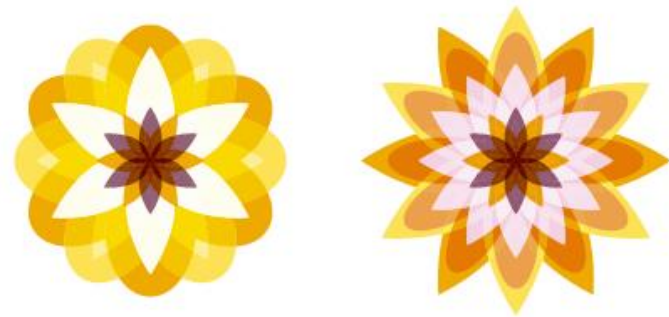


Chapter 06

이미지 예술



1. 이미지

- 마이크를 스크린에 등장시킵시다. 이미지부터 준비해 볼까요?
- 이미지를 웹사이트 (https://github.com/jjin300/start_python/tree/images)에서 다운받으세요.
- 파일 이름은 Mike_umbrella.png입니다.
- 포토그래픽 이미지는 JPEG를 쓰는 게 좋지만 이런 이미지들은 PNG를 쓰는 것이 좋습니다.
- 이 이미지는 크기가 가로 170픽셀, 세로 192픽셀입니다.
- 반드시 170픽셀, 192픽셀일 필요는 없지만 비슷하긴 해야 합니다.
- 이 이미지는 사실 배경이 흰색인 직사각형입니다.
- 검은색 박스 안에 넣으면 오른쪽처럼 보일 것입니다.
- 파이썬 사용 이미지의 배경은 흰색인 것이 좋습니다.
- `screen.fill()` 함수를 쓰면 배경이 흰색으로 채워지거든요.
- 배경이 검정색이라면 흰색 직사각형 모양이 눈에 띄겠지요.



1. 이미지

- 파이썬 파일이 저장된 폴더 안에 images라는 새로운 폴더를 만들고, Mike_umbrella.png를 넣으세요.
- 앞으로 만들 게임에서 사용될 모든 이미지가 images 폴더에 저장될 거예요.
- 게임마다 새로 폴더를 만들고 싶다면 새 게임 파일 폴더 안에도 images 폴더를 만들어야겠지요.
- 만약 폴더 이름을 images라고 하지 않을 거라면, 코드에서 폴더 이름을 찾아 바꾼 이름으로 수정해야 합니다.



2. 마이크 이미지 불러오기

- 먼저 마이크 이미지를 게임에 불러와야 합니다.
- 이 일은 앞에서 빗방울을 만드는 코드를 썼던, 프로그램의 셋업에서 해야 합니다.
- 셋업 가장 아래 다음 코드를 추가하세요.

```
mike_umbrella_image = pygame.image.load("images/Mike_umbrella.png").convert()
```

- `mike_umbrella_image`라는 이름의 객체를 만드는 코드입니다.
- `mike_umbrella image`는 변수기 때문에 뭐라고 이름 붙여도 상관없지만, 그래도 가장 직관적인 이름을 붙이는 게 좋지요. 일단 프로그램이 이미지에 붙이는 이름이라고 생각하세요.
- `pygame.image.load()` 함수는 이미지를 파이게임으로 가져오고, `convert()` 함수는 이미지를 컴퓨터의 그래픽 카드가 읽을 수 있는 포맷으로 바꿔 줍니다.
- 둘 다 이미지를 불러올 때마다 자동으로 일어나는 일인데, 셋업에 써 두면 파이썬이 이미지를 불러올 때마다 쓰는 시간과 노력을 줄일 수 있습니다.

3. 마이크 클래스

- 이제 마이크 클래스를 만듭시다.
- 앞에서 다룬 빗방울 클래스 기억나나요?
- 빗방울 클래스를 만든 이유는 최소한의 코드로 모든 빗방울을 만들고 제어할 수 있기 때문이었습니다.
- 마이크는 하나뿐이지만 마이크 클래스도 만드는 것이 좋습니다.
- 마이크와 관련된 모든 것을 한 곳에 모아놓을 수 있으니까요.
- 아래 코드는 빗방울 클래스 위나 아래에 쓰세요.

```
class Mike:
    def __init__(self):
        self.x = 300
        self.y = 400
    def draw(self):
        screen.blit(mike_umbrella_image, (self.x, self.y))
```

