند و شامل گرافیک رایانه ای و کتابخانه های صدا است که قابل استفاده در برنامه نویسی به</p>



	<p>زبان پایتون هستند که من در این پروژه از آن برای پخش موسیقی در حین بازی استفاده کرده ام چرا که دارای مزیت های بیشتری نسبت به دیگر ماژول های مربوط به پخش صوت است .</p>
random	<p>ماژول random در پایتون یکی از ماژول های پر کاربردی است که می توان از آن در جهت تولید اعداد تصادفی، انتخاب کارکتر های تصادفی و... استفاده نمود. این ماژول بیشتر در برنامه های مربوط به تولید رشته و عدد تصادفی مانند ساخت پسورد ها مورد استفاده قرار میگیرد که البته من از آن برای ساخت اعداد تصادفی با احتمال های نابرابر در خانه های تصادفی در ابتدای بازی استفاده کرده ام.</p>
OS	<p>OS یکی از ماژول های کاربردی در پایتون می باشد. این ماژول امکان استفاده از برخی قابلیت های وابسته به سیستم عامل را فراهم می آورد؛ مانند گرفتن مسیر دایرکتوری برنامه که البته من از آن برای ذخیره امتیازات کاربر استفاده کرده ام.</p>



شرح چگونگی عملکرد بازی 2048

این بازی با استفاده از ماژول های متفاوت و با مرتبط کردن آن ها به وسیله توابع وساختار های مختلف کار می کند.

درواقع کد این بازی به دو بخش تقسیم خواهد شد. در قسمت اول تمامی عملکردهای مربوط به طراحی و ویژگی های اصلی بازی مانند ابزارها، اندازه ویندوز، رنگ های پس زمینه و شبکه شامل بازی و فونت ها و رنگ های حروف و اعداد تنظیم می شود و در بخش دوم، توابع مربوط به عملکرد بازی، محاسبه امتیازات و پایان بازی ایجاد می شود. البته این برنامه علاوه بر فیچر های خواسته شده دارای فیچر های اضافه ای هم هست که در ادامه از آنها مطلع و به بررسی آنها می پردازیم.

قبل از شروع باید ابتدا ماژول های لازم را نصب کنیم .

طراحی بازی

اولین کاری که باید انجام دهیم این است که ماژول های مورد نیاز برای ساخت بازی را با کمک دستور import به محیط برنامه اضافه کنیم . سپس، برخی از متغیرها مقداردهی اولیه می شوند: متغیرهای مربوط به رنگ شبکه، اندازه قلم و رنگ فونت در اینجا مقدار دهی می شوند.

```
from tkinter import messagebox
import tkinter as tk
from tkinter import *
from pygame import mixer
import random
import os
import pygame

# Design numbers & words
GRID_COLOR = "#c2b3a9"
EMPTY_CELL_COLOR = "#a89283"
SCORE_LABEL_FONT = ("ROSEMARY", 22, "bold")
TITLE_FONT = ("ROSEMARY", 29)
DESCRIPTION_FONT = ("Century", 14)
SCORE_FONT = ("ROSEMARY", 22)
CELL_NUMBER_COLORS = {2: "#ffffff", 4: "#ffffff", 8: "#ffffff"}
CELL_NUMBER_FONTS = ("ROSEMARY", 22)
CELL_COLORS = {2: "#b2cd44",
```



```
4: "#78ba3f",
8: "#3ab073",
16: "#2da8e1",
32: "#3566b0",
64: "#514597",
128: "#7d4294",
256: "#c3308b",
512: "#dc2e4e",
1024: "#e04932",
2048: "#e86325",
4096: "#ffd900"}
```

برای برنامه های مربوط به طراحی و عملکرد بازی یک کلاس به نام Game_2048 ایجاد می کنیم . در این کلاس تمام توابع لازم برای ایجاد این بازی تعریف می شوند . این کلاس یک frame از ماژول Tkinter را به عنوان ورودی خود می گیرد و سپس تابع _init_ تعریف می شود. در این تابع اندازه و عنوان پنجره اصلی بازی تنظیم می شود و همچنین، اندازه شبکه (بازی کلاسیک 4x4 است و از ما هم همین اندازه خواسته شده اما می توان آن را به هر اندازه دیگری تغییر داد) و بیشترین امتیاز برای برنده شدن (2048 بالاترین مقدار اصلی این بازی است و به همین دلیل هم به این اسم نام گذاری شده و بعد از این امتیاز هم کاربر می تواند به بازی خود ادامه دهد البته که هستن افرادی که تا امتیازات بالاتر هم پیش رفتند.) ، دکمه هایی که به منظور کار های مختلف (راهنما ، معرفی سازنده و برنامه ، پخش و توقف موزیک) تعریف شده است. در این تابع دکمه های مربوط به حرکت در آوردن کاشی های بازی به این صورت تعریف شده است : (D برای حرکت کاشی ها به سمت راست ، A برای حرکت کاشی ها به سمت چپ ، S برای حرکت کاشی ها به سمت پایین و W برای حرکت کاشی ها به سمت بالا) و کلید های کاربردی جهت ایجاد بازی جدید و خروج در نظر گرفته شده که به این صورت عمل می کنند: (R برای ایجاد بازی جدید و Q جهت خروج از بازی) و در ادامه توابع مورد نیاز برای اجرای بازی فراخوانی شده اند.

آخرین تابع mainloop() از Tkinter وارد شده است. این تابع برای عملکرد بازی نقش حیاتی دارد زیرا منتظر event هایی هست که ما برای برنامه در نظر گرفته ایم ، به عنوان مثال، حرکت به چپ راست و... و رابط کاربری گرافیکی را مطابق



منطق ایجاد شده در طول برنامه به روز می کند و درواقع اگر از این عملکرد استفاده نشود، هیچ چیزی نمایش داده نمی شود و بازی ما اجرا نمی شود.

```
class Game_2048(tk.Frame):
    def __init__(user):
        # Set main window
        tk.Frame.__init__(user)
        user.grid()
        user.master.title("2048 Game :)")
        user.main_grid = tk.Frame(user, bg=GRID_COLOR, bd=3,
width=100, height=100)
        user.main_grid.grid(padx=(14, 0), pady=(175, 0))

        # 2048Game functions and parameters
        # Top value to play the game
        user.top_value = 2048

        # Grid size
        user.grid_size = 4

        # Main window position
        user.sw = user.master.winfo_screenwidth()
        user.sh = user.master.winfo_screenheight()

        # Game initialization
        user.make_GUI()
        user.start_game()

        # Buttons
        user.guide_button()
        user.about_us_button()
        user.play_music_button()
        user.mute_music_button()

        # Defining button to start new game
        user.master.bind('<r>', user.restart_message)

        # Defining button to close game
        user.master.bind('<q>', user.quit_message)
```



```
# Defining buttons to play
user.master.bind('<a>', user.left)
user.master.bind('<s>', user.down)
user.master.bind('<d>', user.right)
user.master.bind('<w>', user.up)

user.mainloop()
```

