<[파이썬 수학] 제8장. 똑똑한 반복 마법, for 루프!>

선생님: (똑같은 카드 100장을 보여주며) 얘들아, 안녕! 만약 이 100장의 카드에 적힌 내용을 하나씩 화면에 보여줘야 한다면 어떻게 할까? `print(카드[0])`, `print(카드[1])`, `print(카드[2])`… 이렇게 100번이나 코드를 쓰는 건 너무 힘들고 지루하겠지?

코딩새싹 🌱: 네! 생각만 해도 팔이 아파요! 😫 혹시 더 쉬운 방법이 있나요?

선생님: 물론! 이럴 때 쓰는 것이 바로 "알아서 순서대로 착착 처리해줘!" 라고 시키는 반복 마법, **for 반복문(for loop)**이란다! `for` 반복문은 리스트나 문자열처럼 순서가 있는 데이터의 첫 번째 항목부터 마지막 항목까지 하나씩 꺼내보면서, 우리가 시킨 일을 반복적으로 처리해준단다.

<1. for 반복문 마법 주문 파헤치기!>

선생님: `for` 반복문은 마치 마법 주문처럼 정해진 형식이 있어. 하나씩 자세히 뜯어보자!

```
for  chwilowy_pseudonim in kolekcja_danych:
작업_do_wykonania
```

궁금이 💡: `for`, `in`…? 조금 복잡해 보여요. 각 단어는 무슨 뜻이에요?

선생님: 아주 좋은 질문이야! 이 마법 주문을 우리가 조립하는 '자동 기계'에 비유해볼게.

```
for num in nums:
print(num ** 2)
```

1. **`for`**: "지금부터 반복 작업을 시작하겠다!" 라고 선언하는 주문이야.
2. **`num`**: 이건 우리가 **임시로 붙여주는 별명**이야. 기계가 `nums`라는 컨베이어 벨트에서 물건을 하나 집었을 때, 그 물건을 잠시 동안 "num"이라고 부르는 거지. `item`, `x` 등 어떤 이름이든 괜찮아!
3. **`in`**: "~안에서" 라는 뜻이야. 즉, "어디에 있는 물건들을 가져올 거냐면…" 하고 알려주는 거지.
4. **`nums`**: 바로 물건들이 놓여있는 **컨베이어 벨트**! 여기에는 리스트나 문자열처럼 순서가 있는 데이터가 온단다.
5. **콜론 (`:`)**: 이건 정말 중요해! "자, 이제부터 반복할 작업 목록을 알려줄게!" 라는 뜻의 마침표 같은 거야. 이 콜론을 빼먹으면 마법이 작동하지 않아!
6. **들여쓰기(Indentation)**: 콜론 다음 줄부터 **반드시 안으로 들여 써야 해!** (보통 스페이스 4칸 또는 Tab키 사용). 이렇게 들여쓰기 된 부분만이 반복될 작업 공간이야. 들여쓰기를 안 하면 컴퓨터는 어떤 일을 반복해야 할지 알 수가 없단다!

코딩새싹 🌱: 아하! `for num in nums:` 는 "nums 라는 컨베이어 벨트에서 물건을 하나씩 꺼내서, 그걸 num이라고 부를게!" 라는 뜻이고, 들여쓰기 된 `print(num ** 2)`가 그 물건(num)을 가지고 할 작업이군요!

<2. for 반복문 실제 사용해보기!>

선생님: 맞아! 그럼 이제 이 자동 기계가 얼마나 일을 잘하는지 직접 보자!

```
# 숫자 리스트를 반복하며 각 숫자를 제곱하기
nums = [1, 7, 3, 4, 5, 10]
print("--- 각 숫자의 제곱 ---")
for num in nums:
  print(num ** 2)

# 문자열을 반복하며 각 글자를 두 번씩 찍기
print("\n--- 글자 두 번씩 찍기 ---")
for letter in "나는 파이썬을 배우고 있습니다.":
  print(letter * 2)

# 문장이 담긴 리스트를 반복하며 각 문장을 3번씩 찍기
my_list = ["왕새우는 천재다!!!", "게다가 잘생겼어", "성격도 짱"]
print("\n--- 문장 세 번씩 외치기 ---")
for i in my_list:
  print(i * 3)

# 문장을 단어 단위로 쪼개서 반복하기
sentence = "나는 파이썬을 배우고 있습니다."
words = sentence.split() # 띄어쓰기로 쪼개서 단어 리스트를 만들고,
print("\n--- 단어 하나씩 보기 ---")
for word in words: # 그 리스트를 반복!
  print(word)
```

궁금이 💡: 와! for 반복문 하나로 숫자 리스트, 문자열, 단어 리스트까지 모두 다 처리할 수 있네요! 순서만

있다면 뭐든지 컨베이어 벨트에 올릴 수 있는 거군요!

<3. 결과를 차곡차곡 쌓는 '누적' 패턴!>

선생님: 맞아! for 반복문의 진짜 강력함은 단순히 하나씩 보여주는 것뿐만 아니라, 결과를 차곡차곡 쌓아서 최종 결과를 만들어낼 때 나타난단다. 이걸 '누적(accumulator) 패턴'이라고 해. 마치 돼지 저금통에 동전을 하나씩 넣는 것과 같아!

[미션 1] sum() 함수 없이 리스트의 총합 구하기!

선생님: 자, 이 숫자 리스트의 총합을 구하려면 어떻게 해야 할까? 먼저, 텅 빈 저금통이 하나 필요하겠지?

```
num_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
total = 0 # 1. 0원이 들어있는 텅 빈 저금통(total)을 준비!

# 2. 컨베이어 벨트(num_list)에서 동전(i)을 하나씩 꺼낸다!
for i in num_list:
  # 3. 꺼낸 동전을 저금통에 넣는다! (원래 있던 돈 + 새 동전)
  total = total + i # total += i 라고 줄여 쓸 수도 있어!

print("총 합계는?", total)
```

코딩새싹 🌱: 아하! total이라는 저금통을 0으로 만들어놓고, for문이 돌 때마다 i라는 동전을 계속 저금통에 더해주는 거군요!

[미션 2] 리스트에 있는 모든 숫자의 곱 구하기!

궁금이 💡: 그럼 모든 숫자를 곱하는 것도 비슷하게 할 수 있겠네요! 그런데… 곱하기를 할 때 저금통은 0으로 시작하면 안 될 것 같아요! 어떤 수든 0을 곱하면 0이 되어버리니까요!

선생님: (감탄하며) 이야, 정말 대단한 통찰력이야! 바로 그거란다. 덧셈의 저금통은 0에서 시작하지만, 곱셈의 저금통은 **1**에서 시작해야 해! 어떤 수든 1을 곱해야 자기 자신이 유지되니까.

```
num_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
product = 1 # 1. 곱셈을 위한 저금통은 반드시 1로 시작!

for i in num_list:
```

```
    product = product * i # product *= i 라고 줄여 쓸 수도 있어!

print("모두 곱한 값은?", product)
```

선생님: (흐뭇하게) 자, 오늘 우리는 파이썬의 가장 중요한 마법 중 하나인 `for` 반복문을 배웠어. `for` 별명 `in` 컨베이어벨트: 라는 주문과, 반복할 작업은 꼭 **들여쓰기**를 해야 한다는 규칙! 그리고 결과를 차곡차곡 쌓아가는 '누적 패턴'까지! 이제 여러분은 단순하고 반복적인 작업은 모두 컴퓨터에게 맡길 수 있는 강력한 마법사가 되었단다! 오늘 정말 수고 많았어! 🚀