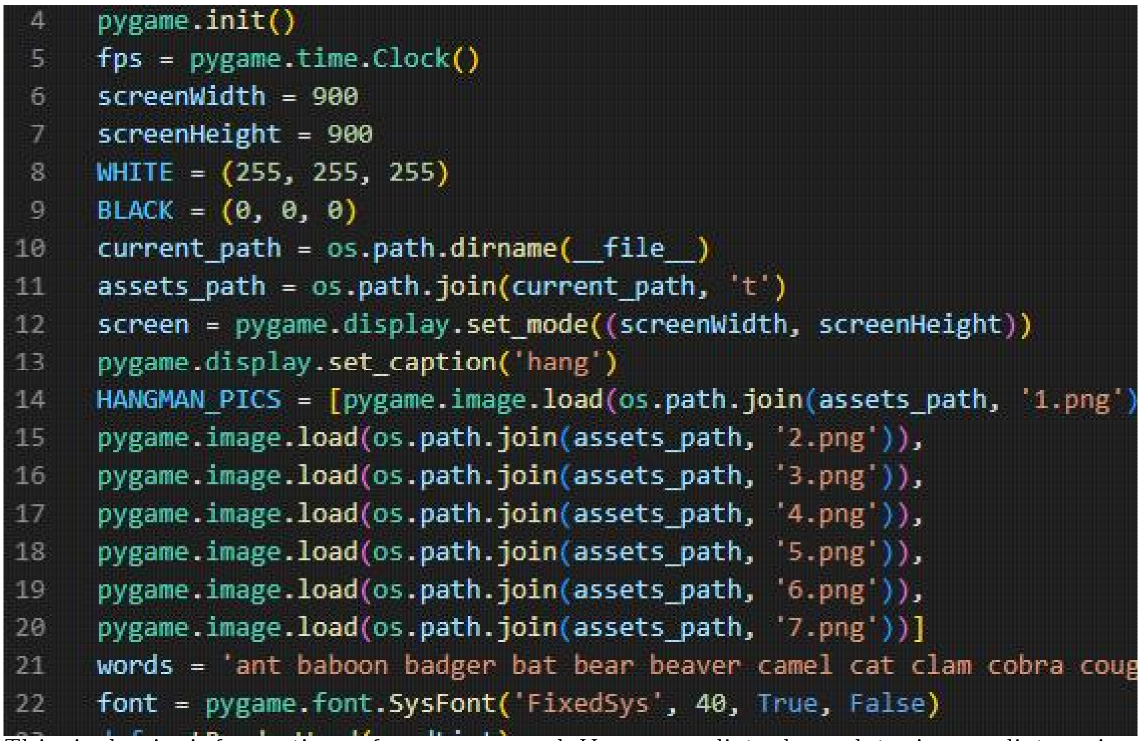
report_hw2(extra. no using input functin.)

