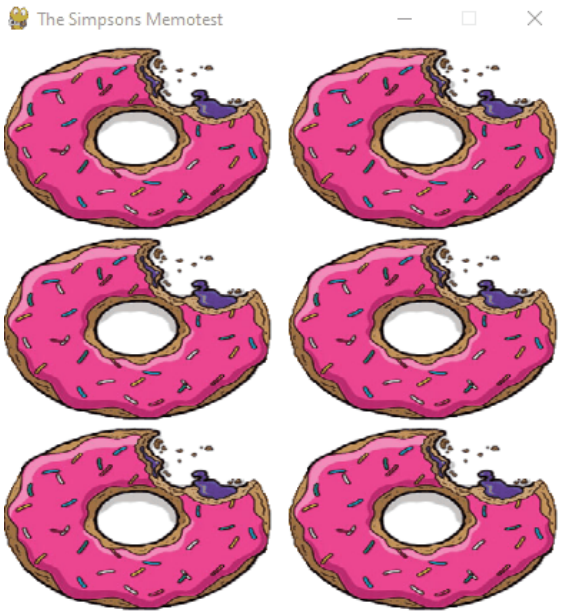
Memotest 0.1

El Memotest, también conocido como el juego de la memoria, es un juego de mesa que consiste en emparejar cartas o fichas iguales que están boca abajo. El objetivo principal del juego es recordar la ubicación de las tarjetas y encontrar todas las parejas en el menor número de intentos posible



En esta primera versión las reglas del juego son:

- El iniciar el juego todas las tarjetas comienzan “boca abajo”
- Cuando el jugador hace clic sobre alguna de las tarjetas, la misma se “da vuelta” y muestra la imagen correspondiente.
- Luego tiene como máximo 3 segundos para hacer clic en una segunda tarjeta (superado este tiempo la primera tarjeta debe voltearse nuevamente). En cada turno solo se pueden voltear dos tarjetas.
- Cuando el jugador hace clic sobre la segunda tarjeta si coincide con la misma figura que la primera, ambas tarjetas deben quedar descubiertas, caso contrario se deberán voltear ambas nuevamente
- El juego finaliza en los siguientes casos:
    - **Resultado ganador:** cuando el jugador descubre todos los pares iguales de tarjetas.

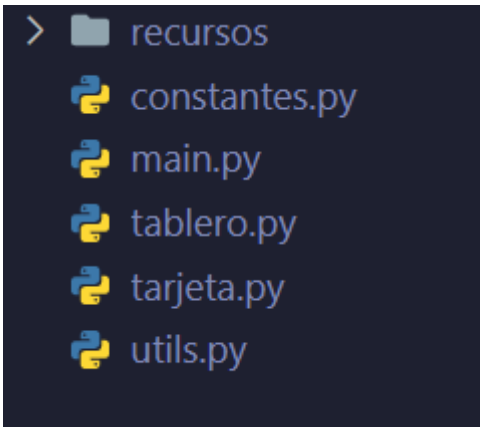
    - **Resultado perdedor:** la cantidad de veces que el jugador intentó voltear pares de tarjetas superó a la constante CANTIDAD_INTENTOS.

    - **Resultado perdedor:** el tiempo de juego superó a la cantidad de segundos establecida en la constante TIEMPO_JUEGO

Para comenzar a trabajar con este proyecto junto con este enunciado van a recibir una carpeta con recursos y una estructura básica de archivos

Estructura del proyecto

Dentro de la carpeta de trabajo nos vamos a encontrar con los siguientes archivos y carpeta que nos van a permitir trabajar de forma más clara y ordenada:



- **constantes.py**: este archivo vamos a guardar todos los valores de configuración inicial que no van a cambiar a los largo de todo el juego. Por ejemplo: ANCHO_LARGO, LARGO_PANTALLA, FPS, etc.

- **main.py**: aca vamos a tener el loop principal del juego (*game loop*) en donde vamos a realizar la configuración inicial, manejar los eventos, la lógica del juego y dibujar los elementos en la ventana

- **tablero.py**: en este archivo vamos a desarrollar todas funciones que estén relacionadas al tablero de juego, por ejemplo la función que nos permite inicializar un tablero nuevo, la que actualizar el estado, etc.

