Chapter 14

공 클래스

1. 공 클래스 – 공의 위치

- 공 클래스의 코드를 다시 봅시다.

```
class Ball:
    def __init__(self):
        self.dx = 12
        self.dy = 0
        self.x = 475
        self.y = 275

    def move(self):
        self.x += self.dx
        self.y += self.dy

    def draw(self):
        screen.blit(ball_image, self.x, self.y))
```

1. 공 클래스 – 공의 위치

- 공 클래스의 기본 코드입니다.
- __init__() 함수는 공에게 위치와 속도를 줍니다.
- 공은 스크린 한 가운데에서 출발, 루프를 1번 돌 때마다 오른쪽으로 12픽셀씩 움직입니다.
- 하지만 공 이미지의 좌표는 가장 위 왼쪽의 좌표이므로, (475, 275)는 스크린 중심에서 25픽셀만큼 왼쪽으로, 또 25픽셀 만큼 위쪽으로 이동한 위치입니다.
- 스크린의 가로 길이는 1000, 세로 길이는 600이기 때문입니다.
- 공은 50x50픽셀이므로, 공의 중심은 이제 스크린의 중심과 같게 됩니다.

2. 공 클래스 – move(), draw() 함수

- move() 함수는 공을 움직입니다. 더 설명할 필요는 없겠죠?
- draw() 함수는 공 이미지를 화면에 표시합니다.
- dx와 dy는 뒤에서 계산하겠습니다.
- dx와 dy는 실수이므로 self.x와 self.y도 실수가 됩니다.
- 파이썬은 좌표에 실수를 쓰면 가끔 불평합니다.
- 항상 그렇진 않지만, 그래도 만일을 대비해서 int() 함수를 이용하여 실수를 정수로 반올림합시다.
- 예를 들어 int(3.4)라고 쓰면 파이썬은 3.4를 정수로 만들어 3을 돌려줍니다.
- 따라서 draw() 함수는 아래와 같이 고쳐야 합니다.

```
def draw(self):
    screen.blit(ball_image,(int(self.x), int(self.y)))
```

2. 공 클래스 – move(), draw() 함수

- 지금까지는 우리가 전에 이미 다 배운 것들입니다.
- int() 함수만 처음 나온 것이지요.
- 그래서 게임이 어딘가 심심합니다.
- 충돌 감지를 추가한 뒤 재미있게 바꿔 봅시다.

3. 첫 번째 충돌

- 충돌을 두 종류 만들어야 합니다.
- 첫 번째, 공이 스크린의 위 또는 아래와 충돌했을 때 튕기도록 해야 합니다.
- 두 번째, 공이 배트에 맞을 때도 튕겨 나와야 합니다.
- 첫 번째가 더 쉬우니 그것부터 하지요.
- 스크린 끝에 닿은 악당이 튕기도록 했던 것과 거의 똑같습니다.
- 공 클래스 안에 튕기기 bounce 함수를 만들겠습니다.

```
def bounce(self):
    if self.y <= 0 or self.y >= 550:
        self.dy *= -1
```

- 공이 스크린의 맨 위나 아래에 닿으면 dy의 부호를 바꿉니다.
- __init__()함수 안에서 self.dy를 12 또는 -12로 정한 다음 프로그램을 실행시켜 보면 확인할 수 있습니다.
- 그러려면 bounce 함수도 불러와야 하지요. 뒤에 나옵니다.

4. 두 번째 충돌

- 공이 배트에 닿았을 때도 튕기게 합시다.

```
def bounce(self):
    if self.y <= 0 or self.y >= 550:
        self.dy *= -1

    for bat in bats:
        if pygame.Rect(bat.x,bat.y,6,80).colliderect(self.x,self.y,50,50):
            self.dx *= -1
```

- for 루프는 배트 리스트 안에서 각각의 배트에 대해 if 문을 실행합니다.
- if 문에는 pygame.Rect().colliderect() 함수가 있습니다.
- 이 함수는 두 직사각형의 충돌을 감지합니다.
- pygame.Rect() 기억나지요?
- 이 함수로 보이지 않는 직사각형을 만들겠습니다.
- 공 이미지와 배트 이미지가 닿았는지 직접 확인할 수 없기 때문에, 공과 배트를 각각 보이지 않는 직사각형에 겹쳐놓고, 두 직사각형이 겹쳐졌는지 확인할 것입니다.