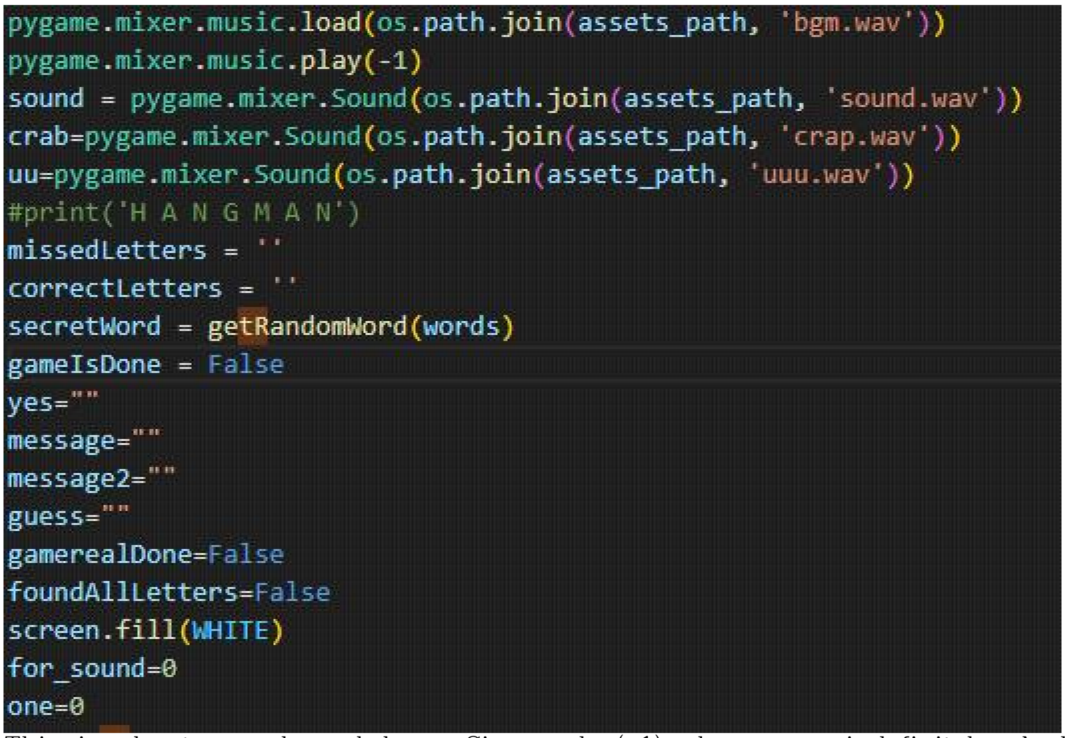
20171300 지용환

Most of the functions used in this assignment (those for Hangman calculation) are already covered in class, so I will skip them. Instead, I will focus on the new changes.



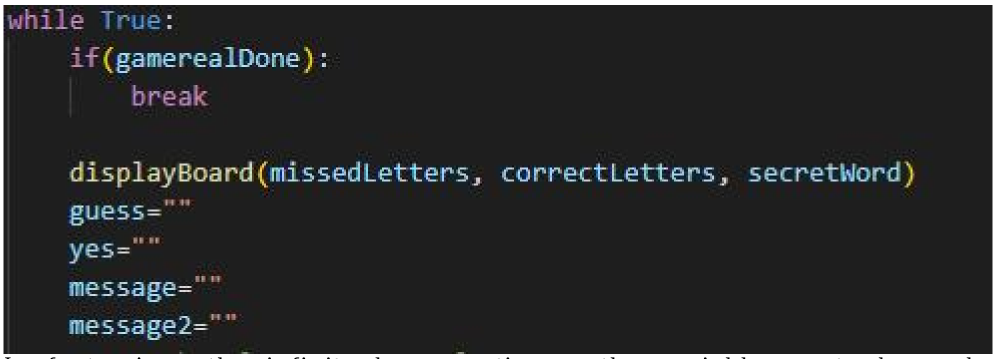
```
pygame.init()
fps = pygame.time.Clock()
screenWidth = 900
screenHeight = 900
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
current_path = os.path.dirname(__file__)
assets_path = os.path.join(current_path, 't')
screen = pygame.display.set_mode((screenWidth, screenHeight))
pygame.display.set_caption('hang')
HANGMAN_PICS = [pygame.image.load(os.path.join(assets_path, '1.png')
pygame.image.load(os.path.join(assets_path, '2.png')),
pygame.image.load(os.path.join(assets_path, '3.png')),
pygame.image.load(os.path.join(assets_path, '4.png')),
pygame.image.load(os.path.join(assets_path, '5.png')),
pygame.image.load(os.path.join(assets_path, '6.png')),
pygame.image.load(os.path.join(assets_path, '7.png'))]
words = 'ant baboon badger bat bear beaver camel cat clam cobra coug
font = pygame.font.SysFont('FixedSys', 40, True, False)
```

This is basic information of pygame, and Hangman list chaned to image list, using pygame,image.load.



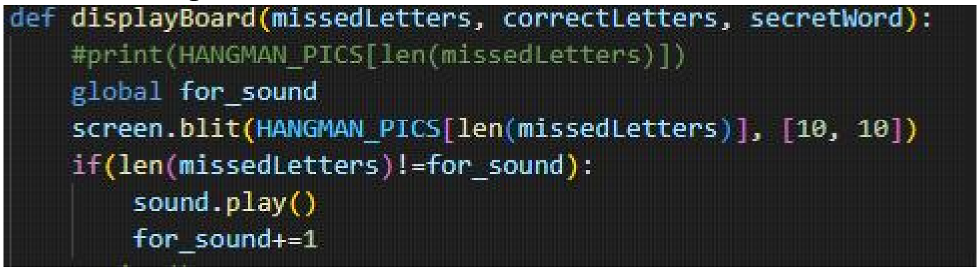
```
pygame.mixer.music.load(os.path.join(assets_path, 'bgm.wav'))
pygame.mixer.music.play(-1)
sound = pygame.mixer.Sound(os.path.join(assets_path, 'sound.wav'))
crab=pygame.mixer.Sound(os.path.join(assets_path, 'crap.wav'))
uu=pygame.mixer.Sound(os.path.join(assets_path, 'uuu.wav'))
#print('H A N G M A N')
missedLetters = ''
correctLetters = ''
secretWord = getRandomWord(words)
gameIsDone = False
yes=""
message=""
message2=""
guess=""
gamerealDone=False
foundAllLetters=False
screen.fill(WHITE)
for_sound=0
one=0
```

