〈[파이썬 알고리즘 탐험] 심화편 제2장: 경우의 수와 확률, itertools로 정복하기!〉

선생님: (여러 가지 색깔의 카드와 주사위를 보여주며) 얘들아, 안녕! 우리가 '5명의 학생 중 2명을 뽑아 줄을 세우는 경우'나 '주사위 2개를 던졌을 때 나올 수 있는 모든 조합' 같은 경우의 수를 계산하려면 머리가 조금 아플 수 있지? 파이썬에는 이런 모든 경우의 수를 마법처럼 쫙~ 뽑아내 주는 **itertools** 라는 아주 편리한 도구 상자가 있단다!

〈1. 순서가 중요해! 순열(Permutations)〉

선생님: 순열은 주어진 묶음에서 몇 개를 뽑아 '순서대로 나열'하는 모든 경우를 말해. 예를 들어, 반장-부반장을 뽑는 것처럼, (왕새우, 너구리)와 (너구리, 왕새우)는 서로 다른 경우로 취급되지!

```
import itertools # itertools 마법 도구 상자 소환!

# 1부터 5까지의 숫자 중 2개를 뽑아 줄을 세우는 모든 경우! (순서 중요!)
event_permutations = list(itertools.permutations(range(1, 6), 2))
print("--- 1~5 중 2개를 뽑는 순열 ---")
print(event_permutations)
print(f"경우의 수: {len(event_permutations)}가지") # 5 * 4 = 20가지

# "왕새우", "너구리", "고래" 중 2명을 뽑아 줄 세우기!
event_names = list(itertools.permutations(["왕새우", "너구리", "고래"], 2))
print("\n--- 3명 중 2명을 뽑는 순열 ---")
print(event_names)
```

코딩새싹 🌱: 와! `itertools.permutations()` 하니까 정말 모든 경우를 다 보여주네요! (1, 2)도 있고 (2, 1)도 있어요! 순서가 정말 중요하군요!

〈2. 각 그룹에서 하나씩! 데카르트 곱(Product)〉

선생님: 이번엔 **데카르트 곱**이라는 건데, 이건 각 그룹에서 하나씩 뽑아서 만들 수 있는 모든 짝꿍을 찾는 거야. 예를 들어, 주사위 두 개를 던지는 것처럼 말이지! 첫 번째 주사위의 눈과 두 번째 주사위의 눈은 서로에게 영향을 주지 않지?

```
# 주사위 2개를 던져서 나올 수 있는 모든 경우!
# 첫 번째 주사위 눈: [1, 2, 3, 4, 5, 6]
# 두 번째 주사위 눈: [1, 2, 3, 4, 5, 6]
# 이 두 그룹에서 하나씩 뽑아 만들 수 있는 모든 짝꿍!
event_product = list(itertools.product(range(1, 7), range(1, 7)))
print("\n--- 주사위 2개를 던졌을 때 모든 경우 ---")
print(event_product)
print(f"경우의 수: {len(event_product)}가지") # 6 * 6 = 36가지
```

궁금이 💡: product는 순열이랑 조금 다르네요! (1, 1)처럼 같은 숫자가 다시 나올 수도 있고, 각 그룹에서 하나씩 뽑는다는 느낌이 강해요! 마치 옷장에서 셔츠 하나, 바지 하나를 고르는 것처럼요!

<3. 순서는 상관없어! 조합(Combinations)>

선생님: 마지막으로 **조합**은 순서에 상관없이 그냥 '뽑기만' 하는 모든 경우를 말해. 예를 들어, 4명의 친구 중 청소 당번 2명을 뽑는 것처럼, (A, B)를 뽑는 것과 (B, A)를 뽑는 것은 결국 똑같은 청소 당번 조가 되는 거지!

```
# 1부터 4까지 숫자 중 2개를 순서에 상관없이 뽑는 모든 경우!
event_combinations = list(itertools.combinations(range(1, 5), 2))
print("\n--- 1~4 중 2개를 뽑는 조합 ---")
print(event_combinations)
print(f"경우의 수: {len(event_combinations)}가지") # (4 * 3) / 2 = 6가지
```

코딩새싹 🌱: 아! 이번엔 (1, 2)는 있는데 (2, 1)은 없네요! 순서가 상관없으니까 하나로만 취급하는 거군요! 순열, 데카르트 곱, 조합… 이제 차이를 알겠어요!

