<[파이썬 그림 마법] 제2장. 달려라, 터틀! 신나는 거북이 경주 게임!>

선생님: (출발 신호용 깃발을 흔들며) 얘들아, 안녕! 지난 시간에는 우리 터틀 친구에게 그림을 그리게 하는 법을 배웠지? 오늘은 그 터틀 친구들을 여러 명 불러서, 누가 가장 빠른지 신나는 경주를 시켜볼 거야! 생각만 해도 두근거리지 않니?

코딩새싹 🌱: 와! 거북이들이 경주를 해요? 우리가 직접 게임을 만드는 거예요? 정말 재미있겠어요!

선생님: 물론이지! 자, 경주 게임을 만들려면 무엇이 필요할까? 먼저 멋진 경주장을 만들어야겠지!

<1. 씽씽쌩쌩! 경주장 준비하기!>

선생님: 터틀 그래픽 도구들을 다시 한번 소환하고, 경주를 위한 넓은 스크린과 배경을 설정하자!

```
# !pip install ColabTurtlePlus # Colab 환경에서 이 도구 상자를 설치하는 마법 주문! (이미 했다면 통과!)
from ColabTurtlePlus.Turtle import * # 터틀 그림 마법 도구를 모두 가져와!
import random # 거북이의 움직임을 예측불허하게 만들 random 마법사도 소환!

clearscreen() # 혹시 이전에 그린 게 있다면 깨끗이 지우고 시작!

# 스크린 만들기 (가로 800, 세로 500, 배경색은 초록색 잔디밭!)
screen_width = 800
screen_height = 500

screen = Screen()
screen.setup(screen_width, screen_height)
screen.bgcolor("green")

# 경주장의 제목과 결승선을 그릴 특별한 터틀을 하나 만들자!
drawing_turtle = Turtle() # 이 터틀은 그림만 그리고 경주에는 참가 안 해!
drawing_turtle.speed(15) # 그림 그리는 속도!
```

궁금이 💡: random은 왜 불러왔어요? 거북이들이 똑같이 움직이면 재미없으니까, 무작위로 움직이게 하려는 건가요?

선생님: 예리한데! 맞아, random.randint(최소값, 최대값) 함수를 쓰면 특정 범위 안의 숫자를 무작위로 뽑아낼 수 있어. 이걸로 거북이들의 전진 거리를 정해줄 거란다.

선생님: 자, 이제 drawing_turtle로 경주장 제목을 쓰고, 멋진 경기장 트랙과 결승선을 그려보자!

```
# 타이틀 쓰기
drawing_turtle.penup() # 펜을 들고 이동해야 선이 안 그려지겠지?
drawing_turtle.goto(-130, 215) # 제목을 쓸 위치로 이동 (화면 위쪽 중앙쯤)
drawing_turtle.color("white") # 흰색 글씨로!
drawing_turtle.write("거북이 경주!", font=("Arial", 30, "bold")) # 글씨 쓰기!

# 운동장(경주 트랙) 그리기
drawing_turtle.goto(-screen_width/2 + 50, screen_height/2 - 50) # 트랙 시작 위치로 이동
drawing_turtle.pendown() # 이제 펜을 내리고 그림 시작!
drawing_turtle.color("peru") # 흙 색깔 트랙!
drawing_turtle.begin_fill() # 색칠 준비!
for i in range(2): # 사각형 그리는 반복문 (가로, 세로, 가로, 세로)
  drawing_turtle.forward(700) # 가로 700
  drawing_turtle.right(90)    # 오른쪽으로 90도 회전
  drawing_turtle.forward(400) # 세로 400
  drawing_turtle.right(90)
drawing_turtle.end_fill() # 색칠 끝!

# 결승선 그리기
# drawing_turtle.forward(600) # 이 부분은 원본 코드에서 트랙과 이어지므로, 위치 조정이 필요할 수 있어요.
#                             # 여기서는 트랙 오른쪽 끝에 결승선을 그린다고 가정할게요.
drawing_turtle.penup()
drawing_turtle.goto(300, screen_height/2 - 50) # 결승선 시작 위치 (트랙 오른쪽 위)
drawing_turtle.pendown()
drawing_turtle.color("black") # 검은색 결승선
drawing_turtle.pensize(5)     # 펜 두께는 5로!
```

```
drawing_turtle.right(90)      # 아래로 그리기 위해 방향 전환
drawing_turtle.forward(400)   # 트랙 높이만큼 쭉!
drawing_turtle.hideturtle()   # 그림 다 그린 drawing_turtle은 이제 숨기자!
```

코딩새싹 🌱: drawing_turtle.write()로 글씨도 쓸 수 있네요! hideturtle()은 거북이를 숨기는 마법인가요?

