<[파이썬 제어 마법] 제1장. 조건이 만족될 때까지! while 반복문>

선생님: (빙글빙글 돌아가는 팽이를 보여주며) 얘들아, 안녕! 우리가 for 반복문은 리스트나 range()처럼 정해진 횟수만큼, 또는 컬렉션의 모든 항목을 훑어볼 때 아주 유용했지?

```
# for 반복문 복습: 리스트의 모든 항목을 순서대로 출력
# for i in [1, 2, 3, 4]:
#   print(i)
```

선생님: 그런데 만약 "정해진 횟수가 아니라, **어떤 조건이 만족되는 동안** 계속해서 특정 작업을 반복하고 싶다"면 어떨까? 예를 들어, "배터리가 다 닳을 때까지 로봇이 청소한다!" 라거나, "초록불이 켜져 있는 동안 계속 길을 건넌다!" 처럼 말이야.

코딩새싹 🌱: 오! for문은 몇 번 반복할지 미리 아는 느낌인데, 이건 조건에 따라 반복 횟수가 달라질 수 있겠네요!

선생님: 바로 그거야! 이럴 때 사용하는 마법 주문이 바로 **while 반복문**이란다! while은 "~하는 동안에" 라는 뜻을 가지고 있지.

<1. while 반복문 마법 주문 살펴보기!>

선생님: while 반복문의 기본 구조는 이렇단다.

```
while 조건문:
반복해서_실행할_코드
```

1. **while**: "지금부터 이 조건이 True인 동안 계속 반복하겠다!" 라고 선언하는 주문이야.
2. **조건문**: True 또는 False로 평가될 수 있는 조건이야. (우리가 if문에서 썼던 그 조건들!)
3. **콜론 (:)**: if나 for처럼, "자, 이제부터 반복할 작업 목록이야!" 라는 뜻.
4. **들여쓰기(Indentation)**: 콜론 다음 줄부터 **반드시 안으로 들여 써야 하는** 반복 작업 공간!

궁금이 💡: 그럼 while 뒤에 있는 조건이 True면 계속 반복하고, 언젠가 False가 되면 반복을 멈추는 거군요!

선생님: 정확해! 그럼 첫 번째 while 반복문을 만나볼까?

```
cnt = 0 # 1. 카운터 변수(cnt)를 0으로 시작!
running = True # 2. '계속 실행 중'이라는 깃발(running)을 True로 설정!
```

```
print("--- running 깃발을 이용한 while 루프 ---")
while running: # 3. running 깃발이 True인 동안 계속 반복!
  print("왕새우")
  cnt += 1 # 카운터를 1씩 증가
  print(cnt)
  if cnt == 5: # 4. 만약 카운터가 5가 되면,
    running = False # 깃발을 False로 바꿔서 다음번엔 멈추도록!

print("루프 끝!")
```

코딩새싹 🌱: 아하! running이라는 변수가 스위치 역할을 하는 거네요! cnt가 5가 되면 running이 False로 바뀌니까, while running: 조건이 더 이상 True가 아니게 돼서 반복이 멈추는 거군요!

궁금이 💡: 만약에 cnt += 1 이나 if cnt == 5: running = False 같은 부분이 없으면 어떻게 돼요?

선생님: 아주 중요한 질문이야! 만약 조건이 영원히 False로 바뀌지 않는다면, while 반복문은 영원히 끝나지 않는 **무한 루프**에 빠지게 된단다! 😱 그래서 while문을 쓸 때는 조건이 언젠가는 False가 되도록 만드는 장치를 꼭 넣어줘야 해!

<2. for 반복문처럼 쓰기: while 문으로 1부터 4까지 출력!>

선생님: 자, 그럼 이 while문으로 for i in [1, 2, 3, 4]: print(i)와 똑같은 결과를 만들어볼까?

```
# 방법 1: running 깃발 사용하기
print("\n--- while로 1~4 출력 (깃발 사용) ---")
num = 1 # 1. 숫자를 1부터 시작
running = True
while running:
  print(num)
  num += 1 # 숫자를 1씩 증가
  if num > 4: # 숫자가 4를 넘어가면
    running = False # 멈춤!

# 방법 2: 더 간결하게! 조건문에 직접 비교하기
```

```
print("\n--- while로 1~4 출력 (직접 비교) ---")
num = 1 # 다시 1부터 시작
while num < 5: # num이 5보다 작은 동안 계속! (즉, 1, 2, 3, 4까지)
  print(num)
  num += 1 # 숫자를 1씩 증가시켜서 언젠가는 num < 5 조건이 False가 되도록!
```

코딩새싹 🌱: 두 번째 방법이 훨씬 깔끔한데요! num < 5 라는 조건이 직접적이니까 이해하기 쉬워요!

<3. 탈출 마법! while True와 break!>

선생님: 때로는 "일단 무한히 반복하다가, 특별한 상황이 되면 그때 빠져나가고 싶어!" 할 때가 있어. 그럴 때 while True: 와 **break** 라는 탈출 마법을 함께 쓴단다. while True:는 조건이 항상 True니까 이론상 영원히 도는 루프를 만들지.

```
print("\n--- while True 와 break 로 0~4 출력 ---")
cnt = 0
while True: # 조건이 항상 True! 일단 무한 반복 시작!
  print(cnt)
  cnt += 1
  if cnt == 5: # 만약 cnt가 5가 되면,
    break # 여기서 'break!' 탈출 마법을 외치면 즉시 반복문을 빠져나간다!
print("탈출 성공!")
```

궁금이 💡: break는 반복문을 강제로 끝내는 비상 탈출 버튼 같은 거네요! while True랑 같이 쓰면, 반복 자체는 무한정 하되, 특정 if 조건에서 탈출하는 구조를 만들 수 있겠어요!

<4. for 반복문 vs. while 반복문: 언제 무엇을 쓸까?>

선생님: for와 while은 둘 다 반복을 위한 마법이지만, 약간의 쓰임새 차이가 있어.

- **for 반복문**: 리스트, 문자열, range()처럼 **반복할 횟수나 대상이 명확할 때** 주로 사용해. "이 컬렉션의 모든 항목에 대해 한 번씩~"
- **while 반복문**: **특정 조건이 만족되는 동안 계속 반복**하고 싶을 때, 또는 반복 횟수를 미리 알기 어려울 때 주로 사용해. "이 조건이 참인 동안에는 계속~" (예: 게임에서 사용자가 '종료'를 누를 때까지, 파일을 다 읽을 때까지 등)

선생님: (팽이를 다시 돌리며) 자, 오늘 우리는 `while` 반복문이라는 새로운 반복 마법을 배웠어. 조건에 따라 반복을 계속하거나 멈추고, `break`로 탈출하는 방법까지! 이제 여러분은 `for`와 `while`이라는 두 가지 강력한 반복 도구를 모두 다룰 수 있게 되었으니, 더 복잡하고 재미있는 프로그램도 만들 수 있을 거야! 오늘 수업도 정말 훌륭했어! 💫