3. 마이크 클래스

- 마이크 클래스는 매우 간단합니다.
- `__init__()`함수로 `self.x`와 `self.y`를 정의합니다.
- `Draw()`함수는 `screen.blit()` 함수를 사용해 이미지를 스크린에 전송(`blit`)합니다.
- `screen.blit()` 함수는 전송할 객체와 전송될 좌표, 2개의 인자를 갖습니다.
- 따라서 여기에서는 `mike_umbrella_image`를 스크린 위의 (`self.x`, `self.y`), 즉 (300, 400)에 보여 줍니다.
 - 이미지를 화면에 '전송한다`blit`'는 것은 이미지를 화면에 표시한다는 뜻입니다.
 - 앞에서는 파이게임 안의 모양 그리기(원 그리기, 직사각형 그리기) 함수들을 사용해서 이미지를 그렸지요.
 - 직접 이미지 파일을 불러와 화면에 표시하지는 않았습니다.
 - 하지만 지금처럼 진짜 이미지를 사용할 때는 반드시 이미지를 전송해야 합니다.
 - 텍스트도 전송할 수 있습니다.

3. 마이크 클래스

- 자 이제 마이크 클래스 만들기가 끝났습니다.
- 아직 마이크 인스턴스는 안 만들었지만요.
- 빗방울의 경우 모든 인스턴스를 저장할 리스트를 만들었지만, 지금은 인스턴스가 1개이므로 리스트를 만들 필요는 없습니다.
- 따라서 빗방울 리스트 바로 밑에 아래처럼 씁니다.

```
raindrops = []  
mike = Mike()
```

- 마이크 클래스Mike() 의 유일한 인스턴스인 마이크mike를 만드는 코드입니다.

3. 마이크 클래스

- 아직 게임 루프에서 draw() 함수를 불러오지 않았기 때문에 마이크는 스크린에 보이지 않을 것입니다.
- draw() 함수가 이미지를 표시해야 합니다.
- 아래 코드를 screen.fill() 바로 밑에 쓰세요.
- 마이크는 스크린의 앞, 빗방울의 뒤에 그려져야 하니까요.
- 들여쓰기는 1번만 합니다.

```
screen.fill((255,255,255))  
mike.draw()
```

- 이제 우산을 쓴 마이크가 스크린에 나타날 것입니다.
- 앗, 그런데 비가 마이크의 우산을 그냥 통과하네요!

3. 마이크 클래스

rain



3. 마이크 클래스



4. 충돌감지

- 마이크 클래스에 함수를 하나 추가해 봅시다.

```
class Mike:  
(중략)  
    def hit_by(self, raindrop):  
        return pygame.Rect(self.x, self.y, 170, 192).collidepoint((raindrop.x, raindrop.y))
```

- 먼저 hit_by() 함수를 살펴보겠습니다.
- hit_by() 함수에는 self 말고 또 다른 인자가 있습니다.
- 바로 빗방울 raindrop입니다.
- while 루프가 돌아가면서 하나의 빗방울을 떨어뜨리면, 이 빗방울이 hit_by() 함수의 인자로 주어집니다.

4. 충돌감지

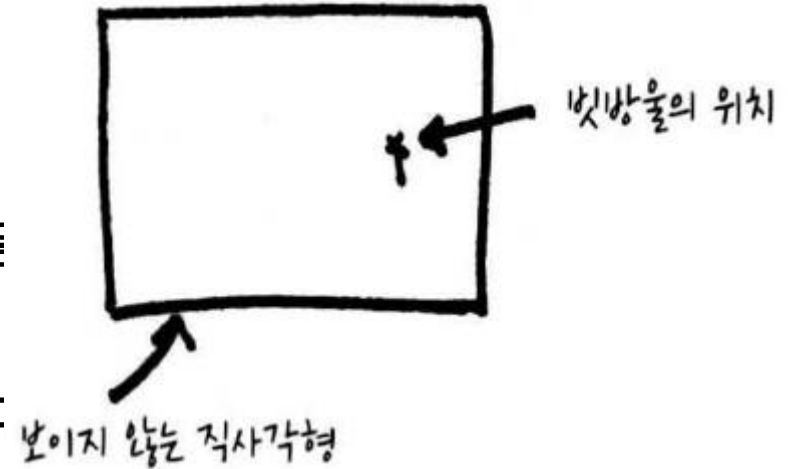
- 마이크 클래스에 함수를 하나 추가해 봅시다.

```
class Mike:  
(중략)  
    def hit_by(self, raindrop):  
        return pygame.Rect(self.x, self.y, 170, 192).collidepoint((raindrop.x, raindrop.y))
```