선생님: 맞아! 임무를 다한 거북이는 숨겨서 화면을 깔끔하게 만드는 거지.

<2. 경주 준비! 개성 만점 거북이 선수들 등장!>

선생님: 멋진 경주장이 완성됐으니, 이제 선수들을 불러 모아야겠지? 여러 마리의 거북이를 각기 다른 색깔과 출발선에 배치해볼 거야. 일일이 만드는 것보다 for 반복문과 enumerate를 쓰면 훨씬 똑똑하게 만들 수 있단다!

```
colors = ["blue", "pink", "purple", "red"] # 우리 선수들의 색깔 목록!
turtles = [] # 선수 거북이들을 담을 빈 리스트!

# enumerate를 사용하면 리스트의 번호표(i)와 내용물(color)을 한 번에 꺼낼 수 있어!
for i, color in enumerate(colors):
  turtle = Turtle() # 새 거북이 선수 등장!
  turtle.color(color) # 목록에서 꺼낸 색깔로 지정!
  turtle.shape("turtle") # 모양은 거북이!
  turtle.shapesize(1.5) # 크기는 1.5배로 살짝 크게!
  turtle.penup()
  # 출발선 위치 정하기: i가 0, 1, 2, 3으로 변할 때 y좌표가 150, 50, -50, -150으로 정렬되도록!
  turtle.goto(-300, 150 - (100 * i))
  turtle.pendown()
  turtles.append(turtle) # 완성된 거북이 선수를 turtles 리스트에 추가!
```

궁금이 💡: 150 - (100 * i) 이 부분 정말 clever(똑똑한데요)! i가 0일 땐 150, 1일 땐 50, 이렇게 거북이들이 서로 다른 레인에 설 수 있게 y좌표를 자동으로 계산해 주네요! enumerate 덕분에 번호표 i를 이렇게 쓸 수 있는 거고요!

선생님: 정확해! 이렇게 하면 거북이가 10마리가 되어도 코드는 거의 그대로겠지?

<3. 시작! 결승선을 향해 달려라! - while 반복문으로 경주 진행>

선생님: 자, 모든 준비가 끝났다! 이제 경주를 시작할 시간이야! "어떤 거북이 한 마리라도 결승선(x좌표 230 부근)에 닿을 때까지" 경주를 계속해야 하니, 이럴 땐 `while` 반복문이 제격이지!

```
# 결승선 x좌표 (화면 오른쪽 끝 근처)
finish_line_x = screen_width/2 - 70 # 예: 800/2 - 70 = 330 (코드는 230으로 되어있으니, 그림에 맞게 조절!)
# 원본 코드의 230을 사용한다고 가정
finish_line_x = 230

# 모든 거북이가 결승선 전에 있는 동안 계속 반복!
# 각 거북이의 x좌표는 turtle.xcor()로 알 수 있어.
game_is_on = True # 경기가 진행 중인지 알려주는 깃발
while game_is_on:
  # 모든 거북이가 결승선에 도달했는지 또는 넘어섰는지 확인
  all_turtles_finished = True # 일단 모두 끝났다고 가정
  for t in turtles:
      if t.xcor() < finish_line_x: # 아직 결승선 전인 거북이가 한 마리라도 있다면
          all_turtles_finished = False # 모두 끝난 게 아님!
          break # 더 확인할 필요 없음

  if all_turtles_finished: # 만약 모든 거북이가 결승선을 통과했다면 (이 경우는 거의 없음, 한 마리만 통과하
      game_is_on = False # 게임 종료 (실제로는 아래 승자 판별 로직에서 종료됨)
      break

  # 각 거북이를 무작위 거리만큼 앞으로 전진!
  for t in turtles:
    # 만약 한 거북이가 이미 결승선을 넘었다면, 더 이상 움직이지 않게 할 수도 있어
    # 여기서는 일단 모두 움직이게 하고, 아래에서 승자를 판별
    if t.xcor() < finish_line_x: # 아직 결승선 전인 거북이만 움직이게
        random_distance = random.randint(1, 15) # 1에서 15 사이의 무작위 숫자만큼!
```

```
        t.forward(random_distance)

    # 한 마리라도 결승선에 도달하면 게임을 멈추고 승자 판별로 넘어가야 해
    if t.xcor() >= finish_line_x:
        game_is_on = False # 깃발을 내려서 while 루프를 멈추도록!
        # break # for 루프를 빠져나감 (선택 사항, 없어도 while 조건 때문에 다음 턴에 멈춤)

  # (선택) 만약 모든 거북이가 동시에 결승선을 넘는 극적인 상황을 만들고 싶지 않다면,
  # 한 턴에 한 거북이만 결승선을 넘을 가능성이 높으므로, game_is_on 플래그만으로도 충분하다.
```

