Made using:
python 3.9.6
pygame 2.0.1

Setup:

-Add images/music to the assets folder.

-In your project folder (e.g. demo), add new chapter directories with your scripts to scene_list.py. Scene classes require an __init__() with 3 variables, and run().

-To change the names of the project folder, the starting chapter, or the starting scene, edit the variables in main.py.

-Use the configuration file for various general settings, as well as setting character images and map coordinates (to get user input).

Script Command Options:

* Background
* Dialog Box
* Narration Box
* Menu
* Portraits/Images
* Map
* Change Scenes
* Music, Sound Effects, Shake Screen, Pause, variables
* Coordinator
* Scene transitions
* Characters (RPG)

Backgrounds:

-Background images are scaled to: 1430 x 950 (30px width remains offscreen, and is used for shake_screen)

Set the background in the __init__:

```
self.background = 'background_image.jpg'
```

-If self.background = None, then a solid color is used.
-You can also set the background image or color in the __init__.

Add text on background:

```
self.game.display_bg_text(text, x, y, color='White', font='Georgia', size=75, bold=0)
```

-The x, y can be the int pixel coordinates, or a percent of screen size as a str: "40%". You can also use 'center' for the x coordinate.

Remove the background text:

```
self.game.clear_bg_text()
```

Dialog Box:

-Dialog box takes a list of quotes. Each quote is a list of 3-4 items: the character name, character image index, quote-text, and (optional) choices. Choices must be in the last quote:

```
self.game.display_dialog([
    ['name', 0, 'quote'],
    ['name', 0, 'quote', ['Yes', 'No']]
])
```

-If you want no name tag, use an empty string '', and same with image index for no image:

```
self.game.display_dialog([['', '', 'quote']])
```

Color:
-Text color (for a single quote only):

Add 'RED:' to the start of the quote. (Check configuration file for color name options)

-Text color (dialog box setting):

```
self.game.dialog_boxes['default'].color = 'Red'
```

Note: any pygame color can be used, or RGB value (255,0,0)

-Background color (dialog box setting):

```
self.game.dialog_boxes['default'].surface.background_color = 'Black'
```

Move the dialog box to top of screen:

```
self.game.dialog_box.move_dialog_up()
```

Move it back down:

```
self.game.dialog_box.move_dialog_down()
```

Prevent automatic removal of dialog box:

```
self.game.display_dialog(remove=False)
self.game.dialog_box.remove_dialog_box()
```

-These two commands are useful when used with transitions.

Advanced Dialog Box options:

-Change settings of dialog box in the configuration file. Use the below options to change mid script or the box dimensions.

Font:

```
self.game.dialog_boxes['default'].change_font(
    font_name=None,
    text_size=None,
    color=None,
    name_tag=False)
```

-To change for the nametag, use: name_tag=True

Toggle typing animation:

```
self.game.dialog_box.gradual_typing = False
```

Typing speed:

```
self.game.dialog_box.typing_speed = 15
```

-This is the default in milliseconds. A higher number will increase the delay time between characters.

To change the size, location, or background color of the box and choices menu, use any of the optional parameters:
-Note: The first 4 parameters can take either an integer or percent (of screen size) as a string, ex: "10.5%"

```
self.game.dialog_box.config_surface(
    x=None,
    y=None,
    width=None,
    height=None,
    background_color=None,
    border_color=None,
    border_width=None
)
```

To change the name tag, add 'name_tag=True' and any of the other optional parameters:

```
self.game.dialog_box.config_surface(
        x=None,
        y=None,
        background_color=None,
        border_color=None,
        border_width=None,
        name_tag=True
)
```

Note: width and height of name tag are set automatically according to the size of the name.

To create and switch to a new dialog box:

```
self.game.create_dialog(
        'new_name',
        x=None,
        y=None,
        width=None,
        height=None,
        background_color=None,
        border_color=None,
        border_width=None,
        name_tag_x=None,
        name_tag_y=None
)

self.game.dialog_boxes['new_name'].change_font(
        font_name=None,
        text_size=None,
        color=None,
        name_tag=False)

self.game.switch_dialog('new_name')
```

-'new_name' becomes new default. Use `self.game.switch_dialog('default')` to switch back.

Narration Box:

Narration Boxes take a list of text lists. Text can be of any length, but more than one text list will create a new Narration Box to display if you don't wish to fill the page.

```
self.game.display_narration(text)
```

To get user input, simply use the choices parameter:

```
self.game.display_narration(text, choices=["Yes", "No"])
```

The optional 'remove=True' will keep the Narration box with text up after the user clicks to continue.
To remove it, display another narration box or use:

```
self.game.narration_box.remove_narration_box()
```

Color:
-Text color (single quote only):

Start the quote with "RED:" (Check configuration file for color name options, or to add pygame colors)

-Text color (all):

```
self.game.narration_box.color = 'Red'
```

Note: Any pygame color can be used, or RGB value (255,0,0)

-Background color (all):

```
self.game.narration_box.surface.background_color = 'black'
```

Advanced:

-Change settings for Narration Box in the configuration file: To change mid script or box dimensions, use the below options:

Toggle typing animation:

```
self.game.narration_box.gradual_typing = False
```

Typing speed:

```
self.game.narration_box.typing_speed = 2
```

-This is the default in milliseconds, a higher number will increase the delay between characters.

