<[파이썬 알고리즘 탐험] 제7장. 수학 문제 자판기 만들기! 클래스로 모든 것을 하나로!>

선생님: (작은 자판기 모형을 보여주며) 얘들아, 안녕! 이 자판기는 동전을 넣고 버튼을 누르면 우리가 원하는 음료수가 뽕! 하고 나오지? 오늘 우리는 파이썬 클래스를 이용해서, 버튼을 누르면 수학 문제가 뽕! 나오고, 또 다른 버튼을 누르면 정답이 뽕! 하고 나오는 '수학 문제 자판기'를 만들어 볼 거야!

코딩새싹 🌱: 와! 우리가 직접 수학 문제 내는 기계를 만드는 거예요? 정말 신기한데요!

선생님: 그럼! 이 자판기를 만들려면 먼저 '설계도(클래스)'가 필요하겠지? 'NaturalNumber'라는 이름의 설계도를 한번 살펴보자. 그리고 이 설계도 안에서는 우리가 예전에 만들었던 소수 판별, 약수 구하기, 최대공약수, 최소공배수 함수들을 재료처럼 사용할 거란다!

<1. 자판기의 핵심 부품들: 우리가 만든 함수들 복습!>

선생님: 먼저, 우리 자판기가 문제를 풀 때 사용할 핵심 알고리즘 함수들을 다시 한번 떠올려볼까? (넘파이의 sqrt를 사용한 is_prime 함수 포함)

```
import numpy as np # 넘파이의 강력한 수학 도구와 랜덤 기능을 사용하기 위해!

def is_prime(n): # 소수 판별 함수
  if n < 2: return False
  for i in range(2, int(np.sqrt(n)) + 1): # 제곱근까지만 확인!
    if n % i == 0: return False
  return True

def divisors(n): # 약수 구하는 함수
  result = []
  for i in range(1, int(n ** 0.5) + 1):
    if n % i == 0:
      result.append(i)
      if i * i != n: result.append(n//i) # 짝꿍 약수 추가 (중복 방지)
  result.sort()
  return result

def gcd(a, b): # 최대공약수 (유클리드 호제법)
  while b != 0: a, b = b, a % b
  return a

def lcm(a, b): # 최소공배수 (GCD 활용!)
  return int(a * b / gcd(a, b)) # a*b를 gcd로 나눈 몫!
```

궁금이 💡: 아! is_prime 함수에서 np.sqrt()를 쓰니까 더 넘파이스러워졌네요! 이 함수들이 우리 자판기의 '두뇌' 역할을 하겠군요!

<2. 자판기 설계도: NaturalNumber 클래스 파헤치기!>

선생님: 자, 이제 이 함수들을 사용하는 NaturalNumber 클래스 설계도를 자세히 보자!

```
class NaturalNumber:
  # 1. 자판기가 만들어질 때(인스턴스 생성 시) 실행되는 초기 설정! (__init__)
  def __init__(self):
    # 어떤 종류의 문제를 낼지 무작위로 선택 (0: 소수/약수, 1: 최대공약수, 2: 최소공배수)
    self.select = np.random.randint(3) # 0, 1, 2 셋 중 하나의 숫자를 뽑아줘!
    # 문제에 사용할 숫자들도 1부터 99 사이에서 무작위로 뽑자!
    self.n1 = np.random.randint(1, 100)
    self.n2 = np.random.randint(1, 100) # n2는 최대공약수/최소공배수 문제에만 쓰여.
    print("✨ 새로운 수학 문제 자판기 준비 완료! ✨")

  # 2. "문제 내줘!" 버튼 (question 메소드)
  def question(self):
    print("\n--- 오늘의 문제! ---")
    if self.select == 0: # 만약 0번 문제가 선택됐다면,
      print(f"문제: {self.n1}이(가) 소수인지 판별하고, 소수가 아니라면 약수를 모두 구하세요.")
    elif self.select == 1: # 만약 1번 문제가 선택됐다면,
      print(f"문제: 두 수 {self.n1}와(과) {self.n2}의 최대공약수를 구하세요.")
    else: # 그 외에는 (2번 문제가 선택됐다면)
      print(f"문제: 두 수 {self.n1}와(과) {self.n2}의 최소공배수를 구하세요.")

  # 3. "정답 알려줘!" 버튼 (answer 메소드)
  def answer(self):
    print("\n--- 정답 공개! ---")
    if self.select == 0: # 0번 문제의 정답은?
      if is_prime(self.n1): # is_prime 함수로 n1이 소수인지 확인!
        print(f"정답: {self.n1}은(는) 소수입니다.")
      else:
        divs = divisors(self.n1) # divisors 함수로 n1의 약수 구하기!
        print(f"정답: {self.n1}은(는) 소수가 아닙니다.")
        print(f"      {self.n1}의 약수는 {divs} 입니다.")

    elif self.select == 1: # 1번 문제의 정답은?
      # gcd 함수로 최대공약수 구하기!
      print(f"정답: {self.n1}와(과) {self.n2}의 최대공약수는 {gcd(self.n1, self.n2)}입니다.")

    else: # 2번 문제의 정답은?
```

```
        # lcm 함수로 최소공배수 구하기!
        print(f"정답: {self.n1}와(과) {self.n2}의 최소공배수는 {lcm(self.n1, self.n2)}입니다.")
```