- `pygame.Rect()`는 직사각형의 좌표를 저장하는 클래스입니다.
- 실제로 직사각형을 그리는 것이 아니라 스크린의 특정 위치에 보이지 않는 직사각형을 만듭니다.
- 보이지 않는 직사각형에 어떤 점이 닿았는지 안 닿았는지 여부를 `collidepoint()` 함수가 감지합니다.
- `collidepoint()` 함수는 `Rect` 클래스에 들어 있는 함수입니다. (이해가 잘 안 돼도 괜찮습니다.)

4. 충돌감지

- 보이지 않는 직사각형 Rect을 마이크 이미지
- 이 일은 마이크 클래스 안에서 해야 합니다.
- 마이크 이미지의 좌표인 `self.x`와 `self.y`를 사용
- 마이크 이미지와 같은 가로 170, 세로 192의
미지의 바로 앞에 가공의 직사각형을 만듭니다
- `collidepoint()`는 튜플tuple로 주어지는 좌표 한 쌍을 인자로 갖습니다.
- 좌표는 특정 빗방울의 x좌표와 y좌표입니다.
- 따라서 보이지 않는 직사각형 위에 특정 빗방울이 겹치는지 아닌지를 확인할 수
있습니다.
- 그 빗방울의 x, y좌표가 `Rect()`로 만든 직사각형과 겹치는지에 따라 `hit_by()` 함
수가 참 또는 거짓을 돌려줍니다.
- 만약 참이라면 해당 빗방울을 빗방울들 리스트에서 지워야 합니다.



4. 충돌감지

```
While 1:
(중략)
    i=0
    while i < len(raindrops):
        raindrops[i].move()
        raindrops[i].draw()
        if raindrops[i].off_screen() or mike.hit_by(raindrops[i]):
            del raindrops[i]
            i -= 1
    i += 1
```

- 빗방울 지우기 코드에 마이크 관련 내용을 넣었습니다.
- 이제 단순히 스크린에서 빗방울이 사라졌는지 뿐만 아니라 마이크와 충돌했는지 까지 확인합니다.
- 만약 둘 중에 하나라도 참이라면 빗방울에게 작별을 고해야겠지요.

4. 충돌감지



5. 리스트와 튜플

- 지금까지 리스트를 많이 써 봤습니다 사실 리스트는 쉽게 만들 수 있습니다.

```
raindrops = []
```

- 라고 쓰기만 하면 빗방울들raindrops이라는 리스트가 만들어지지요.
- 그 다음 append() 함수를 사용해 쉽게 이것저것 추가할 수 있었습니다.
- del을 사용해 리스트에서 삭제할 수도 있고요.
- 앞에서 튜플에 대해 이야기한 것을 기억하나요?
- 색깔은 항상 튜플로 표시합니다.
- 튜플은 기본적으로 내용을 바꿀 수 없는 리스트입니다.
- 추가나 삭제할 수 없습니다.
- 색깔 튜플에 뭘 더 집어넣겠다는 것 자체가 말이 안 되지요.

5. 리스트와 튜플

- 튜플을 만드는 방법은 리스트를 만드는 법과 같습니다.
- 대괄호[] 대신 소괄호()를 쓴다는 점만 빼고요.
- 또한 튜플에서는 소괄호 사이에 쉼표를 최소한 1개 이상 써야 합니다. 비어 있지 않은 한 말이지요.
- 따라서 항목이 1 개 있는 튜플은 다음과 같이 씁니다.

```
tupleA=(32,)
```
- 만약 tupleA=(32)라고 쓰면 파이썬은 이것 튜플이 아니라 뭔가 수학적인 거라고 오해합니다.