- **tarjeta.py**: idem anterior, pero en este caso a funciones relacionadas con las tarjetas, por ejemplo en este archivo vamos a encontrar la función que nos va a permitir crear una nueva tarjeta.

- **utils.py**: en este archivo vamos a tener funciones de uso general que nos van a permitir ahorrar tiempo y repetir código. Les dejamos algunas ya listas para usar

para cargar sonidos, música y generar textos

- **Carpeta recursos:** dentro de esta carpeta vamos a encontrar las imágenes y sonidos que vamos a utilizar en este proyecto.
  Para las tarjetas contamos con 8 imágenes distintas que se encuentran nombradas *01.png* al *08.png.* El archivo *00.png* corresponde a la tarjeta volteada (“boca abajo”)

Crear tarjeta

1. Dentro del archivo *tarjeta.py* vamos a encontrar la firma de la función *crear_tarjeta()*. Para lo cual nosotros debemos realizar el desarrollo de la función, cuyo objetivo será crear y retornar un diccionario el cual va a representar a una tarjeta con todos sus datos de inicialización cargados.

   El diccionario deberá contar con los siguientes pares clave-valor:

   - ***superficie*:** vamos a guardar la superficie (*pygame.Surface*) obtenida de cargar la imagen que se encuentra en *CARPETA_RECURSOS* + *nombre_imagen* (parámetro recibido en la función)
     Para poder mostrarla en el tamaño correcto vamos a necesitar acomodarla (escalarla) al tamaño de la tarjeta mediante *pygame.transform.scale()*

   - ***superficie_escondida*:** idem al valor anterior, pero en este caso vamos a guardar la superficie correspondiente a la imagen *CARPETA_RECURSOS* + *nombre_imagen_escondida* (que recibimos por parámetro)

   - ***identificador*:** es un número entero que nos va permitir identificar cada tarjeta. Las tarjetas que tengan la misma imagen (*superficie*) van a tener el mismo identificador

   - ***visible***: valor booleano que indica si la tarjeta debe mostrar la imagen de la tarjeta “dada vuelta” (es decir la imagen correspondiente a *superficie_escondida*) o mostrar la imagen correspondiente a la tarjeta (imagen correspondiente a *superficie*) . Valor inicial: *False*

   - ***descubierto*:** valor booleano que nos indica si esa tarjeta ya fue descubierta su pareja. Valor inicial: *False*

   - ***rectangulo*:** vamos a guardar el rectángulo obtenido mediante *get_rect()* del valor que tengamos dentro de *superficie*

2. Por último antes de retornar el diccionario que acabamos de crear vamos a posicionar el rectángulo que tenemos dentro del valor *rectangulo* en la posición x e y que recibimos por parámetro en este funcion

Crear lista de tarjetas

3. Dentro del archivo *tablero.py* vamos a encontrar la firma de la función *generar_lista_tarjetas()*. Para lo cual nosotros debemos realizar el desarrollo de la misma.
   El objetivo de esta función es retornar una lista con las tarjetas necesarias para poder jugar una partida.

   Cada tarjeta las vamos a crear llamando a la función que creamos en el apartado anterior (que se encuentra en *tarjeta.crear_tarjeta()*) pasándole los parámetros necesarios:

   - ***identificador*:** dentro de la función *generar_lista_tarjetas()* vamos a encontrar una lista llamada *lista_id* que se genera a partir de lo que nos devuelve la función *generar_lista_ids_tarjetas().* Esta lista contiene una serie de enteros (desde 1 hasta la constante *CANTIDAD_TARJETAS_UNICAS*) ordenados de forma aleatoria y repetidos dos veces (es decir vamos a tener dos veces el número 1, dos veces el número 2, etc). Por ejemplo:
     [5, 6, 5, 2, 1, 4, 6, 1, 3, 4, 3, 2]

