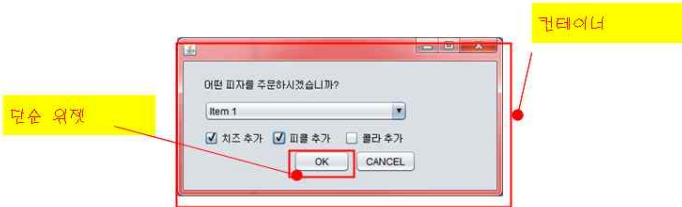


■ 서식 1 : 결과보고서 요약본

# Project Based Learning 결과보고서 요약본

교과목명	국문	인공지능을위한파이썬		
	영문	Python for AI		
PBL 관련 능력단위 (능력단위코드)		PBL 관련 능력단위요소 (능력단위요소코드)		
파이썬언어와패키지 (YCS-2001020903_20v1)		파이썬언어구현하기		
		파이썬패키지사용하기		
학년 반	2학년 1반	조원	이은우	
프로젝트 주제	Tkinter 모듈을 이용한 GUI, 간단한 게임 만들기			
지도교수	심효선	산업체 참가여부	<input type="checkbox"/> 유 <input checked="" type="checkbox"/> 무	
참여학생	이은우(202025016)			
작품 개요 (주제 선정 이유)	2학년에 올라가면 기계학습을 배우게 되는데 그중에 최종 결과를 사용자에게 보여주기 위하여 UI를 작성하여야 한다. 파이썬에서는 tkinter 모듈을 제공하여 GUI프로그램을 작성할 수 있게 해준다. 파이썬을 이용하여 GUI의 일반적인 구조를 이해하고, 배치관리자를 사용하여 화면의 레이아웃을 만들고, 위젯의 콜백함수를 이용하여 이벤트를 처리하고, 캔버스에 다양한 도형을 그려서 애니메이션까지 동작 시킬 수 있는 코딩 능력을 키우는 것을 목표로 한다.			
작품 구조도 (문제점 제시 및 개선방안)	<p>작품은 동작하는 게임을 작성하는 것으로 다음의 내용을 포함해야 한다.</p> <p>1) 게임이 제공하는 기능에 대한 신규 아이디어 설명</p> <ul style="list-style-type: none"><li>기존 실습 코드에서는 단순히 깨뜨릴 블록들과 움직이는 공 하나, 공을 튕길 패들 하나로 벽돌만 깨는 게임으로 코드가 짜여있다. 여기서 실수로 공을 놓치게 되면 화면상에서 공이 사라지고 게임은 계속되고 있어 다시 시작할수도, 게임을 이어서 할 수도 없는 상태에 이르게 된다.</li><li>때문에 우리는 이 기존 코드에서 실수하여 공을 놓쳤을때를 대비해 3번의 기회(목숨 시스템)과 그 기회 마저도 다 잃었을 경우 재도전을 할 수 있는 기능을 추가하려고 한다. 또 단순히 벽돌을 깨기만 하면 재미없을 것을 우려하여 벽돌을 깰 때마다 점수가 올라가는 점수판 기능도 추가하려 한다.</li></ul> <p>2) 게임이 제공하는 UI 화면에 대한 설명</p> <ul style="list-style-type: none"><li>화면 좌측 상단에 점수와 남은 목숨(기회)를 보여주는 UI를 추가하여 게임도중 확인하기 쉽게 기능을 추가하였다.</li><li>공을 놓쳐 남은 목숨(기회)이 모두 소모될때까지 벽돌이 남아있다면 Game Over라는 글자와 지금까지 벽돌을 깬 점수를 출력하여 게임이 끝났음을 알려준다. 여기서 새로운 게임을 하기위해 “스페이스바를</li></ul>			