اولین تابع ایجاد شده تابع `make_GUI` است. در داخل این تابع اندازه شبکه کاشی ها، رنگ و اندازه سلول ها، فونت شماره ها، اندازه و رنگ و موقعیت صفحه بازی بر روی صفحه نمایش کامپیوتر (موقعیت نسبت به هر سیستمی تنظیم می شود به صورت خودکار) تنظیم شده است و از دستور `resizable` به منظور تغییر ناپذیر بودن پنجره توسط کاربر استفاده شده است. در این قسمت عنوان پنجره ی مورد نظر را هم تعیین کرده ایم ، توضیح کوتاه و مختصر درمورد هدف بازی را نیز در قالب یک لیبل بر روی صفحه تنظیم کرده ایم و همچنین امتیاز فعلی بازیکن و رکورد او (بهترین امتیاز، این مقدار را در یک فایل `.ini` به نام `bestscore` ذخیره میکنیم تا همیشه برنامه به آن دسترسی داشته باشد و هر بار که بازیکن از آن امتیاز بیشتری دریافت می کند آپدیت شود و رکورد جدید در صفح نمایش داده شود) تعریف شده است و در این قسمت به منظور تنظیم گرافیک بازی از متغیر هایی که در اول برنامه تعریف کرده بودیم استفاده کرده ایم.

```
# Functions to set game desing
def make_GUI(user):
    user.cells = []
    # Creating the grid
    for i in range(user.grid_size):
        row = []
        for j in range(user.grid_size):
            cell_frame = tk.Frame(user.main_grid,
bg=EMPTY_CELL_COLOR, width=80, height=80)
            cell_frame.grid(row=i, column=j, padx=5,
pady=5)
            cell_number = tk.Label(user.main_grid,
bg=EMPTY_CELL_COLOR)
            cell_number.grid(row=i, column=j)
```



```
        cell_data = {'frame': cell_frame, "number":
cell_number}
        row.append(cell_data)
        user.cells.append(row)

# Game position in screen
w = user.grid_size*99
h = (user.grid_size+2)*93
x = (user.sw - w)/2
y = (user.sh - h)/2
user.master.geometry('%dx%d+%d+%d' % (w, h, x, y))
user.master.resizable(False,False)
user.configure(bg='#f0eddf')

# Game title
act_frame = tk.Frame(user)
act_frame.place(relx=0.16, rely=0.1, anchor="center",)
tk.Label(
    act_frame,
    text="2048",
    font=TITLE_FONT,
    bg='#ecc400',
    fg='white',
    width=4,
    height=2,
).grid(row=0)

# Description
description_frame = tk.Frame(user)
description_frame.place(relx=0.52, rely=0.285,
anchor="center")
tk.Label(
    description_frame,
    text = "join the numbers & get to the 2048 tile!!",
    font = DESCRIPTION_FONT,
    fg='#787167',
    bg='#f0eddf',
).grid(row=0)
```



```
# Game current score and best score
user.score = 0
user.bstScore = 0
# Save best score
if os.path.exists("bestscore.ini"):
    with open("bestscore.ini", "r") as f:
        user.bstScore = int(f.read())

# Score
score_frame = tk.Frame(user)
score_frame.place(relx=0.5, y=46, anchor="center")
tk.Label(
    score_frame,
    text="Score",
    font=SCORE_LABEL_FONT,
).grid(row=0)
user.score_label = tk.Label(score_frame,
text=user.score, font=SCORE_FONT)
user.score_label.grid(row=1)

# Record
record_frame = tk.Frame(user)
record_frame.place(relx=0.84, y=46, anchor="center")
tk.Label(
    record_frame,
    text="Record",
    font=SCORE_LABEL_FONT,
).grid(row=0)
user.record_label = tk.Label(record_frame,
text=user.bstScore, font=SCORE_FONT)
user.record_label.grid(row=2)
```

دو تابع بعدی برای شروع بازی است. تابع `new_game` تابع `make_GUI` را برای مقداردهی اولیه یک بازی جدید در `frame` فراخوانی می کند و تابع `start_game` دو عدد 2 و 4 را با احتمال های به ترتیب 0.8 و 0.2 را در موقعیت های



تصادفی روی شبکه کاشی ها که به عنوان یک ماتریس در نظر گرفته شده است اضافه می کند که این کار توسط ماژول random و برای افزودن اعداد صحیح از randint استفاده کرده ایم و در این قسمت هم از مقادیری که برای پارامترها در اول بازی نوشته ایم به منظور تنظیم گرافیک برنامه استفاده می کنیم و در ادامه با استفاده از ماژول pygame موسیقی را از پوشه musics لود و پخش می کنیم که به دلیل آنکه چندین موسیقی همزمان باهم در حال پخش هستند می توانیم از channel استفاده کنیم تا به کانال های مختلف تقسیم شود و در هنگام پخش یک صوت ، صوت دیگر تا انتهای آن صوت متوقف نشود و همراه باهم پخش شوند اگر می خواستیم از قطعه کد

```
mixer.init()
```

```
mixer.music.load('musics/musics/Approaching Dusk.mp3')
```

```
mixer.music.play(-1)
```

استفاده کنیم قطعا به مشکل بر می خوردیم.

```
# Function for restart game
def new_game(user):
    user.make_GUI()
    user.start_game()

# Create new game
def start_game(user):
    # Place the first random number in a random position
    user.matrix = [[0]*user.grid_size for _ in
range(user.grid_size)]
    row = random.randint(0, user.grid_size-1)
    col = random.randint(0, user.grid_size-1)
    user.matrix[row][col] = 2
    user.cells[row][col]["frame"].configure(bg=CELL_COLORS[
2])
    user.cells[row][col]["number"].configure(
        bg=CELL_COLORS[2],
        fg=CELL_NUMBER_COLORS[2],
        font=CELL_NUMBER_FONTS,
```




```
        text="2"
    )

    # Place the second random number in an empty random
    position
    while(user.matrix[row][col] != 0):
        row = random.randint(0, user.grid_size-1)
        col = random.randint(0, user.grid_size-1)
    user.matrix[row][col] = 2
    user.cells[row][col]["frame"].configure(bg=CELL_COLORS[
2])
    user.cells[row][col]["number"].configure(
        bg=CELL_COLORS[2],
        fg=CELL_NUMBER_COLORS[2],
        font=CELL_NUMBER_FONTS,
        text="2"
    )
    user.score = 0

    # Play music in background of
    game
    pygame.mixer.Channel(0).play(pygame.mixer.Sound('musics
/Approaching Dusk.mp3'), loops= -1)
```

در این قسمت توابع `about_us` و `guide_message` این وظیفه را به عهده دارند تا پیام هایی به منظور راهنمای بازی و معرفی را نمایش دهند که این عمل با کلیک کردن بازیکن بر روی دکمه های `About us` و `How to play` انجام می شود . در ادامه باید این دکمه هارا به توسط توابع `about_us_button` و `guide_button` تعریف کنیم و در این توابع محل قرار گیری دکمه ها برروی صفحه اصلی و همچنین موقعیت و اندازه صفحه نمایش پیام ها تنظیم گردد.

```
# Show message for how to play
def guide_message(user):
    messagebox.showinfo("Game Guide","Hi, thanks for choose
this game:)\n\
    -Press 'A', 'S', 'D', 'W' on keyboard to move left,down,right
&
    up.\n\
```



```
-Press'Q' on keybord to exit the game and'R'to start
new game. "
)

# Button for help player how can play whith this game
def guide_button(user):
    g_button = Button(user,
                        text="How to play",
                        bg="#a09999",
                        fg="white",
                        font=("Century", 8, 'bold'),
                        command=user.guide_message,
                        width=10
                        )
    g_button.place(relx=0.84, rely=0.21,
anchor="center")

# show message for introduction creator
def about_us(user):
    messagebox.showinfo('About Me', 'Hello, Im Fatemeh
Rahmani a third semester student of computer engineering at the
Faculty of Technology and Engineering - East of Guilan.\n\n\
This is a game for Data Structures project.')

# Button for introduction
def about_us_button(user):
    about_button = Button(user,
                           text="About us",
                           bg="#cca993",
                           fg="white",
                           font=("Century", 8, 'bold'),
                           command=user.about_us,
                           width=10)
    about_button.place(relx=0.50, rely=0.21,
anchor="center")
```



ممکن هست در هنگام بازی بخواهید که موسیقی زمینه بازی قطع شود اما صدای حرکت کاشی ها را همچنان بشنوید در نتیجه باید برای این مساله هم فکری کرد.

