1 The freight (shipping) container identification is covered by the ISO 6346 standard. Under ISO 6346, each container is labelled with an 11-character code (four letters + seven digits) in which the last digit is a “check” digit that is computed from the leftmost 4 letters and 6 digits, according to a fixed rule. For example, in MSKU3881107, the final “7” is the check digit.

The check digit is calculated by applying the following algorithm, on the left-most 4 letters and 6 digits of a freight container identification:

1. The four letters are replaced with corresponding numbers. The letters A through Z correspond to the numbers 10 through 38, with all multiples of 11 skipped.

2. The 10 numbers are multiplied by successive powers of two, starting from 2^0 to 2^9, and added together.

3. Take the remainder when dividing by 11, and if the result is 10, it is taken to be zero.

For example, given the 4 characters and 6 digits MSKU388110:

Step 1: 24 30 21 32 3 8 8 1 1 0

Step 2: $24\times2^0 + 30\times2^1 + 21\times2^2 + 32\times2^3 + 3\times2^4 + 8\times2^5 + 8\times2^6 + 1\times2^7 + 1\times2^8 + 0\times2^9 =$ 1624

Step 3: 1624 / 11 = 147 remainder 7, hence check digit = 7

For each of the sub-tasks, add a comment statement, at the beginning of the code using the hash symbol ‘#’, to indicate the sub-task the program code belongs to, for example:

```
In [1]:   # Task 1.1
          Program code

          Output:
```

Task 1.1

Write a function `task1_1(input_value)` that returns a string.

The function should:

- validate that the parameter `input_value` is a string of 4 letters followed by 7 digits
- return `True` if the above rule is met, otherwise return `False`.

Task 1.2

Create four tests to test your function. Display the input value used for each test, and its result.

Your four input values should be:

- a string containing only integers
- a string containing only letters
- a valid string
- a string of incorrect length.

Test your function with your four input values by calling it using the following statement:

```
print(taskl_l(input_value))
```

Task 1.3

Write a function `taskl_3(input_value)` that returns the calculated check digit.

The function should:

- take in an `input_value` containing the left-most 4 letters and 6 digits of the 11-character code
- calculate the check digit
- return the calculated check digit.

Task 1.4

A full 11-character freight container identification can be validated by removing the check digit, calculating the check digit from the remaining 10 alphanumeric characters and comparing it to the removed digit.

Write a function `task1_4(input_value)` that returns a Boolean indicating whether the check digit is valid. The function should:

- validate the parameter `input_value` by calling your function from **Task 1.1**
- return `False` if `input_value` is invalid
- remove the rightmost digit
- use your function from **Task 1.3** to calculate the check digit from the remaining string
- return `True` if the calculated check digit is the same as the check digit in the identification code and `False` otherwise.

Test your function with the following statements:

```
print(task1_4('PONU2079674'))

print(task1_4('XONU2079674'))
```

Task 1.5

The rule specified by ISO 6346 for computing the check digit is designed so that accidental changes or misreading of a single digit in a code can by identified as it will also change the check digit. Hence, most such errors can be caught easily using the check digit. However, the check digit can sometimes fail to detect a change in the code.

Create two test cases such that one test case is different from the other test case by a single character yet both test cases produce the same check digit.

Test your function with the following statements:

```
print(task1_4(test_case_1))

print(task1_4(test_case_2))
```

Both statements should return `True`.

Save your Jupyter notebook for Task 1.