Documentation

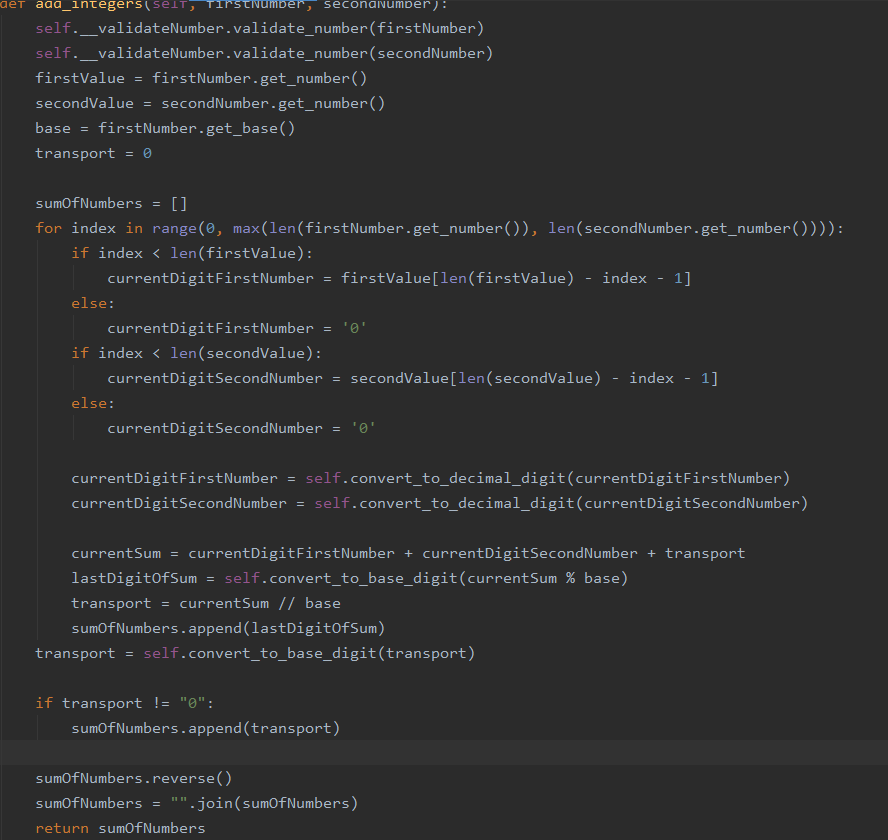
1. The problem statement

Create an application which implements algorithms for:

* Arithmetic operations for integers: addition, subtraction, multiplication by one digit and division by one digit, in a base p, where p is of {2, 3, …, 16};
* Conversions of natural numbers between two bases p and q of {2, 3, …, 16}, using the substitution method, the successive divisions method or the rapid conversions method( which will be applied for 2 bases p and q of {2, 4, 6, 8, 16}.

The application will have a menu such that all operations and conversion methods can be verified separately.

1. The used algorithms in pseudo-code
2. The addition of 2 numbers of the same base

 The algorithm begins with the validation of the two numbers from the user. In the case that one of the digits is bigger or equal to the base, then an error will be raised. Otherwise, the algorithm will continue. The two numbers are transmitted as objects( containing the actual value and the base, the value being a string and the base an integer). I will also define a transport digit and a list of digits of the sum. The digits will be calculated in the reverse order.

I will go through the numbers, from right to left, repeating this process until I reached all the digits of the number with the maximum length.

The first step of the loop is to retain the current digits of both number, situated at the same index, the digits being retained in the source base.

After, I will convert both of the digits in their decimal values.