در این بخش توابع `play_music` و `mute_music` وظیفه پخش مجدد و توقف موسیقی را بر عهده دارند (از قطعه کد `loops = -1` به این منظور استفاده شده که موسیقی ما همانند یک لوپ هر بار پس از خاتمه دوباره از اول شروع شود و تا زمانی که بازی تمام نشده ای فرایند تکرار می شود) که برای این عمل نیاز هست تا دو تابع دیگر نیز تعریف کنیم که دو دکمه را بر روی صفحه به وجود آورند که بازیکن با کلیک بر روی آنها موسیقی را پخش یا قطع کند در نتیجه دو تابع `play_music_button` و `mute_music_button` این وظیفه را به عهده دارند همچنین برای نمایش بهتر و زیبا تر از قطعه کدی استفاده کردیم که تصویر دو بلند گو را در پس زمینه این دکمه ها لود کند تا بازی جلوه زیبا تری داشته باشد. البته از کلید واژه `global` به این منظور استفاده شده تا اگر به هر دلیلی این تصویر شناسایی نشد توسط این کد به خوبی تعریف و لود شود.

```
# Play music in background of game
def play_music(user):
    pygame.mixer.Channel(0).play(pygame.mixer.Sound('musics/Approaching Dusk.mp3'), loops= -1)

# Mute background sound
def mute_music(user):
    pygame.mixer.Channel(0).stop()

# Button for play music in background of game
def play_music_button(user):
    global play_photo
    play_photo = PhotoImage(file="images/play.png")
    playBtn = Button(user, image=play_photo,
command=user.play_music, bg='#f0eddf', bd=0)
    playBtn.image = play_photo
    playBtn.place(relx=0.11, rely=0.23, anchor="center")

# Button for mute music in background of game
def mute_music_button(user):
    global mute_photo
    mute_photo = PhotoImage(file="images/mute.png")
```



```
muteBtn = Button(user, image=mute_photo,
command=user.mute_music, bg='#f0eddf', bd=0)
muteBtn.image = mute_photo
muteBtn.place(relx=0.21, rely=0.23, anchor="center")
```

در این قسمت دو تابع quit_message و restart_message این وظیفه را به عهده دارند که از بازیکن بپرسد که آیا مایل به ترک و یا ایجاد یک بازی جدید هست یا نه و در هر چهار صورت ممکنه واکنش معقولی به خواسته بازیکن نشان دهد و البته در ایجاد یک بازی جدید برنامه چندین بار از بازیکن میپرسد که آیا از بازی به وجود آمده راضی هست یا نه و می خواهد یک بازی دیگر به وجود بیاید یا همین بازی ای که به وجود آمده خوب است و اگر بازیکن بزدن دکمه NO برنامه مطلع سازد که از بازی به وجود آمده راضی است ، این پیغام از بین می رود و او می تواند به بازی خود بپردازد.

```
# For ask player wants quite game or continue
def quit_message(user, event):
    q_win=Tk()
    frame1 = Frame(q_win, highlightbackground="green",
highlightcolor="green",highlightthickness=1, bd=0)
    frame1.pack()
    q_win.overridereirect(1)
    q_win.geometry("200x70+650+400")
    label = Label(frame1, text="Are you sure you want to
quit game?")
    label.pack()
    yes_button = Button(frame1, text="Yes", bg="light
blue", fg="dark blue",command=quit, width=10)
    yes_button.pack(padx=10, pady=10 , side=LEFT)
    no_button = Button(frame1, text="No", bg="light blue",
fg="dark blue",command=q_win.destroy, width=10)
    no_button.pack(padx=10, pady=10, side=LEFT)
    q_win.mainloop()

# For ask player wants restart game or no
def restart_message(user, event):
    r_win=Tk()
```



```
frame1 = Frame(r_win, highlightbackground="green",
highlightcolor="green",highlightthickness=1, bd=0)
frame1.pack()
r_win.overridedirect(1)
r_win.geometry("240x70+650+400")
label = Label(frame1, text="Are you sure you want to
start a new game?")
label.pack()

yes_button = Button(frame1, text="Yes", bg="light
blue", fg="dark blue", command=user.new_game, width=10)
yes_button.pack(padx=13, pady=10 , side=LEFT)
no_button = Button(frame1, text="No", bg="light blue",
fg="dark blue", command=r_win.destroy, width=10)
no_button.pack(padx=25, pady=10, side=LEFT)
r_win.mainloop()
```

عملکرد بازی

توابع زیر قسمتی هستند که عملکرد بازی را تعریف می کنند. برای ایجاد منطق بازی، ماتریسی به اندازه شبکه کاشی هایی که در رابط کاربری گرافیکی بود تعریف می کنیم. سپس چهار تابع را ایجاد میکنیم، یکی برای هر کدام از حرکت های ممکن در بازی (چپ، راست، بالا و پایین). باتوجه به کلید فشار داده شده چند مرحله باید انجام شود که توابع زیر مراحل هستند که با توجه به این حرکات مورد استفاده قرار می گیرند و در کنارهم باعث به وجود آمدن حرکات و پیشبرد آنها می شود .

اولین تابع تعریف شده stack است. این تابع مقادیر موجود در ماتریس را در ستون سمت چپ قرار می دهد بنابراین هر عدد در ماتریس به اولین سلول خالی در ستون سمت چپ در همان ردیف منتقل می شود و ماتریس را اپدیت می کند و بر می گرداند.

```
# Stack of number
def stack(user):
```



```
new_matrix = [[0] * user.grid_size for _ in
range(user.grid_size)]
for row in range(user.grid_size):
    fill_position = 0
    for col in range(user.grid_size):
        if user.matrix[row][col] != 0:
            new_matrix[row][fill_position] =
user.matrix[row][col]
            fill_position += 1
    user.matrix = new_matrix
```

تابع Combine دو عدد را در ماتریس ترکیب می کند. برای ترکیب دو عدد، آن دو باید مقدار یکسانی داشته باشند و باهم یکی باشند. به عنوان مثال، برای ترکیب، باید دو تا عدد 2 با هم باشد تا ترکیب شوند و یک عدد 4 تشکیل شود و... ولی اگر یک عدد 2 و دیگری به عنوان مثال 4 باشد باهم ترکیب نمی شوند. پس از ترکیب اعداد حالا نوبت آن است که امتیاز فعلی و در صورت ممکن رکورد بازیکن آپدیت شود و مقادیر جدید به نمایش گذاشته شود و البته بیشترین امتیاز بازیکن در یک فایل ذخیره شود تا همیشه امتیاز بازیکن نصب به آن سنجیده شود. همچنین از قطعه کدی به منظور افکت صوتی هنگام ترکیب اعداد استفاده شده تا جلوه زیبا تری داشته باشد.

```
# Combine equal numbers
def combine(user):
    for row in range(user.grid_size):
        for col in range(user.grid_size-1):
            if (user.matrix[row][col] != 0) and
(user.matrix[row][col] == user.matrix[row][col + 1]):
                user.matrix[row][col] *= 2
                user.matrix[row][col + 1] = 0
                user.score += user.matrix[row][col]
                if user.score > user.bstScore:
                    user.bstScore = user.score
                    with open("bestscore.ini", "w") as f:
                        f.write(str(user.bstScore))

# Sound effect for combine
tiles
```



```
pygame.mixer.Channel(1).play(pygame.mixer.Sound('musics/combine.mp3'))
```

برای اعمال Combine و stack ، بدون توجه به جهت (چپ، راست، بالا یا پایین)، سلول‌های روی ماتریس باید موقعیت خود را تغییر دهند، بستگی به حرکت دارد. پس برای انجام این کار، دو تابع دیگر باید ایجاد شود.