6. 구름 띄우기

- 구름을 하나 만들어 봅시다.
- 먼저 이미지가 필요합니다.
- 아까처럼 이미지는 직접 그리거나 웹사이트에서 다운받아야 합니다.
- 파일 이름을 cloud.png로 정하겠습니다.
- 프로그램에 이미지를 불러오기 위해 다음 코드를 프로그램의 셋업에 씁니다.

```
cloud_image = pygame.image.load("images/cloud.png").convert()
```

6. 구름 띄우기

- 이제 구름 클래스를 만듭니다.
- 빗방울이나 마이크 클래스 위 또는 아래에 쓰세요.

```
class Cloud:  
    def __init__(self):  
        self.x = 300  
        self.y = 50  
  
    def draw(self):  
        screen.blit(cloud_image, (self.x, self.y))
```

- `__init__()` 함수에서 구름의 X, Y좌표를 정합니다.
- `Draw()` 함수에서 구름 이미지를 전송합니다.

6. 구름 띄우기

- 이제 구름 클래스의 인스턴스를 만듭니다.
- 마이크 클래스의 인스턴스처럼 만들면 됩니다.
- `mike = Mike()` 밑에 쓰세요. 이번에도 들여쓰기는 하지 않습니다.

```
mike = Mike()  
cloud = Cloud()
```

- 마지막으로 구름의 `draw()` 함수를 불러옵니다.
- 이 코드는 게임 루프 안에서 마이크의 `draw()` 함수를 불러오는 코드 바로 밑에 씁니다.

```
mike.draw()  
cloud.draw()
```

- 손쉽게 구름 하나를 만들었습니다.

7. 구름에서 떨어지는 빗방울

- 이제 비슷한 구름들을 더 만들고 구름에서 빗방울이 떨어지게 해 봅시다.
- 그런데 지금은 게임루프 안에서 빗방울들을 만듭니다.
- 따라서 `screen.fill()` 함수 바로 위의 아래 코드는 삭제하든지 `#`을 달아서 실행되지 않게 해야 합니다.

```
raindrops.append(Raindrop())
```

- 빗방울이 빗방울들 리스트에 추가되면, 빗방울 클래스의 `__init__()` 함수는 빗방울을 아무 위치에나 놓습니다.
- 이 두 가지 일을 모두 구름 클래스 안으로 집어넣읍시다.
- 구름 클래스 안에 함수를 하나 만들고, 이 함수가 새로운 빗방울을 만든 뒤 빗방울을 놓을 위치를 알아서 정하게 합니다.
- 그 다음 빗방울 클래스에게 해당 위치를 알려 줍니다.

7. 구름에서 떨어지는 빗방울

- 따라서 빗방울 클래스의 `__init__()` 함수를 다음과 같이 수정합니다.

```
class Raindrop:
    def __init__(self, x, y):
        self.x = random.randint(0, 1000) x
        self.y = -5 y
        self.speed = random.randint(5, 18)
```

- `__init__()` 함수에게 2개의 인자 `x`, `y`를 줍니다.
- 빗방울이 만들어지면 이 새로운 값(`x`와 `y`)을 줍니다.
- 그다음에 `self.x`를 `x`와 같게 만들고, `self.y`를 `y`와 같게 만듭니다.
- 이런 방법으로 다른 곳에서 만든 좌표를 빗방울에게 줄 수 있습니다.

7. 구름에서 떨어지는 빗방울

- 이제 구름 클래스에 다음 함수를 추가합니다.
- 그럼 빗방울도 만들 수 있고, 위치도 정해집니다.
 - ‘빗방울 클래스에서 빗방울을 만들어야 하는 거 아닌가?’ 생각할 수도 있지만, 그럴 수는 없습니다.
 - 빗방울 클래스는 이미 만들어진 빗방울들에 대해서만 작업할 수 있습니다.
 - 클래스의 인스턴스는 클래스 밖에서 만들어야 합니다.

```
class Cloud:
```

```
(중략)
```

```
    def rain(self):
```

```
        raindrops.append(Raindrop(random.randint(self.x,self.x+300),self.y+100))
```

- 2번째 줄을 간단히 하면 아래와 같습니다.

```
raindrops.append(Raindrop(x, y))
```

7. 구름에서 떨어지는 빗방울

- Raindrop()은 빗방울 생성자constructor입니다.
- 빗방울 생성자는 빗방울 클래스의 인스턴스를 만듭니다.
- 빗방울 클래스의 인스턴스를 빗방울들 리스트에 추가하지요.
- 이렇게 만들어진 인스턴스는 x, y 2개의 값을 받습니다.

```
class Cloud:
```

```
(중략)
```

```
    def rain(self):
```

```
        raindrops.append(Raindrop(random.randint(self.x,self.x+300),self.y+100))
```

