Exercises

Exercise 1

Write a program that produces the following output:

```
000000001
000000022
000000333
000004444
000055555
000666666
007777777
088888888
999999999
```

Exercise 2

Write a program that asks the user to provide different numbers of banknotes and coins, from €50 to €1. Then output the total sum in euros. For example:

```
Enter number of 50 euro banknotes: 10
Enter number of 20 euro banknotes: 20
Enter number of 10 euro banknotes: 30
Enter number of 5 euro banknotes: 40
Enter number of 2 euro coins: 50
Enter number 1 euro coints: 60
You have 1560 euros.
```

Exercise 3

A quadratic equation is an equation of the form: $ax^2 + bx + c$

The roots of a quadratic equation are given by the formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Note that if $b^2 - 4ac < 0$, then no real-valued solutions exist. Write a program that asks the user to enter values for *a, b ,c,* then prints the solutions of the quadratic equation they define, if they exist. If they do not exist, it should output an appropriate message.

Exercise 4

Write a program that calculates the first five terms of the harmonic sequence, that is,

the numbers:

$$1$$

$$1 + \frac{1}{2}$$

$$1 + \frac{1}{2} + \frac{1}{3}$$

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4}$$

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5}$$

These are the numbers:

```
1
1.5
1.8333333333333333
2.083333333333333
2.283333333333333
```

Exercise 5

The pronic numbers are those that are the product of two consecutive integers, so that they have the form *n(n+1)*. For example, the first six pronic numbers are:

$$1 \times 2 = 2$$

$$2 \times 3 = 6$$

$$3 \times 4 = 12$$

$$4 \times 5 = 20$$

$$5 \times 6 = 30$$

$$6 \times 7 = 42$$

Write a program that asks the user the number of pronic numbers to output, then goes on and prints them:

```
Enter number of pronic numbers: 10
2 6 12 20 30 42 56 72 90 110
```

Exercise 6

The Greek Tax Identification Number (TIN) consists of 9 digits. The last digit is a check digit. It is calculated as follows:

1. We remove the check digit, so that we are left with an 8-digit number.
2. We take the 8 digits one by one, from the right to the left. We multiply each digit by the power of 2 corresponding to its position: the first from the right will be multiplied by 2^1, the second will be multiplied by2^2, and so on.
3. We take these powers and we sum them.
4. We calculate the remainder of this sum by 11.
5. We take this remainder and we calculate its remainder by 10. The result must equal the check digit.

For example, let us say we have the TIN 090034337. The check digit is 7. The other digits are 09003433. We have:

$$3 \times 2^1 + 3 \times 2^2 + 4 \times 2^3 + 3 \times 2^4 + 9 \times 2^7 =$$
$$3 \times 2 + 3 \times 4 + 4 \times 8 + 3 \times 16 + 9 \times 128 =$$
$$6 + 12 + 32 + 48 + 1152 = 1250$$

Then, *1250 mod 11 =7* and *7 mod 10=7* .

Write a program that will ask the user for TIN and will respond whether it is correct or not. For example:

```
Enter Tax Identification Number: 090034337
Tax Identification Number valid.

Enter Tax Identification Number: 090034336
Tax Identification Number not valid.
```

Exercise 7

A simple way to check binary data is the so-called *parity bit*. A byte consists of 8 bits, so that we can use the last bit to check whether the previous 7 are OK. We can do that by checking that the sum of the 1 bits is an even number (this is actually called *even parity*; we might require that the sum of 1 bits is odd, which is called *odd parity*). For example, see the following table where we see the first 7 bits of some numbers, the number of 1 bits in them, the full 8-bit number (including the parity) and the number of 1 bits in the byte.

First 7 bits	Number of 1s	8 bits (with parity)	Number of 1s
0000000	0	00000000	0
1010001	3	10100011	4
1101001	4	11010010	4

Write a program that asks the user for an 8-bit binary number and replies whether the parity bit checks OK. For example:

```
Enter binary number: 01010101
Parity check OK.

Enter binary number: 11010101
Parity check not OK.
```

Exercise 8

Write a program that asks the user for a 10-digit number and will then print it in two lines. The first line will contain the numbers in the odd positions and the second line the numbers in the even positions. For example:

```
Enter 10 digit number: 1234567890
1 3 5 7 9
 2 4 6 8 0
```

Take care so that the numbers line up in columns *exactly* as in the above example: each number must be in a column by itself.

Exercise 9

Write a program that asks the user for a 9-digit number and then prints it in three lines. Each line must contain three digits. For example:

```
Enter 9 digit number: 123456789
1  4  7
 2  5  8
  3  6  9
```

Take care so that the numbers line up in columns *exactly* as in the above example: each number must be in a column by itself.

References

https://github.com/codeandwork/courses/blob/master/courses/prep/pythonTutorial2.ipynb

https://github.com/codeandwork/courses/blob/master/courses/prep/pythonTutorial1.ipynb