```
""" -*- coding: utf-8 -*-
If we all list the nutural numbers beloe 10 that are multiples of 3 or 5 we get
3,5,6,9. the sum of these multiples is 23.
find the sum of the all the multiples of 3 or 5 below 1000.
"""
sum_=0
for i in range(1,1000):
    if i%3==0 or i%5==0:
        sum_=sum_+i
print sum_

# The answer is  233168


"""
using list comprehension in python:
>>> sum([x for x in range(1000) if x % 3== 0 or x % 5== 0])
233168
"""




"""
Simple version in Python, using sets.
sum(set(range(0,1000,3))|set(range(0,1000,5)))
"""





"""
Numbers divisible by 33 follow sequence 3,6,9,12,15⋯3,6,9,12,15⋯ or 3n3n
Numbers divisible by 55 follow sequence 5,10,15,20,25⋯5,10,15,20,25⋯ or 5n5n
Numbers divisible by 1515 follow sequence: 15,30,45,60,75⋯15,30,45,60,75⋯ or 15n15n


We can add the numbers divisible by 33 to the numbers divisible by 55, and then
subtract the numbers divisible by 1515 to compensate for doublecounting.

Let's look at the 33's as an example. If we wanted to add up consecutive terms that
are divisible by 33, notice that we're adding together a bunch of 3n3n terms:
3(1)+3(2)+3(3)+3(4)+3(5)+⋯
3(1)+3(2)+3(3)+3(4)+3(5)+⋯

which is the same as
3(1+2+3+4+5+⋯)
3(1+2+3+4+5+⋯)
```

That inner sum can be simplified. The sum of consecutive integers from 11 to cc is:
∑k=1ck=c(c+1)2
∑k=1ck=c(c+1)2

The next logical question to ask is "Okay, but how do we know what cc should be?" cc, in this case, represents the count of numbers under NN divisible by dd. This is c=⌊N−1d⌋c=⌊N−1d⌋.

Finally, we plug in 10001000 for NN and solve the summations for dd = 33, 55, and 1515.

The sum of all 3n3n is 3t(t+1)23t(t+1)2 where t=⌊1000−13⌋t=⌊1000−13⌋
The sum of all 5n5n is 5f(f+1)25f(f+1)2 where f=⌊1000−15⌋f=⌊1000−15⌋
The sum of all 15n15n is 15x(x+1)215x(x+1)2 where x=⌊1000−115⌋x=⌊1000−115⌋

Add the first two of these, and subtract the third.

Python Code

```
def PE1(N):
    t=(N-1)//3; f=(N-1)//5; x=(N-1)//15
    return 3*t*(t+1)/2 + 5*f*(f+1)/2 - 15*x*(x+1)/2

print PE1(1000)

#233168
```

The answer follows a nice pattern, too, for powers of 1010:

103:ans=233168103:ans=233168
104:ans=23331668104:ans=23331668
105:ans=2333316668105:ans=2333316668
106:ans=233333166668106:ans=233333166668
107:ans=23333331666668107:ans=23333331666668
108:ans=2333333316666668108:ans=2333333316666668
109:ans=233333333166666668109:ans=233333333166666668
1010:ans=233333333316666666681010:ans=23333333331666666668
etc
"""

```
"""
def PE1(a, b, n):
    def f(x, n):
        fl = int((n-1)/x)
        return x * fl * (fl+1)
    return (f(a,n) + f(b,n) - f(a*b,n))/2
"""




"""
big = list(range(1, 1000))
new = []
for i in big:
    if i % 3 == 0 or i % 5 == 0:
        new.append(i)

print(sum(new))"""




"""
Probably not the first to post an answer in Python but here you go:
I'm just happy that I've found some problems to solve with programming.
#declare value
value = 0
#test each integer between 0 and 1000, not including 1000.
for i in range(1, 1000):
    # if meets criteria add i to value
    if i % 3 == 0 or i % 5 == 0:
        value += i
# when all integers are processed print value a.k.a answer.
print(value)

# same thing in one line.
# this is in no way smarter, it just uses fewer lines and is a pain to read.
print(sum([i for i in range(1, 1000) if i % 3 == 0 or i % 5 == 0]))
"""
```

```
"""
Solved with Python:

x = 1
y = 0

while x < 1000:
    if x%3 == 0 or x%5 == 0:
        y = y+x
    x+=1

print y
"""
```