7. 구름에서 떨어지는 빗방울

- 앞쪽에서 하나의 빗방울이 만들어 질 때 빗방울 클래스의 `__init__()` 함수가 `x, y` 2개의 인자를 요구한다고 했습니다.
- 위 코드에서 `x`는 `random.randint(self.x, self.x+300)`로 주었습니다.
- 구름의 `x`좌표인 `self.x`와 `self.x+300` 사이 임의의 숫자를 정하라는 뜻입니다.
- 구름의 `X`좌표는 구름의 왼쪽 끝을 말합니다.
- 구름의 가로 길이는 300이므로 오른쪽 끝은 `self.x+300`입니다.
- 따라서 구름의 양 끝 사이의 임의의 숫자를 선택하게 됩니다.
- `y`값은 `self.y+100`으로 고정합니다. `self.y`는 구름의 가장 윗부분의 `y`좌표입니다.
- 구름의 높이는 108픽셀입니다. 따라서 `self.y+100`은 구름의 가장 밑부분에서 아주 조금 올라온 곳입니다. (`y`축은 아래 방향이 양수니까요!)
- 이제 게임 루프에서 이 함수를 불러와야 합니다.
- `draw()` 함수를 불러 올 때 같이하면 됩니다.

```
cloud.draw()  
cloud.rain()
```

8. 화살표 키로 구름 움직이기

- 이제 구름을 움직입니다.
- 눌린 키들 리스트 `pressed_keys` list가 필요합니다.
- 이 코드는 끝내기 `QUIT` 섹션 밑에 씁니다.
- 끝내기와는 별 관계 없지만, 이런 코드는 보통 끝낼 때 쓰니까요.

```
for event in pygame.event.get():  
    if event.type == QUIT:  
        sys.exit()
```

```
pressed_keys = pygame.key.get_pressed()
```



8. 화살표 키로 구름 움직이기

- 구름의 움직임 제어를 위해 구름 클래스 안에서 move 함수를 만듭시다.

Class Cloud:

(중략)

```
def move(self):  
    if pressed_keys[K_RIGHT]:  
        self.x += 1  
    if pressed_keys[K_LEFT]:  
        self.x -= 1
```

- 원을 움직이던 코드와 비슷하지요? xpos 대신 self.x를 썼을 뿐입니다.
- 작동 방식도 같습니다.

8. 화살표 키로 구름 움직이기

- 이제 게임 루프에서 구름이 다른 함수들과 같이 `move()` 함수를 불러오도록 추가합니다.

```
cloud.draw()  
cloud.rain()  
cloud.move()
```

- 위 코드에서 `rain()` 함수는 루프당 1번 불러오고, 빗방울 1개만 내보냅니다.
- 빗방울 여러 개를 내보내려면 구름 클래스 안의 `rain()` 함수를 이렇게 바꾸세요.

```
def rain(self):  
    for i in range(10):  
        raindrops.append(Raindrop(random.randint(self.x,self.x+300),self.y+100))
```

- `for` 루프는 10까지의 모든 1 값을 따라 돌면서 각 루프당 1번씩 마지막 줄을 실행합니다.
- 따라서 `rain()` 함수가 1번 불러올 때마다 빗방울 10개가 생깁니다.

9. 우산을 쓰지 않은 마이크

- 이제 간단한 애니메이션을 만들어 봅시다.
- 비 올 때는 우산이 필요하지만, 구름이 가버리고 빗방울도 떨어지지 않으면 필요 없습니다.
- 비가 올 때만 마이크가 우산을 들도록 만듭시다.
- 먼저 마이크의 두 번째 이미지를 불러옵니다.
- 오른쪽 이미지는 Mike.png이고, 웹사이트에 있습니다.
- 프로그램의 셋업에 아래 코드를 써서 불러옵니다.



```
mike_image = pygame.image.load("images/Mike.png").convert()
```