4. 두 번째 충돌

- Rect() 함수에 4개의 인자를 줍니다.
  - bat.x, bat.y, 6, 80은 순서대로 배트의 x좌표, 배트의 y좌표, 배트의 가로 길이, 배트의 세로 길이입니다.
- 이렇게 배트에 겹쳐진, 보이지 않는 직사각형을 만들었습니다.
- 이제 위 직사각형이 공 이미지에 겹쳐진, 보이지 않는 직사각형과 충돌했는지 물어봐야 합니다.
- 두 번째 직사각형은 colliderect() 함수에게 self.x, self.y, 50, 50을 줘서 만듭니다
  - 이 인자들은 공의 좌표와 크기입니다.
- pygame.Rect().colliderect() 함수가 충돌을 감지하면, if 문은 참값을 돌려줍니다
- 그러면 self.dx에 -1을 곱해 방향을 반대로 만듭니다.
- 배트와 부딪힌 공이 튕기는 거예요.

4. 두 번째 충돌

- 마지막으로, 게임 루프 안에 함수를 부르는 코드를 넣으세요.
- 공의 move() 함수나 draw() 함수를 부르는 곳에 쓰면 됩니다.

```
ball.move()
ball.draw()
ball.bounce()
```

4. 두 번째 충돌

- 공과 배트가 어느 방향으로 움직이는지는 중요하지 않습니다.
- Rect()는 공에 대한 정보를 가질 수 있고, colliderect()는 배트에 대한 정보를 가질 수 있습니다.
- 우리는 공 클래스 안에 있으므로, self는 항상 공을 의미합니다.
- 공과 배트의 인자를 바꿔도 된다는 뜻입니다.
- Rect()와 colliderect()에 2개의 인자만 써도 됩니다.
- x좌표, y좌표, 가로의 길이, 세로의 길이를 각각 쓰는 대신, 그냥 좌표와 크기만 써도 됩니다.
- 이렇게 써도 비슷해 보입니다.

4. 두 번째 충돌

- **colliderect() 함수를 예를 들어 봅시다.**

```
colliderect((self.x,self,y),(50,50))
```

- **colliderect() 함수의 두 인자는 모두 튜플입니다.**
- **별 의미 없는 것처럼 보이지만, 파이게임에는 객체의 크기를 구할 수 있는 get_size() 함수가 있습니다.**
- **따라서 다음과 같이 ball_image.get_size()라고 써도 됩니다.**
- **그러면 (50, 50)이라는 튜플을 돌려줄 것입니다.**
- **따라서, 위 코드를 다음과 같이 바꿔도 됩니다.**

```
colliderect((self.x,self,y),bail_image.get_size())
```

4. 두 번째 충돌


```
collideрect((self.x,self,y),bail_image.get_size())
```

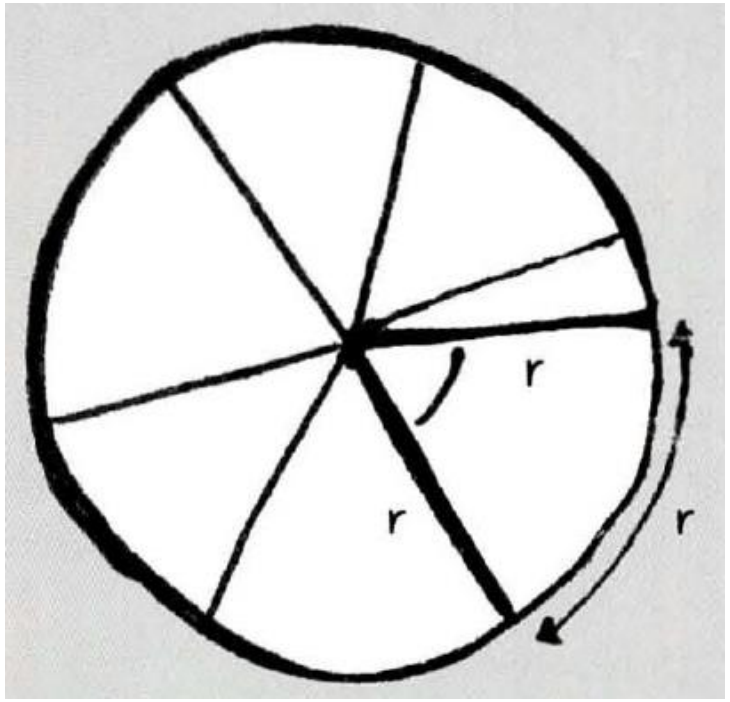
- 이렇게 쓰면 공의 크기를 모를 때 굳이 찾아보지 않아도 됩니다.
- 공의 크기를 바꿨을 때 함수를 수정할 필요도 없지요.
- 다음에는 프로그램이 실행되는 동안 객체의 크기가 바뀌는 게임을 만듭니다.
- 그럴 때 get_size()나 기타 유사한 함수들을 매우 유용하게 사용할 수 있습니다.
- 하지만 배트의 크기는 이 방법으로 구할 수 없습니다.
- 배트에는 연결된 객체가 없기 때문입니다.
- 배트는 이미지가 아니라 파이게임한테 직선을 계속 그리라고 시켜서 만드는 거거든요.