This is about sound, and bgm. Since .play(-1) play songs indefinitely. And yes, message, message2, guess are variables that was intentionally made in advance to keep text displayed in the window. Simiarly gamerealDone, foundAllletters are variables to check condition in pygame. When using pygame, regardless of input (all input functions were removed for extra in the first place), infinite loops continue to run at an extremely fast speed, so you have to define variables in advance and change these variables in the loop according to the event. Then work can be done.



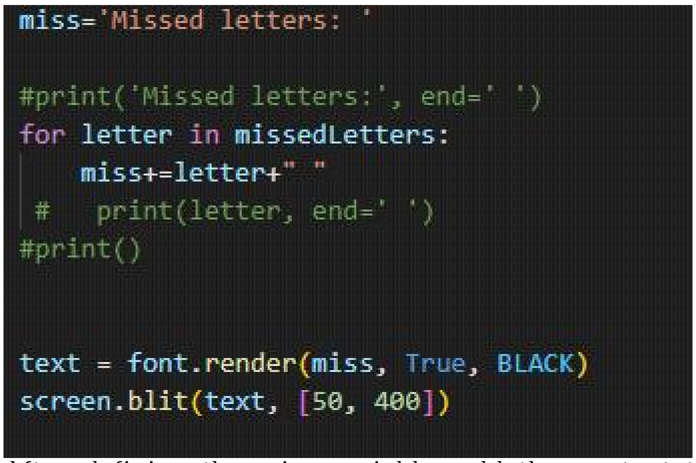
```
while True:
    if(gamerealDone):
        break

    displayBoard(missedLetters, correctLetters, secretWord)
    guess=""
    yes=""
    message=""
    message2=""
```

In fact, since the infinite loop continues, the variable must always be initialized after executing the event once.



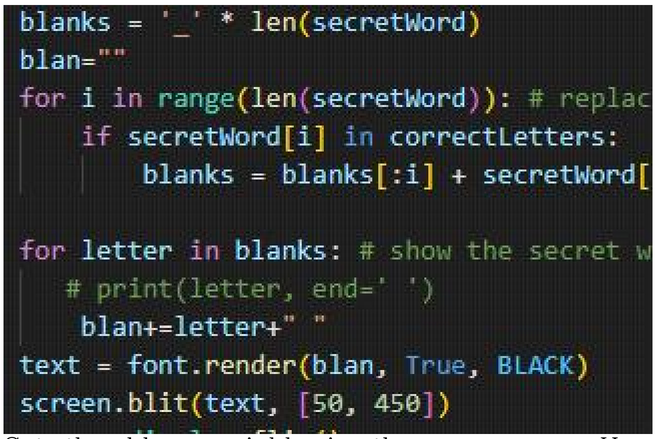
```
def displayBoard(missedLetters, correctLetters, secretWord):
    #print(HANGMAN_PICS[len(missedLetters)])
    global for_sound
    screen.blit(HANGMAN_PICS[len(missedLetters)], [10, 10])
    if(len(missedLetters)!=for_sound):
        sound.play()
        for_sound+=1
```

This is displayBoard function. First, draw image in window using blit. And for_sound global variable is variable for sound. If len(miss..) != for_sound, this mean missedletters is added, this mean you’re along. So, in this situation, sound.play(), and add 1 to the sound variable to prevent the song from playing infinitely.



