Euler 0032

The Problem:

Pandigital numbers are numbers that have 1-n shown exactly once.
7254 is unusual because 39 * 186 = 7254, which is pandigital through the whole calculation.

What is the sum of all products that can generate these pandigital calculations?

Considerations and Approach:

Seeing that it is 1-9 pandigital, it would be good to skip multiplicand and multipliers that contain 0s.

Assuming a * b = c, we can increment b up until the total digit length is > 9, then we increment a and reset b back to a. When b is reset and a is incremented we can check if total digit length is > 9 and break

The Code:

```
break_threshold = 1000000000 #just something comfy to break

cycles = 0
pands = set()
a = 1
b = 1

#only break if the len of the all the digits is greater than 9 (pigeonhole)
while cycles < break_threshold and len(str(a)) + len(str(b)) + len(str(a*b)) <= 9:
    current_str = str(a) + str(b) +str(a*b)
    #iterate b until the current string length exceeds 9
```

```
    while len(current_str) <= 9:
        if len(current_str) == 9:
            #we don't accept 0s, we are looking for 1-9
            if '0' not in current_str:
                #all digits appear once so the sum of counts should be 9 (9 numbers)
                if sum([current_str.count(x) for x in current_str]) == 9:
                    pands.add(a*b)
        b += 1
        current_str = str(a) + str(b) + str(a*b)

    #increment the cycle
    a += 1
    b = a
    cycles += 1

print(pands, cycles)
print(sum(pands))
```

Revision #3
Created 2026-02-26 21:37:33 UTC by Maxwell
Updated 2026-02-26 21:46:45 UTC by Maxwell