5. 라디안

- 게임을 재미있게 만들기 위해 처음으로 할 일은, 시작할 때 공이 임의의 각도에서 나오게 하는 것입니다.
- 이건 앞에서 해 보지 않은 거니까 자세히 설명하겠습니다.
- 프로그래머들은 각도를 다룰 때 도(°)로 표시하지 않고 라디안(radian)으로 표시합니다.
- 수학자들이 라디안을 쓰기 때문이죠.
- 사실 프로그래밍은 수학의 한 분야라고 할 수 있습니다.
- 삼각형에서 각도를 측정할 때나 빌딩 숲 사이에서 어떤 방향으로 가야 할지 알고 싶을 때는 60분법을 사용하는 게 좋지만 조금 더 멀리, 수학의 고지대(highlands) 안으로 들어가려면, 라디안이 더 편리합니다.

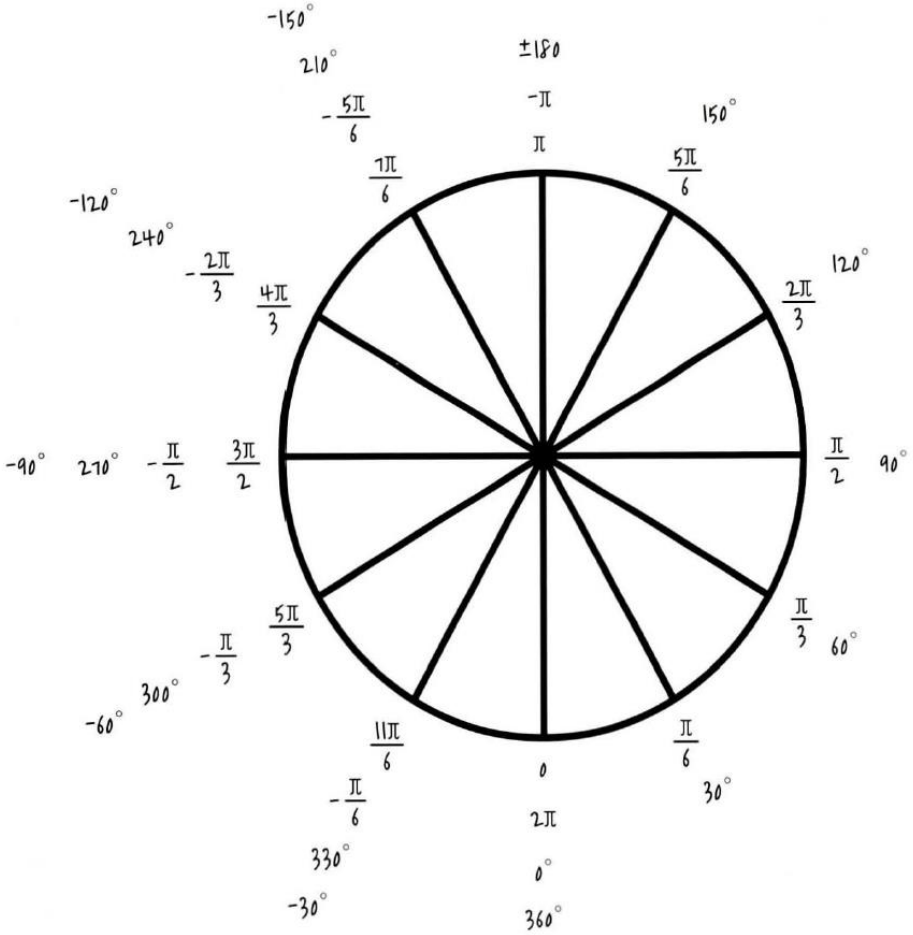
5. 라디안

- 라디안은 각의 단위입니다.
- 온도를 섭씨나 화씨로 표현하는 것처럼 각도는 60분법이나 라디안으로 표시할 수 있습니다
- 호의 길이가 반지름의 길이와 같을 때, 그 부채꼴의 중심각이 1 라디안입니다.
  - 약 57.3°입니다.
- 원의 둘레는 2π입니다.
- 따라서 한 원의 중심각은 2π 라디안이 됩니다.
- 즉, 2π 라디안은 360°와 같습니다.
- 따라서 180°는 π라디안과 같습니다.



5. 라디안

- 각 구역은 π/6만큼 증가합니다.
- 0에서 시작해 1/6π, 2/6π, 3/6π 이런 식으로 쓸 수도 있습니다.
- 아래 그림의 숫자들은 기약분수로 쓴 것 뿐이랍니다.



6. 임의의 방향에서 나오는 공

- 공 클래스 안의 __init__() 함수를 다음과 같이 고칩니다.

```