تابع reverse ماتریس را در جهت مخالف می‌خواند. به این معنا که در ماتریس جدید، اولین ستون آخرین ستون ماتریس اصلی خواهد بود. که این با استفاده از append محیا می‌شود و در آخر هم ماتریس آپدیت می‌شود.

```
def reverse(user):  
    new_matrix = []  
    for row in range(user.grid_size):  
        new_matrix.append([])  
        for col in range(user.grid_size):  
            new_matrix[row].append(user.matrix[row][(user.grid_size-1) - col])  
    user.matrix = new_matrix
```

تابع transpose ماتریس را جابجا می‌کند به این معنا که در ماتریس جدید، ستون اول اولین سطر از ماتریس اصلی خواهد بود که در واقع با قطعه کد `new_matrix[row][col] = user.matrix[col][row]` جای این دو عوض می‌شود و در انتها هم ماتریس آپدیت می‌شود.

```
# Transpose function  
def transpose(user):  
    new_matrix = [[0]*user.grid_size for _ in range(user.grid_size)]  
    for row in range(user.grid_size):  
        for col in range(user.grid_size):  
            new_matrix[row][col] = user.matrix[col][row]
```



```
user.matrix = new_matrix
```

توابع بعدی به روز رسانی فریم بازی و امتیاز بعد از انجام هرحرکت توسط بازیکن می پردازند. در ابتدا، تابع `add_new_tile` در ماتریس، در یک خانه خالی تصادفی، عدد 2 یا 4 را اضافه می کند که احتمال اضافه شدن 2 در آن خیلی بیشتر از 4 است. برای ایجاد یک خانه تصادفی مهم ترین عمل را قطعه کد

```
row = random.randint(0, user.grid_size-1)
```

```
col = random.randint(0, user.grid_size-1)
```

انجام می دهد که وظیفه پیدا کردن خانه ای تصادفی به عهده تابع `random` است و با استفاده از قطعه کد

```
random.choice([2, 4], p=[0.8, 0.2])
```

و متد `choice` که در آن استفاده شده تابع از بین عدد 2 و 4 با احتمال به ترتیب 0.8 و 0.2 یکی از این دو را در خانه تصادفی ای که در بالا انتخاب شد قرار می دهد.

```
# Add new number in a random position
def add_new_tile(user):
    if any(0 in row for row in user.matrix):
        row = random.randint(0, user.grid_size-1)
        col = random.randint(0, user.grid_size-1)
        while(user.matrix[row][col] != 0):
            row = random.randint(0, user.grid_size-1)
            col = random.randint(0, user.grid_size-1)
        # Display numbers 2 and 4 with probabilities of 0.8
        and 0.2
        user.matrix[row][col] = random.choice([2, 2, 2, 2,
        2, 2, 2, 2, 4, 4])
```




تابع `update_GUI` اساساً امتیاز فعلی بازی و رنگ کاشی ها را در شبکه به روز می کند البته با استفاده از متد `configure` و `bg` که مخفف `background` است گرافیکی که در ابتدای بازی برای هر خانه و عددی در نظر داشتیم اعمال می شود. اگر امتیاز فعلی بازیکن که کسب کرده بیشتر از رکورد قبلی او باشد، این راهم آپدیت می کند و به بازیکن نمایش می دهد.

```
# Functions to update de GUI
def update_GUI(user):
    cell_text_color = 0
    cell_cell_color = 0
    for row in range(user.grid_size):
        for col in range(user.grid_size):
            cell_value = user.matrix[row][col]
            if cell_value == 0:
                user.cells[row][col]["frame"].configure(bg=EMPTY_CELL_COLOR)
                user.cells[row][col]["number"].configure(bg=EMPTY_CELL_COLOR, text="")
            else:
                if cell_value >= 8:
                    cell_text_color = 8
                else:
                    cell_text_color = cell_value
                if cell_value >= 4096:
                    cell_cell_color = 4096
                else:
                    cell_cell_color = cell_value

                user.cells[row][col]["frame"].configure(bg=CELL_COLORS[cell_cell_color])
                user.cells[row][col]["number"].configure(
                    bg=CELL_COLORS[cell_cell_color],
                    fg=CELL_NUMBER_COLORS[cell_text_color],
                    font=CELL_NUMBER_FONTS,
                    text=str(cell_value))
            user.score_label.configure(text=user.score)
            user.record_label.configure(text=user.bstScore)
            user.update_idletasks()
```



سه تابع زیر برای بررسی زمان پایان بازی است، یا به دلیل رسیدن به هدف یا به دلیل اینکه حرکت دیگری در شبکه وجود ندارد. تابع any_move بررسی می‌کند که آیا ترکیب احتمالی (همان حرکت)، افقی یا عمودی وجود دارد یا خیر. اگر حرکت دیگری وجود داشته باشد، true برمی‌گرداند ولی در غیر این صورت false برمی‌گرداند.

```
# Check for possibles moves
def any_move(user):
    for i in range(user.grid_size):
        for j in range(user.grid_size-1):
            if user.matrix[i][j] == user.matrix[i][j + 1]
or \
            user.matrix[j][i] == user.matrix[j + 1][i]:
                return True
    return False
```

تابع game_over دو چیز را بررسی می‌کند: آیا به کاشی 2048 رسیده است یا خیر و اینکه آیا حرکت احتمالی در ماتریس وجود دارد یا خیر. اگر به 2048 رسید، یک پاپ آپ با پیام رسیدن به هدف و شما برنده شده اید ظاهر می‌شود، همچنین top_value به بعدی یعنی (4096) به روز می‌شود و می‌توانید ادامه بازی را انتخاب کنید که آیا می‌خواهید ادامه بدهید یا نه. اگر هیچ حرکتی در ماتریس وجود نداشته باشد، یک پنجره با پیام game over ظاهر می‌شود که یعنی بازی تمام شده و همچنین در انتها پس از باخت بازیکن آخرین امتیاز او و همچنین رکوردش در ترمینال چاپ می‌شود.

```
# Check for game over
def game_over(user):
    # Check if to value is reached
    if any(user.top_value in row for row in user.matrix):
        text = f"You did {user.top_value}!!"
        user.popup(text, text + "Continue?")
        user.top_value = user.top_value*2
    # Check if there are no more moves in the grid say game
over
```



```
elif not any(0 in row for row in user.matrix) and not
user.any_move():
    user.popup("Game Over:", "\t\tGame Over!!\t\t\nbut
you can try it again later :)")
    pygame.mixer.Channel(2).play(pygame.mixer.Sound('mu
sics/game_over.mp3'))
    print("Record:", str(user.bstScore))
    print("Score:", str(user.score))
```

تابع popup پیغامی را برای بازیکن نمایش می دهد که البته اطلاعات ورودی را از تابع game_over می گیرد و برای دو صورت مشاهده 2048 و یا باخت سوال می پرسد که اگر جواب بله بود پیغام حذف شده و به ادامه بازی و یا شروع بازی جدید می پردازد ولی اگر جواب خیر بود صفحه بازی به کلی بسته می شود و از محیط برنامه خارج می شود..

```
# Create popup for game over and win
def popup(user, win_title, win_message):
    popup_win = tk.Toplevel()
    popup_win.wm_title(win_title)
    w = 280
    h = 90
    x = (user.sw - w)/2
    y = (user.sh - h)/2
    popup_win.geometry('%dx%d+%d+%d' % (w, h, x, y))
    l = tk.Label(popup_win, text=win_message)
    l.pack(padx=11, pady=3)
    ok_btn = tk.Button(popup_win, text="Ok", bg="light
blue", fg="dark blue", command=popup_win.destroy, width=10)
    no_btn = tk.Button(popup_win, text="NO", bg="light
blue", fg="dark blue", command=popup_win.quit, width=10)
    popup_win.master.resizable(False, False)
    ok_btn.pack(padx=26, pady=5, side=LEFT)
    no_btn.pack(padx=30, pady=5, side=LEFT)
```

