DOCUMENTATION

- The **sys module** in **Python** provides various functions and variables that are used to manipulate different parts of the **Python** runtime environment. To use this module first of all we need to import it.

```
import sys
```

- The **random module** is a built-in module in python that you can use to make random numbers.

```
import random
```

- Pygame is a library which is used to make games using Python.

```
import pygame
```

- There is a common technique to 'import PyGame locals' at the start of a program, like this:

```
from pygame.locals import * # Basic pygame imports
```

- **pygame.display.set_mode** is used to initialize a window or screen for display.
  Syntax:

```
pygame.display.set_mode((width,height))
```

- pygame.init() initialize all imported pygame modules.
  Syntax:

```
pygame.init()
```

- **pygame.display.set_caption** is used to set the current window caption.
  Syntax:

```
pygame.display.set_caption('Your_Title_Here')
```

- **pygame.image.load** load new image from a file.
  Syntax:

```
pygame.image.load('image_path')
```

- To rotate the image we use the **pygame.transform.rotate(image, degree)** method where we pass the **image** that we are going to rotate and the **degree** by which rotation is to be done.
  Syntax:

```
pygame.transform.rotate(image, degree)
```

- In order to play music/audio files in pygame, pygame.mixer is used (pygame module for loading and playing sounds).
  Syntax:

```
pygame.mixer.Sound('audio_path')
```

- **convert_alpha** change the pixel format of an image including per pixel alphas. In digital images, each **pixel** contains color information (such as **values** describing intensity of red, green, and blue) and also contains a **value** for its opacity known as its '**alpha**' **value**. An **alpha value** of 1 means totally opaque, and an **alpha value** of 0 means totally transparent.
  Syntax:

```
convert_alpha()
```

- **convert** change the pixel format of an image.
  Syntax:

```
convert()
```

  NOTE: Both convert( ) and convert_alpha( ) are used for fast blitting of images in the screen. In simple words we can say that the image is optimized for the game using these functions.

- To get height of the image use **image.get_height()** method**,** here image is the variable in which image object is stored.
  Syntax:

```
image.get_height()
```

- Similarly, to get width of the image we use **image.get_width()** method, here image is the variable in which image object is stored.
  Syntax:

```
image.get_width()
```

- Pygame will register all events from the user into an event queue which can be received with the code **pygame.event.get()**
  Syntax:

```
pygame.event.get()
```

  Every element in this queue is an Event object and they'll all have the attribute **type**, which is an integer representing what kind of event it is. In the pygame module there are predefined integer constants representing the type.

- Call **pygame**. **quit**() to shut down **pygame** (this includes closing the window) Call **sys.exit()** to shut down the program (this exits the infinite loop).
  Syntax:

```
pygame.quit()
```

```
sys.exit()
```

- SCREEN.blit(*image*, *(left*, *top)*) draw the image to the screen at the given position.
  Syntax:

```
SCREEN.blit(image, (left, top))
```

- **pygame.display.update()** allows to update a portion of the screen, instead of the entire area of the screen.
  Syntax:

```
pygame.display.update()
```

- To control the number of frames we will use **pygame.time.Clock.tick( )** function.
  Syntax:

```
pygame.time.Clock.tick(N)
```

  The above syntax means that for every second at most **'N'** frames should pass.

- **SOME BASIC PYGAME EVENTS:**

  a) Use **pygame.KEYDOWN** and **pygame.KEYUP** to detect if a key is physically pressed down or released.
  b) **pygame.QUIT** is sent when the user clicks the window's "X" button.
  c) **K_SPACE** used when space is pressed on keyboard.
  d) **K_UP** used when up arrow is pressed on keyboard.
  e) **K_ESCAPE** used when escape key is pressed on keyboard.

- The randrange() method returns a randomly selected element from the specified range.
  Syntax:

```
random.randrange(start, stop, step)
```

Parameter	Description
start	Optional. An integer specifying at which position to start. Default 0
stop	Required. An integer specifying at which position to end.
step	Optional. An integer specifying the incrementation. Default 1

- The **play()** method is used to play the wav and mp3 file.
  Syntax:

```
play()
```

- The min() function returns the item with the lowest value, or the item with the lowest value in an iterable.
  Syntax:

```
min(iterable)
```

- The zip() function takes iterables (can be zero or more), aggregates them in a tuple, and return it.
  Syntax:

```
zip(iterator1, iterator2, iterator3 ...)
```