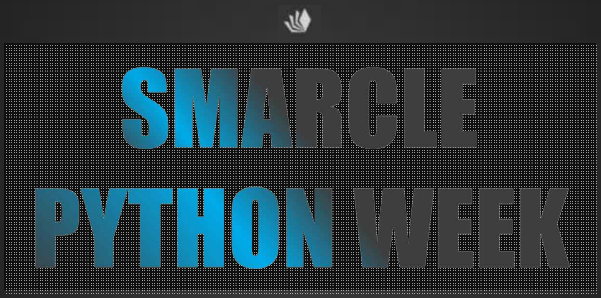
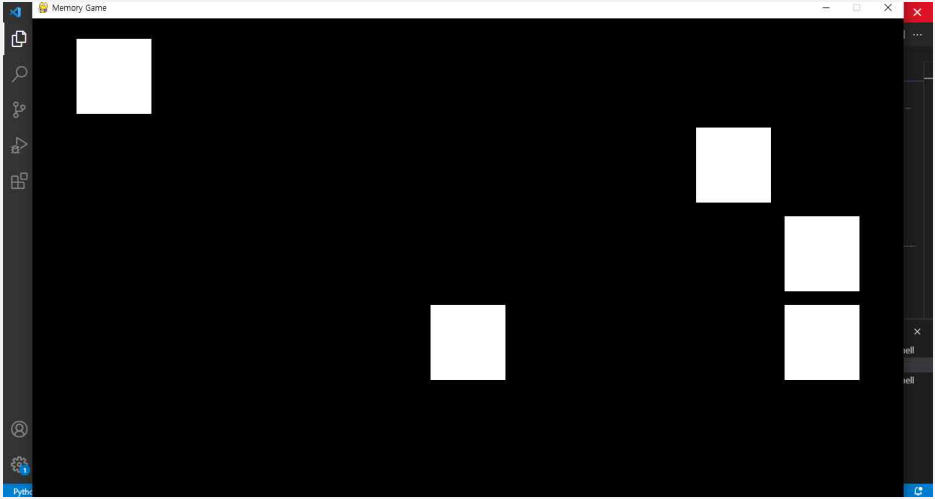

SMARCLE
PYTHON WEEK

파이썬 위크 주제 :

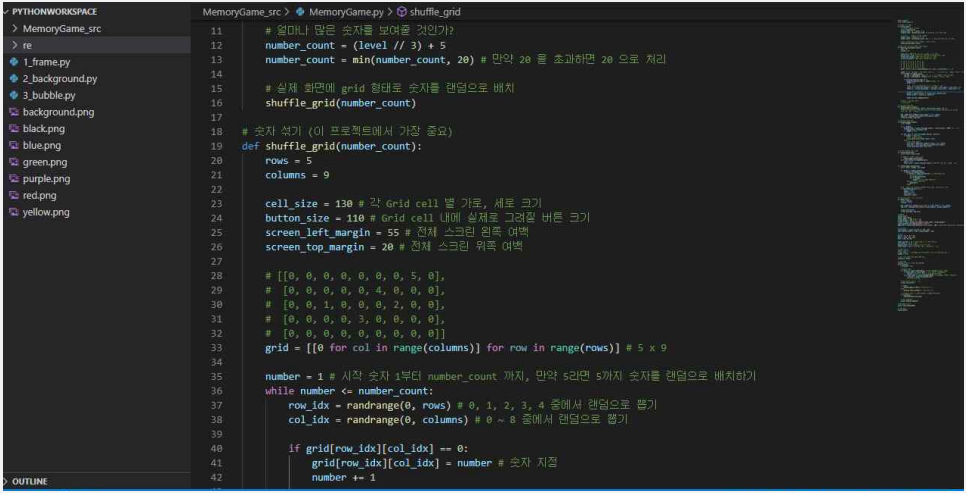
파이썬으로 만들어보는 게임 모음집!