     La idea es utilizar estos números como identificadores de cada par de tarjetas

   - ***nombre_imagen**:* recuerden como se encontraban nombrados los archivos de las imágenes que vamos a utilizar para las tarjetas: "*01.png*", "*02.png*". Podemos aprovechar esto y generar automáticamente los nombres de los archivos de la imágenes haciéndolos coincidir con el número de identificador

   - ***nombre_imagen_escondida:*** en este parámetro no tenemos mucho que pensar, se trata del nombre de la imagen que corresponde a la tarjeta *boca abajo* (en nuestro juego, la imagen de la dona)

   - **Posición horizontal y vertical (argumentos *x* e *y*):** en este caso se trata de dos valores enteros que representan la ubicación a modo de coordenadas cartesianas de la tarjeta dentro del tablero.
     Tener en cuenta que el origen de coordenadas (es decir la posición (0,0)) se encuentra en la esquina superior izquierda. Cuando aumenta la posición de X la tarjeta se desplaza hacia la derecha y cuando aumenta la posición de Y se desplaza hacia abajo.

     Para hacerles ganar tiempo les dejamos un algoritmo básico para obtener esos valores. Se trata de dos bucles *for* anidados, uno para calcular el valor de X y otro para calcular el valor de Y.

Crear tablero

4. Dentro del archivo *tablero.py* vamos a encontrar la firma de la función *crear_tablero()*. Para lo cual nosotros debemos realizar el desarrollo de la misma. El objetivo de esta función es retornar un diccionario el cual va a representar a un tablero de juego con todos sus datos de inicialización cargados.

   El diccionario deberá contar con los siguientes pares clave-valor:

   - ***tarjetas*:** vamos a guardar la listas de tarjetas del juego que estamos por comenzar. La misma se genera por medio de la función que creamos en el apartado anterior llamada *generar_lista_tarjetas()*

   - ***tiempo_ultimo_destape:*** se trata de un valor entero que representa una referencia temporal desde la última vez que se hizo click sobre una tarjeta. Más adelante vamos a ver como actualizar este valor (para lo cual vamos a utilizar la función *pygame.time.get_ticks()*). Valor inicial: 0

   - ***primer_tarjeta_seleccionada:*** lo vamos a utilizar para guardar el diccionario que representa la primer tarjeta seleccionada (más adelante vamos a actualizar su valor cuando hagamos clic sobre alguna tarjeta). Valor inicial: *None*

   - ***segunda_tarjeta_seleccionada:*** idem al valor anterior pero en este caso para la segunda tarjeta a la que le hagamos click. Valor inicial: *None*

Dibujar tablero

5. Dentro del archivo *tablero.py* vamos a encontrar la firma de la función *dibujar_tablero()*. Para lo cual nosotros debemos realizar el desarrollo de la misma. El objetivo de esta función es fusionar / imprimir en la ventana del juego principal todas las tarjetas en sus posiciones correspondientes y según su estado

   Esta función recibe los siguientes parámetros:

   - ***tablero:*** es el diccionario con los datos cargados del tablero actual que se está jugando

   - **pantalla_juego:** es la superficie que representa la ventana principal del juego (que obtenemos cuando hacemos *pygame.display.set_mode* en el archivo *main.py*)

Para lograr nuestro objetivo vamos a necesitar iterar la lista de tarjetas contenida dentro del diccionario *tablero* y en función del valor de la clave *visible* de cada *tarjeta* debemos mostrar la superficie correspondiente a la tarjeta "*dada vuelta"* o la imagen correspondiente.