9. 우산을 쓰지 않은 마이크

- 비가 멈추고 1초 뒤에 마이크가 우산을 내려놓도록 하려면 현재 시각을 알려 줄 시계가 필요합니다.
- 마이크 위로 마지막 빗방울이 떨어진 시각을 기록해야 하니까요. 그래야 1초가 지난 뒤 마이크 이미지를 바꾸라고 할 수 있거든요.
- 마이크가 마지막으로 빗방울에 맞은 시각을 저장할 변수도 추가해야 합니다.
- 이 변수는 프로그램의 셋업에 씁니다. 당연히 들여쓰기는 하지 않습니다.

```
last_hit_time = 0
```

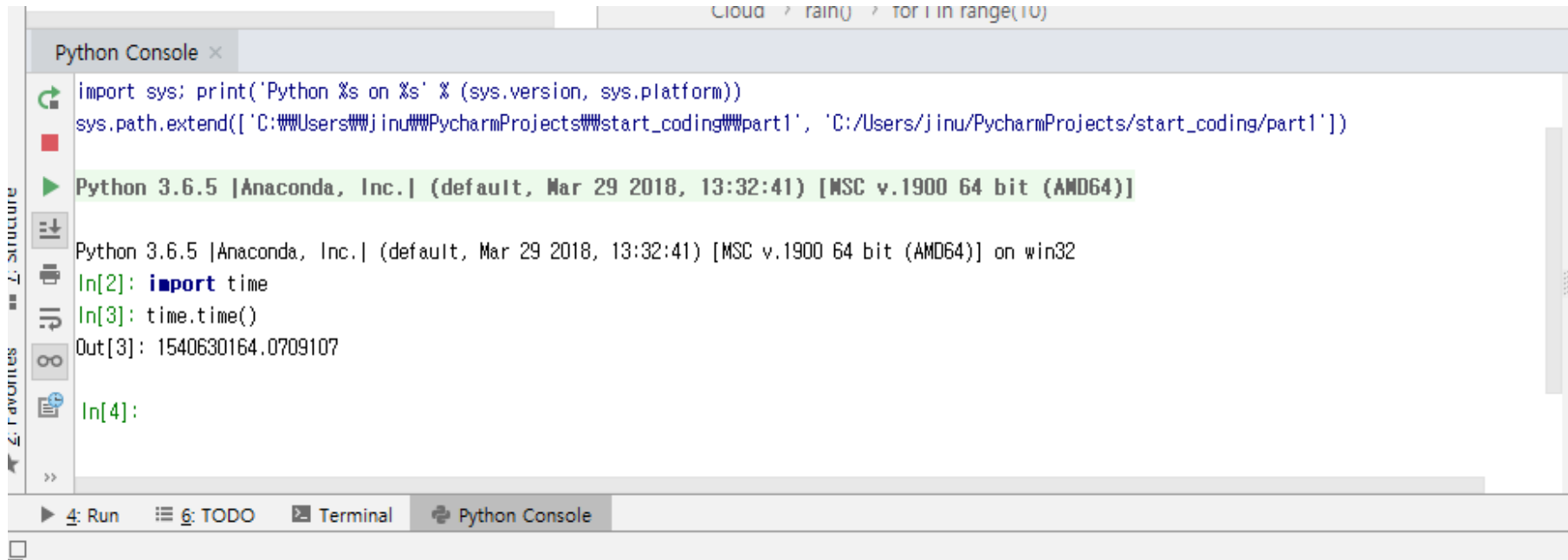
- 여기서 질문! 변수는 클래스 안에 만들어야 할까요? 아니면 셋업 또는 게임 루프 안에 만들어야 할까요?

9. 우산을 쓰지 않은 마이크

- 일반적으로 클래스의 인스턴스가 사용하는 변수는 클래스 안에서 만듭니다.
`self.x`나 `self.y`처럼 말이지요.
- 매 게임 루프마다 리셋되거나 1처럼 게임 루프 안에서만 사용되는 변수는 게임 루프 안에 만듭니다.
- 클래스와 게임 루프 안에서 동시에 사용되거나, 게임 루프 안에서만 사용되는데 매 루프마다 업데이트되지만 리셋되지는 않는 변수는 셋업에서 만듭니다.
- `last_hit_time`은 게임 루프와 클래스 안에서 사용되며 매 루프마다 업데이트되는 변수입니다.
- 따라서 셋업에 만듭니다.