코딩새싹 🌱: `while game_is_on:` 이라는 조건으로 반복하고, 거북이들이 `random.randint(1, 15)`만큼 앞으로 가네요! 정말 누가 이길지 알 수가 없겠어요! `t.xcor()`는 거북이의 x좌표를 알려주는 건가요?

선생님: 맞아! 각 거북이의 x좌표(`t.xcor()`)가 결승선 x좌표(`finish_line_x`)보다 작을 동안은 계속 앞으로 나아가고, 한 마리라도 결승선을 넘으면 `game_is_on` 깃발이 `False`가 되면서 `while` 루프가 멈추게 된단다.

<4. 승자 발표! - if-elif-else로 우승 거북이 확인!>

선생님: `while` 루프가 끝났다는 건, 누군가 결승선을 통과했다는 뜻이지! 이제 어떤 색깔의 거북이가 우승했는지 확인하고 발표해야 해! 이럴 때 `if-elif-else`가 딱이지.

```
# 우승자를 발표하는 함수를 만들어두면 편하겠지?
def announce_winner(color, text):
  drawing_turtle.penup() # 아까 숨겼던 drawing_turtle을 다시 활용!
  drawing_turtle.goto(-150, 0) # 화면 중앙쯤에 메시지를 쓰자
  drawing_turtle.color(color)
  drawing_turtle.write(text, font=("Arial", 30, "bold"))
  drawing_turtle.showturtle() # 잠깐 보여줬다가 다시 숨길 수도 있고!

# 각 거북이의 최종 x좌표를 확인하여 우승자 판별
# (가장 멀리 간 거북이를 찾는 로직이 더 정확하지만, 여기서는 먼저 도달한 거북이 기준)
winner_found = False
for i, t in enumerate(turtles):
```

```
    if t.xcor() >= finish_line_x:
        winner_color = colors[i] # i번째 거북이의 색깔
        announce_winner(winner_color, f"{winner_color.capitalize()} 거북 승리!")
        winner_found = True
        break # 첫 번째 승자가 나오면 더 이상 확인할 필요 없음

# 만약 동시 도착 등으로 위 로직에서 승자가 안 가려졌다면 (실제로는 거의 발생 안 함)
if not winner_found:
    announce_winner("Gray", "아슬아슬 무승부!")
```

궁금이 💡: `colors[i]` 이렇게 해서 `enumerate`로 얻은 인덱스 `i`로 해당 거북이의 색깔을 바로 알 수 있네요! 그리고 `announce_winner` 함수로 승리 메시지를 보여주고요! 정말 멋진 게임이에요!

<5. 보너스 마법 복습!>

선생님: 이 게임을 만들면서 우리는 `while` 반복문, `enumerate`, 튜플 언패킹 같은 유용한 파이썬 마법들을 자연스럽게 사용했어. 잠깐 복습해볼까?

```
# while 반복문 복습 (game_on 깃발)
# cnt = 0
# game_on = True
# while game_on:
#   print(cnt)
#   cnt += 1
#   if cnt == 3: game_on = False # 0, 1, 2 출력 후 종료

# while 반복문 복습 (cnt < 조건)
# cnt = 0
# while cnt < 3: # 0, 1, 2 출력 후 종료
#   print(cnt)
#   cnt += 1
```

```
# enumerate 복습
# for i, color in enumerate(colors): # colors는 ["blue", "pink", "purple", "red"]
#   print(i, color) # 0 blue, 1 pink, 2 purple, 3 red

# 튜플 언패킹 복습
# name, age = ("왕새우", 1000)
# print(name) # 왕새우
# print(age)  # 1000
```

선생님: (깃발을 흔들며) 자, 오늘 우리는 터틀과 random, for, while, if, 함수, 리스트, enumerate 등 정말 많은 마법을 조합해서 멋진 거북이 경주 게임을 완성했어! 코드가 그림이 되고, 그림이 게임이 되는 놀라운 경험이었지? 이제 여러분은 어떤 아이디어든 파이썬으로 현실로 만들 수 있는 강력한 마법사가 되었단다! 정말 대단해! 🏆🐢