def __init__(self):
    d = (math.pi/3)*random.random()+(math.pi/3)+math.pi*random.randint(0,1)
    self.dx = 12 math.sin(d)*12
    self.dy = 0 math.cos(d)*12
    self.x = 475
    self.y = 275
```

- 파이썬의 math와 random 모듈을 써야 하므로, 프로그램의 1번째 줄에서 이 모듈들을 가져옵니다.

```
import pygame, sys, math, random
```

6. 임의의 방향에서 나오는 공

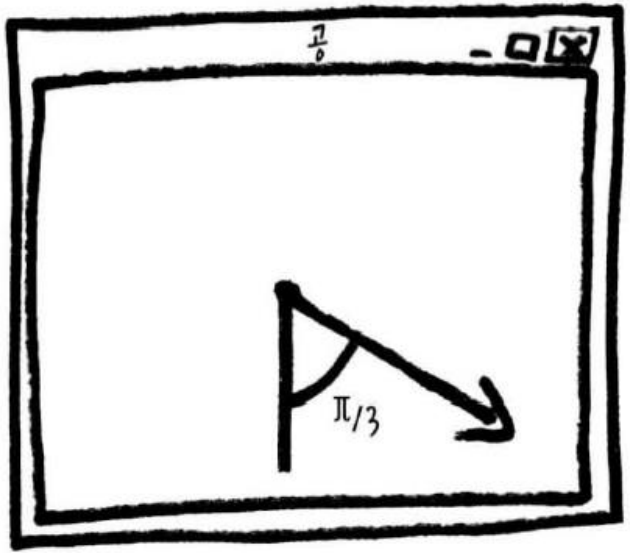
- 실행시키면 임의의 방향으로 향하는 공을 볼 수 있습니다.
- 왜 그런지 하나하나 살펴볼까요? “d"가 들어 있는 코드를 봅시다.

```
d = (math.pi/3)*random.random() + (math.pi/3) + math.pi*random.randint(0,1)
```

- d는 방향(direction)을 의미합니다.
- 라디안으로 표시한 각도입니다.
- __init__() 함수 밖에서는 쓰지 않을 거니까 “self”는 필요 없습니다.
- 파이썬의 math 모듈에서만 pi가 정의돼 있기 때문에, 그냥 pi라고 쓰면 안 되고 math.pi라고 써야 합니다.
- 이런 이유로 1번째 줄에서 math 모듈을 가져온 거랍니다.
- 변수 d는 시작할 때 공이 나오는 각도입니다.
- 스크린의 위나 아래보다, 왼쪽이나 오른쪽을 향하게 하는 것이 좋습니다.

