2장 Manim 기본 예제들 살펴보기

Manim을 처음 실행을 하여 보도록 하자.

클래스와 메소드의 개념이 필요는 하지만 우선 실행을 하여 보도록 하자. 첫 번째 매개 변수로 클래스의 인스턴스가 넘어오게 됩니다. 이 첫 번째 매개 변수의 이름은 보통 관행적으로 self라고 하며, 인스턴스 메서드는 이 self를 통해 인스턴스 속성(attribute)에 접근하거나 다른 인스턴스 메서드를 호출할 수 있다. 뿐만 아니라 self를 통해, 클래스 속성에 접근하거나 클래스 메서드를 호출할 수도 있다.

우선 원 하나를 그려보려고 한다.

소스코드를 보자.

manim 패키지의 모든 모듈을 가져온다.

```
from manim import *
```

CreateCircle란 이름의 클래스를 정의하고 Scene에 상속한다. 왜 상속을 해야 할까? 보통 상속은 기존 클래스를 변경하지 않고 기능을 추가하거나 기존 기능을 변경하려고 할 때 사용한다. "클래스에 기능을 추가하고 싶으면 기존 클래스를 수정하면 되는데 왜 굳이 상속을 받아서 처리해야 하지?" 라는 의문이 들 수도 있다. 하지만 기존 클래스가 라이브러리 형태로 제공되거나 수정이 허용되지 않는 상황이라면 상속을 사용해야 한다.

```
class CreateCircle(Scene):
```

클래스 CreateCircle 안에 메소드 construct를 def 키워드를 사용하여 정의한다. 파이썬은 클래스의 메소드를 정의할 때 self를 명시한다. 메소드를 불러올 때 self는 자동으로 전달된다. self를 사용함으로 클래스 안에 정의한 멤버에 접근할 수 있게 된다. 또한 파이썬에서는 중괄호 { }을 사용하여 메소드를 묶지 않고 들여쓰기를 하여 이를 구분한다. 코드 블럭을 구성하기 위해 if, for, class, def 등등을 작성하면서 나오는 : 다음 아랫줄은 반드시 들여쓰기를 해야 한다. 들여쓰기의 방법은 한 칸, 두 칸, 네 칸, 탭 등 여러 가지 방식이 있다. 보통은 네 칸 또는 탭을 사용한다. 중요한 것은 같은 블록 내에서는 들여쓰기 칸 수가 같아야 한다. 위반시에는 "IndentationError: unexpected indent"라는 에러를 출력한다.

▌ def construct(self):

이제 manim 패키지에서 불러들인 Circle() 모듈을 이용하여 원을 그리려고 한다. 아래와 같이 두 탭 들여쓰기를 하여 작성하자.

▌ self.add(Circle())

Circle()의 () 안에 아무것도 코딩을 하지 않으면 기본값인 중심이 원점이고 반지름 1이며 테두리 색이 빨간색인 원을 그린다. 나중에 매개변수, 메소드, 속성 등에 대하여 논의하자. 지금은 그냥 넘어가자.

전체 코딩은 아래와 같다.

```
1  from manim import * #manim 패키지의 모든 모듈 가져오기.
2
3  class CreateCircle(Scene): #클래스 만들기
4      def construct(self):
5          self.add(Circle())
```

그림 1

그리고 아래에 있는 터미널 창에 ①과 같이

```
manim -p lecture_1.py
```

을 입력하고 엔터를 쳐 실행을 하자. 그러면 아래와 같이 실행이 된다.