<최종 도전! 주사위 합이 소수가 될 확률은?>

선생님: 자, 이제 이 마법 도구들을 이용해서 실제 확률 문제를 풀어보자! **"주사위를 두 번 던져서 나온 눈의 합이 소수(prime number)가 될 확률은?"**

궁금이 💡: 확률을 구하려면, (우리가 원하는 사건의 경우의 수) / (일어날 수 있는 모든 경우의 수)를 계산해야 해요!

1단계: 모든 경우의 수 구하기 (표본 공간)
선생님: 주사위를 두 번 던졌을 때 나올 수 있는 모든 경우는 방금 뭘로 구했지?

코딩새싹 🌱: itertools.product요! 총 36가지 경우가 있었어요!

```
total_event = list(itertools.product(range(1, 7), range(1, 7)))
```

2단계: 우리가 원하는 사건(두 눈의 합이 소수)의 경우의 수 구하기
선생님: 이제 저 36가지 경우를 하나씩 보면서, 두 눈의 합이 소수인지 확인하면 되겠지? 우리 '소수 판별 탐정 함수'가 다시 활약할 시간이군!

```
# 소수 판별 함수 (복습)
def is_prime(n):
  if n < 2: return False # 1과 자기 자신만 약수로 가지려면, 2보다 작으면 안돼!
  for i in range(2, n):
    if n % i == 0:
      return False # 중간에 약수가 발견되면 소수가 아님!
  return True # 모든 검사를 통과하면 소수!

# 방법 1: 중첩 for 문으로 직접 구하기 (itertools 없이)
# add_primes = []
# for i in range(1, 7):
#   for j in range(1, 7):
#     if is_prime(i + j):
```

```
#       add_primes.append((i, j))
# print(len(add_primes)) # 15가지

# 방법 2: itertools와 리스트 내포로 훨씬 멋지게!
# total_event (36가지 모든 경우)에서,
# 각 경우(i)의 합(sum(i))이 소수(is_prime)인 경우만 골라내줘!
prime_sum_events = [i for i in total_event if is_prime(sum(i))]

print("\n--- 두 눈의 합이 소수인 경우들 ---")
print(prime_sum_events)
print(f"경우의 수: {len(prime_sum_events)}가지") # 15가지
```

3단계: 확률 계산하기!

선생님: 자, 모든 경우의 수는 36가지, 우리가 원하는 사건의 경우의 수는 15가지! 그럼 확률은?

궁금이 💡: 15 / 36 이네요! Fraction으로 표현하면 더 깔끔하겠어요!

```
from fractions import Fraction # 분수 마법사 소환!

probability = Fraction(len(prime_sum_events), len(total_event))
print(f"\n두 눈의 합이 소수가 될 확률은 {len(prime_sum_events)} / {len(total_event)} = {probability} 입니
```

선생님: (흐뭇하게 웃으며) 완벽해! 오늘 우리는 itertools라는 강력한 마법 도구 상자를 열어서 순열, 데카르트 곱, 조합이라는 경우의 수의 세 가지 보물을 손에 넣었어. 그리고 이걸 이용해서 복잡한 확률 문제까지 아주 우아하게 해결했지! 이렇게 파이썬 도구를 활용하면, 복잡한 수학 문제도 마치 게임처럼 즐겁게 풀어나갈 수 있단다! 오늘 탐험도 정말 대단했어! 💯