파이썬 거북이, 알록달록 집 짓고 색칠하기! 🏠🎨🐢💨 (대소문자 설명 추가, 주석 제거 버전)

등장인물:

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

슬기 선생님: 여러분, 오늘도 즐거운 코딩 시간! 지난 시간에는 거북이가 펜을 들었다 놨다 하고, 좌표로 순간이동하는 마법까지 배웠죠? 오늘은 그 마법들을 총동원해서, 예쁜 집을 짓고 그 집에 알록달록 색깔까지 칠해보는 건축가가 되어 볼 거예요!

코딩이: 와! 건축가요? 제가 직접 집을 짓고 색칠까지 한다고요? 너무 신나요! `jane` 거북이, 오늘 잘 부탁해!

슬기 선생님: (웃음) 좋아요! 먼저 우리 건축 현장을 깨끗하게 정리하고, 특별한 도구를 하나 더 준비해 볼게요. 바로 도화면 자체를 관리하는 `Screen` 객체랍니다.

```
clearscreen()


screen = Screen()
screen.bgcolor("green")
```

지혜: 어? 선생님, `Turtle()` 말고 `Screen()`이라는 새로운 게 나왔어요! `screen = Screen()` 이건 `jane = Turtle()`처럼 `Screen`이라는 설계도로 `screen`이라는 실제 도화면을 만드는 건가요? `screen` 말고 `my_drawing_paper = Screen()` 이렇게 이름 지어도 돼요? 그리고 왜 `Screen`은 대문자로 시작하고, `screen`은 소문자로 써요? 꼭 그렇게 해야 하나요? 가끔 `Screen = Screen()` 이렇게 쓰거나 `Screen.bgcolor("green")` 이렇게 쓰고 싶을 때도 있는데요!

슬기 선생님: 와, 지혜 학생! 정말 중요한 질문들을 한꺼번에 해줬어요! 아주 훌륭해요! 하나씩 차근차근 설명해 줄게요. 먼저, 지혜 학생이 이해한 대로 `Screen()`은 우리 거북이가 그림을 그리는 도화면 전체를 다루는 설계도, 즉 **클래스(Class)**예요. 그리고 `screen = Screen()`이라고 하면, 그 `Screen` 설계도로 만들어진 실제 도화면, 즉 **객체(Object) 또는 인스턴스(Instance)**가 생기고, 그 객체에 `screen`이라는 이름표를 붙여준 거죠. `my_drawing_paper = Screen()`처럼 다른 이름을 붙여줘도 물론 괜찮아요!

자, 이제 대소문자 이야기로 넘어가 볼게요. 파이썬은 **대소문자를 아주 엄격하게 구분하는 언어**예요. 그래서 `Screen`과 `screen`은 완전히 다른 것으로 받아들인답니다.

일반적으로 약속된 규칙이 있어요.

- **`Screen`처럼 첫 글자를 대문자로 시작하는 이름**은 방금 말한 **클래스(설계도)**의 이름으로 많이 사용돼요. "이건 무언가를 만드는 설계도야!"라는 표시 같은 거죠. `Turtle`도 마찬가지였죠?
- **`screen`처럼 모든 글자를 소문자로 쓰거나, 단어 사이에 밑줄(_)을 넣는 이름** (`my_drawing_paper`)은 클래스로부터 만들어진 **객체(실제 물건)에 붙이는 이름표(변수)**로 많이 사용돼요.

그래서 screen = Screen()이라고 쓰는 것은 " Screen이라는 설계도로 물건(객체)을 하나 만들고, 그 물건에 screen이라는 이름표를 붙여줘!" 라는 아주 정확한 표현이랍니다.

지혜: 아하! Screen은 설계도 이름이고, screen은 그 설계도로 만든 실제 도화지 이름이군요! 그럼 Screen = Screen() 이렇게 쓰면 안 되는 거예요?

슬기 선생님: 아주 좋은 질문이에요! 만약 Screen = Screen()이라고 쓰면, 파이썬은 "어? Screen이라는 이름표에다가 Screen 설계도로 만든 객체를 넣으라고?" 하면서 좀 헷갈릴 수 있어요. 원래 Screen은 설계도를 가리키는 이름이었는데, 이제는 실제 도화지 객체를 가리키게 되는 거죠. 이렇게 하면 나중에 또 다른 도화지를 만들고 싶을 때 Screen() 설계도를 찾기 어려워질 수 있어요. 그래서 설계도 이름(클래스 이름)과 객체 이름(변수 이름)은 다르게 해주는 것이 훨씬 좋답니다. 규칙을 지키면 코드가 덜 헷갈리거든요.

코딩이: 그럼 Screen.bgcolor("green") 이렇게 설계도에 바로 색칠하라고 하면 안 돼요?

슬기 선생님: 코딩이 학생도 좋은 질문을 했어요! bgcolor("green") 같은 명령은 "배경색을 초록색으로 바꿔줘!"라는 건데, 이건 실제 도화지 **객체**에게 내리는 명령이에요. 설계도 자체에는 색깔이 없잖아요? 설계도로 만든 실제 도화지에 색을 칠하는 거죠. 그래서 Screen.bgcolor("green")이라고 하면 "어떤 도화지의 배경색을 바꿔야 할지 모르겠어!" 하면서 오류가 날 수 있어요. 반드시 screen.bgcolor("green")처럼 실제 도화지 객체인 screen에게 명령을 내려야 한답니다. " screen아, 네 배경색을 초록색으로 바꿔!" 이렇게요.

지혜: 이제 확실히 알겠어요! Screen은 설계도, screen은 설계도로 만든 진짜 도화지! 그리고 명령은 진짜 도화지한테 내려야 하는군요!