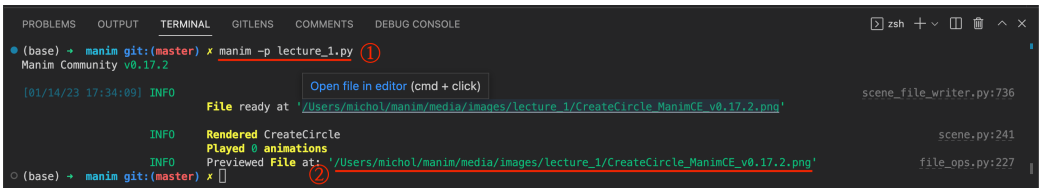

그림 2

②을 윈도우는 'Ctrl + 클릭', 맥은 'command + 클릭'을 하면 실행이 된다. 터미널 창에 이미 옵션 -p를 넣어 실행시켰으므로 자동 실행된다. 파일 위치도 확인할 수 있다. 위의 manim 예는 애니메이션이 없는 단지 원만 추가한 것이라 그림 파일인 png로 결과가 저장된다.

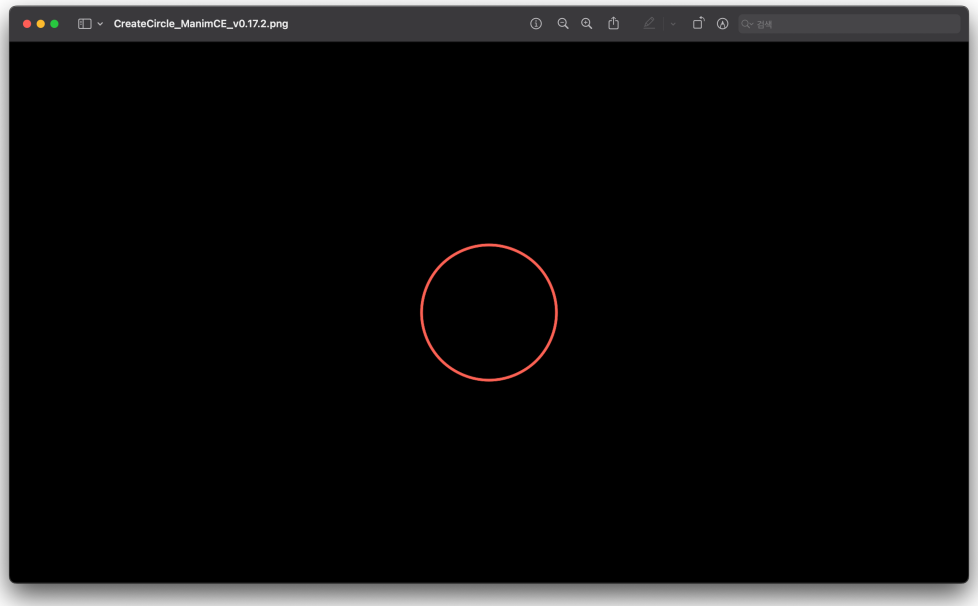

그림 3

컴퓨터 프로그래머들은 이러한 코딩을 그리 좋아 하지 않는다. cir이란 객체(instance)에 Circle() 모듈을 만든다. 여러 개의 같은 객체를 만들어 사용할 수도 있다.

```
    cir = Circle()
    self.add(cir)
```

전체 코드는 아래와 같다.

```
from manim import * #manim 패키지의 모든 모듈 가져오기.

class CreateCircle(Scene): #클래스 만들기
    def construct(self):
        cir = Circle()
        self.add(cir)
```

그림 4

위에서와 같이 터미널 창에서 실행을 시키면 같은 실행 결과를 얻는다.
이제 lecture_1.py에 또 다른 클래스 정사각형 클래스를 만들어 보자.

```
class CreateSquare(Scene):
    def construct(self):
```

- sqr =Square()
- self.add(sqr)

전체 코드는 다음과 같다.

```python
from manim import * #manim 패키지의 모든 모듈 가져오기.

class CreateCircle(Scene): #클래스 만들기
    def construct(self):
        cir = Circle()
        self.add(cir)

class CreateSquare(Scene):
    def construct(self):
        sqr = Square()
        self.add(sqr)
```

그림 5

그리고 아래의 터미널에서 실행을 하여 보자.

```
manim -p lecture_1.py
```

그러면 아래와 같이 선택의 메세지가 나온다.