Para fundir / imprimir una superficie sobre otra debemos llamar a *blit* desde la superficie que queremos utilizar como base (en este caso *pantalla_juego*). *blit* recibe dos argumentos obligatorios:

- La segunda superficie (es decir la que queremos dibujar sobre la superficie base). En nuestro caso la superficie correspondiente a la tarjeta (ya sea si está visible o *dada vuelta*)

- Las coordenadas o rectángulo donde queremos ubicarlo dentro de la superficie base (en nuestro caso el rectángulo correspondiente a la tarjeta lo tenemos dentro de la clave *rectangulo* de la misma)

Detectar colisión

6. Dentro del archivo *tablero.py* vamos a encontrar la firma de la función *detectar_colision()*. Para lo cual nosotros debemos realizar el desarrollo de la función.
   El objetivo principal de esta función identificar la tarjeta sobre la cual se hizo *click* con el mouse

   Esta función recibe los siguientes parámetros:

   - ***tablero:*** es el diccionario con los datos cargados del tablero actual que se está jugando

   - **Coordenadas con la posición donde se hizo click:** se trata de una tupla con los valores de las coordenadas de la posición donde se hizo click con el mouse (vamos a obtener este valor dentro del *game loop* principal cuando hacemos el manejo de los eventos)

   Para lograr nuestro objetivo vamos a necesitar iterar la lista de tarjetas contenida dentro de *tablero* para luego llamar al collidepoint() del rectángulo de cada tarjeta y pasarle como argumento las coordenadas obtenidas de la posición del mouse

   rectangulo.collidepoint((x, y)) nos va a devolver *True* cuando esas coordenadas que le pasamos caen (o *colisionan*) sobre nuestro rectángulo.

   Una vez que determinemos sobre que tarjeta se hizo click, vamos a tener que actualizar los siguientes valores:

   - Pasamos *visible* a *True* de la tarjeta en la cual se hizo click

   - Si es que se trata de la primer tarjeta del turno (en un turno solo podemos voltear dos tarjetas) entonces en *tablero["primer_tarjeta_seleccionada"]* guardamos la tarjeta seleccionada

- Caso contrario si se trata de la segunda tarjeta entonces actualizamos el valor de *tablero["segunda_tarjeta_seleccionada"]* con el valor de la tarjeta seleccionada.

- Por último actualizamos también el valor de *tablero["tiempo_ultimo_destape"]* con el valor obtenido de *pygame.time.get_ticks()*)

Nota: probablemente antes de realizar todos estos cambios vamos a necesitar validar que no estemos intentando destapar más de 2 tarjetas.

Comparar tarjetas

7. Dentro del archivo *tablero.py* vamos a encontrar la firma de la función *comparar_tarjetas()*. Para lo cual nosotros debemos realizar el desarrollo de la función.

   El objetivo de esta función es bastante simple, debe comparar

Actualizar tablero

8. Dentro del archivo *tablero.py* vamos a encontrar la firma de la función *actualizar_tablero()*. Para lo cual nosotros debemos realizar el desarrollo de la misma.

   El objetivo de esta función es actualizar el estado de las distintas tarjetas del tablero en relación a las reglas del juego

   La función va a recibir como parámetro el diccionario que representa el *tablero* del juego actual

   Una de las primeras cosas que debemos verificar es si paso más del tiempo permitido en el que podemos tener una tarjeta visible. Para hacer esto debemos obtener el tiempo actual mediante *pygame.time.get_ticks()* y luego calcular el tiempo transcurrido desde el ultimo destape. Esto lo podemos hacer calculando la diferencia que tenemos con el dato guardado en *tablero["tiempo_ultimo_destape"]*.

   Si este tiempo supera el valor que tenemos guardado en la constante TIEMPO_MOVIMIENTO entonces debemos realizar las siguientes acciones:

   - **tablero["tiempo_ultimo_destape"]:** lo volvemos a 0

   - **Volver a tapar todas las tarjetas que no fueron descubiertas aún:** para hacerlo podemos iterar la lista de tarjetas, verificar si el valor de *descubierto*

es *False* y en ese caso cambiamos el valor de *visible* a *False*

- ***tablero["primer_tarjeta_seleccionada"]*:** lo volvemos a dejar en *None*
- ***tablero["segunda_tarjeta_seleccionada"]:*** lo volvemos a dejar en *None*

Otras de las cosas que vamos a necesitar verificar es si las *primer_tarjeta_seleccionada* y *segunda_tarjeta_seleccionada* tienen el mismo identificador, en caso de tenerlo debemos cambiarle el valor a *visible* a *True* y descubierto a *True*