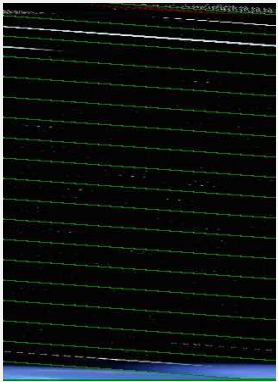

3팀 : 이윤서, 유혁재


Memory Game
3
5
4
1
2


Memory Game

# SMARCLE PYTHON WEEK





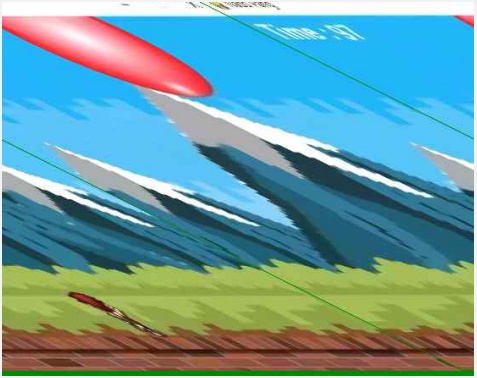
2) 슈팅 게임

# SMARCLE PYTHON WEEK

```
import pygame
import sys
import random
from time import sleep


padWidth = 480
padHeight = 640
rockImage = ['rock01.png', 'rock02.png', 'rock03.png', 'rock04.png', 'rock05.png', \
             'rock06.png', 'rock07.png', 'rock08.png', 'rock09.png', 'rock10.png', \
             'rock11.png', 'rock12.png', 'rock13.png', 'rock14.png', 'rock15.png', \
             'rock16.png', 'rock17.png', 'rock18.png', 'rock19.png', 'rock20.png', \
             'rock21.png', 'rock22.png', 'rock23.png', 'rock24.png', 'rock25.png', \
             'rock26.png', 'rock27.png', 'rock28.png', 'rock29.png', 'rock30.png']
explosionSound = ['explosion01.wav', 'explosion02.wav', 'explosion03.wav', 'explosion04.wav']

def writeScore(count):
    global gamePad
    font = pygame.font.Font('NanumGothic.ttf', 20)
    text = font.render('파괴한 운석 수: ' + str(count), True, (255, 255, 255))
    gamePad.blit(text, (10, 0))


def writePassed(count):
    global gamePad
    font = pygame.font.Font('NanumGothic.ttf', 20)
    text = font.render('놓친 운석 수: ' + str(count), True, (255, 0, 0))
    gamePad.blit(text, (300, 0))


def writeMessage(text):
    global gamePad, gameOverSound
    textfont = pygame.font.Font('NanumGothic.ttf', 60)
    text = textfont.render(text, True, (255, 0, 0))
    textpos = text.get_rect()
    textpos.center = (padWidth/2, padHeight/2)
    gamePad.blit(text, textpos)
    pygame.display.update()
    pygame.mixer.music.stop()
    gameOverSound.play()
    sleep(2)
    pygame.mixer.music.play(-1)
    runGame()

def crash():
    global gamePad
    writeMessage('GAME OVER!')

def gameOver():
    global gamePad
    writeMessage('GAME OVER!')

def drawObject(obj, x, y):
    global gamePad
    gamePad.blit(obj, (x, y))

def initGame():
    global gamePad, clock, background, fighter, missile, explosion, missileSound, gameOverSound
    pygame.init()
    gamePad = pygame.display.set_mode((padWidth,padHeight))
    pygame.display.set_caption('PyShooting')
    background = pygame.image.load('background.png')
    fighter = pygame.image.load('fighter.png')
    missile = pygame.image.load('missile.png')
    explosion = pygame.image.load('explosion.png')
    pygame.mixer.music.load('music.wav')
```

```
def runGame():
    global gamePad, clock, background, fighter, missile, explosion, missileSound

    fighterSize = fighter.get_rect().size
    fighterWidth = fighterSize[0]
    fighterHeight = fighterSize[1]

    x = padWidth * 0.45
    y = padHeight * 0.9
    fighterX = 0

    missileXY = []

    rock = pygame.image.load(random.choice(rockImage))
    rockSize = rock.get_rect().size
    rockWidth = rockSize[0]
    rockHeight = rockSize[1]
    destroySound = pygame.mixer.Sound(random.choice(explosionSound))

    rockX = random.randrange(0, padWidth - rockWidth)
    rockY = 0
    rockSpeed = 2

    isShot = False
    shotCount = 0