To change the font:

```
self.game.dialog_box.change_font(font_name=None, text_size=None, color=None)
```

To change the size, placement, and background colors, fill in any of the optional parameters:

```
self.game.narration_box.config_surface(
    x=None,
    y=None,
    width=None,
    height=None,
    background_color=None,
    border_color=None,
    border_width=None
)
```

-Can use int or percent (of screen size) ex: "10.5%"

Menu

Create a menu:

```
self.game.create_menu(
    name,
    items,
    x,
    y,
    color="Black",
    size=50,
    spacing=10,
    align='left',
    bold=0,
    font='georgia'
)
```

To add a background box behind menu text:

```
self.game.menus['menu_name'].add_bg(
    padding=20,
    bg_color='white',
    border_color='Grey',
    border_width=0
)
```

To enable or disable a menu item using the index:

```
self.game.menus['menu_name'].disable_enable_menu_item(0)
```

Move in from sides (animation):

```
self.game.menus['menu_name'].move_in_left(remove=False)
```

```
self.game.menus['menu_name'].remove_menu()
self.game.menus['menu_name'].move_in_right()
```

Portraits/Images

Add an image to screen:

```
self.game.portraits['example'].blit_image(self, x=0, y=0, dialog='left')
```

-x can be: 'right' 'center' 'left' or int
-y can be: 'bottom' or 'top' or int
-dialog can be: 'left' 'right' or 'center' (automatic x y placement above dialog box)

Animations:

```
self.game.portraits['example'].move_in_left(y='default')
self.game.portraits['example'].move_in_right(y='default')
self.game.portraits['example'].move_out_left()
self.game.portraits['example'].move_out_right()
```

Change size:

```
self.game.portraits['example'].upscale(-20)
```

Remove image:

```
self.game.portraits['example'].remove()
```

Maps

Display a map:

```
self.game.display_map('map_name')  -> 'name of location'
```

-input returned is stored in self.game.input_return

No input, reset the previously chosen location, or set a starting location:

```
self.game.display_map('map_name', display_only=True, reset_loc=True, set_loc='loc_name')
```

Show/hide a location during script:

```
self.game.show_map_loc('map_name', 'loc_name')
self.game.hide_map_loc('map_name', 'loc_name')
```

Remove map:

```
self.game.remove_map()
```

To disable input only:

```
self.game.remove_map(keep_display=True)
```

Configure a map's settings:

```
self.game.maps['map_name'].config_map(
        x=15,
        y=15,
        width="80%",
        height="80%",
        border_color='Grey',
        border_width=4,
        txt_color='white',
        txt_bg='Black',
        font_size=50,
        font='georgia',
        bold=1,
        dot_radius=30,
        pad_multi=None
)
```

Scenes

To change scene:

```
self.game.change_scene('chapter_name', 'scene_name')
```

-'chapter_name' is the directory name inside project folder. 'scene_name' is the name for the class specified in scene_list.py.

Various

Music:

```
self.game.play_song(
        song_name,
        volume=None,
        stop=False,
        pause=False,
        unpause=False,
        repetition=-1
)
```

-volume is a number between 0 and 1

Check if a song is currently playing using pygame:

```
pygame.mixer.music.get_busy() -> T/F
```

Sound Effect:

```
self.game.sounds['sound_name'].play()
```

Shake Screen:

```
self.game.shake_screen()
```

Does not pause script (to use with dialog) but can be paused with `self.game.game_loop_input(15)`

A pause without input:

```
self.game.game_loop_input(num)  # num is number of loops. 30 = 1 second
pygame.time.delay(1000)  # 1 second
```

To create a variable to use, can use the flag_vars dictionary (can also be added in configuration file):

```
self.game.flag_vars["new_var"] = False
```

Coordinator

The map locations in configuration uses percent of screen rather than coordinates, so that it will scale to screen size changes. To quickly get the locations, you can use the Coordinator, simply put the code at the start of your script and click on the map where you want the location, and the coordinates will print to the console. (Coordinator.py can also be run directly)

```
self.game.coordinator.coordinate('map_name')
```

Transitions

```
self.game.fade_out()
self.game.fade_in()
self.game.slide_left('background_image.jpg')
```

```
self.game.slide_right('background_image.jpg')
```

Characters

Note: To add RPG elements to game. Not really implemented yet.

Adding a character to dict in configuration.py:

```
{'Name': Character(preset='knight')}
```

-preset options: 'knight', 'archer', 'thief', 'white mage', 'black mage'

Without a preset:

```
{'Name': character(name=None, hp=0, mp=0, attack=0, defense=0, speed=0, target=0)}
```