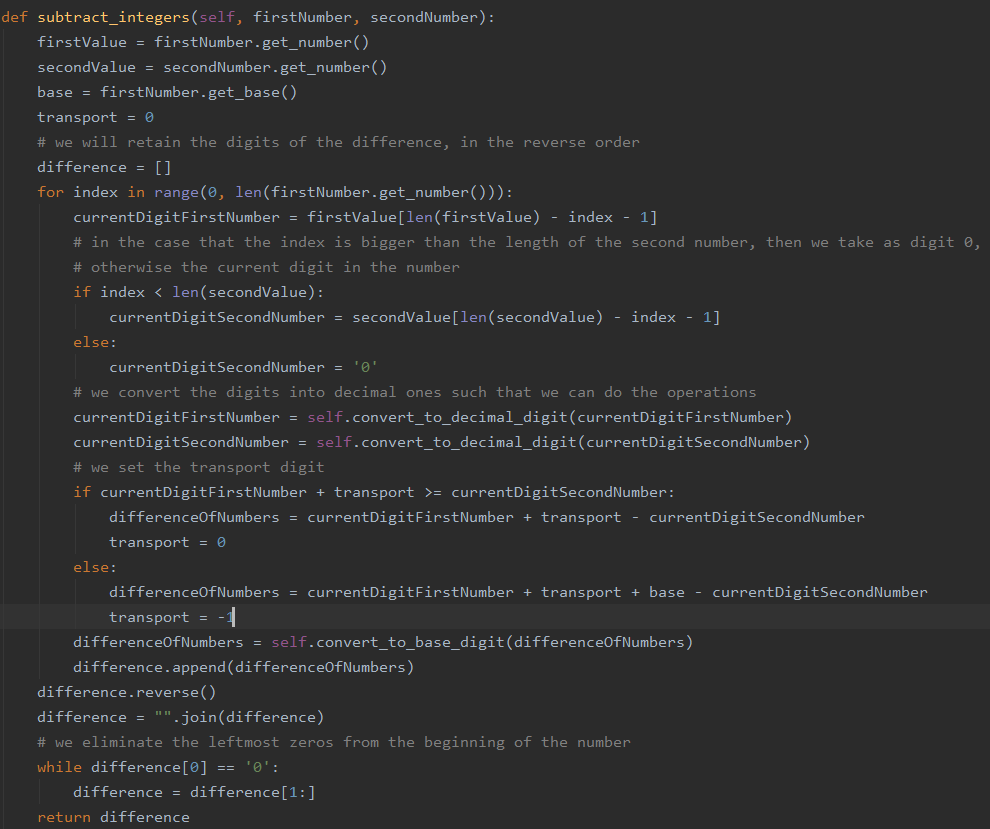
The third step is to calculate the current sum, which is the sum between the 2 digits and the transport one.

At the next step, I will calculate the last digit in sum, which will be appended to the result list( after it is converted into the source base), and the quotient will be retained as the transport digit.

After the loop is over, I will append the transport digit( only if it is non-zero) to the list of digits of the result. The list will be reversed, and then I will convert it to a string.

The algorithm returns the sum of the 2 initial numbers.

1. The subtraction of 2 numbers of the same base

 The function will have 2 parameters, the 2 numbers which will be given by the user. Both of them are objects, containing a string as the value of the number and a base as an integer. We will also set a transport digit, which will take the values 0 or -1 depending on the current digits. Also we will take a list of digits, where we will retain, in the reverse order, the digits of the difference.

We will go through both numbers, from right to left, by a number of times equal to the length of the first number, being the biggest one.

Firstly we will set the current digits of the numbers.

On the second hand, those 2 digits will be converted to their decimal value, in order to be able to do the operations.