    rockPassed = 0

    onGame = False
    while not onGame:
        for event in pygame.event.get():
            if event.type in [pygame.QUIT]:
                pygame.quit()
                sys.exit()

            if event.type in [pygame.KEYDOWN]:
                if event.key == pygame.K_LEFT:
                    fighterX -= 5

                elif event.key == pygame.K_RIGHT:
                    fighterX += 5

                elif event.key == pygame.K_SPACE:
                    missileSound.play()
                    missileX = x + fighterWidth/2
                    missileY = y - fighterHeight
                    missileXY.append([missileX, missileY])

            if event.type in [pygame.KEYUP]:
                if event.key == pygame.K_LEFT or event.key == pygame.K_RIGHT:
                    fighterX = 0

        drawObject(background, 0, 0)

        x+= fighterX
        if x < 0:
            x = 0
        elif x > padWidth - fighterWidth:
            x = padWidth - fighterWidth

        if y < rockY + rockHeight:
            if(rockX > x and rockX < x + fighterWidth) or \
                    (rockX + rockWidth > x and rockX + rockWidth < x + fighterWidth):
                crash()
```

```
        drawObject(fighter, x, y)

        if len(missileXY) != 0:
            for i, bxy in enumerate(missileXY):
                bxy[1] -= 10
                missileXY[i][1] = bxy[1]

                if bxy[1] < rockY:
                    if bxy[0] > rockX and bxy[0] < rockX + rockWidth:
                        missileXY.remove(bxy)
                        isShot = True
                        shotCount += 1


                if bxy[1] <= 0:
                    try:
                        missileXY.remove(bxy)
                    except:
                        pass
        if len(missileXY) != 0:
            for bx, by in missileXY:
                drawObject(missile, bx, by)

        writeScore(shotCount)

        rockY += rockSpeed

        if rockY > padHeight:
            rock = pygame.image.load(random.choice(rockImage))
            rockSize = rock.get_rect().size
            rockWidth = rockSize[0]
            rockHeight = rockSize[1]
            rockX = random.randrange(0, padWidth - rockWidth)
            rockY = 0
            rockPassed +=1

        if rockPassed == 3:
            gameOver()

        writePassed(rockPassed)

        if isShot:
            drawObject(explosion, rockX, rockY)
            destroySound.play()

            rock = pygame.image.load(random.choice(rockImage))
            rockSize = rock.get_rect().size
            rockWidth = rockSize[0]
            rockHeight = rockSize[1]
            rockX = random.randrange(0, padWidth - rockWidth)
            rockY = 0
            destroySound = pygame.mixer.Sound(random.choice(explosionSound))
            isShot = False

            rockSpeed += 0.2
            if rockSpeed >=10:
                rockSpeed = 10

        drawObject(rock, rockX, rockY)

        pygame.display.update()

        clock.tick(60)

initGame()
runGame()
```