	<p>리 재도전하기“ 라는 UI를 추가해 주었다.</p> <ul style="list-style-type: none"> <li>반대로 벽돌을 모두 깨면 게임에서 승리하게 되고. 벽돌을 깬 점수와 함께 “승리!“ 라는 문구를 출력하여 게임에서 이겼음을 알려준다. 마찬가지로 새 게임을 시작할수있게 ”새로운 게임“이라 써진 버튼을 누르면 새로운 게임을 할수있게 안내하는 UI를 추가하였다.</li> </ul> <p>3) 동작이 가능하도록 한 구현 방법</p> <ul style="list-style-type: none"> <li>ui에 관한 것은 대부분이 BrickBreaker 클래스에 새로운 함수를 정의하여 추가되어있다. 게임 시작전 간단한 게임 설명과 게임시작 버튼을 눌러 게임을 시작하게 하는 “show_start_screen()함수“와 버튼을 누르면 해당 시작화면과 시작버튼을 없애고 start() 함수를 불러오는 “start_game()함수”, 새로운 UI[점수와 목숨]을 추가해줄 “update_ui()함수”, 게임에서 졌음을 알려주는 “game_over()함수”, 게임에서 이겼음을 알려주는 “game_won()함수”, 게임 재시작시 처음에 나왔던 화면을 만들어주는 “restart_game()함수”등을 정의하여 추가하였다.</li> <li>그 외적인 부분은 공의 속도를 약간 증가시키거나, 패들의 폭을 넓혀난이도를 조금 줄이는 약간의 수정작업과 벽돌을 깨때마다 점수를 계산하는 반복문 추가, 목숨[기회]를 계산하는 조건문 추가, 게임 승리 조건인 벽돌이 모두 깨졌는지 삭제된 벽돌을 필터링하는 조건문등을 통해 제시했던 아이디어들을 구현시켰다.</li> </ul> <p>4) 최종 동작 화면</p>
<p>관련 이론</p>	<p>마지막 결과물 제작 부분에서 소개예정</p> <ul style="list-style-type: none"> <li>▶ Tkinter</li> <li>▶ 단순 위젯(Button, Canavs)</li> <li>▶ 컨테이너 컴포넌트(Frame, Toplevel, LabelFrame, PanedWindow)</li> </ul> 
<p>결과물 제작 (문제점 개선사항)</p>	<p>1. 과제 아이디어 제시</p> <ul style="list-style-type: none"> <li>- 기존 게임에서 미흡한 기능을 개선 <ul style="list-style-type: none"> <li>• 게임 오버 화면 출력을 추가</li> <li>• 게임 시작전 간단한 조작 설명</li> <li>• 점수 시스템을 추가하여 성취욕구를 자극시킴</li> <li>• 목숨[기회] 시스템을 추가하여 게임이 금방 끝나지 않게 해줌.</li> </ul> </li> <li>- 완성되지 않은 UI를 개선, 새로운 기능, UI를 추가한다 <ul style="list-style-type: none"> <li>• 현재 점수와 남은 기회(목숨)을 표시하는 UI추가</li> </ul> </li> </ul>

- 첫 화면에 게임에 대한 간단한 설명과 시작 버튼을 추가
- 게임을 이겼을시 이겼다는 텍스트와 점수를 출력하고 새로운 게임을 시작하는 버튼 추가
- 게임을 졌을시 졌다는 텍스트와 점수를 출력하고 재도전 하는 버튼을 추가

#### - 명확한 구현 기능 정의

- 3번 구현 항목에서 설명

## 2. 동작 시나리오 및 UI방안 제시

#### - 어떻게 동작하는지 설명

- 게임의 목표는 3x9줄로 생성된 벽돌을 움직이는 공을 이용해 모두 깨는 것이 승리 조건이다. 그렇기에 키보드의 화살표 버튼을 입력 받아 공을 충돌시킬 패들을 좌,우로 움직이고 공은 화면의 정 가운데 부분에서 움직이기 시작하며, 패들과 충돌하게되면 움직이는 반대방향으로 움직이게 된다.

#### - 어떤 UI가 만들어 지는지 설명

- 게임 시작시 첫 화면에 간단한 게임설명과 게임 시작 버튼 UI가 만들어짐

### 벽돌깨기 게임



키보드에서 왼쪽, 오른쪽 화살표를 누르면 패들이 해당 방향으로 움직입니다.

벽돌 하나를 깨때마다 점수 10점이 누적되며, 260점이 만점입니다.

시작버튼을 누르면 게임이 바로 시작됩니다.

