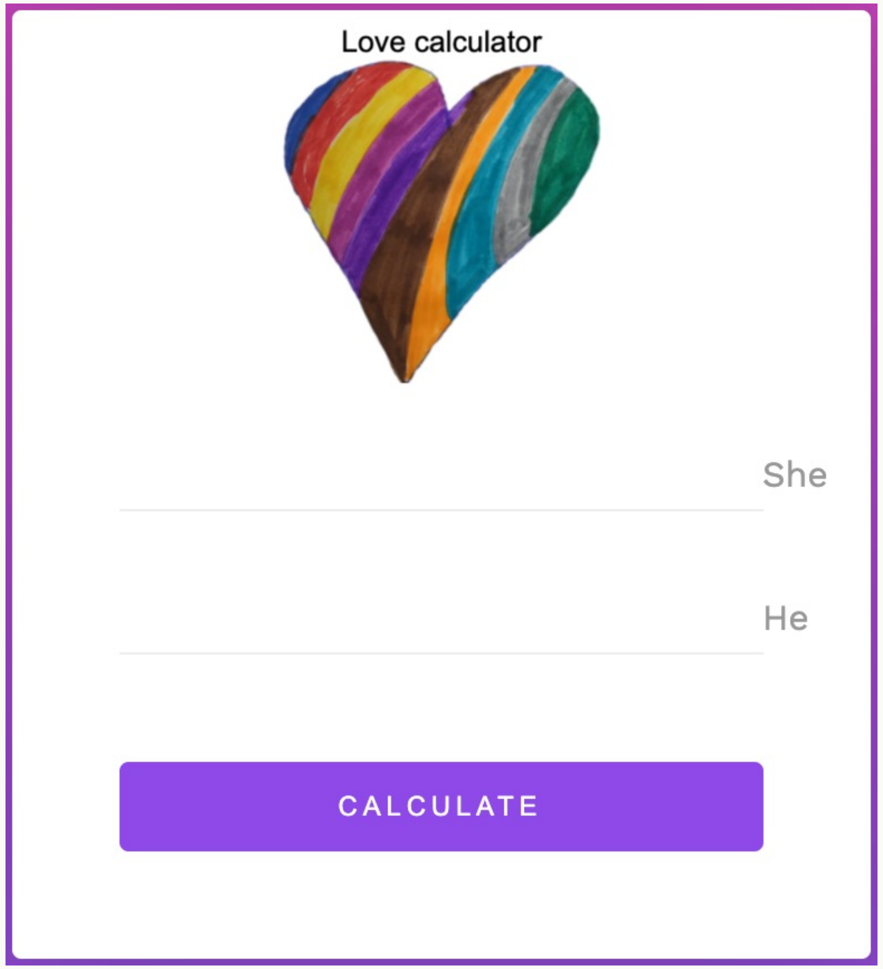
Introduction into Python

Tomislav Jakopec, PhD, assistant professor, University of Osijek, Croatia

Problem

Solution

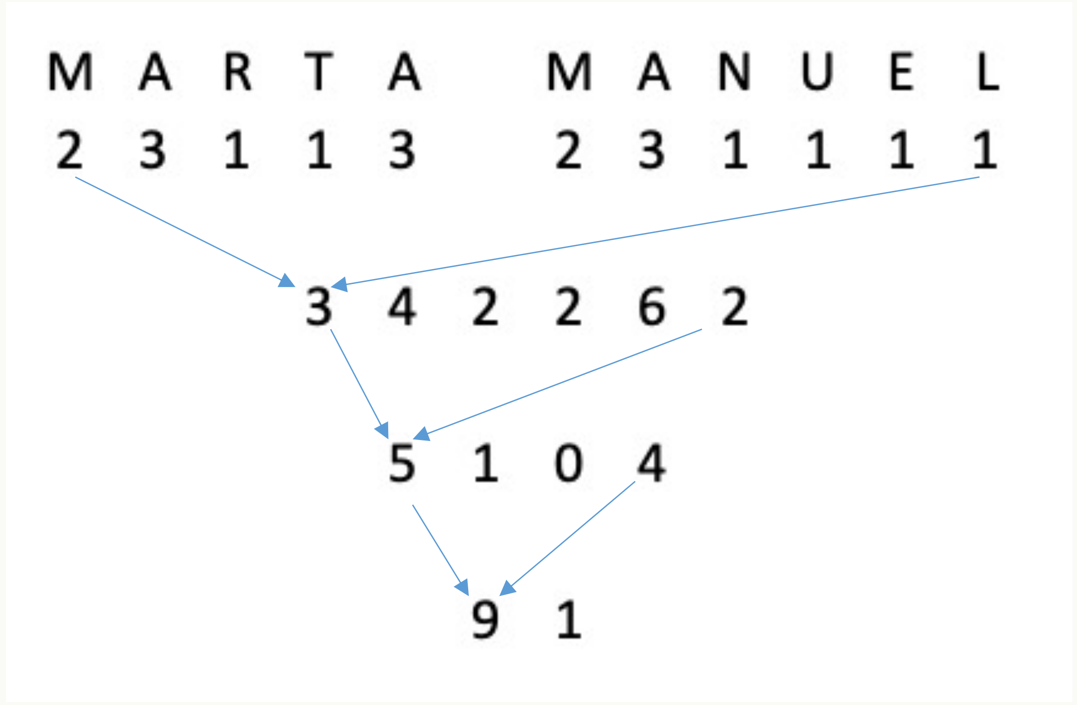

Love calculator