6. 임의의 방향에서 나오는 공

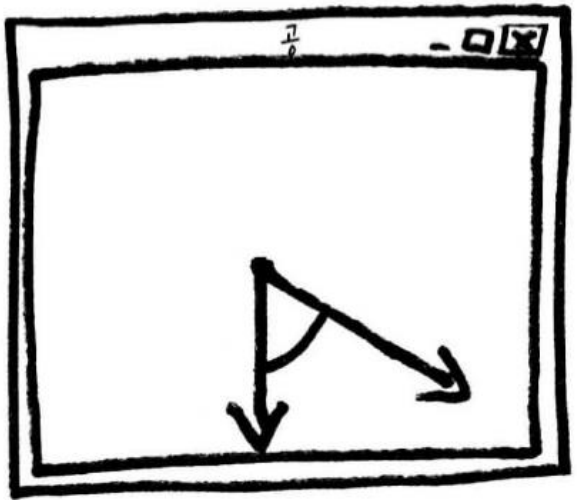
- d의 값에 따른, 공이 나오는 각도를 생각해 봅시다.
- math.pi/3이라고 쓰면 π/3라디안, 즉 60도 방향에서 나옵니다.
- x축은 오른쪽으로 가는 방향이 양수이고 y축은 내려가는 방향이 양수라고 했던 것 기억나나요?
- 따라서 크기가 0인 각은 아래로 수직 방향을 가리키고, 시계 반대 방향으로 갈수록 크기가 커집니다.
- 즉, 크기가 π/3라디안인 각 d는 아래 그림과 같습니다.



6. 임의의 방향에서 나오는 공


```
(math.pi/3)*random.random()
```

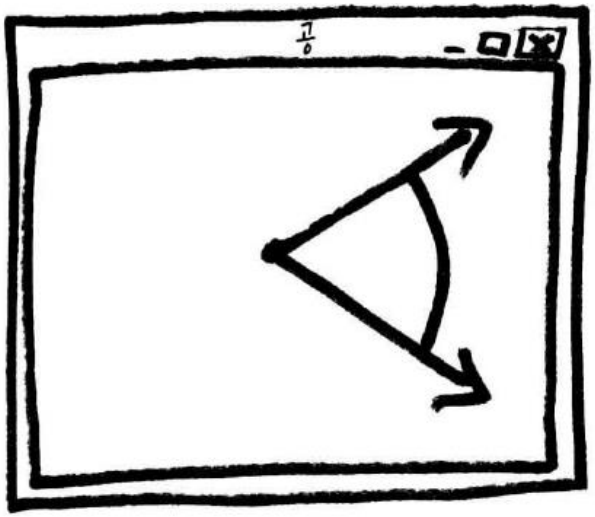
- random.random() 함수는 0과 1 사이의 임의의 수를 줍니다.
- 예를 들어 0.4572943478340743 같은 수를 얻게 되지요.
- 따라서 math.pi/3*random.random()은 0과 π/3 라디안 사이의 각을 줄 것입니다.
- 0과 60도 사이의 각이지요.
- 따라서 math.pi/3에 random.random()을 곱하면 아래 그림과 같은 각이 됩니다



6. 임의의 방향에서 나오는 공


```
(math.pi/3)*random.random()+(math.pi/3)
```

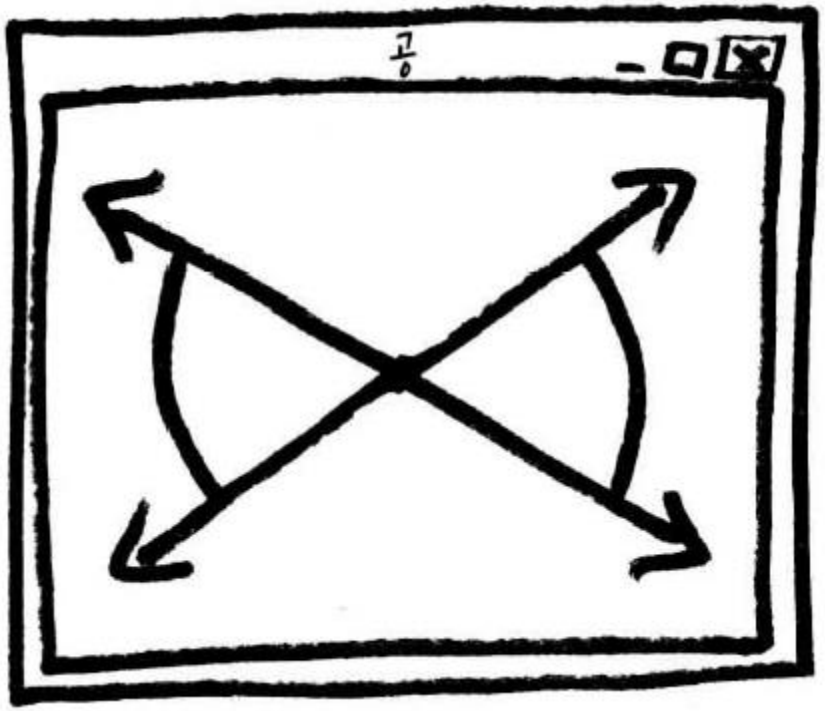
- 그 다음 (math.pi/3)을 더한다고 했으므로, 다시 π/3라디안, 즉 60도를 더하면 그림과 같이 바뀔 것입니다.