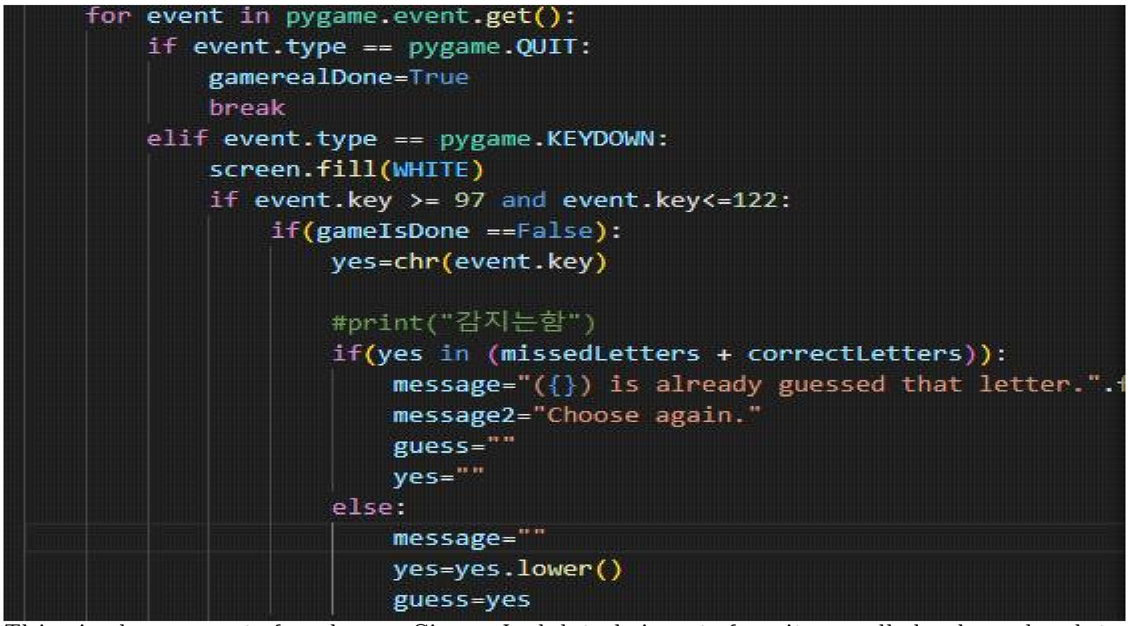
```
miss='Missed letters: '

#print('Missed letters:', end=' ')
for letter in missedLetters:
    miss+=letter+" "
#   print(letter, end=' ')
#print()


text = font.render(miss, True, BLACK)
screen.blit(text, [50, 400])
```

After defining the miss variable, add the content to the miss variable, and print it out with blit when everything is added.



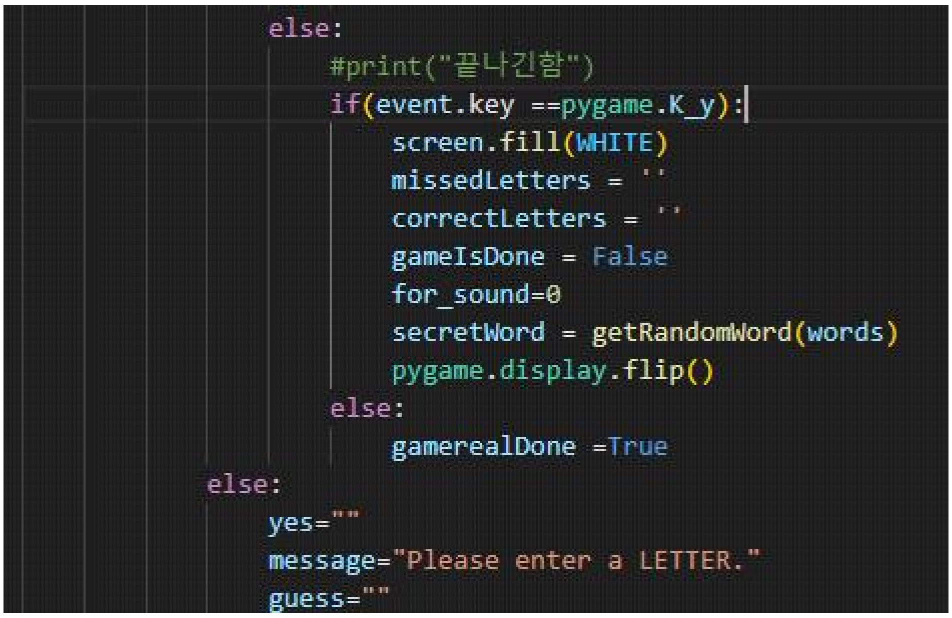
```
blanks = '_' * len(secretWord)
blan=""
for i in range(len(secretWord)): # replac
    if secretWord[i] in correctLetters:
        blanks = blanks[:i] + secretWord[

for letter in blanks: # show the secret w
   # print(letter, end=' ')
    blan+=letter+" "
text = font.render(blan, True, BLACK)
screen.blit(text, [50, 450])
```

Set the blan variable in the same way. Up to this point, we can display the hangman's picture, miss letter, and matched letter on the screen.