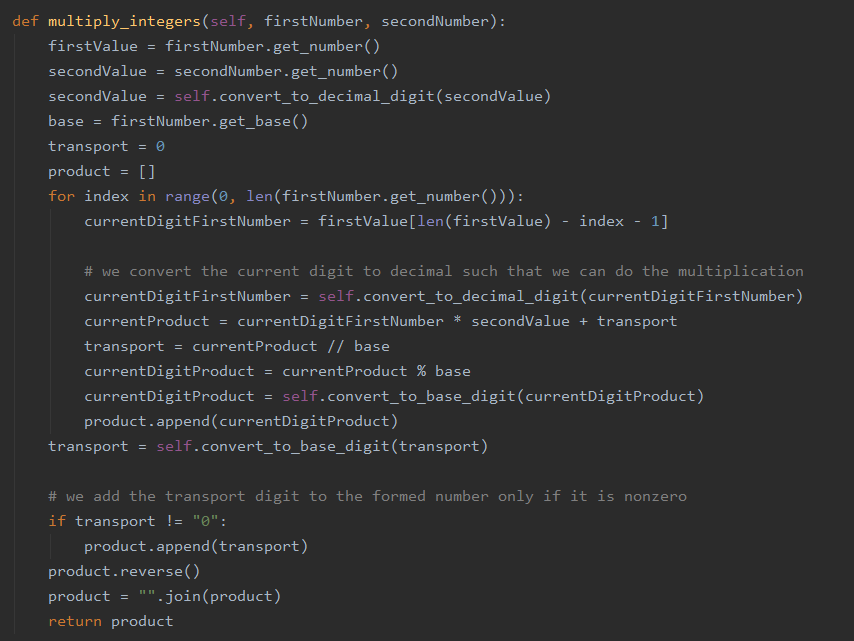
The third step is to set the transport digit. So, if the sum between the current digit of the first number and the transport digit is bigger than the current digit of the second number, then the transport digit will be 0. Otherwise, it will be set to -1.

On the fourth hand, we will calculate the current difference and the digit will be appended to the list of digits, after it was converted to the source base.

Considering the fact that the digits are retained in the reverse order, we will reverse the list and then we will convert it to string.

Moreover, there is the chance that the first digits will be 0, so we will delete them.

After all these operations we will return the difference as a string.

1. The multiplication between a number and a digit of the same base

The functions gets 2 parameters, both being objects( each of them is formed of a string which represents the value of the number and an integer, which is the base).

We will also set a transport digit and the list of digits of the number, digits which will be retained in the reverse order.

We will go through each digit of the first number( the second one is a digit), from left to right. We will do this loop for a number of times equal to the length of the first number.

Firstly we will retain the current digit in the number and convert it to its decimal value.

On the second hand, I will do the product between the current digit and the second number, adding to it the transport digit.

At the first step I will append to the list of digits the remainder between the current result and the source base, the transport base being the quotient of this operation.

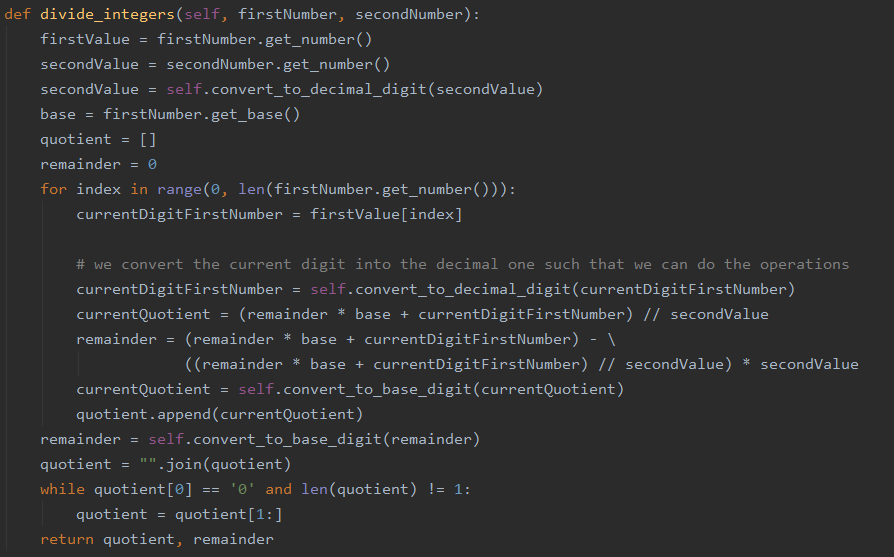
The remainder will be also converted to the value in the source base before appending it.

After the loop is over, we will see if the transport is non-zero or not. If it is not, then we will also append it to the list of the digits of the product.

Getting the digits in the reverse order, we will do the reverse of the list and then transform it to a string.

At the end we will return the product, which is a string.