그림 6

정사각형을 실행하려면 2번을 입력하고 엔터를 누른다. 정사각형이 실행되고

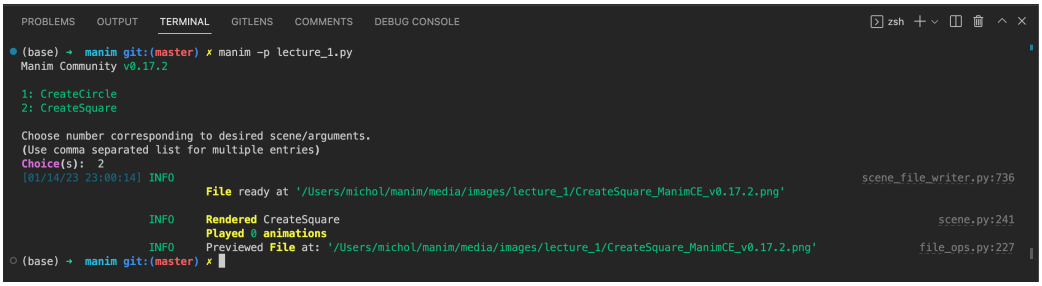

그림 7

그러면 정사각형의 모양이 그려진다.

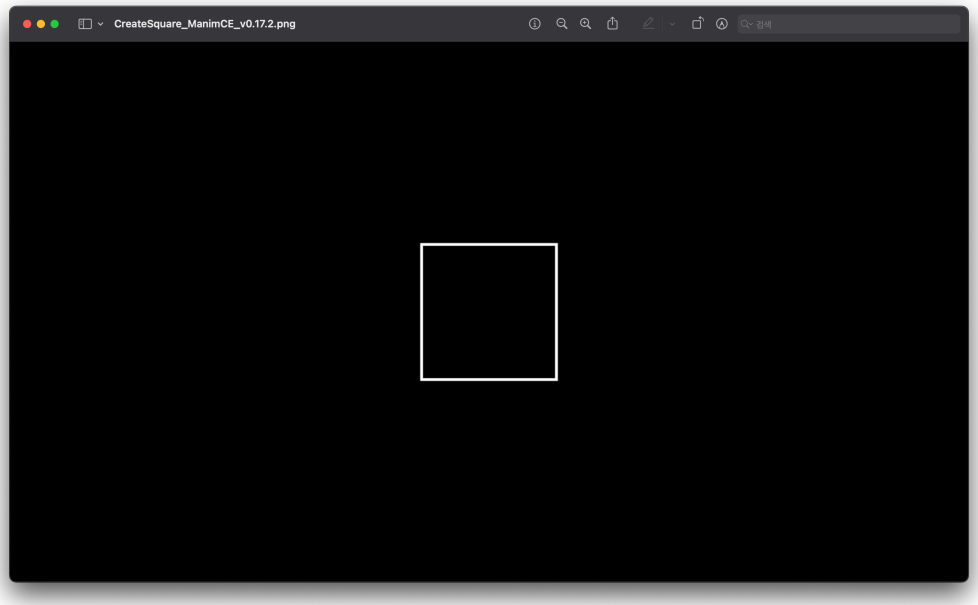

그림 8

둘 중 선택을 하지 않고 CreateCircle 클래스를 실행하려면 다음과 같이 입력하고 실행을 한다.

```
manim -p lecture_1.py CreateCircle
```