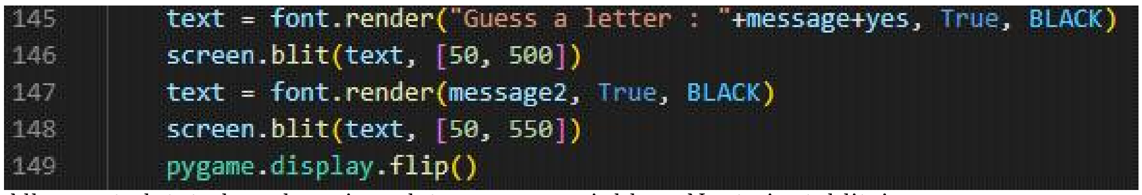
```
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            gamerealDone=True
            break
        elif event.type == pygame.KEYDOWN:
            screen.fill(WHITE)
            if event.key >= 97 and event.key<=122:
                if(gameIsDone ==False):
                    yes=chr(event.key)

                    #print("감지는함")
                    if(yes in (missedLetters + correctLetters)):
                        message="({}) is already guessed that letter.".f
                        message2="Choose again."
                        guess=""
                        yes=""
                    else:
                        message=""
                        yes=yes.lower()
                        guess=yes
```

This is key event for loop. Since I deleted input funciton, all keyboard-related exceptions were handled here. Exception are guessing already guessed letter, don’t enter alphbet. Using 97<event.key<122 , this detected alphabet keyboard. In this case, if the game isn't over, it detects an already guessed exception, or assigns the pressed keyboard event to the guess variable.



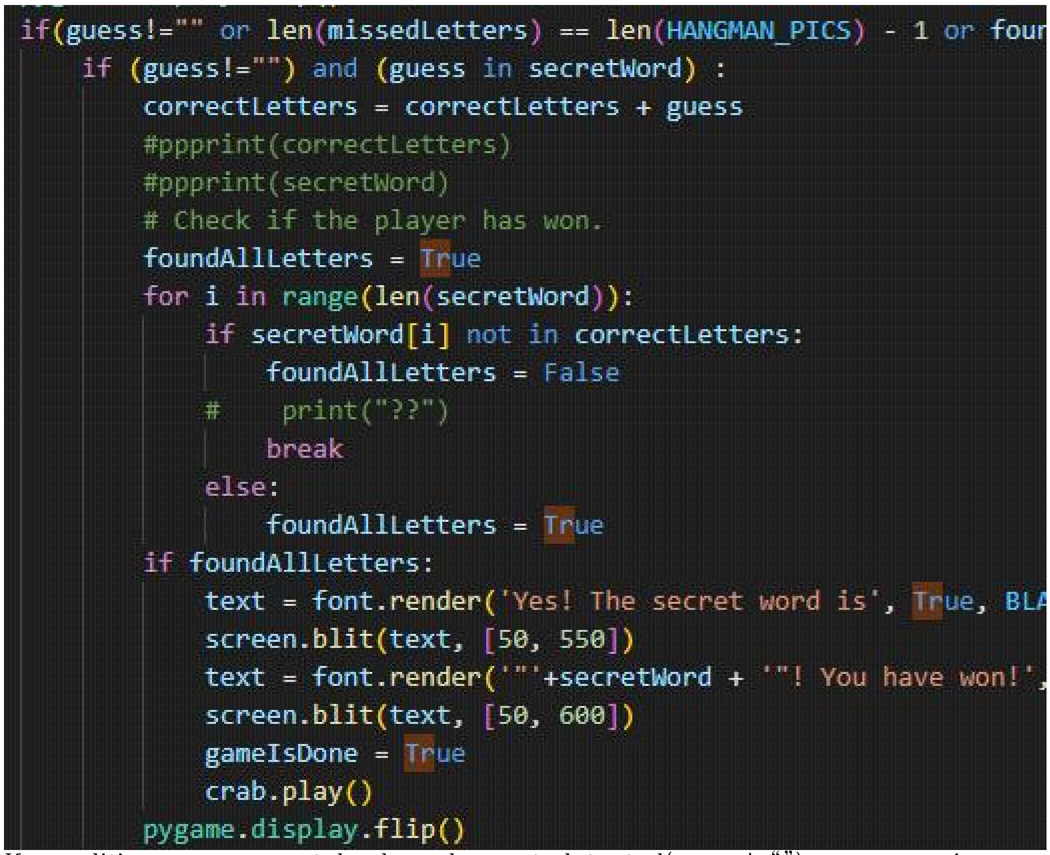
```
                else:
                    #print("끝나긴함")
                    if(event.key ==pygame.K_y):
                        screen.fill(WHITE)
                        missedLetters = ''
                        correctLetters = ''
                        gameIsDone = False
                        for_sound=0
                        secretWord = getRandomWord(words)
                        pygame.display.flip()
                    else:
                        gamerealDone =True
            else:
                yes=""
                message="Please enter a LETTER."
                guess=""
```

