def print_output():
    total = 0
    for number in range(9, 20):
        if number % 2 == 0 or number % 3 == 0:
            total = total + 1
    print(total)

def main():
    print_output()

main()
Complete the loops

Complete the for...in range() loop so that the output is:

for number in range():
    print(number, end = " ")
print()

Complete the for...in range() loop so that the output is:

for value in range():
    print(value, end = " ")
print()
An amount doubles each year. Using a for...in range() loop complete the `double_each_year()` function which prints the growth of the parameter, (start_amt) for the given number of years (num_years). The first line printed by the function is the starting amount.

Each line of the output is numbered starting from the number 1. The function returns the final amount.

```python
def double_each_year(start_amt, num_years):

def main():
    print("After 4 years: ", double_each_year(24, 4))
    print("After 3 years: ", double_each_year(235, 3))
    print("After 5 years: ", double_each_year(15, 5))

main()
```

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>After 4 years: 384</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Starting with: 235</td>
<td>1: 470</td>
<td>2: 940</td>
<td>3: 1880</td>
<td></td>
</tr>
<tr>
<td></td>
<td>After 3 years: 1880</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Starting with: 15</td>
<td>1: 30</td>
<td>2: 60</td>
<td>3: 120</td>
<td>4: 240</td>
</tr>
<tr>
<td></td>
<td>After 5 years: 480</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Complete the function

Using a for...in range() loop complete the print_series() function which prints a series of numbers starting from the parameter value, start_num. The second number printed is the first number plus 1, the third number is the second number plus 2, the fourth number is the third number plus 3, and so on, e.g., a series of 8 numbers starting from the number 2 is:

```
2   3   5   8   12  17  23  30
+1  +2  +3  +4  +5  +6  +7
```

```
def print_series(start_num, how_many):
    # Implementation

def main():
    print_series(2, 8)
    print_series(5, 12)
    print_series(16, 9)
main()
```
Convert the following while loop into a for...in range() loop:

counter = 12
while counter < 260:
    print(counter)
    counter = counter + 10

Convert the following for...in range() loop into a while loop:

for num in range(45, 3, -5):
    print(num * 2)
Do the following two loops give the same output? If not, what is the difference in output and what change needs to be made in order to make the outputs of the two loops the same?

```
top = 6
count = 0
total = 0

for bottom in range(0, top + 1, 2):
    count = count + 1
    total = total + top + bottom

print("count:", count,"sum:", total)
```

```
top = 6
bottom = 0
count = 0
total = 0

while bottom < top:
    count = count + 1
    total = total + top + bottom
    bottom = bottom + 2

print("count:", count,"sum:", total)
```
Complete the function

A perfect number is an integer that is equal to the sum of its divisors (including 1, excluding the number itself), e.g., $28 = 1 + 2 + 4 + 7 + 14$. Complete the `get_sum_of_divisors()` function using a `for...in range()` loop for the iteration.

```python
def get_sum_of_divisors(number):
    pass

def check_perfection(number):
    if number == get_sum_of_divisors(number):
        print(number, "is a perfect number")
    else:
        print(number, "is NOT a perfect number")

def main():
    check_perfection(28)
    check_perfection(54)
    check_perfection(496)

main()
```

28 is a perfect number
54 is NOT a perfect number
496 is a perfect number
The `get_series_sum()` function returns the sum of the desired number of terms of the series:

\[
\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots.
\]

E.g., `get_series_sum(4)` returns the sum of one half plus one quarter plus one eighth plus one sixteenth. Complete the function

```python
def get_series_sum(num_terms):
    # Insert the function implementation here

def main():
    for num in range(1, 10):
        comment = f"Terms: {num}, sum:" + str(get_series_sum(num))
        print(comment, get_series_sum(num))
main()
```