1. The division between a number and a digit of the same base

 The functions gets 2 parameters, both being objects( each of them is formed of a string which represents the value of the number and an integer, which is the base).

We will also set the quotient list, which will retain the digits in the right order and a remainder.

We will go through each digit of the first number, from left to right.

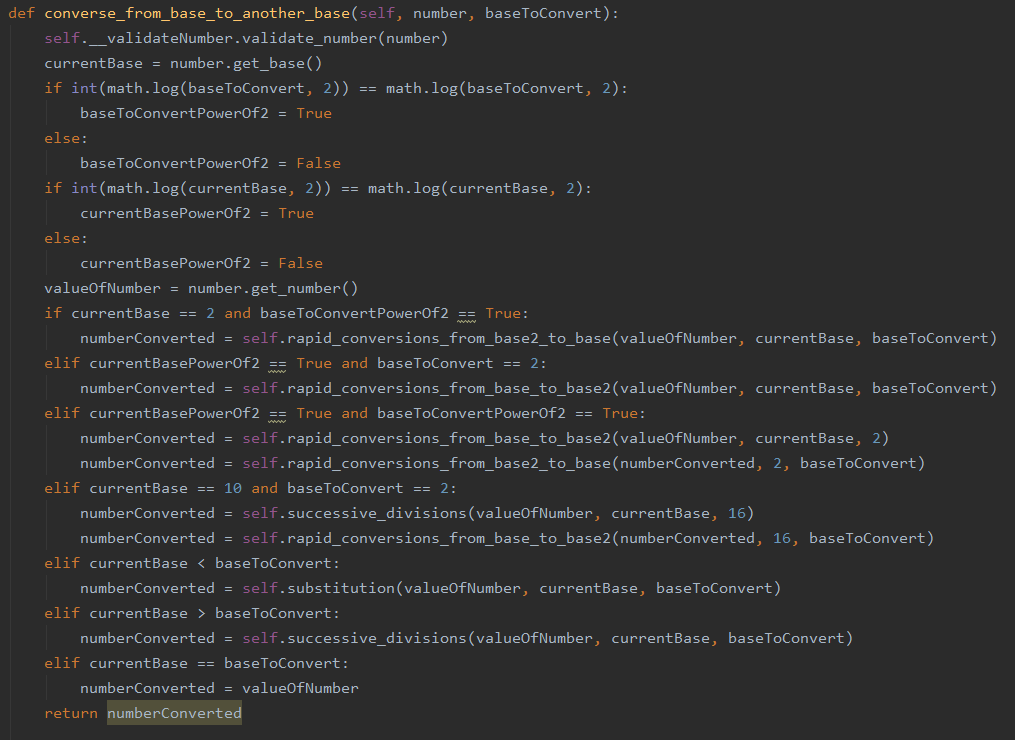
Firstly we will retain the current digit and then we will convert it to its decimal value.

On the second hand, the quotient and the remainder will be calculated using these formulas, where c is the quotient, t represents the remainder, a is the current digit of the first number, b is the second number and m is the length of the first number.



Thirdly, the quotient will be converted to the source base and then it will be appended to the list of digits. After the loop is over, the remainder will also be converted to the source base and the list of digits will be transformed in a string. We will also remove the insignificant leftmost digits of the quotient. The quotient and the remainder will be returned from the function.