She

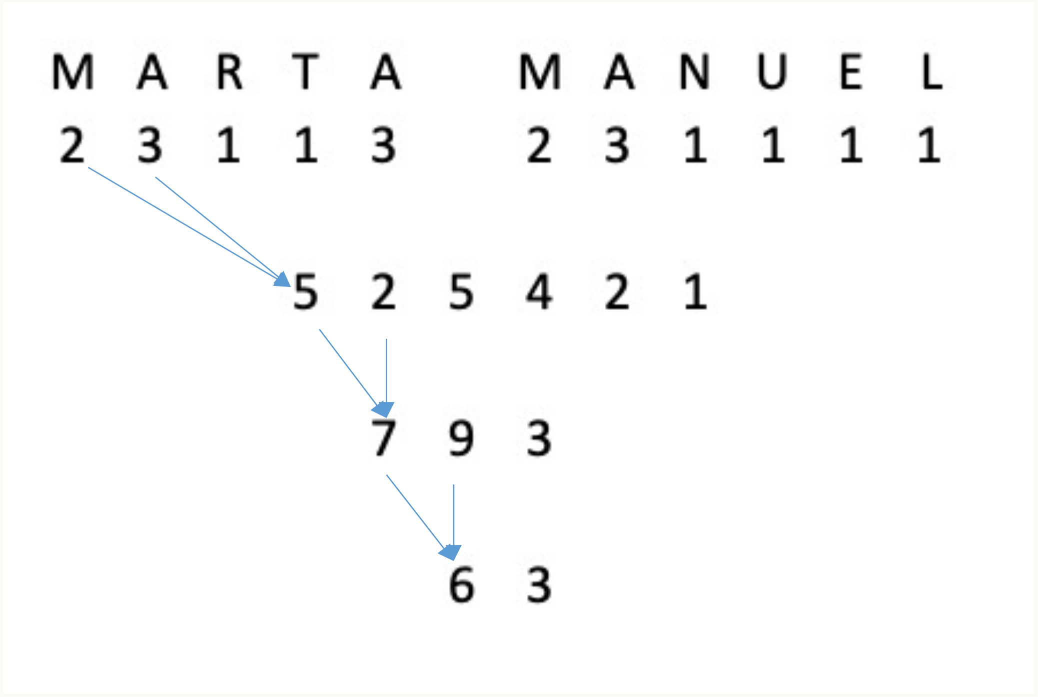
He

Algorithms as a solution

Algorithm 1



Algorithm 2



Algorithm n

Algorithms as a solution

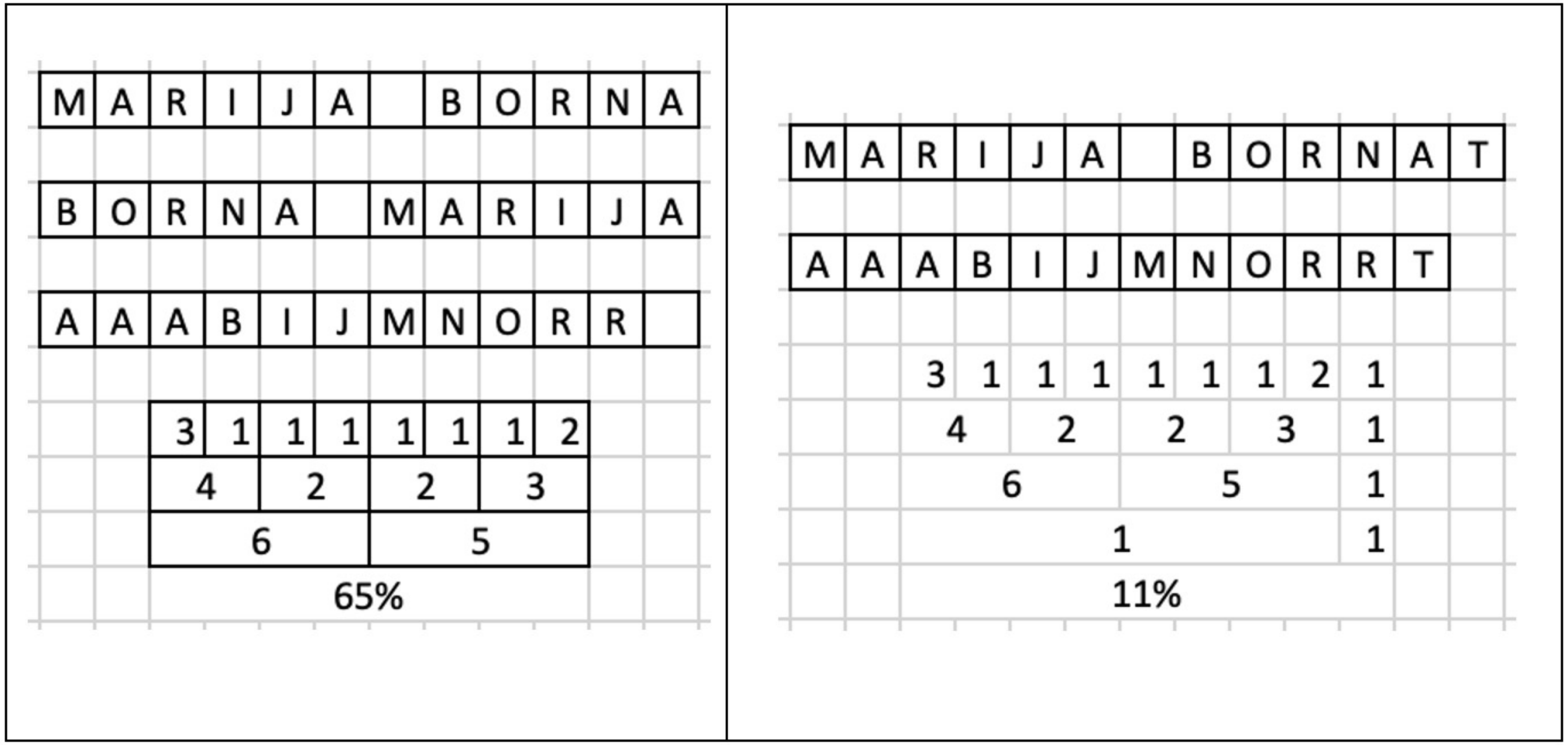
Algorithm 1

```
M A R T A   M A N U E L
2 3 1 1 3   2 3 1 1 1 1

      3 4 2 2 6 2

        5 1 0 4

          9 1
```

Algorithm 1

```
M A N U E L   M A R T A
2 3 1 1 1 1   2 3 1 1 3

      5 4 2 4 3 1

          6 7 6

          1 2 7

            3 7
```