6. 임의의 방향에서 나오는 공


```
(math.pi/3)*random.random()+(math.pi/3)+math.pi*random.choice(0,1)
```

- random.choice() 함수는 임의의 정수를 골라 줍니다.
- 전에 악당의 속도를 임의로 바꾸기 위해서 사용했지요.
- 여기서는 0과 1 사이에서 고르니까 0 또는 1을 돌려주겠네요.
- 거기에 math.pi를 곱하면 0 x π 또는 1 × π 둘 중 하나가 됩니다.
- 0 또는 π 가 나오겠군요. 그것을 더합니다.
- 0도 또는 180도를 더하는 것과 같아요.
- 이제 각 d는 그림과 같이 바뀝니다.

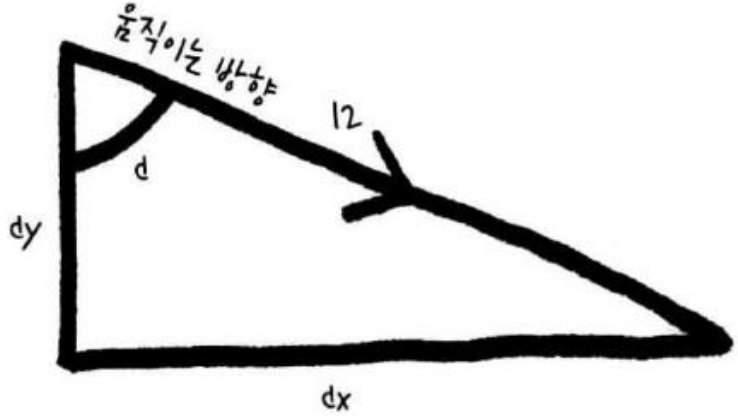


6. 임의의 방향에서 나오는 공

- 이제 방향을 정했으니, 공을 그 방향으로 12픽셀만큼 움직이도록 해야 하는데, 파이썬은 물체를 x방향이나 y방향으로만 이동시킵니다.
- 다른 방향으로 움직이게 하려면 다음 두 가지를 섞어 써야 합니다.

```
self.dx=math.sin(d)*12
self.dy=math.cos(d)*12
```

- 각 d를 알면 우리는 삼각비를 사용하여 x와 y의 값을 바꿀 수 있습니다.
- 그러니까 dx와 dy의 값을 구해야 합니다.
- dx는 대변의 길이, dy는 밑변의 길이입니다.



6. 임의의 방향에서 나오는 공

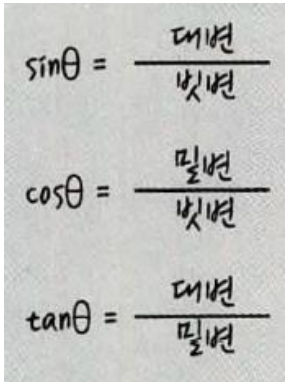
- 우리는 각 d의 값을 알고 있습니다. 직접 정했으니까요.
- 어떻게 하면 매 루프를 돌 때마다 공이 빗변의 길이 (12픽셀)만큼 움직이도록 할 수 있을까요?
- 참고로 여기서 0은 수직 방향입니다. (다음 쪽의 설명에서는 0이 오른쪽 방향이지요.)