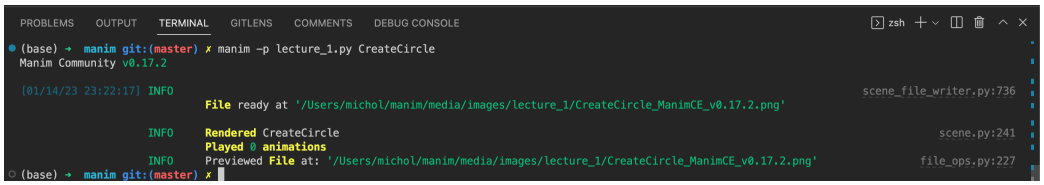

그림 9

CreateSquare 클래스를 실행하려면 다음과 같이 입력을 하고 실행을 한다.

```
manim -p lecture_1.py CreateSquare
```

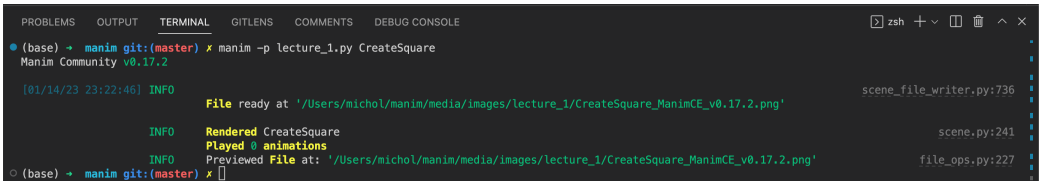

그림 10

이번에는 그림이 아니라 애니메이션을 만들어 보자. 원을 한 점에서 커져서 원이되는 애니메이션을 제작하여 보자.

위의 예제에서 다른 것은 그래도 하고 self.add(cir) 대신 아래와 같이 수정을 하자.

▌self.play(GrowFromCenter(cir))

그리고 manim -p lecture_1.py를 터미널 창에 입력을 하고 실행을 시키자.

그림 11

그러면 다음과 같은 애니메이션이 구현된다. 렌더링이 아무 오류 없이 실행되며 mp4 동영상의 위치를 볼 수 있다.

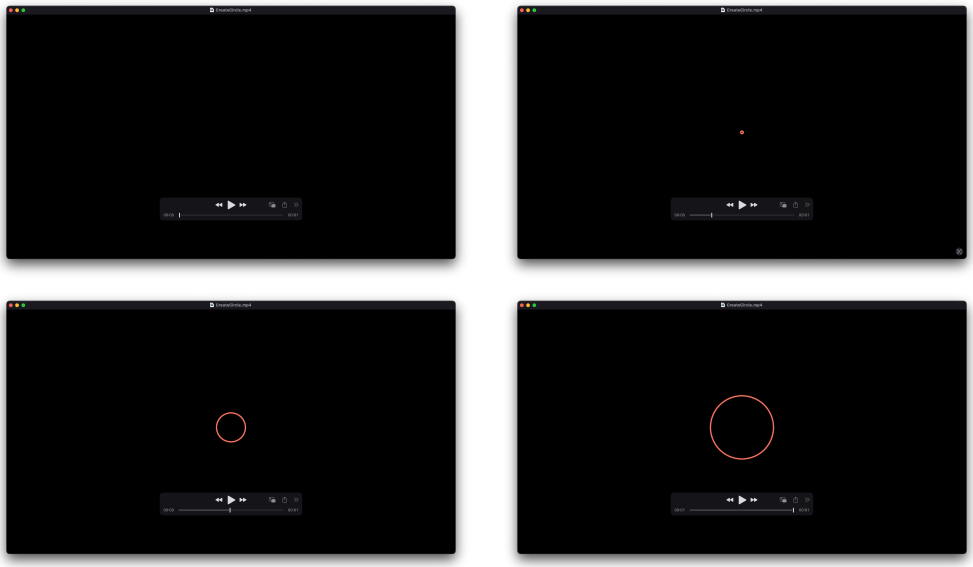

그림 12

Manim으로 수학 애니메이션을 구현하는 기초적인 방법을 익혔다. 물론 Tex과 sox의 나래이션 부분은 필요한 부분에서 다시 설명을 하겠다.