게임 시작

- 좌측 상단 화면에는 얻은 점수와 남은 목숨(기회)를 표시하는 UI를 만듦

점수: 0

목숨: 3



- 남은 목숨을 모두 소모할때까지 벽돌을 모두 깨지 못했다면 게임에서 졌으므로, Game Over 텍스트를 띄우고, 지금까지 얻은 최고 점수를 표시한다. 그리고 재도전을 위해 재도전 버튼을 띄우는 UI를 만듦

Game Over

최고 점수: 0

재도전

- 벽돌을 모두 깼을시 승리하였으므로 WIN 텍스트를 띄우고 벽돌을 모두 깬 개수인 270점을 띄우며 새로운 게임을 시작할 수 있게 새로운 게임 버튼을 생성한다.



### 3. 구현

- 주요한 소스코드 제시 및 간단한 설명

```
def show_start_screen(self):
    self.canvas.create_text(self.width / 2, self.height / 2 - 70, text="벽돌깨기 게임", fill="black",
font=("Arial", 24), tags="start_screen")
    self.canvas.create_text(self.width / 2, self.height / 2, text="키보드에서 왼쪽, 오른쪽 화살표를 누르면 패들이 해당 방향으로 움직입니다.", fill="black", font=("Arial", 13), tags="start_screen")
    self.canvas.create_text(self.width / 2, self.height / 2 + 30, text="벽돌 하나를 깨 때마다 점수 10점이 누적되며, 260점이 만점입니다.", fill="black", font=("Arial", 13), tags="start_screen")
    self.canvas.create_text(self.width / 2, self.height / 2 + 70, text="시작버튼을 누르면 게임이 바로 시작됩니다.", fill="black", font=("Arial", 13), tags="start_screen")
    self.start_button = Button(self, text="게임 시작", command=self.start_game)
    self.start_button.place(x=self.width / 2 - 40, y=self.height / 2 + 90)
```

- Show\_start\_screen() 함수는 게임 시작시 게임에 대한 간단한 설명과 게임을 시작할 수 있는 버튼을 생성해주는 함수이다. 윈도우 위에 Canvas위젯을 생성하고, 게임 설명 부분은 canvas.create\_text를 이용하여 폰트와 크기, 스타일을 지정하고 마지막 부분에 tags를 start\_screen이라는 태그를 지정해준다. 이유는 start\_game()함수에서 호출하게 됨, 그리고 게임시작 버튼은 버튼 위젯을 이용하여 생성하게 된다. x,y값을 조절하여 가운데에 위치하게 해준다.

```
def start_game(self):
    self.start_button.destroy()
    self.canvas.delete("start_screen")
    self.start()
```

- Start\_game() 함수는 게임시작 시 클릭하는 게임 시작 버튼을 없애고, start\_screen 태그가 붙은 Canvas 위젯들을 제거한다. 여기서 게임 시작시 보였던 간단한 설명을 의미하고 있다. 그리고 마지막으로 start() 함수를 호출한다.

```
def start(self):
    if not self.is_game_over and not self.is_game_won:
        self.game_loop()
```

- Start() 함수는 본격적으로 게임을 시작시키는 함수이며, 게임 오버상태/게임 승리상태가 아니면 game\_loop() 함수를 호출하는 조건을 생성한다.

```
def update_ui(self):
    self.canvas.delete("ui")
    self.canvas.create_text(50, 20, text=f"점수: {self.score}", fill="black", tags="ui")
    self.canvas.create_text(150, 20, text=f"목숨: {self.lives}", fill="black", tags="ui")
    self.canvas.tag_raise("ui")
```

- Update\_ui() 함수는 ui태그가 달린 canvas 위젯을 제거하고[사실상 초기화를 의미], 점수와 목숨을 보여주는 text를 생성한다. 추가적으로 ui라는 태그를 붙인다.

```
def game_over(self):
    self.is_game_over = True
    self.canvas.create_text(self.width/2, self.height/2, text="Game Over", fill="red", font=(
"Arial", 24), tags="ui")
    self.canvas.create_text(self.width/2, self.height/2 + 30, text=f"최고 점수: {self.score}",
    fill="black", font=("Arial", 18), tags="ui")
    self.retry_button = Button(self, text="재도전", command=self.restart_game)
    self.retry_button.place(x=self.width/2 - 50, y=self.height/2 + 60)
```

