1 Finding particular sequences of prime numbers

Write a program that finds all sequences of 6 consecutive prime 5-digit numbers, so of the form $(a, b, c, d, e, f)$ with $b = a + 2$, $c = b + 4$, $d = c + 6$, $e = d + 8$, and $f = e + 10$. So $a$, $b$, $c$, $d$ and $e$ are all 5-digit prime numbers and no number between $a$ and $b$, between $b$ and $c$, between $c$ and $d$, between $d$ and $e$, and between $e$ and $f$ is prime.

The expected output is:

The solutions are:

```
13901 13903 13907 13913 13921 13931
21557 21559 21563 21569 21577 21587
28277 28279 28283 28289 28297 28307
55661 55663 55667 55673 55681 55691
68897 68899 68903 68909 68917 68927
```

2 Decoding a multiplication

Write a program that decodes all multiplications of the form

```
* * *
x * *
--------
* * *
* * *
--------
* * *
```

such that the sum of all digits in all 4 columns is constant.

The expected output is:

```
411 * 13 = 5343, all columns adding up to 10.
425 * 23 = 9775, all columns adding up to 18.
```
3 Finding particular sequences of triples

Write a program that finds all triples of positive integers \((i, j, k)\) such that \(i, j\) and \(k\) are two digit numbers, no digit occurs more than once in \(i, j\) and \(k\), and the set of digits that occur in \(i, j\) or \(k\) is equal to the set of digits that occur in the product of \(i, j\) and \(k\).

The expected output is:

- \(20 \times 79 \times 81 = 127980\)
- \(21 \times 76 \times 80 = 127680\)
- \(28 \times 71 \times 90 = 178920\)
- \(31 \times 60 \times 74 = 137640\)
- \(40 \times 72 \times 86 = 247680\)
- \(46 \times 72 \times 89 = 294768\)
- \(49 \times 50 \times 81 = 198450\)
- \(56 \times 87 \times 94 = 457968\)
4 Decoding a sequence of operations

Write a program that finds all possible ways of inserting + and – signs in the sequence 123456789 (at most one sign before any digit) such that the resulting arithmetic expression evaluates to 100.

Here are a few hints.

- 1 can either be preceded by –, or optionally be preceded by +; so 1 starts a negative or a positive number.
- All other digits can be preceded by – and start a new number to be subtracted to the running sum, or be preceded by + and start a new number to be added to the running sum, or not be preceded by any sign and be part of a number which it is not the leftmost digit of. That gives $3^8$ possibilities for all digits from 2 to 9. We can generate a number $N$ in $[0, 3^8 − 1]$. Then we can:
  - consider the remainder division of $N$ by 3 to decide which of the three possibilities applies to 2;
  - consider the remainder division of $\frac{N}{3}$ by 3 to decide which of the three possibilities applies to 3;
  - consider the remainder division of $\frac{N}{3^2}$ by 3 to decide which of the three possibilities applies to 4;
  - ...

The expected output is (the ordering could be different):

```
1 + 23 - 4 + 5 + 6 + 78 - 9 = 100
123 - 4 - 5 - 6 - 7 + 8 - 9 = 100
123 + 45 - 67 + 8 - 9 = 100
123 + 4 - 5 + 67 - 89 = 100
12 + 3 + 4 + 5 - 6 - 7 + 89 = 100
123 - 45 - 67 + 89 = 100
12 - 3 - 4 + 5 - 6 + 7 + 89 = 100
1 + 2 + 34 - 5 + 67 - 8 + 9 = 100
1 + 2 + 3 - 4 + 5 + 6 + 78 + 9 = 100
-1 + 2 - 3 + 4 + 5 + 6 + 78 + 9 = 100
12 + 3 - 4 + 5 + 67 + 8 + 9 = 100
1 + 23 - 4 + 56 + 7 + 8 + 9 = 100
```

Consider writing two versions: one that generates and evaluates the expression on the left hand side of the equality “by hand”, and one that generates and evaluates the expression on the left hand side of the equality using the builtin `eval()` function.
5 Encoding sets of integers as integers

Write a program that implements a number of functions to decode and decode a set $S$ of integers as an integer $N$, letting the positions of the bits set to 1 in the binary representation of $N$ (counting from 0 and starting from the right) correspond to the members of $S$. For instance, in binary the integer 76 reads as 1001100, and the bits set to 1 in 1001100 are at position 2, 3 and 6 (counting from 0 and starting from the right), so 76 encodes the set \{2, 3, 6\}.