코딩새싹 🌱: `__init__` 함수에서 `np.random.randint()`로 `self.select`, `self.n1`, `self.n2`를 무작위로 정하니까, 자판기를 만들 때마다 다른 문제가 나오겠네요! 정말 자판기 같아요!

궁금이 💡: 그리고 `question` 메소드랑 `answer` 메소드 안에서 `self.select` 값에 따라 `if-elif-else`로 다른 문제를 내거나 다른 정답 계산법을 사용하는군요! `self.n1`이나 `self.n2`는 자판기(인스턴스)가 만들어질 때 정해진 그 숫자들이고요!

선생님: 바로 그거야! `self`를 통해 각 자판기 인스턴스는 자신만의 문제 유형(`self.select`)과 숫자들(`self.n1`, `self.n2`)을 기억하고, 그에 맞는 질문과 답변을 할 수 있는 거지. 마치 자판기마다 다른 음료수가 들어있는 것처럼!

<3. 수학 문제 자판기, 실제로 사용해보기!>

선생님: 자, 그럼 이 `NaturalNumber` 설계도로 실제 자판기(인스턴스)를 하나 만들어서 문제를 풀어볼까?

```
# 수학 문제 자판기 q1을 하나 만든다! (이때 __init__이 실행되면서 문제가 준비됨)
q1 = NaturalNumber()

# q1 자판기야, 문제 내줘!
q1.question()

# q1 자판기야, 정답 알려줘!
q1.answer()

print("\n--- 또 다른 문제! ---")
# 새로운 자판기 q2를 만들면, q1과는 다른 문제가 나오겠지?
q2 = NaturalNumber()
q2.question()
q2.answer()
```

코딩새싹 🌱: 와! `q1 = NaturalNumber()` 하니까 바로 "새로운 수학 문제 자판기 준비 완료!" 메시지가 뜨고, `q1.question()`이랑 `q1.answer()`를 부르니까 알아서 다른 문제랑 답이 나와요! `q2`는 또 다른 문제고요!

궁금이 💡: 이렇게 클래스로 만들어두니까, 우리가 예전에 만들었던 함수들을 마치 부품처럼 가져다가 더 크고 재미있는 프로그램을 만들 수 있게 되는 거군요! 정말 멋져요!

선생님: (흐뭇하게 웃으며) 맞아! 오늘 우리는 지금까지 배운 알고리즘 함수들을 클래스라는 멋진 틀 안에 담아서, 스스로 문제를 내고 푸는 '수학 문제 자판기'를 만들어봤어. 이렇게 클래스를 이용하면 관련된 데이터(숫자들, 문제 유형)와 기능(문제 내기, 답하기)을 하나로 깔끔하게 묶어서 관리할 수 있단다. 이게 바로 객체 지향 프로그래밍의 강력한 힘 중 하나지!

오늘 "파이썬 알고리즘 탐험"의 마지막 장까지 함께한 여러분, 정말 대단해! 이제 여러분은 파이썬으로 생각하고, 문제를 해결하고, 심지어 새로운 세계를 창조하는 진짜 마법사가 되었단다! 🧙‍♀️⊠🎉