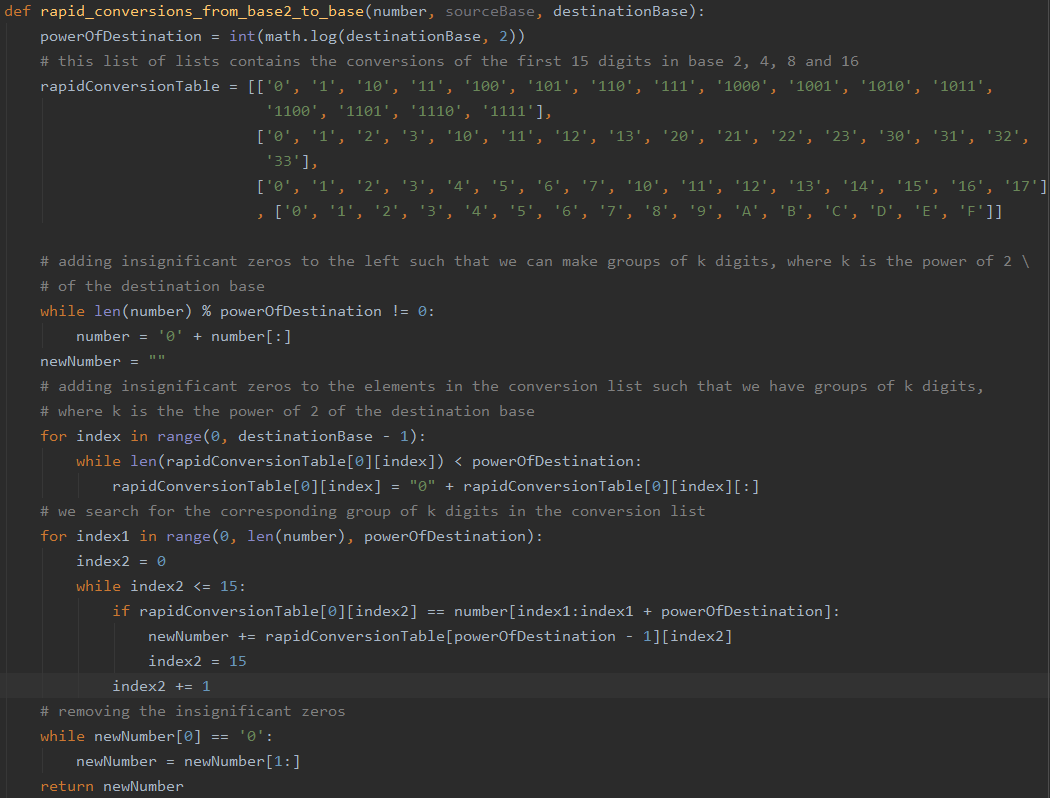
1. Conversions

There are 7 cases I have used in order to convert a number from a base to another base. Depending on the relation between the 2 bases I have chosen the most efficient method.

The cases are:

1. The source base is 2 and the destination base is also a power of 2( in this case I have used the rapid conversions from base 2 to another base, also power of 2)
2. The source base is a power of 2 and the destination base is 2( in this case I have used the rapid conversions from a base power of 2 to base 2)
3. Both the source and the destination source are powers of 2, different from 2( in this case I have used the rapid conversions, firstly from the source base to base and secondly from base 2 to the destination source)
4. The source base is 10 and the destination base is 2( in the case I have used firstly the successive divisions method, converting the number from base 10 to base 16 and then I have used the rapid conversions from base 16 to base 2)
5. The source base is smaller than the destination base( in this case I have used the substitution method)
6. The source base is bigger than the destination base( in this case I have used the successive divisions method)
7. The source base is equal to the destination base( in this case the number remains the same)

In the end, I have returned the converted number.

1. Rapid conversions from base 2 to a destination base power of 2

I will firstly calculate the power of the destination base. I will then set the rapid conversions table, which is a list of lists. The first list in the list contains the binary representation of the numbers from 0 to 16. The second list contains the representation in base 4 of the numbers from 0 to 16. The third list contains the representation in base 8 of the numbers from 0 to 16. The fourth list contains the representation in base 16 of the numbers from 0 to 16.

Considering the fact that we will need to make groups of k binary digits, where k is the power of the destination base, we will add insignificant zeros to the left to the initial number.

Moreover, we will also add insignificant zeros to the numbers in the rapid conversions table of the binary representation, such that we have groups of k or more binary digits.

We will then go through the number, k digits at one time and we will search for that certain group of binary digits in the rapid conversions table. When we find it, we will add in the new number the correspondent in the destination base.

At the end we will remove the insignificant zeros from the left of the number and then return it.

