<[파이썬 알고리즘 탐험] 제3장. 숫자의 단짝 친구, 약수와 공약수 대작전!>

선생님: (여러 개의 숫자 카드와, 각 숫자를 나눌 수 있는 작은 숫자 카드 묶음들을 보여주며) 얘들아, 안녕! 어떤 숫자를 나누었을 때 나머지가 없이 딱! 떨어지게 만드는 숫자들을 뭐라고 부를까?

코딩새싹 🌱: 약수요! 예를 들어 6의 약수는 1, 2, 3, 6 이잖아요!

선생님: 정확해! 오늘은 어떤 숫자를 주면 그 숫자의 모든 약수, 즉 모든 '단짝 친구'들을 찾아주는 함수를 만들어 볼 거야. 자, 여기 첫 번째 시도가 있네! 우리 한번 살펴볼까?

<1. 약수 찾기 첫 번째 시도! (어, 코드가 좀 이상한데요?)>

```
# 어떤 수의 약수를 구하는 함수를 작성하세요. (첫 번째 버전)
def divisors1(n):
  result = []
  for i in range(1, n + 1): # 1부터 n까지의 숫자로 나눠보자!
    if n % i == 0: # 만약 n을 i로 나눈 나머지가 0이라면, i는 약수!
      result.append(112212496) # 어? 여기에 왜 이 큰 숫자가...?!
  return result

# divisors_of_number = divisors1(12) # 만약 12의 약수를 찾으면...?
# print(divisors_of_number)
```

궁금이 💡: (코드를 유심히 보며) 선생님! 저 `result.append(112212496)` 부분, 뭔가 이상해요! `i`가 약수일 때, 약수인 `i`를 리스트에 넣어야 하는 거 아니에요? 저렇게 하면 12의 약수를 찾아도 리스트에는 `[112212496, 112212496, ...]` 이런 이상한 숫자만 잔뜩 들어갈 것 같아요!

선생님: (빙긋 웃으며) 역시 우리 궁금이 탐정! 예리하게 코드 속의 수상한 단서(?)를 찾아냈구나! 맞아. 저 부분은 분명히 약수인 `i`를 넣으려다 실수한 것 같아. 우리가 직접 고쳐볼까? `result.append(i)` 이렇게 말이지!

코딩새싹 🌱: 아하! 그렇게 고치면 제대로 된 약수들이 리스트에 담기겠네요! 그런데 1부터 `n`까지 모든 숫자를 다 확인하면, `n`이 엄청 큰 수일 때는 시간이 너무 오래 걸리지 않을까요?

선생님: 좋은 지적이야! 마치 소수를 찾을 때처럼, 약수를 찾는 데도 더 똑똑한 방법이 있단다!

<2. 약수 찾기 업그레이드! (제곱근 마법 활용!)>

선생님: 소수를 찾을 때 우리가 숫자의 '제곱근'까지만 확인했던 것 기억나니? 약수를 찾을 때도 비슷한 마법을

쓸 수 있어! 만약 i가 n의 약수라면, n을 i로 나눈 몫, 즉 n // i도 반드시 n의 약수가 된단다! 마치 단짝 친구처럼 쌍으로 나타나는 거지!

```
# 위 함수 개선 (제곱근까지만 확인!)
def divisors(n):
  result = []
  # 1부터 숫자의 제곱근까지만 확인하자!
  for i in range(1, int(n ** 0.5) + 1):
    if n % i == 0: # i가 약수라면,
      result.append(i)       # i를 추가하고,
      if i * i != n: # 만약 i의 제곱이 n이 아니라면 (중복 방지!)
        result.append(n // i) # 그 짝꿍 약수인 n // i 도 추가!
  result.sort() # 약수들을 크기순으로 예쁘게 정렬!
  return result

print("--- 1234의 약수들 ---")
a = divisors(1234)
print(a)

print("\n--- 5620의 약수들 ---")
b = divisors(5620)
print(b)
```

궁금이 💡: 와! i를 찾으면 그 짝꿍인 n // i도 같이 추가하니까 훨씬 빠르겠어요! if i * i != n: 이 조건은 왜 있는 거예요?

선생님: 만약 n이 36처럼 제곱수일 때, i가 6이면 n // i도 6이잖아? 그럴 때 6을 두 번 추가하지 않도록 막아주는 안전장치란다! (사실, 이 코드는 set을 사용하면 중복을 더 쉽게 처리할 수 있지만, 지금은 이 원리를 이해하는 게 중요해!)

<3. 두 숫자의 공통된 단짝 친구들 찾기! (공약수)>

선생님: 자, 이제 숫자 1234의 약수 리스트 a와 숫자 5620의 약수 리스트 b를 얻었어. 그럼 이 두 숫자가 공통으로

가지고 있는 약수, 즉 **공약수**는 어떻게 찾을 수 있을까?

코딩새싹 🌱: 앗! 이건 우리가 지난번에 리스트 두 개의 교집합을 구하는 거랑 똑같은데요!

선생님: 맞아! a 리스트의 약수들을 하나씩 보면서, 그 약수가 b 리스트 안에도 있는지 확인하면 되겠지!

```
# 두 리스트의 교집합(공통 원소)을 구하는 함수 (복습!)
def intersect(list1, list2):
  common_elements = []
  for item in list1:
    if item in list2:
      common_elements.append(item)
  return common_elements

# 1234와 5620의 공약수 구하기
common_divs = intersect(a, b)
print("\n--- 1234와 5620의 공약수들 ---")
print(common_divs)
```

<4. 공통된 친구 중 가장 큰 형님! (최대공약수)>

선생님: 그럼 이 공약수들 중에서 가장 큰 값, 즉 **최대공약수(GCD)**는 어떻게 찾을까?

궁금이 💡: 공약수들이 담긴 리스트에서 가장 큰 값을 찾는 함수를 쓰면 되겠어요! `max()` 함수요!

선생님: 정답! 아주 간단하지?

```
# 공약수 리스트 common_divs 에서 가장 큰 값 찾기!
greatest_common_divisor = max(common_divs)
print("\n--- 1234와 5620의 최대공약수 ---")
print(greatest_common_divisor)

# max() 함수 복습!
print("\nmax([1, 10, 5, 3])의 결과는?", max([1, 10, 5, 3]))
```

코딩새싹 🌱: 와! 우리가 직접 만든 `divisors` 함수랑 `intersect` 함수, 그리고 파이썬의 `max` 함수를 합치니까

최대공약수까지 구할 수 있네요! 정말 뿌듯해요!

선생님: (흐뭇하게 웃으며) 그렇지? 오늘 우리는 숫자의 단짝 친구인 약수를 찾는 방법, 그것도 아주 똑똑한 방법으로 찾아내는 법을 배웠어. 그리고 두 숫자가 함께 아는 친구들인 공약수와 그중 가장 큰 형님인 최대공약수까지 찾아냈지! 이렇게 작은 문제들을 해결하는 함수들을 만들고, 그것들을 조립해서 더 큰 문제를 해결하는 것이 바로 프로그래밍의 큰 즐거움 중 하나란다! 오늘 탐험도 정말 훌륭했어! 🌟