توابع مربوط به حرکات



عملکردهای زیر زمانی که هر یک از حروف A,S,D,W روی صفحه کلید فشار داده می شود، فراخوانی می شوند. تابع left را فراخوانی می کند. این حرکت به دلیل نحوه تعریف توابع قبلی (stack.Combine) می توان گفت ساده ترین حرکت است چرا که منطق اصلی به شرح زیر است (این برای سه حرکت دیگر با مقداری تغییر قبل و بعد صدق می کند): ابتدا پشته شکل میگیرد، سپس در صورت موجود بودن عددی ترکیب می شود و سپس یک پشته دیگر (اگر ترکیبی ساخته شده بود، برای انباشته شدن) . سپس، رابط کاربری گرافیکی به روز می شود و بررسی می شود که آیا بازی تمام شده است یا نه و همچنین با استفاده از mixer.music افکت صوتی ای که برای حرکت به سمت چپ در نظر گرفته ایم لود و سپس پخش می شود و با استفاده از قطعه کد user.after(500, mixer.music.stop) صوت پس از 5 ثانیه متوقف می شود.

```
mixer.init()
# Stacking for move left
def left(user, event):
    user.stack()
    user.combine()
    user.stack()
    user.add_new_tile()
    user.update_GUI()
    user.game_over()
# Sound effect for move tiles
mixer.music.load('musics/left_right.mpeg')
mixer.music.play(-1)
user.after(500, mixer.music.stop)
```

فشاردن کلید D تابع right را فرا می خواند. قبل از منطق اصلی، ما باید جهت ماتریس را تغییر دهیم. بنابراین، ابتدا ماتریس معکوس می شود، سپس منطق اصلی اتفاق می افتد (stack ، Combine ، stack) و سپس ماتریس دوباره معکوس می شود تا ماتریس اصلی به دست آید. سپس، رابط کاربری گرافیکی به روز می شود و بررسی می شود که آیا بازی تمام شده است یا نه و همچنین با استفاده از mixer.music افکت صوتی ای که برای حرکت به سمت راست در نظر گرفته ایم لود و سپس پخش می شود و با استفاده از قطعه کد user.after(500, mixer.music.stop) صوت پس از 5 ثانیه متوقف می شود.



```
# Stacking for move right
def right(user, event):
    user.reverse()
    user.stack()
    user.combine()
    user.stack()
    user.reverse()
    user.add_new_tile()
    user.update_GUI()
    user.game_over()
# Sound effect for move tiles
mixer.music.load('musics/left_right.mpeg')
mixer.music.play(-1)
user.after(500, mixer.music.stop)
```

حرکت بعدی حرکت به سمت بالا است و همانطور که می توانید حدس بزنید، فشردن کلید W تابع up را فراخوانی می کند. مشابه حرکت right، قبل از منطق اصلی باید تغییری در ماتریس انجام شود تا بتوان آن را اعمال کرد. در این مورد، ماتریس باید جابجا شود. بنابراین، ماتریس جابجا می شود، سپس منطق اصلی اعمال می شود (stack، Combine، stack) و سپس ماتریس دوباره جابجا می شود. در نهایت، برابر با حرکات قبلی، توابع روز رسانی رابط کاربری گرافیکی و بررسی اینکه آیا بازی تمام شده است یا نه فراخوانی می شوند و همچنین با استفاده از mixer.music افکت صوتی ای که برای حرکت به سمت بالا در نظر گرفته ایم لود و سپس پخش می شود و با استفاده از قطعه کد user.after(500, mixer.music.stop) صوت پس از 5 ثانیه متوقف می شود.

```
# Stacking for move up
def up(user, event):
    user.transpose()
    user.stack()
    user.combine()
    user.stack()
    user.transpose()
    user.add_new_tile()
    user.update_GUI()
    user.game_over()
```



```
# Sound effect for move tiles
mixer.music.load('musics/up_down.mpeg')
mixer.music.play(-1)
user.after(500, mixer.music.stop)
```

آخرین تابع برای حرکت به سمت پایین است. این عمل کمی مشکل است زیرا برای اعمال منطق اصلی نیاز به تغییر بیشتر قبل و بعد دارد. برای این حرکت، ماتریس جابجا و معکوس می شود، سپس منطق اصلی. برای به دست آوردن ماتریس اصلی باید تبدیل ها را برگردانیم، بنابراین ماتریس معکوس شده و دوباره به آن ترتیب جابجا می شود. مانند حرکات قبلی، توابع به روز رسانی رابط کاربری گرافیکی و بررسی اینکه آیا به هدف رسیده است (یا هیچ حرکتی وجود ندارد) فراخوانی می شود و همچنین با استفاده از mixer.music افکت صوتی ای که برای حرکت به سمت پایین در نظر گرفته ایم لود و سپس پخش می شود و با استفاده از قطعه کد user.after(500, mixer.music.stop) صوت پس از 5 ثانیه متوقف می شود.

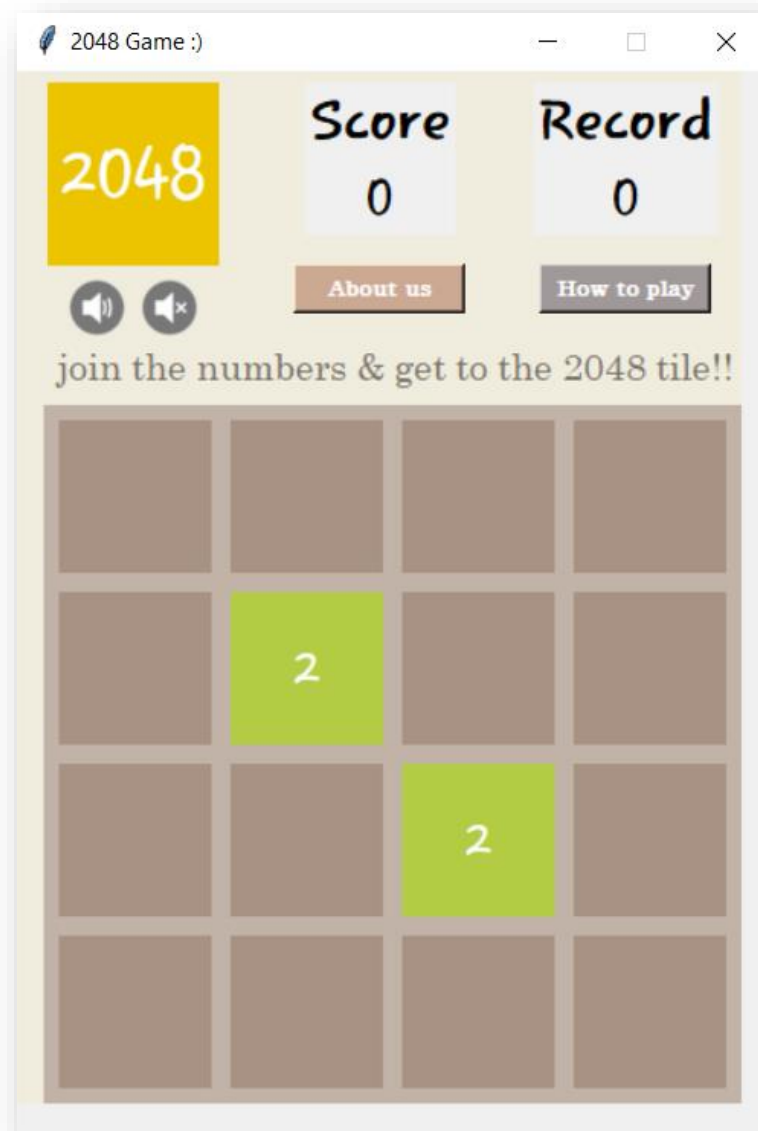
```
# Stacking for move down
def down(user, event):
    user.transpose()
    user.reverse()
    user.stack()
    user.combine()
    user.stack()
    user.reverse()
    user.transpose()
    user.add_new_tile()
    user.update_GUI()
    user.game_over()
    # Sound effect for move tiles
    mixer.music.load('musics/up_down.mpeg')
    mixer.music.play(-1)
    user.after(500, mixer.music.stop)
    user.game_over()
```

در نهایت در اسکریپت، خارج از کلاس Game_2048 یک تابع (main) تعریف می کنیم که کلاسی که ایجاد کردیم را فراخوانی کند.



```
if __name__ == "__main__":  
    Game_2048()
```

برای اجرای بازی اسکریپت را اجرا می کنیم و بازی نمایش داده می شود و حالا امیدوارم که از بازی لذت ببرید!





