파이썬 거북이, for 반복문 중첩으로 그리는 신기한 패턴과 별 잔치! 🐢🌟✨겹겹

등장인물:

- **슬기 선생님:** 친절하고 유머러스한 코딩 선생님
- **지혜:** 똑똑하고 질문이 많은 학생
- **코딩이:** 코딩을 처음 배우지만 호기심 많은 학생

슬기 선생님: 여러분, 안녕하세요! for 반복문으로 멋진 그림들을 그려봤죠? 오늘은 for 반복문 안에 또 다른 for 반복문을 넣는, 아주 특별한 **이중 for문 (Nested for loop)** 마법을 배워볼 거예요! 이 마법을 사용하면 더욱 복잡하고 화려한 패턴과 그림들을 만들 수 있답니다!

코딩이: 이중 for문이요? for문 안에 for문이 또 들어간다고요? 마치 상자 안에 상자가 들어있는 것 같아요! 그걸로 뭘 할 수 있어요?

슬기 선생님: (웃음) 맞아요, 코딩이 학생! 상자 속의 상자처럼, 반복 안에 또 다른 반복을 넣는 거예요. 먼저 간단한 예시로 이중 for문이 어떻게 돌아가는지 살펴볼까요?

1. 이중 for문의 기본 원리 탐험!

```
for i in range(6):              # 바깥쪽 for문: i는 0부터 5까지 변해요.
  for j in ["a", "b", "c"]:  # 안쪽 for문: j는 "a", "b", "c"를 차례로 가져요.
    print(i, j)              # i와 j를 함께 출력!
```

지혜: 와! 결과가 신기하게 나와요! i가 0일 때 j가 “a”, “b”, “c” 순서대로 나오고, 그 다음에 i가 1로 바뀌니까 또 j가 “a”, “b”, “c” 이렇게 나와요!

슬기 선생님: 정확해요, 지혜 학생! 이중 for문은 이렇게 작동해요.

1. **바깥쪽 for문** (`for i in range(6):`)이 먼저 한 바퀴 돌기 시작해요. i에 첫 번째 값인 0이 들어가죠.
2. 그 상태에서 **안쪽 for문** (`for j in ["a", "b", "c"]:`)을 만나요.
3. 안쪽 for문은 바깥쪽 i가 0인 상태에서 **자신의 반복을 끝까지 다 돌아요.** 즉, j에 “a”가 들어가서 `print(0, "a")` 실행, 그 다음 j에 “b”가 들어가서 `print(0, "b")` 실행, 마지막으로 j에 “c”가 들어가서 `print(0, "c")`를 실행하죠.
4. 안쪽 for문이 반복을 다 마치면, 다시 바깥쪽 for문으로 돌아가서 i에 다음 값인 1이 들어가요.
5. 그리고 다시 안쪽 for문은 j에 “a”, “b”, “c”를 차례로 넣으면서 자신의 반복을 끝까지 또 다 돌죠. (`print(1, "a")`, `print(1, "b")`, `print(1, "c")`) 이런 식으로 바깥쪽 for문이 한 번 변할 때마다, 안쪽 for문은 처음부터 끝까지 자신의 모든 반복을 수행하는 거예요. 마치 시계의 시침(바깥쪽)이 한 칸 움직일 때 분침(안쪽)이 한 바퀴를 다 도는 것과 비슷하답니다!

코딩이: 아하! 바깥쪽 for문이 대장이고, 안쪽 for문은 대장이 한 번 명령할 때마다 자기 할 일을 처음부터 끝까지 다 하는 부하 같아요!

슬기 선생님: (웃음) 아주 좋은 비유예요, 코딩이 학생! 자, 그럼 이 이중 for문 마법으로 우리 james 거북이와 함께 화려한 별 잔치를 열어볼까요?

2. 이중 for문으로 그리는 알록달록 별 잔치!

```
clearscreen()

james = Turtle()
james.speed(10)
james.pensize(5)

james.penup()
james.goto(-100, 100) # 별들을 그릴 시작 위치로 이동
james.pendown()

# for x in range(3): # 바깥쪽 대장 for문! (3번 반복)
  # for i in ["green", "yellow", "black", "blue", "gray", "orange"]: # 안쪽 색깔 담당 for문! (6가지 색)
    # james.color(i)
    # james.begin_fill()
    # for j in range(5):   # 가장 안쪽 별 그리기 담당 for문! (별 꼭짓점 5개)
      # james.forward(100)
      # james.right(144)
    # james.end_fill()
    # james.right(20)      # 다음 별을 그리기 위해 살짝 방향 틀기
    # james.forward(50)    # 다음 별을 그릴 위치로 조금 이동
```

(선생님이 위 코드의 주석을 풀면서 한 단계씩 설명한다)

슬기 선생님: 자, 여기 아주 멋진 코드가 있어요. 맨 바깥에 `for x in range(3):` 이라는 대장 for문이 있죠?

지혜: 네! x가 0, 1, 2 이렇게 세 번 반복되겠네요! 그런데 선생님, 어떤 친구가 `for x in (3):` 이렇게 썼다가 오류가 났다고 했어요. range를 안 쓰면 왜 안 돼요?

슬기 선생님: 아주 중요한 질문이에요! for 반복문은 "어떤 범위 안에서" 또는 "어떤 목록 안에서" 값을 하나씩 꺼내오면서 반복해요. `range(3)`은 `0, 1, 2`라는 숫자 목록(정확히는 반복 가능한 객체)을 만들어줘서 for문이 세 번 반복될 수 있도록 해줘요. 하지만 그냥 `(3)`이라고 쓰면 이건 숫자 3일 뿐, 여러 번 반복할 수 있는 목록이 아니랍니다. 그래서 파이썬이 "어? 반복할 수 있는 목록을 줘야 하는데 숫자 하나만 줬네?" 하면서 TypeError 같은 오류를 내는 거예요. for문에는 꼭 `range()`나 리스트처럼 반복 가능한 친구들을 넣어줘야 한답니다!

