Question 1

Write a program which will find all numbers which are **divisible by 7** but are not a multiple of 5, between 2000 and 3200 (both included).

The numbers obtained should be printed in a comma-separated sequence on a single line.

Hints:

Consider use range(#begin, #end) method

Question 2

Write a program which can compute the **factorial** of a given number.

The results should be printed in a comma-separated sequence on a single line.

Suppose the following input is supplied to the program:

8

Then, the output should be:

40320

Hints:

e.g Factorial 8 is calculated as 8x7x6x5x4x3x2x1

In case of input data being supplied to the question, it should be assumed to be a console input.

Question 3

Write and test a function which takes one argument (a year) and returns **True** if the year is a leap year, or **False** otherwise. The seed of the function is already sown in the skeleton code (below).

Note: we've also prepared a short testing code, which you can use to test your function. The code uses two lists – one with the test data, and the other containing the expected results. The code will tell you if any of your results are invalid.

```
def IsYearLeap(year):
#
# put your code here
#

testdata = [1900, 2000, 2016, 1987]
testresults = [False, True, True, False]

for i in range(len(testdata)):
     yr = testdata[i]
     print(yr,"->",end="")
     result = IsYearLeap(yr)
if result == testresults[i]:
     print("OK")
else:
     print("Failed")
```

Question 4

Write and test a function which takes two arguments (a year and a month) and returns the number of days for the given month/year pair (yes, we know that only February is sensitive to the year value, but we want our function to be universal). The initial part of the function is ready. Now, convince the function to return None if its arguments don't make sense.

Of course, you can (to be honest: you should!) use the previously written and tested function. It may be very helpful (we cannot say anything more, sorry). We encourage you to use a list filled with the months' lengths. You can create it inside the function – this trick will significantly shorten the code.

We've prepared a testing code. Expand it to include more test cases.

```
def IsYearLeap(year):
#
# your code is already here
#
def DaysInMonth(year,month):
#
# put your new code here
#
testyears = [1900, 2000, 2016, 1987]
testmonths = [ 2, 2, 1, 11]
testresults = [28, 29, 31, 30]
for i in range(len(testyears)):
    yr = testyears[i]
    mo = testmonths[i]
    print(yr,mo,"->",end="")
    result = DaysInMonth(yr,mo)
if result == testresults[i]:
    print("OK")
else:
    print("Failed")
```

Question 5

Write and test a function which takes three arguments (a year, a month, and a day of the month) and returns the corresponding day of the year, or returns **None** if any of the arguments is invalid.

Use the previously written and tested functions. Add some test cases to the code. Our test is only a beginning.

```
def IsYearLeap(year):
#
# the already written code
#
def DaysInMonth(year,month):
#
# your code is already here
#
def DayOfYear(year,month,day):
#
# put your code here
#
print(DayOfYear(2000,12,31))
```

Question 6

*A natural number is **prime** if it is greater than 1 and has no divisors other than 1 and itself.*

Complicated? Not at all. Look: 8 isn't a prime number, as you can divide it by 2 and 4 (we can't use divisors equal to 1 and 8 as the definition prohibits this). On the other hand, 7 is a prime number, as we can't find any legal divisors for it.

Write a function checking whether a number is prime or not.

The function:

- is called **IsPrime**;
- takes one argument (the value to check)
- returns **True** if the argument is a prime number, and **False** otherwise.

Hint: try to divide the argument by all subsequent values (starting from 2) and check the remainder – if it's zero, your number cannot be a prime; think carefully about when you should stop the process.
*If you need to know the square root of any value, you can utilize the ** operator. Remember: the square root of x is the same as x ** 0.5 (x raised to the power of 1.5).*

Complete the code presented below.
Run your code and check whether your output is the same as ours.

```
def IsPrime(num):
#
# put your code here
#
for i in range(20):
    if IsPrime(i + 1):
        print(i,end=" ")
print()
```

Example output

```
2 3 5 7 11 13 17 19
```