3) 오락실 게임

# SMARCLE PYTHON WEEK

```python
import os
import pygame

# 기본 초기화
pygame.init()

# 화면 크기 설정
screen_width = 640 # 가로 크기
screen_height = 480 # 세로 크기
screen = pygame.display.set_mode((screen_width, screen_height))

# 화면 타이틀 설정
pygame.display.set_caption("Splitting a ball") # 게임 이름

# FPS
clock = pygame.time.Clock()

current_path = os.path.dirname(__file__) # 현재 파일의 위치 반환
image_path = os.path.join(current_path, "images") # images 폴더 위치 반환

# 배경 생성
background = pygame.image.load(os.path.join(image_path, "background.png"))

# 스테이지 생성
stage = pygame.image.load(os.path.join(image_path, "stage.png"))
stage_size = stage.get_rect().size
stage_height = stage_size[1]

# 캐릭터 생성
character = pygame.image.load(os.path.join(image_path, "character.png"))
character_size = character.get_rect().size
character_width = character_size[0]
character_height = character_size[1]
character_x_pos = (screen_width / 2) - (character_width / 2)
character_y_pos = screen_height - character_height - stage_height

# 캐릭터 이동 방향
character_to_x = 0

# 캐릭터 이동 속도
character_speed = 5
```

```python
# 무기 생성
weapon = pygame.image.load(os.path.join(image_path, "weapon.png"))
weapon_size = weapon.get_rect().size
weapon_width = weapon_size[0]

# 무기는 한 번에 여러 발 발사 가능
weapons = []

# 무기 이동 속도
weapon_speed = 10

# 공 생성
ball_images = [
    pygame.image.load(os.path.join(image_path, "balloon1.png")),
    pygame.image.load(os.path.join(image_path, "balloon2.png")),
    pygame.image.load(os.path.join(image_path, "balloon3.png")),
    pygame.image.load(os.path.join(image_path, "balloon4.png"))]

# 공 크기에 따른 최초 속도
ball_speed_y = [-18, -15, -12, -9]

# 공들
balls = []

# 최초 발생하는 큰 공 추가
balls.append({
    "pos_x" : 50,
    "pos_y" : 50,
    "img_idx" : 0,
    "to_x" : 3,
    "to_y" : -6,
    "init_spd_y": ball_speed_y[0]})


# 사라질 무기, 공 정보 저장 변수
weapon_to_remove = -1
ball_to_remove = -1

#  Font 정의
game_font = pygame.font.Font(None, 40)
total_time = 100
```

```python
# 이벤트 루프
running = True
while running:
    dt = clock.tick(30)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                character_to_x -= character_speed
            elif event.key == pygame.K_RIGHT:
                character_to_x += character_speed
            elif event.key == pygame.K_SPACE:
                weapon_x_pos = character_x_pos + (character_width / 2) - (weapon_width / 2)
                weapon_y_pos = character_y_pos
                weapons.append([weapon_x_pos, weapon_y_pos])

        if event.type == pygame.KEYUP:
            if event.key == pygame.K_LEFT or event.key == pygame.K_RIGHT:
                character_to_x = 0

    # 캐릭터 위치 정의
    character_x_pos += character_to_x

    if character_x_pos < 0:
        character_x_pos = 0
    elif character_x_pos > screen_width - character_width:
        character_x_pos = screen_width - character_width

    # 무기 위치 조정
    weapons = [ [w[0], w[1] - weapon_speed] for w in weapons]

    # 천장에 닿은 무기 제거
    weapons = [ [w[0], w[1]] for w in weapons if w[1] > 0]

    # 공 위치 정의
    for ball_idx, ball_val in enumerate(balls):
        ball_pos_x = ball_val["pos_x"]
        ball_pos_y = ball_val["pos_y"]
        ball_img_idx = ball_val["img_idx"]
```