슬기 선생님: 맞아요! 이 규칙을 잘 기억하면 앞으로 코딩할 때 헷갈리는 일이 훨씬 줄어들 거예요. 자, 그럼 우리 screen 도화면의 배경색이 초록색으로 변했으니, 이제 jane 건축가 거북이를 불러올 차례죠?

```
jane = Turtle()
jane.speed(10)
jane.pensize(5)
jane.color("yellow", "blue")
```

지혜: jane.speed(10)은 거북이가 움직이는 속도를 정하는 건가요? 숫자가 클수록 빠른 거예요? 그리고 jane.color("yellow", "blue")는 색깔이 두 개나 들어가 있어요! 이건 뭐예요?

슬기 선생님: 예리한 질문이에요! jane.speed()는 거북이가 그림 그리는 애니메이션 속도를 조절해요. 0이 가장 빠르고, 1(느림)부터 10(빠름)까지 숫자가 커질수록 빨라진답니다. 10도 꽤 빠른 편이죠. 그리고 jane.color("첫 번째 색", "두 번째 색") 이렇게 색을 두 개 넣어주면, **첫 번째 색("yellow")은 거북이가 그리는 선의 색깔**이 되고, **두 번째 색("blue")은 나중에 도형 안을 채울 때 사용할 채우기 색깔**이 된답니다.

코딩이: 오! 선 색이랑 채우는 색을 다르게 할 수 있구나! 신기하다! 그럼 이제 집을 그릴 건데… 어떻게 색칠해요? 페인트 통이라도 주나요?

슬기 선생님: (웃음) 페인트 통 대신에 jane 거북이에게 특별한 명령어를 알려줄 거예요. 바로 begin_fill()과 end_fill() 이랍니다! 이 두 명령어 사이에 그린 도형 안을 아까 정한 채우기 색으로 샥~ 하고 칠해줘요. 마치 한붓그리기처럼, 시작점과 끝점을 잘 정해줘야 예쁘게 칠해진답니다. 자, 먼저 집의 지붕과 오른쪽 벽을 그려볼까요?

```
jane.begin_fill()
jane.back(50)
jane.left(70)
jane.forward(90)
jane.right(70)
jane.forward(200)
jane.right(70)
jane.forward(90)
jane.right(110)
jane.forward(210)
jane.end_fill()
```

(jane 거북이가 노란색 선으로 지붕과 벽을 그리고, 그 안을 파란색으로 채운다)

지혜: 와! `begin_fill()` 하고 그림을 그리니까, `end_fill()` 하는 순간 파란색으로 샥! 하고 칠해졌어요! 진짜 마법 같아요! 선생님, 근데 `begin_fill()`은 언제 해야 하고, `end_fill()`은 언제 해야 하는지 조금 헷갈려요.

슬기 선생님: 아주 중요한 질문이에요! 많은 친구들이 헷갈려 하는 부분이거든요. `begin_fill()`은 **"내가 지금부터 그릴 도형의 내부를 색칠할 거야!"** 라고 선언하는 것과 같아요. 그래서 색칠하고 싶은 도형을 그리기 **바로 직전에** 써줘야 해요. 그리고 `end_fill()`은 **"자, 이제 도형 다 그렸으니까 약속대로 아까 그 도형 안을 칠해줘!"** 라고 명령하는 거예요. 그래서 색칠하고 싶은 도형의 **모든 선을 다 그리고 난 후에** 써줘야 한답니다. 거북이가 그린 선들이 이어져서 하나의 닫힌 공간을 만들어야 예쁘게 칠해져요.

코딩이: 아하! 색칠 시작 버튼(`begin_fill()`) 누르고 그림 그리고, 다 그리면 색칠 완료 버튼(`end_fill()`) 누르는 거네요! 그럼 이제 집 옆에 다른 부분도 그려서 색칠해 볼까요?

슬기 선생님: 좋아요! 이번에는 집 옆에 붙어있는 작은 공간을 그려볼 거예요. 이번에는 선 색은 계속 노란색으로 하고, 채우기 색은 검은색으로 바꿔볼까요?

```
jane.color("yellow", "black")
jane.begin_fill()
jane.left(90)
jane.forward(100)
jane.left(90)
jane.forward(160)
jane.left(90)
jane.forward(100)
jane.end_fill()
```

(jane 거북이가 노란색 선으로 집의 옆 부분을 그리고, 그 안을 검은색으로 채운다)

지혜: 와! 이번엔 검은색으로 칠해졌어요! `color()` 함수로 채우기 색을 바꿀 수 있으니까 여러 가지 색으로 칠할 수 있네요!

코딩이: 초록색 풀밭 위에 파란 지붕과 검은 방이 있는 노란 테두리 집! 제가 만든 거예요! 너무 뿌듯해요!

begin_fill()이랑 end_fill() 위치만 잘 기억하면 되겠어요! 대문자 Screen이랑 소문자 screen도 이제 안 헷갈릴 거예요!

슬기 선생님: 아주 잘했어요, 여러분! 오늘 우리는 Screen 객체와 대소문자 규칙, 도화면 배경색 바꾸기, 거북이 속도 조절, 두 가지 색을 사용하는 color() 함수, 그리고 가장 중요한 begin_fill()과 end_fill()로 도형 안을 예쁘게 채우는 방법까지 배웠어요. 이제 여러분은 정말 멋진 건축가가 될 준비가 된 것 같네요!

지혜 & 코딩이: 네! 선생님! 다음엔 또 어떤 멋진 걸 만들 수 있을지 정말 기대돼요!