코딩이: 아하! range가 반복 마법의 재료였군요! 그럼 저 바깥쪽 x 루프는 지금 코드에서는 별들이 그려지는 위치를 바꾸거나 하진 않는 것 같아요. goto가 바깥에 한 번만 있으니까요.

슬기 선생님: 맞아요, 코딩이 학생. 현재 코드에서는 goto가 바깥에 한 번만 있어서, x 루프가 3번 반복되면서 안쪽의 별 그리는 동작 전체(여러 색깔의 별 부채꼴 모양으로 그리기)를 같은 위치에서 3번 겹쳐 그리게 될 거예요. 만약 x 루프 안에서 `james.goto()`로 위치를 바꿔주거나 하면 3개의 다른 위치에 별 무더기를 그릴 수도 있겠죠! 오늘은 이중 for문의 구조에 집중해 볼게요.

슬기 선생님: 자, 그럼 바깥 x 루프가 한 번 돌 때, 그 안에서 무슨 일이 일어나는지 볼까요? `for i in ["green", "yellow", "black", "blue", "gray", "orange"]:` 이 중간 for문이 있네요!

```
# (바깥 for x in range(3): 가 한 번 실행될 때 아래 내용이 실행됨)
  for i in ["green", "yellow", "black", "blue", "gray", "orange"]: # 안쪽 색깔 담당 for문!
    james.color(i)         # i에 들어온 색깔로 james 변신!
    james.begin_fill()     # "이 색깔로 별 채울 준비!"

    for j in range(5):     # 가장 안쪽 별 그리기 담당 for문!
      james.forward(100)
      james.right(144)
    james.end_fill()       # "이 색깔 별 채우기 완료!"

    james.right(20)        # 다음 별을 그리기 위해 살짝 방향 틀기
    james.forward(50)      # 다음 별을 그릴 위치로 조금 이동
```

지혜: 아! 이 `i` 루프는 색깔 리스트에 있는 색깔들("green", "yellow", …)을 하나씩 `i`에 넣으면서 반복되겠네요! 그럼 초록색 별 하나, 노란색 별 하나, 이렇게 총 6개의 다른 색깔 별이 그려지는 건가요?

슬기 선생님: 정확해요! 그리고 각 색깔의 별 하나를 그리기 위해 가장 안쪽에 `for j in range(5):` 루프가 또 있죠? 이 루프가 바로 우리가 저번에 배운 별의 뾰족한 꼭짓점 5개를 그리는 역할을 하는 거예요. 여기서 `begin_fill()`과 `end_fill()`의 위치가 아주 중요해요! `james.begin_fill()`은 **하나의 별을 그리기 시작하기 직전**(가장 안쪽 `j` 루프 직전, `i` 루프 안)에 와야 그 특정 색깔로 별을 채울 준비를 하겠죠? 그리고 `james.end_fill()`은 **하나의 별 그림이 완성된 직후**(가장 안쪽 `j` 루프가 끝난 직후, `i` 루프 안)에 와야 그 별 하나를 제대로 채울 수 있어요.

코딩이: 만약에 `end_fill()`을 가장 안쪽 `j` 루프 안에 넣으면 어떻게 돼요? 저번에 별 그릴 때 실수했던 것처럼요! 그리고 들여쓰기도 엄청 중요하죠? 헷갈려요!

슬기 선생님: 맞아요, 코딩이 학생! 들여쓰기는 이중, 삼중 for문에서 더욱 중요해져요! 각 for문에 속한 명령어들은 그 for문보다 한 단계 더 안쪽으로 들여써야 "나는 이 for문의 부하야!"라고 표시하는 거랍니다. 만약 `end_fill()`을 가장 안쪽 `j` 루프 안에 넣으면, 별의 꼭짓점 하나 그리고 채우고, 또 하나 그리고 채우고… 이렇게 돼서 완성된 별 모양이 아니라 이상한 조각들만 채워질 거예요. 지금 이 코드에서는 `begin_fill()`과 `end_fill()`이 중간 `i` 루프(색깔 루프) 안에 있으면서, 가장 안쪽 `j` 루프(별 그리기 루프)를 감싸고 있죠? 그래서 각 색깔별로 별 하나가 완성될 때마다 채우기가 실행되는 거랍니다.

(james 거북이가 다양한 색깔의 별들을 부채꼴 모양으로 그리는 것을 3번 반복한다 - 실제로는 겹쳐 그려짐)

지혜: 아! 이중 for문은 바깥 루프가 한 번 돌 때 안쪽 루프가 자기 할 일을 다 하고, 그 안쪽 루프 안에 또 루프가 있으면 그 루프도 자기 할 일을 다 하는 거군요! 정말 겹겹이네요! 그리고 들여쓰기로 누가 누구의 명령어인지 확실히 구분해 줘야 하고요!

슬기 선생님: 정확히 이해했어요! 이중 for문, 심지어 삼중 for문도 이런 원리로 동작한답니다. 처음에는 조금 헷갈릴 수 있지만, 어떤 for문이 언제 실행되고, 그 안의 명령어들은 무엇인지 차근차근 따라가 보면 금방

익숙해질 거예요. 오늘 이중 for문의 매력에 푹 빠져봤죠?

코딩이 & 지혜: 네! 선생님! 조금 복잡하지만, 훨씬 더 멋진 그림을 그릴 수 있을 것 같아요! 이중 for문 마스터에 도전할래요!