Document your code so that running

```python
>>> import question_4
>>> help(question_4)
```

yields the following (the names of the functions you have to implement, their meaning and intended use, and a number of test cases).

Help on module question_4:

NAME

question_4

DESCRIPTION

Performs operations on encodings of a set of (distinct) nonnegative integers \{n_1, \ldots, n_k\} as $2^{n_1} + \ldots + 2^{n_k}$.

FUNCTIONS

cardinality (encoded_set)

Returns the number of elements in the set encoding as the argument.

```python
>>> cardinality(0)
0
>>> cardinality(1)
1
>>> cardinality(76)
3
```

display_encoded_set (encoded_set)

Displays the members of the set encoded by the argument, from smallest to largest element, in increasing order.

```python
>>> display_encoded_set(0)
{}
>>> display_encoded_set(1)
{0}
>>> display_encoded_set(3)
{0, 1}
>>> display_encoded_set(76)
{2, 3, 6}
```
is_in_encoded_set(nonnegative_integer, encoded_set)
    Returns True or False depending on whether the first argument belongs to the set encoded as the second argument.

>>> is_in_encoded_set(0, 0)
False
>>> is_in_encoded_set(0, 1)
True
>>> is_in_encoded_set(3, 76)
True
>>> is_in_encoded_set(4, 76)
False

set_encoding(set_of_nonnegative_integers)
    Encodes a set and returns the encoding. Here an empty set of curly braces provided as argument denotes the empty set, not the empty dictionary.

>>> set_encoding({})
0
>>> set_encoding({0})
1
>>> set_encoding({0, 1})
3
>>> set_encoding({2, 3, 6})
76

End your program with
if __name__ == '__main__':
    import doctest
doctest.testmod()

Recall that thanks to this technique, if the test cases all pass correctly, running
$ python3 question_4.py -v

yields no output, but running
$ python3 question_4.py -v

outputs the tests, all attempted and passed:
$ python3 question_4.py -v
Trying:
    cardinality (0)
Expecting:
    0
ok
Trying:
    cardinality (1)
Expecting:
    1
ok
Trying:
    cardinality (76)
Expecting:
    3
ok
Trying:
    display_encoded_set (0)
Expecting:
    {}
ok
Trying:
    display_encoded_set (1)
Expecting:
    {0}
ok
Trying:
    display_encoded_set (3)
Expecting:
    {0, 1}
ok
Trying:
    display_encoded_set (76)
Expecting:
    {2, 3, 6}
ok
Trying:
    is_in_encoded_set (0, 0)
Expecting:
    False
ok
Trying:
    is_in_encoded_set (0, 1)
Expecting:
    True
ok
Trying:
    is_in_encoded_set(3, 76)
Expecting:
    True
ok
Trying:
    is_in_encoded_set(4, 76)
Expecting:
    False
ok
Trying:
    set_encoding({})
Expecting:
    0
ok
Trying:
    set_encoding({0})
Expecting:
    1
ok
Trying:
    set_encoding({0, 1})
Expecting:
    3
ok
Trying:
    set_encoding({2, 3, 6})
Expecting:
    76
ok
1 items had no tests:
    __main__
4 items passed all tests:
    3 tests in __main__.cardinality
    4 tests in __main__.display_encoded_set
    4 tests in __main__.is_in_encoded_set
    4 tests in __main__.set_encoding
15 tests in 5 items.
15 passed and 0 failed.
Test passed.