Or if game is over, just detect keyboar Y event. This mean new game start. Else, this mean game over. So, gamerealDone variable be True, and then escape first while loop, and quit.



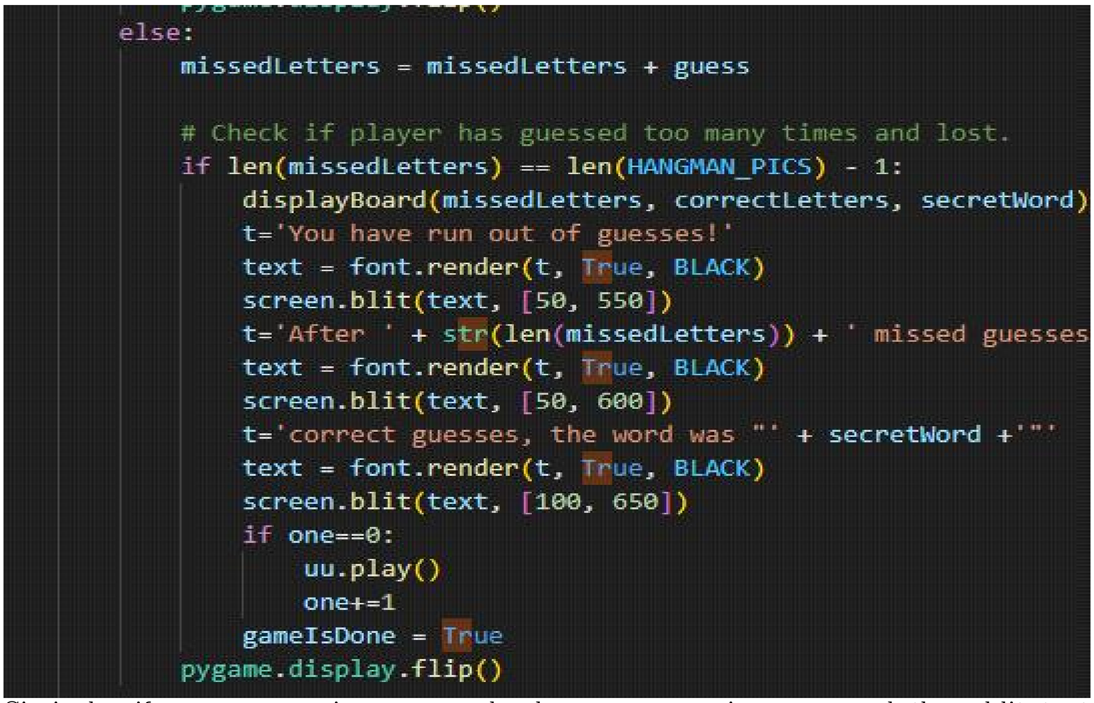
```
        text = font.render("Guess a letter : "+message+yes, True, BLACK)
        screen.blit(text, [50, 500])
        text = font.render(message2, True, BLACK)
        screen.blit(text, [50, 550])
        pygame.display.flip()
```

All event deteted and assinged to some variables. Next, just blit in screen. Next