10. time.time()

- 컴퓨터 안에는 1970년 1월 1일부터 1000분의 1초millisecond 단위로 숫자를 세는 시계가 있습니다.
- 커맨드 라인에서 이 시계를 불러 봅시다



The screenshot shows a Python IDE interface with a 'Python Console' tab. The console displays the following code and output:

```
import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['C:\\Users\\j inu\\PycharmProjects\\start_coding\\part1', 'C:/Users/j inu/PycharmProjects/start_coding/part1'])

Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:32:41) [MSC v.1900 64 bit (AMD64)]

Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:32:41) [MSC v.1900 64 bit (AMD64)] on win32
In[2]: import time
In[3]: time.time()
Out[3]: 1540630164.0709107

In[4]:
```

The bottom of the IDE shows tabs for 'Run', 'TODO', 'Terminal', and 'Python Console'.

- 가장 아래쪽 숫자가 1970년 1월 1일 자정으로부터 지난 초를 나타내는 수입니다 . 앞으로 이 시계를 이용할 것입니다.

10. time.time()

- 파이썬에서 이 시계를 이용하려면 프로그램 맨 첫 줄에서 `time` 모듈을 가져 와야 (`import`) 합니다.

```
import pygame, sys, random, time
```

- 빗방울 제어 `while` 루프(`while 1:` 말고요) 안에는 빗방울을 삭제하는 두 가지 조건이 있습니다.
- 아래 5, 6번째 줄을 보면, 빗방울이 스크린 밖으로 떨어지거나, 마이크에 닿았을 때 삭제됩니다.

```
i=0
while i < len(raindrops):
    raindrops[i].move()
    raindrops[i].draw()
    if raindrops[i].off_screen() or if mike.hit_by(raindrops[i]):
        del raindrops[i]
        i -= 1
    i += 1
```

10. time.time()

- 마이크 위에 빗방울이 떨어질 때만 `last_hit_time`을 `time.time()`으로 정해야 합니다.

```
i=0
while i < len(raindrops):
    raindrops[i].move()
    raindrops[i].draw()
    if raindrops[i].off_screen():
        del raindrops[i]
        i -= 1
    if mike.hit_by(raindrops[i]):
        del raindrops[i]
        last_hit_time = time.time()
        i -= 1
    i += 1
```

- 6번째 줄에서 10번 빗방울이 삭제됐다면, 8번째 줄이 인식하는 빗방울은 새롭게 10번 빗방울이 된 11번 빗방울입니다.
- 우리는 이 빗방울을 움직이지도, 그리지도, 스크린 밖으로 떨어졌는지 확인하지도 않았습니다.

10. time.time()

- 게임 루프 1번에 빗방울 수백 개 중 하나가 없어진다고 해서 뭐가 문제냐고요?
- 지금은 괜찮지만 나중에 총알 같은 걸 다룬다면 문제가 될 수 있습니다.
- 다행히 8번째 줄의 if를 else로 바꾸어 문제를 해결할 수도 있습니다.
- else 문은 기본적으로 if 문이 거짓일 때만 실행되기 때문입니다.

10-1. if-else 문

- 프로그래밍을 하다 보면 아래와 같은 형태로 된 구문을 종종 볼 수 있습니다.

```
if x == 3:  
    위아래로 뛰기  
else:  
    가만히 서 있기
```

- 만약 1 번째 줄이 참이면 이걸 읽은 파이썬은 우리를 위아래로 뛰게 만들 것입니다. 이때 "else" 부분은 무시합니다.
- 하지만 만약 1 번째 줄이 거짓이면 파이썬은 "else" 쪽을 보고 거기 적힌 대로 시키겠지요.
- 파이썬은 이걸 하든 저걸 하든 둘 중 하나를 시킵니다 결코 2개를 동시에 시키지는 않지요.