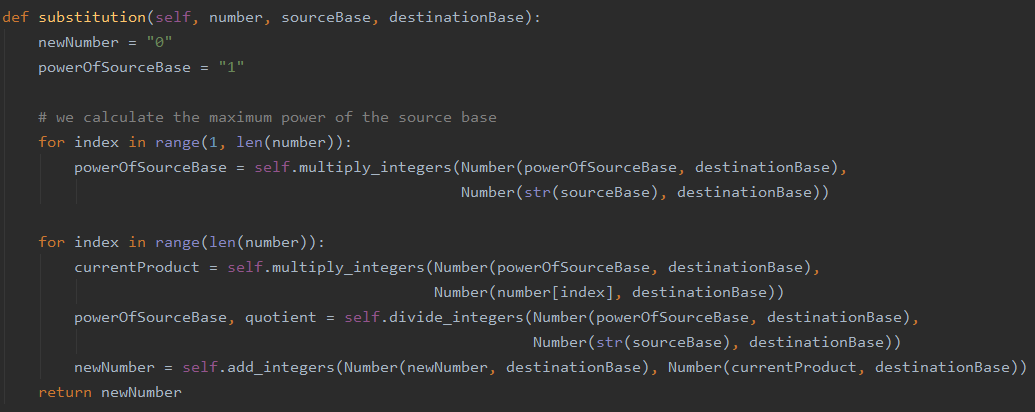
1. Rapid conversions from source base power of 2 to base 2

I will firstly calculate the power of the source base. I will then set the rapid conversions table, which is a list of lists. The first list in the list contains the binary representation of the numbers from 0 to 16. The second list contains the representation in base 4 of the numbers from 0 to 16. The third list contains the representation in base 8 of the numbers from 0 to 16. The fourth list contains the representation in base 16 of the numbers from 0 to 16.

We will also add insignificant zeros to the numbers in the rapid conversions table of the binary representation, such that we have groups of k or more binary digits, where k is the power of 2 of the source base.

We will go through each digit of the number and add in the new number the corresponding k binary digits.

After the loop is over, we will remove the leftmost insignificant zeros and then return the number.



1. The substitution method

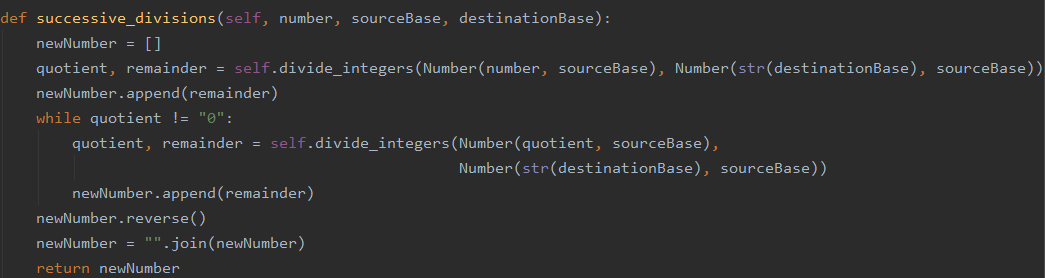
Firstly we will define the number in which we will do the sum of the products( which will be equal to 0) and the power of the source base( which will be initially equal to 1).

I will calculate the biggest power of the source base( if the number has k digits, then the source base will be at the power of k-1). Moreover, all the calculations will be done in the destination base.

I will go through each digit of the number, from left to right. I will then calculate the current product, which is the product between the power of the source base and the current digit. I will add this product to the new number.

After all the current calculations are done, I will divide the power of the source base by the source base.

The result will then be returned, representing the new number.

1. The successive divisions method

We will retain the result in a list, where we will put the remainders in the order of obtaining them, having to reverse it at the end of all operations( which are done in the source base).

We will do the first operation outside the loop, in order to see if the quotient is 0 or not.

If the quotient is not 0 after the first operation, we will do once again a division, but between the quotient and the destination base. The remainder will be added to the list of remainders.

We will continue this process until the quotient is 0. After it is finished we will reverse the list and transform it into a string, which represents the new number.

1. Test data

