Before we start:

**N**: the input of number of attractions

**K**: the input of the maximum attractions allowed to be visited.

**num\_iter**: the number of possible arrangements for this specific **N** and **K**; the number of iterations needed for the program to find the global max.

**remainder**: the number of elements allows in the group that has less elements than **K** (the number of elements in the *special day*)

**glb\_max**: the maximum of all possible outputs. It is what we return

**lcl\_max**: record the maximum from each iteration and is compared to glb\_max after each iteration.

**ini**: arrange on which day we need to visit less attractions than **K**. It tries the *special day* one by one by multiplying **I** and **K** and compare it to **j**.

**i**: keep track of the number of iteration and will never exceed the **num\_iter**.

**j**: a tool act as the slicing index. The number added to **j** decides the size of attractions visited on that day. It will be refresh to 0 in every new iteration. It will never exceed the size of **N**

The reasoning behind all the above arrangement:

In this question, a very simple mathematic problem should be solved first.

If we have **N** attractions and **K** maximum visit per day, we will have at most

**N//K + 1** possible arrangement if **N%K** != 0

**N//K** possible arrangement if **N%K** == 0

Given the fact that we are looking for the maximum score in the fewest day.

We name this number of possible arrangements as **num\_iter**, which also implies the number of iterations needed that allows the program to check all the possible combination.

For example, when **N** == 8, **K** == 3:

Possible arrangement will equal to 8//3 + 1 = 3

The remainder will be 2

The **N//K** or **N//K +1** gets us the number of combination and the remainder tells us how many attractions is allowed in that one *special day* where its size is less than **K**

*Note: N//K or N//K +1 is replaced by the math.ceil(N/K) in python*

So, our possible arrangement with fewest day possible is:

332 or 323 or 233

*Special day*: the day that have less elements than **K**

The **remainder** of **N/K** will be the “*special day*” where it has a smaller number of attractions then the others.

In the example’s case, this *special day* has a magnitude of 2

In the program, we also need to make sure we check all the arrangements.

So, a list of position is needed. Using the above example, we know that we need three positions in the list.

**ini** = [1, 2, 3]

Then we need to make sure that in **ini**[0], we get to put the *special day* at front, which is 233. When **ini**[1], we need to put our *special day* in second position, which is 323, and so on.

However, finding the pattern and creating a list is too complicated. So, we use **ini = i\*K** instead.

**i** keeps track of the process of iteration and it is set to not exceed the **num\_iter**

**j** is a tool we use to keep track of the position of slicing.

When **j** does not equal to **ini**, we slice the day with **K** attractions. When **j** equals to **ini**, which usually only happens once per iteration, we slice the day with **remainder** attraction**.**

In each iteration, the code experience with each combination and record the maximum using **lcl\_max**. After finishing all the iteration, the **glb\_max** represents the max combination amount all and it will be returned as the final output.

However, if **remainder** equals to zero, because **N%K** == 0, the program will just evenly split the **N** attractions into groups of **K** and add up the max in each group.