10-1. if-else 문

- 다중 if-else 문은 다음과 같은 형태입니다

```
if x == 5:
    위아래로 뛰기
elif x == 6:
    빙빙 돌기
elif x == 7:
    구부려서 손을 발가락에 대기
else:
    가만히 있기
```

- 참인 구문을 찾으면 파이썬은 나머지를 무시합니다.
- 만약 x가 601라면 파이썬은 우리를 빙빙 돌게 만들겠지요.
- x가 7과 같은지 물어보지는 않습니다
- 만약 참인 경우를 찾지 못했다면, 파이썬은 "else" 구문 아래에 있는 것을 시킬 것입니다

11. 플래그

- if-else 문을 사용해도 문제가 해결되지만, 플래그flag를 쓰는 것도 가능합니다.
- 플래그는 파이썬에서 조건이 참인지를 가리키기 위해 사용되는 논리 변수입니다.

```
i=0
while i < len(raindrops):
    raindrops[i].move()
    raindrops[i].draw()
    flag = False
    if raindrops[i].off_screen() or
        del raindrops[i]
        i -= 1
        flag = True
    if mike.hit_by(raindrops[i]):
        del raindrops[i]
        i -= 1
        flag = True
        last_hit_time = time.time()
    if flag:
        del raindrops[i]
        i -= 1
    i += 1
```

아아아아아! 아아아아아!
난 곧 죽을 거야! 난 곧 죽을 거야!

아아아아아!
난 곧 죽을 거야! 아아아아아!
난 곧 죽을 거야!

아아아아아!
난 곧 죽을 거야! 아아아아아!
난 곧 죽을 거야!

아아아아아!
난 곧 죽을 거야! 아아아아아!
난 곧 죽을 거야!

<빛방울들의 속마음>

〈빛의 양을 줄이는 속마음〉

11. 플래그

- 각각의 빗방울에 플래그를 꽂습니다.
- 플래그가 거짓이면 빗방울은 삭제되지 않습니다.
- 하지만 다른 조건이 참이 되면 플래그를 참으로 바꿉니다.
- `if flag:` 라는 코드는 만약 플래그가 참이라면 다음에 나오는 코드를 실행하라는 뜻입니다.
- 즉, 두 가지 조건 중 하나라도 참이 되면, 플래그도 참이 돼 빗방울을 삭제하라는 뜻입니다.
- 플래그가 2번 참이 되는 것은 상관없습니다.
- 참이 또 참이 돼도 여전히 참이지요.
- 이제 마이크가 빗방울에 맞을 때마다 `last hit time`은 업데이트 됩니다.
- 다음 쪽의 `draw()` 함수에서 이것을 사용해 봅시다.

12. 우산 쓴 마이크 vs 우산 안쓴 마이크

```
class Mike:
    (중략)
    def draw(self):
        if time.time() > last_hit_time + 1:
            screen.blit(mike_image, (self.x, self.y))
        else:
            screen.blit(mike_umbrella_image, (self.x, self.y))
```

- 마이크 클래스에서 처음 만든 draw() 함수는 꽤 간단했습니다.
- 마이크 이미지를 좌표(self.x, self.y)에 그리기만 했지요.
- 하지만 이제 if time.time() > last_hit_time + 1이라는 조건이 추가됐습니다.
- 빗방울이 마이크 위로 떨어질 때마다, last_hit_time 은 time.time() 으로 업데이트됩니다.
- draw() 함수가 불려올 때, last_hit_time과 time.time() 사이의 차이는 미미합니다.

12. 우산 쓴 마이크 vs 우산 안쓴 마이크

```
class Mike:
    (중략)
    def draw(self):
        if time.time() > last_hit_time + 1:
            screen.blit(mike_image, (self.x, self.y))
        else:
            screen.blit(mike_umbrella_image, (self.x, self.y))
```

- `time.time()`은 `last_hit_time+1` 보다 크지 않을 것이고, 위 조건식의 값은 거짓이 됩니다.
- 따라서 `else` 문이 실행되고 우산을 쓴 마이크 이미지가 나타납니다.

12. 우산 쓴 마이크 vs 우산 안쓴 마이크

```
class Mike:
    (중략)
    def draw(self):
        if time.time() > last_hit_time + 1:
            screen.blit(mike_image, (self.x, self.y))
        else:
            screen.blit(mike_umbrella_image, (self.x, self.y))
```

- 하지만 비가 마이크 위로 떨어지지 않게 되면, last_hit_time의 값은 그대로이고, time.time()은 계속 증가합니다.
- 1초 뒤에는 time.time() 이 last_hit_time+1보다 커지게 됩니다.
- if 문은 참값을 돌려주고, 우산을 쓰지 않은 마이크 이미지가 전송될 것입니다.
- 즉, 1초 동안 비가 오지 않아야 마이크는 우산을 접습니다.
- 물론 1 대신 다른 숫자를 써도 됩니다.



Thank You