نکات مهم برای بکارگیری این برنامه

نکاتی که برای استفاده از این برنامه لازم هست که رعایت شود این هست که ابتدا پایتون را در سیستم خود نصب کنید (ترجیحا پایتون 3.10) ، ماژول هایی که به طور پیشفرض بر روی سیستم نصب نیستند را از طریق ترمینال و با کمک دستور pip install نصب کنید ، اطمینان حاصل کنید که زبان کیبورد شما انگلیسی است چرا که در غیر این صورت قادر به حرکت دادن کاشی ها نخواهید بود و از unmute بودن سیستم صوتی خودتان اطمینان حاصل کنید تا لذت این بازی دوچندان شود.

بعد از انجام کارهای گفته شده بازی را اجرا و لذت ببرید .

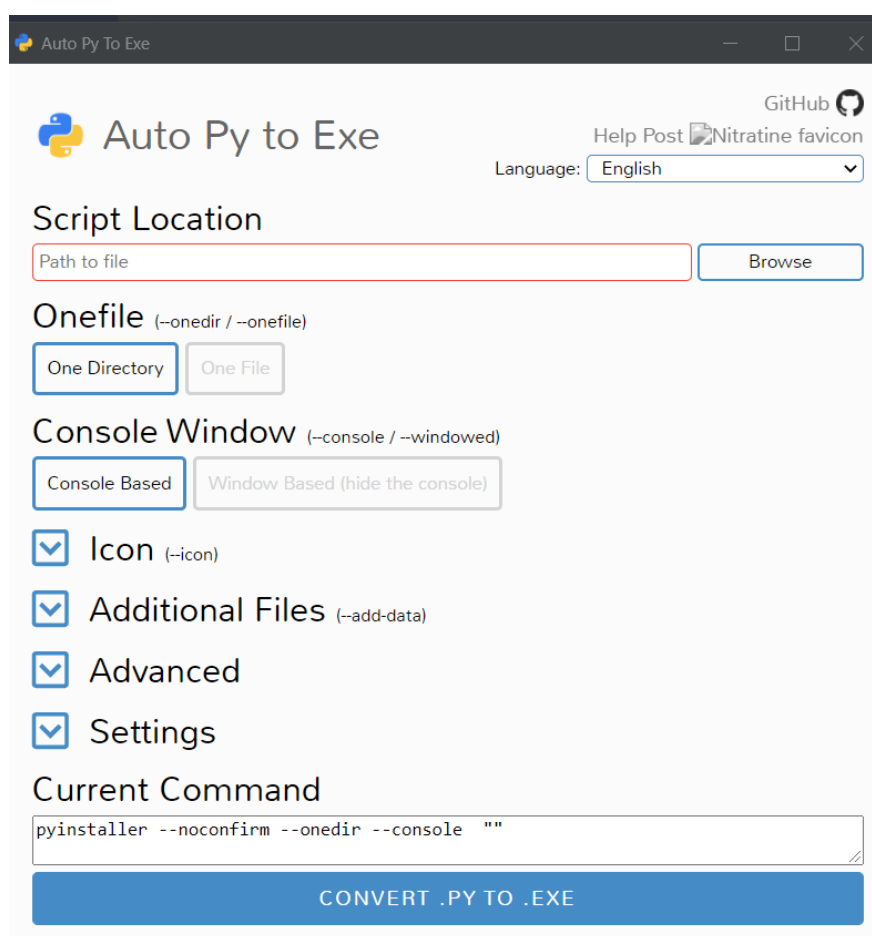
✱امیدوارم که بتونید به کاشی 2048 برسید. 😊

تهیه فایل exe از بازی

شاید خیلی از اوقات این اتفاق براتون رخ داده باشد که می خواهید برنامه ای را که خودتون نوشتید در اختیار کسی قرار بدین که در سیستم خود آن زبان را نصب نکرده و یا به هر دلیلی با برنامه نویسی آشنا نیست در نتیجه این ایده به فکرتون میرسه که یک فایل exe از فایلتون تهیه کنید تا به راحتی آن را در اختیار دیگران قراربدید.

از آنجایی که من در محیط visual studio کد این برنامه را نوشتم تهیه فایل exe طبیعتا با استفاده از pyinstaller محیا نبود حتی بعد از نصب دستی آن در cmd و به ارور بر می خوردم در نتیجه تصمیم گرفتم از راهی به غیر از این راه استفاده کنم.

به این منظور ابتدا با استفاده از `pip install auto-py-to-exe` این convertor را در بخش cmd سیستم نصب کردم و سپس با استفاده از کد `auto-py-to-exe` در cmd برنامه را فراخواندم. برنامه ی مورد نظر



The screenshot shows the 'Auto Py to Exe' application window. It has a title bar with the application name and standard window controls. The interface includes a GitHub logo and a 'Help Post' link. A language dropdown menu is set to 'English'. Under 'Script Location', there is a text input field for the file path and a 'Browse' button. The 'Onefile' section has two buttons: 'One Directory' (selected) and 'One File'. The 'Console Window' section has two buttons: 'Console Based' (selected) and 'Window Based (hide the console)'. There are four checked checkboxes: 'Icon', 'Additional Files', 'Advanced', and 'Settings'. At the bottom, the 'Current Command' field contains the command `pyinstaller --noconfirm --onedir --console ""`. A large blue button labeled 'CONVERT .PY TO .EXE' is at the bottom.



بود که تبدیل فایل py به exe. در آن بسیار آسان است و پس از طی مراحل آپلود فایل سورس کد و تصویر آیکون و... خروجی بازی ما با آیکون زیر ایجاد شد و در نتیجه از این پس می توان تنها با در اختیار گذاشتن همین اپلیکیشن و بدون نصب زبان پایتون از آن استفاده کرد.