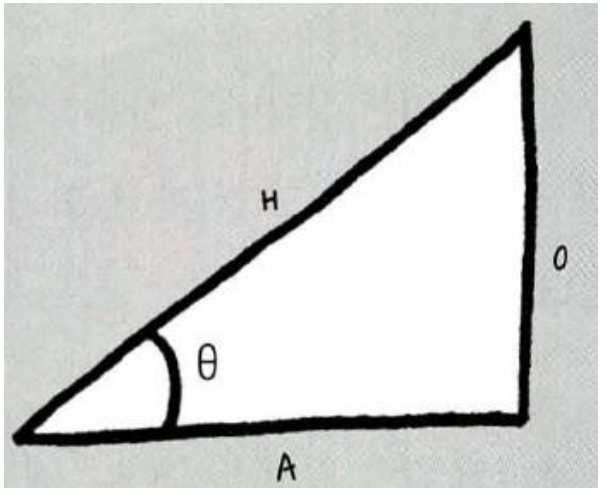
```
dx = sin d x 12
dy = cos d x 12
```

- 삼각비를 이용하면 위와 같은 식을 얻습니다.
- 수학적으로 dx와 dy를 구하면 완전히 임의는 아니지만, 어느 정도 임의적 요소를 가지고 있도록 각을 정할 수 있습니다. (삼각비에 대한 설명은 다음 쪽을 보세요.)

7. 삼각비

- 이미 배운 사람도 있고 아직 안 배운 사람도 있겠지만, 삼각비(sohcahtoa)는 다음 내용을 기억하기 위한 방법입니다
- 삼각비는 영어로 trigonometry 입니다
- sohcahtoa는 영어권 학생들이 삼각비를 쉽게 외우기 위해 쓰는 축약법입니다
- 우리나라에서는 s, c, t를 그림으로 그려서 외우지요





7. 삼각비

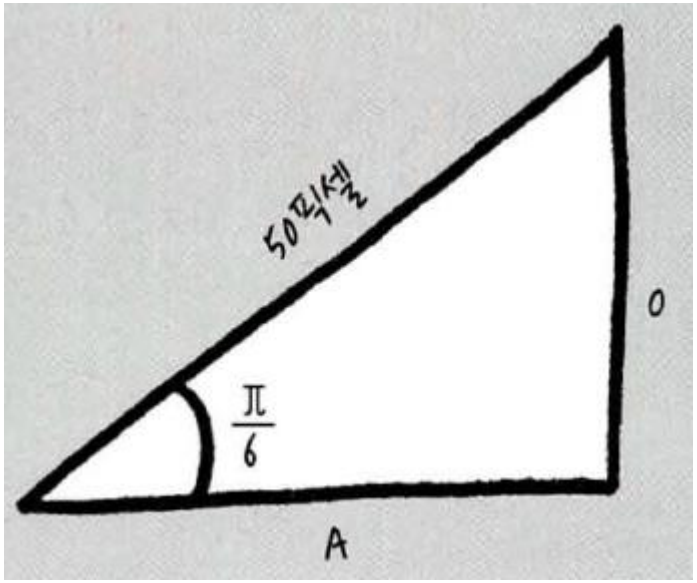
- 한 직각삼각형에 대해, 각 θ와 빗변(Hypotenuse)의 길이 H를 알면, 대변(Opposite)의 길이 O와 밑변(Adjacent)의 길이 A를 구할 수 있습니다

$$\sin\theta = \frac{O}{H} \qquad \cos\theta = \frac{A}{H}$$

$$\Rightarrow O = \sin\theta \times H \qquad \Rightarrow A = \cos\theta \times H$$

$$\Rightarrow O = \sin\frac{\pi}{6} \times 50 \qquad \Rightarrow A = \cos\frac{\pi}{6} \times 50$$