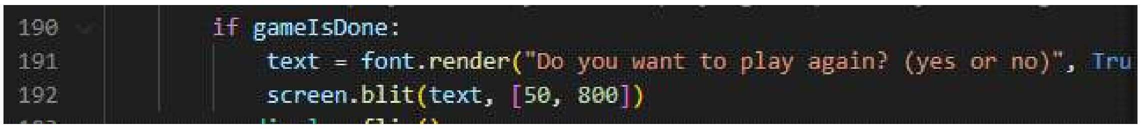
```
if(guess!="" or len(missedLetters) == len(HANGMAN_PICS) - 1 or four
    if (guess!="") and (guess in secretWord) :
        correctLetters = correctLetters + guess
        #ppprint(correctLetters)
        #ppprint(secretWord)
        # Check if the player has won.
        foundAllLetters = True
        for i in range(len(secretWord)):
            if secretWord[i] not in correctLetters:
                foundAllLetters = False
            #    print("??")
                break
            else:
                foundAllLetters = True
        if foundAllLetters:
            text = font.render('Yes! The secret word is', True, BLA
            screen.blit(text, [50, 550])
            text = font.render('"'+secretWord + '"! You have won!',
            screen.blit(text, [50, 600])
            gameIsDone = True
            crab.play()
        pygame.display.flip()
```

If conditions are corret keyboard event detected(guess!=“”) or, game is over(fail or success). The conditions of this if statement are conditions added so that the continuously running pygame can continuously blit text. Next if condtion is you guess corrent secretword. Then using foundAllletters variable, check your guess is perfect matching with secret. If perfect match, blit text on screen. Obviously crab sound begin.



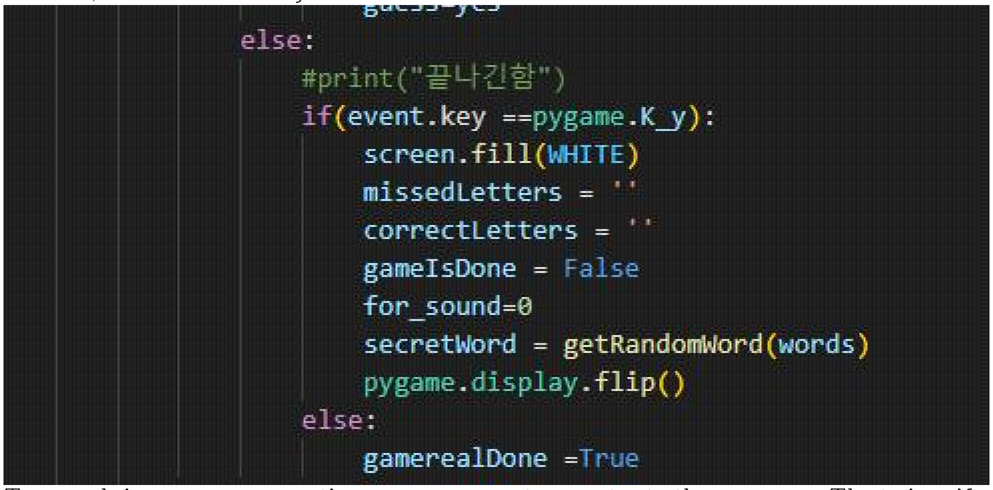
```
else:
    missedLetters = missedLetters + guess

    # Check if player has guessed too many times and lost.
    if len(missedLetters) == len(HANGMAN_PICS) - 1:
        displayBoard(missedLetters, correctLetters, secretWord)
        t='You have run out of guesses!'
        text = font.render(t, True, BLACK)
        screen.blit(text, [50, 550])
        t='After ' + str(len(missedLetters)) + ' missed guesses
        text = font.render(t, True, BLACK)
        screen.blit(text, [50, 600])
        t='correct guesses, the word was "' + secretWord +'"'
        text = font.render(t, True, BLACK)
        screen.blit(text, [100, 650])
        if one==0:
            uu.play()
            one+=1
        gameIsDone = True
    pygame.display.flip()
```

Simiarly, if your guess is wrong, check your game is over, and then blit text on screen. And using one varible, just sound once.



```
if gameIsDone:
    text = font.render("Do you want to play again? (yes or no)", Tru
    screen.blit(text, [50, 800])
```

If gameIsDone variable becomes true, in the above two conditions, then bilt text on screen, and detect key event in this condition.



```
else:
    #print("끝나긴함")
    if(event.key ==pygame.K_y):
        screen.fill(WHITE)
        missedLetters = ''
        correctLetters = ''
        gameIsDone = False
        for_sound=0
        secretWord = getRandomWord(words)
        pygame.display.flip()
    else:
        gamerealDone =True
```

To explain one more time, press y to restart the game. That is, if ‘y’, screen initialization and variable initialization are performed and if other alphabets are pressed, the gamerealDone variable for escaping from the loop statement becomes true.