- Game\_over() 함수는 이름에서도 알 수 있듯이, 게임에서 승리조건을 달성하지 못했을 때 활성화되는 함수이다. 게임에서 지게 되면 Game Over라는 텍스트가 띄워지고, 죽기전까지 얻은 최고점수를 띄우고, 코드 마지막에는 ui라는 태그를 붙여준다. 그리고 재도전을 할 수 있는 재도전 버튼을 버튼 위젯을 통해 만들어준다. 버튼을 누르면 restart\_game() 함수를 호출한다.

```
def game_won(self):
    self.is_game_won = True
    self.canvas.create_text(self.width / 2, self.height / 2, text="WIN!", fill="green", font=(
"Arial", 24), tags="ui")
    self.canvas.create_text(self.width / 2, self.height / 2 + 30, text=f"최고 점수: {self.score}",
    fill="black", font=("Arial", 18), tags="ui")
    self.retry_button = Button(self, text="새로운 게임", command=self.restart_game)
    self.retry_button.place(x=self.width/2 - 50, y=self.height/2 + 60)
```

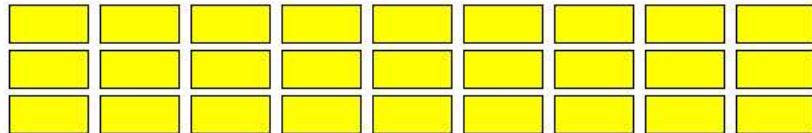
- Game\_won() 함수는 게임에서 이겼을 시 활성화되며, Win이라는 텍스트가 띄워지고, 지금까지 얻은 최고점수를 띄운다. 마찬가지로 마지막에 ui라는 태그를 넣어주며, 새로운 게임을 하기 위해 게임 진행 상황을 초기화 시켜주는 버튼 위젯을 넣어주었다. 버튼을 누르면 restart\_game() 함수를 호출한다.

```
def restart_game(self):
    self.retry_button.destroy()
    self.canvas.delete("all")
    self.shapes.clear()
    self.paddle = Paddle(self.canvas, self.width/2, 450)
    self.shapes[self.paddle.item] = self.paddle
    self.ball = Ball(self.canvas, 310, 200, 10)
    self.score = 0
    self.lives = 3
    self.create_bricks()
    self.update_ui()
    self.is_game_over = False
    self.is_game_won = False
```

- Restart\_game() 함수는 위에서 나왔듯이, 재도전 버튼과 새로운 게임 버튼을 누르면 진행중이던 게임이 초기화되고 새로운 게임을 시작하는 함수이다. 버튼을 눌러 호출당했을때 retry\_button과 Canvas 위젯을 모두 삭제하고, 게임오버되어 재시작하는 것에 대응해, 이전에 있던 벽돌과 공[도형들]을 제거해준다. 그런 다음 새로 패들과 공, 벽돌을 새로 생성하고, 점수와 목숨도 원래대로 복구시켜준다. 또 승리, 게임오버의 상태를 False로 비활성화 시킨다.

#### 4. 완성된 프로그램 동작 사진

점수: 0      목숨: 3



### 벽돌깨기 게임



키보드에서 왼쪽, 오른쪽 화살표를 누르면 패들이 해당 방향으로 움직입니다.

벽돌 하나를 깨 때마다 점수 10점이 누적되며, 260점이 만점입니다.

시작버튼을 누르면 게임이 바로 시작됩니다.

게임 시작



	<div><div>점수: 0      목숨: 1</div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div></div> <div><div>Game Over</div><div>최고 점수: 0</div><div>재도전</div></div> <div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div> <div><div>점수: 270      목숨: 3</div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div> <div><div>WIN!</div><div>최고 점수: 270</div><div>새로운 게임</div></div> <div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>
기대 효과 및 활용방안	<p>기계학습의 결과물은 앱을 통해서 사용자에게 보여지는데 파이썬 언어를 이용하여 앱을 만들고 UI 화면을 작성할 수 있는 코딩능력을 가지게 된다. 파이썬 언어는 UI를 간단하게 제작할 수 있을뿐 아니라 제공되는 풍부한 라이브러리를 이용해서 짧은 시간에 복잡한 기능들을 쉽게 구현할 수 있어서 인공지능의 구현 결과물을 쉽게 만들 수 있게 된다.</p>
<div>2024    년    6    월    14    일</div> <div>지도교수                      심    효    선                      (인)</div>	