کد نهایی بازی 2048

```
from tkinter import messagebox
import tkinter as tk
from tkinter import *
from pygame import mixer
import random
import os
import pygame

# Design numbers & words
GRID_COLOR = "#c2b3a9"
EMPTY_CELL_COLOR = "#a89283"
SCORE_LABEL_FONT = ("ROSEMARY", 22, "bold")
TITLE_FONT = ("ROSEMARY", 29)
DESCRIPTION_FONT = ("Century", 14)
SCORE_FONT = ("ROSEMARY", 22)
CELL_NUMBER_COLORS = {2: "#ffffff", 4: "#ffffff", 8: "#ffffff"}
CELL_NUMBER_FONTS = ("ROSEMARY", 22)
CELL_COLORS = {2: "#b2cd44",
                4: "#78ba3f",
                8: "#3ab073",
                16: "#2da8e1",
                32: "#3566b0",
                64: "#514597",
                128: "#7d4294",
                256: "#c3308b",
                512: "#dc2e4e",
                1024: "#e04932",
                2048: "#e86325",
                4096: "#ffd900"}

class Game_2048(tk.Frame):
    def __init__(user):
        # Set main window
        tk.Frame.__init__(user)
        user.grid()
        user.master.title("2048 Game :)")
```



```
user.main_grid = tk.Frame(user, bg=GRID_COLOR, bd=3,
width=100, height=100)
user.main_grid.grid(padx=(14, 0), pady=(175, 0))

# 2048Game functions and parameters
# Top value to play the game
user.top_value = 2048

# Grid size
user.grid_size = 4

# Main window position
user.sw = user.master.winfo_screenwidth()
user.sh = user.master.winfo_screenheight()

# Game initialization
user.make_GUI()
user.start_game()

# Buttons
user.guide_button()
user.about_us_button()
user.play_music_button()
user.mute_music_button()

# Defining button to start new game
user.master.bind('<r>', user.restart_message)

# Defining button to close game
user.master.bind('<q>', user.quit_message)

# Defining buttons to play
user.master.bind('<a>', user.left)
user.master.bind('<s>', user.down)
user.master.bind('<d>', user.right)
user.master.bind('<w>', user.up)

user.mainloop()
```



```
# Functions to set game desing
def make_GUI(user):
    user.cells = []
    # Creating the grid
    for i in range(user.grid_size):
        row = []
        for j in range(user.grid_size):
            cell_frame = tk.Frame(user.main_grid,
bg=EMPTY_CELL_COLOR, width=80, height=80)
            cell_frame.grid(row=i, column=j, padx=5,
pady=5)
            cell_number = tk.Label(user.main_grid,
bg=EMPTY_CELL_COLOR)
            cell_number.grid(row=i, column=j)
            cell_data = {'frame': cell_frame, "number":
cell_number}
            row.append(cell_data)
        user.cells.append(row)

    # Game position in screen
    w = user.grid_size*99
    h = (user.grid_size+2)*93
    x = (user.sw - w)/2
    y = (user.sh - h)/2
    user.master.geometry('%dx%d+%d+%d' % (w, h, x, y))
    user.master.resizable(False,False)
    user.configure(bg='#f0eddf')

    # Game title
    act_frame = tk.Frame(user)
    act_frame.place(relx=0.16, rely=0.1, anchor="center",)
    tk.Label(
        act_frame,
        text="2048",
        font=TITLE_FONT,
        bg='#ecc400',
        fg='white',
        width=4,
        height=2,
```



```
        ).grid(row=0)

    # Description
    description_frame = tk.Frame(user)
    description_frame.place(relx=0.52, rely=0.285,
anchor="center")
    tk.Label(
        description_frame,
        text = "join the numbers & get to the 2048 tile!!",
        font = DESCRIPTION_FONT,
        fg='#787167',
        bg='#f0eddf',
    ).grid(row=0)

    # Game current score and best score
    user.score = 0
    user.bstScore = 0
    # Save best score
    if os.path.exists("bestscore.ini"):
        with open("bestscore.ini", "r") as f:
            user.bstScore = int(f.read())

    # Score
    score_frame = tk.Frame(user)
    score_frame.place(relx=0.5, y=46, anchor="center")
    tk.Label(
        score_frame,
        text="Score",
        font=SCORE_LABEL_FONT,
    ).grid(row=0)
    user.score_label = tk.Label(score_frame,
text=user.score, font=SCORE_FONT)
    user.score_label.grid(row=1)

    # Record
    record_frame = tk.Frame(user)
    record_frame.place(relx=0.84, y=46, anchor="center")
    tk.Label(
```



```
        record_frame,
        text="Record",
        font=SCORE_LABEL_FONT,
    ).grid(row=0)
    user.record_label = tk.Label(record_frame,
text=user.bstScore, font=SCORE_FONT)
    user.record_label.grid(row=2)

    # Function for restart game
    def new_game(user):
        user.make_GUI()
        user.start_game()

    # Create new game
    def start_game(user):
        # Place the first random number in a random position
        user.matrix = [[0]*user.grid_size for _ in
range(user.grid_size)]
        row = random.randint(0, user.grid_size-1)
        col = random.randint(0, user.grid_size-1)
        user.matrix[row][col] = 2
        user.cells[row][col]["frame"].configure(bg=CELL_COLORS[
2])
        user.cells[row][col]["number"].configure(
            bg=CELL_COLORS[2],
            fg=CELL_NUMBER_COLORS[2],
            font=CELL_NUMBER_FONTS,
            text="2"
        )

        # Place the second random number in an empty random
position
        while(user.matrix[row][col] != 0):
            row = random.randint(0, user.grid_size-1)
            col = random.randint(0, user.grid_size-1)
            user.matrix[row][col] = 2
            user.cells[row][col]["frame"].configure(bg=CELL_COLORS[
2])
            user.cells[row][col]["number"].configure(
```



```
        bg=CELL_COLORS[2],
        fg=CELL_NUMBER_COLORS[2],
        font=CELL_NUMBER_FONTS,
        text="2"
    )
    user.score = 0

    # Play music in background of
game
    pygame.mixer.Channel(0).play(pygame.mixer.Sound('musics
/Approaching Dusk.mp3'), loops= -1)

    # Show message for how to play
    def guide_message(user):
        messagebox.showinfo("Game Guide","Hi, thanks for choose
this game:)\n\
        -Press 'A','S','D','W' on keyboard to move left,down,right
&
        up.\n\
        -Press 'Q' on keyboard to exit the game and 'R' to start
new
        game. "
    )

    # Button for help player how can play with this game
    def guide_button(user):
        g_button = Button(user,
            text="How to play",
            bg="#a09999",
            fg="white",
            font=("Century", 8, 'bold'),
            command=user.guide_message,
            width=10
        )
        g_button.place(relx=0.84, rely=0.21,
anchor="center")

    # show message for introduction creator
    def about_us(user):
```




```
        messagebox.showinfo('About Me', 'Hello, Im Fatemeh
Rahmani a third semester student of computer engineering at the
Faculty of Technology and Engineering - East of Guilan.\n\n\
This is a game for Data Structures project.')

# Button for introduction
def about_us_button(user):
    about_button = Button(user,
                           text="About us",
                           bg="#cca993",
                           fg="white",
                           font=("Century", 8, 'bold'),
                           command=user.about_us,
                           width=10)
    about_button.place(relx=0.50, rely=0.21,
anchor="center")

# Play music in background of game
def play_music(user):
    pygame.mixer.Channel(0).play(pygame.mixer.Sound('musics
/Approaching Dusk.mp3'), loops= -1)

# Mute background sound
def mute_music(user):
    pygame.mixer.Channel(0).stop()

# Button for play music in background of game
def play_music_button(user):
    global play_photo
    play_photo = PhotoImage(file="images/play.png")
    playBtn = Button(user, image=play_photo,
command=user.play_music, bg='#f0eddf', bd=0)
    playBtn.image = play_photo
    playBtn.place(relx=0.11, rely=0.23, anchor="center")

# Button for mute music in background of game
def mute_music_button(user):
    global mute_photo
```



```
    mute_photo = PhotoImage(file="images/mute.png")
    muteBtn = Button(user, image=mute_photo,
command=user.mute_music, bg='#f0eddf', bd=0)
    muteBtn.image = mute_photo
    muteBtn.place(relx=0.21, rely=0.23, anchor="center")

# For ask player wants quite game or continue
def quit_message(user, event):