# SMARCLE PYTHON WEEK

```
    if ball_pos_x < 0 or ball_pos_x > screen_width - ball_width:
        ball_val["to_x"] = ball_val["to_x"] * -1

    if ball_pos_y >= screen_height - stage_height - ball_height:
        ball_val["to_y"] = ball_val["init_spd_y"]

    else:
        ball_val["to_y"] += 0.5

    ball_val["pos_x"] += ball_val["to_x"]
    ball_val["pos_y"] += ball_val["to_y"]

# 캐릭터 rect 정보 업데이트
character_rect = character.get_rect()
character_rect.left = character_x_pos
character_rect.top = character_y_pos

for ball_idx, ball_val in enumerate(balls):
    ball_pos_x = ball_val["pos_x"]
    ball_pos_y = ball_val["pos_y"]
    ball_img_idx = ball_val["img_idx"]

    # 공 rect 정보 업데이트
    ball_rect = ball_images[ball_img_idx].get_rect()
    ball_rect.left = ball_pos_x
    ball_rect.top = ball_pos_y

    # 공과 캐릭터 충돌 처리
    if character_rect.colliderect(ball_rect):
        running = False
        break

    # 공과 무기들 충돌 처리
    for weaopn_idx, weapon_val in enumerate(weapons):
        weapon_pos_x = weapon_val[0]
        weapon_pos_y = weapon_val[1]

        # 무기 rect 정보 업데이트
        weapon_rect = weapon.get_rect()
        weapon_rect.left = weapon_pos_x
        weapon_rect.top = weapon_pos_y

        # 충돌 체크
        if weapon_rect.colliderect(ball_rect):
            weapon_to_remove = weaopn_idx
            ball_to_remove =  ball_idx

            if ball_img_idx < 3:

                ball_width = ball_rect.size[0]
                ball_height = ball_rect.size[1]

                small_ball_rect = ball_images[ball_img_idx + 1].get_rect()
                small_ball_width = small_ball_rect.size[0]
                small_ball_height = small_ball_rect.size[1]

                balls.append({
                    "pos_x" : ball_pos_x + (ball_width / 2) - (small_ball_width / 2),
                    "pos_y" : ball_pos_y + (ball_height / 2) - (small_ball_height / 2),
                    "img_idx" : ball_img_idx + 1,
                    "to_x" : -3,
                    "to_y" : -6,
                    "init_spd_y" : ball_speed_y[ball_img_idx + 1]})

                balls.append({
                    "pos_x" : ball_pos_x + (ball_width / 2) - (small_ball_width / 2),
                    "pos_y" : ball_pos_y + (ball_height / 2) - (small_ball_height / 2),
                    "img_idx" : ball_img_idx + 1,
                    "to_x" : 3,
                    "to_y" : -6,
                    "init_spd_y" : ball_speed_y[ball_img_idx + 1]})

                break
        else:
            continue
        break

# 충돌한 공 or 무기 없애기
if ball_to_remove > -1:
    del balls[ball_to_remove]
    ball_to_remove = -1

if weapon_to_remove > -1:
    del weapons[weapon_to_remove]
    weapon_to_remove = -1

if len(balls) == 0:
    game_result = "Mission Complete"
    running = False


# 갱신
screen.blit(background, (0, 0))

for weapon_x_pos, weapon_y_pos in weapons:
    screen.blit(weapon, (weapon_x_pos, weapon_y_pos))

for idx, val in enumerate(balls):
    ball_pos_x = val["pos_x"]
    ball_pos_y = val["pos_y"]
    ball_img_idx = val["img_idx"]
    screen.blit(ball_images[ball_img_idx],  (ball_pos_x, ball_pos_y))

screen.blit(stage, (0, screen_height - stage_height))
screen.blit(character, (character_x_pos, character_y_pos))

elapsed_time = (pygame.time.get_ticks() - start_ticks) / 1000
timer = game_font.render("Time : {}".format(int(total_time - elapsed_time)), True, (255, 255, 255))
screen.blit(timer, (10, 10))

if total_time - elapsed_time <= 0:
    game_result = "Time Over"
    running = False

pygame.display.update()

msg = game_font.render(game_result, True, (255, 255, 0))
msg_rect = msg.get_rect(center=(int(screen_width / 2), int(screen_height / 2)))
screen.blit(msg, msg_rect)
pygame.display.update()

pygame.time.delay(2000)

pygame.quit()
```

**윤서 :** 팀원이랑 원래 인공지능 활용하는 걸 도전해보고 싶다는 생각에서 시작된 활동인데 게임을 주제에 맞춰서 여러개 만들어야 했기도 했고 인공지능 활용 부분을 완전히 잘 알지는 못해서 도전해보지 못한 점과 내가 직접 아이디어를 짜고 코딩을 한게 아니라 자료들을 참고하며 만든 게임이라는 점이 굉장히 아쉬웠다.
하지만 한줄한줄이 게임에 어떻게 필요한건지 이해하면서 만들어보니 파이썬으로 어떻게 게임을 만들 수 있는지를 배울 수 있었으므로 다음에는 직접 아이디어를 짜서 만들 수 있을 것 같다.

**혁재 :** 평소에도 게임에 관심이 많았었기에 코딩을 통해 게임이 어떻게 만들어지는지, 어떤 코드로 구성되어 있는지에 대한 의문점을 해소할 수 있어서 좋았다. 비록 이번 프로젝트에서는 간단한 게임을 만들었지만 나중에 더 복잡한 게임을 만들 수 있는 용기를 얻어 갔다.
또한 이렇게 간단한 게임이라도 몇 백 줄이 넘는 코드가 필요한 것을 보고 게임 만드는 것이 쉽지만은 않다는 것을 느꼈다.
아쉬운 점이라고 하면 해보고 싶었던 인공지능도 활용을 못해봤고 처음에 목표로 한 주제를 완벽히 소화해내지 못했다는 점이다.
그래도 이번 경험을 통해 파이썬과 더 친해질 수 있었고 더 복잡하고 어려운 프로젝트로 가기 위한 문을 열었다고 생각한다.