$$\Rightarrow O = 25 \qquad \Rightarrow A = 43.3$$



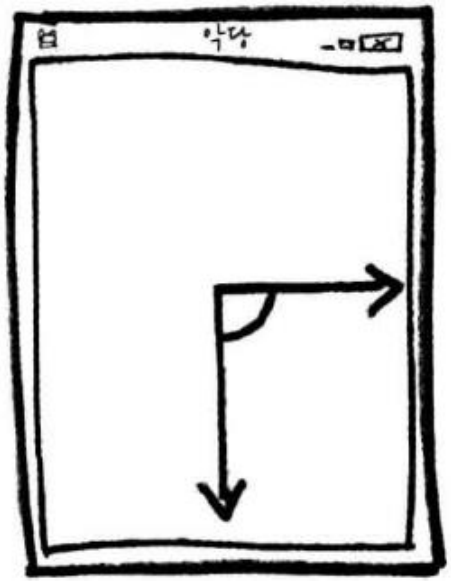
8. 임의의 방향에서 나오는 악당

- 방금 배운 것, 무언가를 임의의 방향에서 나오게 하는 코드는 자주 사용됩니다.
- 아래 코드는 복사하여 다른 프로그램에 붙여 넣어도 됩니다.
- 우주 침략자들을 만들 때 사용할 수도 있겠지요.
- 예를 들어, 이전 게임에서 만들었던 악당 클래스의 __init__() 함수를 다음과 같이 써도 됩니다.

```
class Badguy:
(중략)
    def __init__(self):
        self.x = random.randint(0,570)
        self.y = -100
        d=(math.pi/2)*random.random()-(math.pi/4)
        speed = random.randint(2,6)
        self.dx=math.sin(d)*speed
        self.dy=math.cos(d)*speed
```

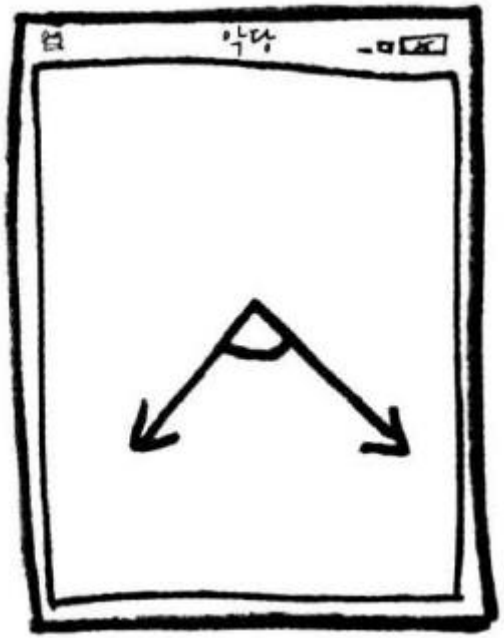
8. 임의의 방향에서 나오는 악당

- 이렇게 쓰려면 1번째 줄에서 모듈을 가져올 때 math 모듈도 추가해야 합니다.
- 실행하면 우리가 했던 것과 별 차이 없이 작동합니다.
- (math.pi/2)*random.random()이라고 쓰면 우리가 원하는 각을 그릴 수 있습니다. (pi/2라디안은 90°입니다.)



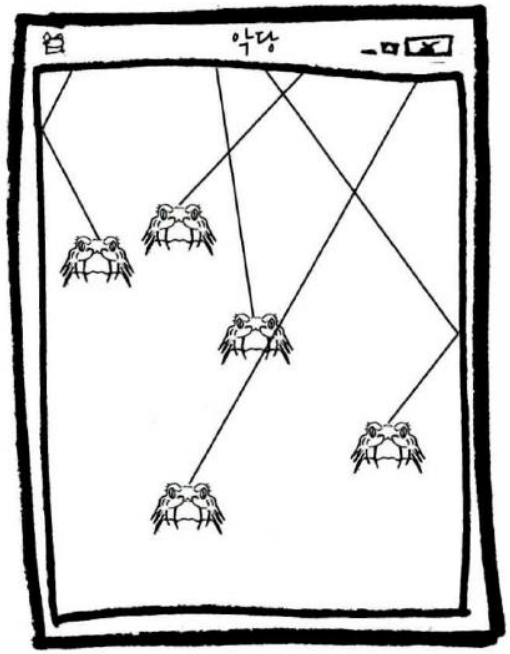
8. 임의의 방향에서 나오는 악당

- -(math.pi/4)라고 써서 각이 아래를 향하게 돌립니다.
- 주어진 각에서 pi/4 라디안만큼 빼면 각을 시계 방향으로 돌릴 수 있습니다.



8. 임의의 방향에서 나오는 악당

- 이제 속도를 임의의 값으로 정하고 삼각비를 사용하여 dx와 dy를 구합니다.
- 시작점을 스크린 위에서 100픽셀 아래, x축으로는 0부터 570 사이의 임의의 수로 정했습니다.
- 이제 move()와 bounce() 함수를 불러서 악당이 임의의 각도로 움직이면서 내려오도록 하면 됩니다.



Thank You