    q_win=Tk()
    frame1 = Frame(q_win, highlightbackground="green",
highlightcolor="green",highlightthickness=1, bd=0)
    frame1.pack()
    q_win.overridedirect(1)
    q_win.geometry("200x70+650+400")
    label = Label(frame1, text="Are you sure you want to
quit game?")
    label.pack()
    yes_button = Button(frame1, text="Yes", bg="light
blue", fg="dark blue",command=quit, width=10)
    yes_button.pack(padx=10, pady=10 , side=LEFT)
    no_button = Button(frame1, text="No", bg="light blue",
fg="dark blue",command=q_win.destroy, width=10)
    no_button.pack(padx=10, pady=10, side=LEFT)
    q_win.mainloop()

# For ask player wants restart game or no
def restart_message(user, event):
    r_win=Tk()
    frame1 = Frame(r_win, highlightbackground="green",
highlightcolor="green",highlightthickness=1, bd=0)
    frame1.pack()
    r_win.overridedirect(1)
    r_win.geometry("240x70+650+400")
    label = Label(frame1, text="Are you sure you want to
start a new game?")
    label.pack()
```



```
yes_button = Button(frame1, text="Yes", bg="light
blue", fg="dark blue", command=user.new_game, width=10)
yes_button.pack(padx=13, pady=10 , side=LEFT)
no_button = Button(frame1, text="No", bg="light blue",
fg="dark blue", command=r_win.destroy, width=10)
no_button.pack(padx=25, pady=10, side=LEFT)
r_win.mainloop()

# Stack of number
def stack(user):
    new_matrix = [[0] * user.grid_size for _ in
range(user.grid_size)]
    for row in range(user.grid_size):
        fill_position = 0
        for col in range(user.grid_size):
            if user.matrix[row][col] != 0:
                new_matrix[row][fill_position] =
user.matrix[row][col]
                fill_position += 1
    user.matrix = new_matrix

# Combine equal numbers
def combine(user):
    for row in range(user.grid_size):
        for col in range(user.grid_size-1):
            if (user.matrix[row][col] != 0) and
(user.matrix[row][col] == user.matrix[row][col + 1]):
                user.matrix[row][col] *= 2
                user.matrix[row][col + 1] = 0
                user.score += user.matrix[row][col]
            if user.score > user.bstScore:
                user.bstScore = user.score
                with open("bestscore.ini", "w") as f:
                    f.write(str(user.bstScore))

# Sound effect for combine
tiles
```



```
pygame.mixer.Channel(1).play(pygame.mixer.Sound('musics/combine.mp3'))

# Reverse function
def reverse(user):
    new_matrix = []
    for row in range(user.grid_size):
        new_matrix.append([])
        for col in range(user.grid_size):
            new_matrix[row].append(user.matrix[row][(user.grid_size-1) - col])
    user.matrix = new_matrix

# Transpose function
def transpose(user):
    new_matrix = [[0]*user.grid_size for _ in range(user.grid_size)]
    for row in range(user.grid_size):
        for col in range(user.grid_size):
            new_matrix[row][col] = user.matrix[col][row]
    user.matrix = new_matrix

# Add new number in a random position
def add_new_tile(user):
    if any(0 in row for row in user.matrix):
        row = random.randint(0, user.grid_size-1)
        col = random.randint(0, user.grid_size-1)
        while(user.matrix[row][col] != 0):
            row = random.randint(0, user.grid_size-1)
            col = random.randint(0, user.grid_size-1)
        # Display numbers 2 and 4 with probabilities of 0.8 and 0.2
        user.matrix[row][col] = random.choice([2, 2, 2, 2, 2, 2, 2, 2, 4, 4])

# Functions to update de GUI
```



```
def update_GUI(user):
    cell_text_color = 0
    cell_cell_color = 0
    for row in range(user.grid_size):
        for col in range(user.grid_size):
            cell_value = user.matrix[row][col]
            if cell_value == 0:
                user.cells[row][col]["frame"].configure(bg=
EMPTY_CELL_COLOR)
                user.cells[row][col]["number"].configure(bg
=EMPTY_CELL_COLOR, text="")
            else:
                if cell_value >= 8:
                    cell_text_color = 8
                else:
                    cell_text_color = cell_value
                if cell_value >= 4096:
                    cell_cell_color = 4096
                else:
                    cell_cell_color = cell_value

                user.cells[row][col]["frame"].configure(bg=
CELL_COLORS[cell_cell_color])
                user.cells[row][col]["number"].configure(
                    bg=CELL_COLORS[cell_cell_color],
                    fg=CELL_NUMBER_COLORS[cell_text_color],
                    font=CELL_NUMBER_FONTS,
                    text=str(cell_value))
            user.score_label.configure(text=user.score)
            user.record_label.configure(text=user.bstScore)
            user.update_idletasks()

# Check for possibles moves
def any_move(user):
    for i in range(user.grid_size):
        for j in range(user.grid_size-1):
            if user.matrix[i][j] == user.matrix[i][j + 1]
or \
```



```
        user.matrix[j][i] == user.matrix[j + 1][i]:
            return True
    return False

# Check for game over
def game_over(user):
    # Check if to value is reached
    if any(user.top_value in row for row in user.matrix):
        text = f"You did {user.top_value}!!"
        user.popup(text, text + "Continue?")
        user.top_value = user.top_value*2
    # Check if there are no more moves in the grid say game
over
    elif not any(0 in row for row in user.matrix) and not
user.any_move():
        user.popup("Game Over:", "\t\tGame Over!!\t\t\nbut
you can try it again later :)")
        pygame.mixer.Channel(2).play(pygame.mixer.Sound('mu
sics/game_over.mp3'))
        print("Record:", str(user.bstScore))
        print("Score:", str(user.score))

# Create popup for game over and win
def popup(user, win_title, win_message):
    popup_win = tk.Toplevel()
    popup_win.wm_title(win_title)
    w = 280
    h = 90
    x = (user.sw - w)/2
    y = (user.sh - h)/2
    popup_win.geometry('%dx%d+%d+%d' % (w, h, x, y))
    l = tk.Label(popup_win, text=win_message)
    l.pack(padx=11, pady=3)
    ok_btn = tk.Button(popup_win, text="Ok", bg="light
blue", fg="dark blue", command=popup_win.destroy, width=10)
    no_btn = tk.Button(popup_win, text="NO", bg="light
blue", fg="dark blue", command=popup_win.quit, width=10)
    popup_win.master.resizable(False, False)
```



```
ok_btn.pack(padx=26, pady=5, side=LEFT)
no_btn.pack(padx=30, pady=5, side=LEFT)

mixer.init()
# Stacking for move left
def left(user, event):
    user.stack()
    user.combine()
    user.stack()
    user.add_new_tile()
    user.update_GUI()
    user.game_over()
    # Sound effect for move tiles
    mixer.music.load('musics/left_right.mpeg')
    mixer.music.play(-1)
    user.after(500, mixer.music.stop)

# Stacking for move right
def right(user, event):
    user.reverse()
    user.stack()
    user.combine()
    user.stack()
    user.reverse()
    user.add_new_tile()
    user.update_GUI()
    user.game_over()
    # Sound effect for move tiles
    mixer.music.load('musics/left_right.mpeg')
    mixer.music.play(-1)
    user.after(500, mixer.music.stop)


# Stacking for move up
def up(user, event):
    user.transpose()
    user.stack()
    user.combine()
    user.stack()
```



```
user.transpose()
user.add_new_tile()
user.update_GUI()
user.game_over()
# Sound effect for move tiles
mixer.music.load('musics/up_down.mpeg')
mixer.music.play(-1)
user.after(500, mixer.music.stop)

# Stacking for move down
def down(user, event):
    user.transpose()
    user.reverse()
    user.stack()
    user.combine()
    user.stack()
    user.reverse()
    user.transpose()
    user.add_new_tile()
    user.update_GUI()
    user.game_over()
    # Sound effect for move tiles
    mixer.music.load('musics/up_down.mpeg')
    mixer.music.play(-1)
    user.after(500, mixer.music.stop)

if __name__ == "__main__":
    Game_2048()
```

لینک ویدئو

https://drive.google.com/file/d/1uJVtvNM6ktKS6_2V1CCdXwqZ1uyMv_yX/view?usp=sharing

به علت بالابودن حجم فایل حتما باید دانلود شود تا فایل پخش شود. 🚦