Proposed solution



AGENCIJA ZA MOBILNOST I PROGRAME EU

Erasmus+

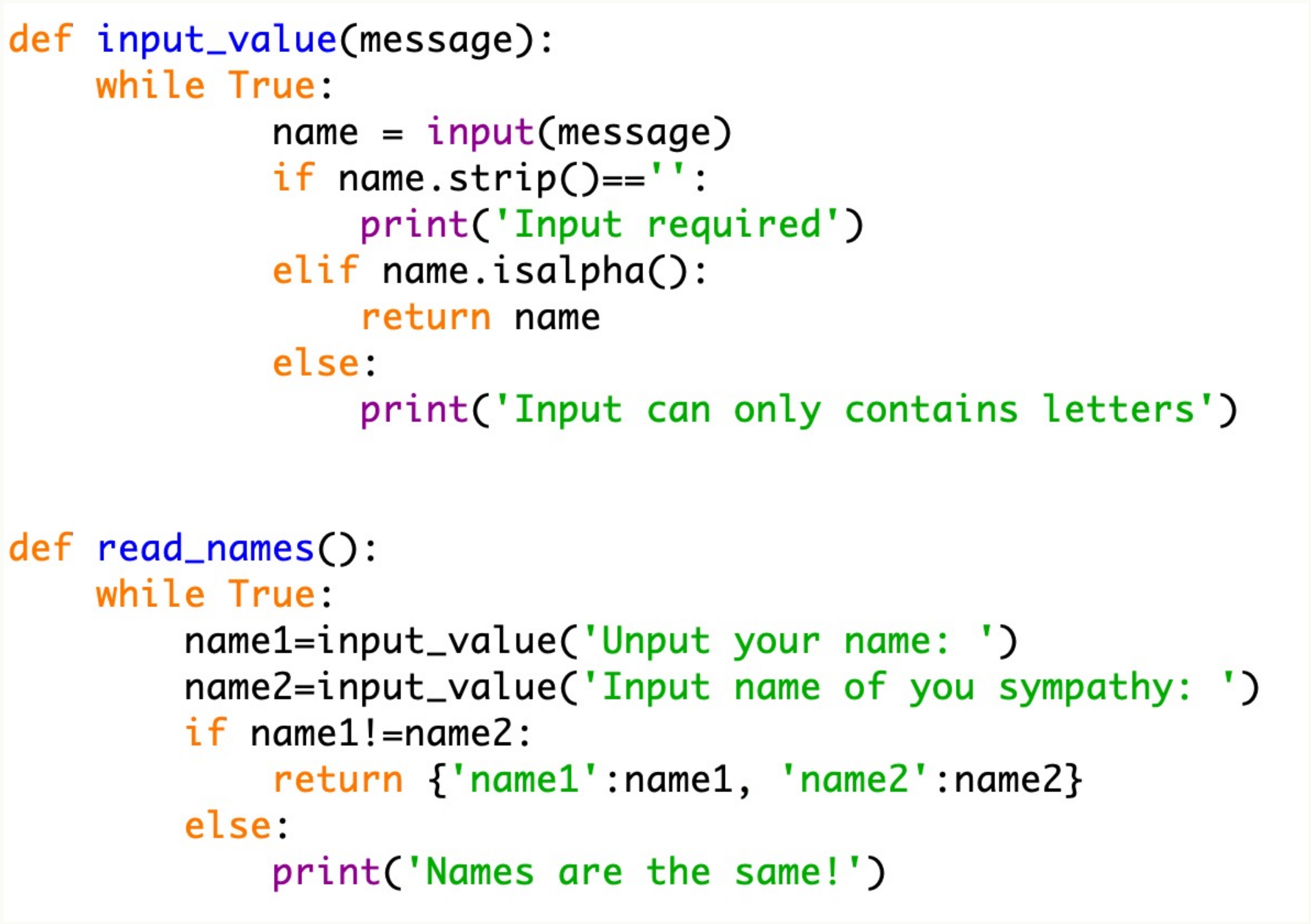
Steps leading to a solution

1. Input two strings (first and second name)
2. All characters will be arranged in a new unique sequence in which the characters will be arranged alphabetically
3. The letters from the created and sorted sequence will be counted and a series of numbers will be created. If a letter appears more than 9 times, only the unit part of the number will be taken
4. The two adjacent numbers will be added together and entered in the row below. If the sum is greater than 9, only the unit part of the number will be taken.
   - This procedure will be repeated until we get a number that is less than 100.
5. The result is a percentage of how much these two love each other

DECriS

1. Input two strings (first and second name)

Programming



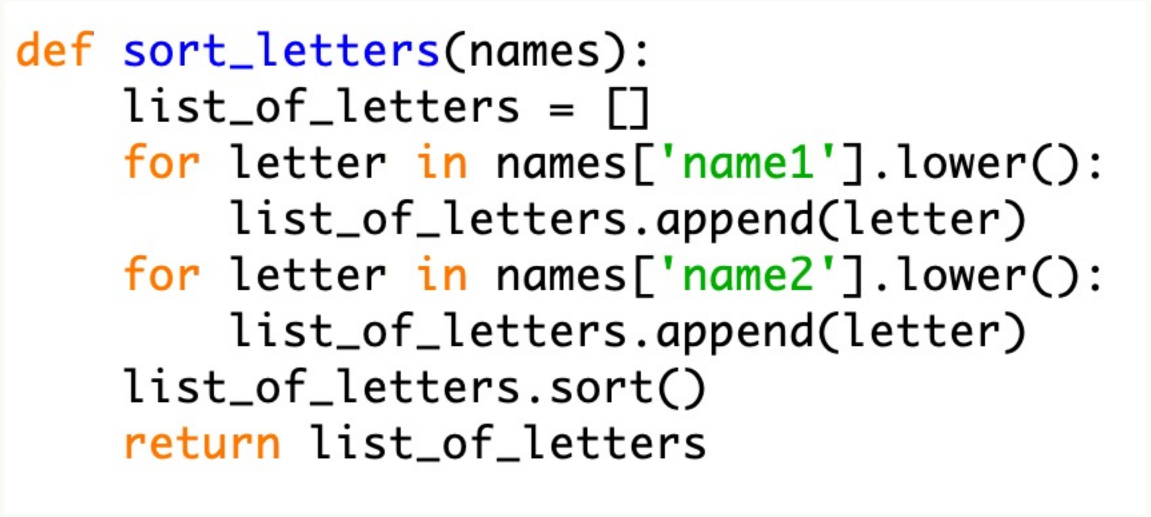
```python
def input_value(message):
    while True:
            name = input(message)
            if name.strip()=='':
                print('Input required')
            elif name.isalpha():
                return name
            else:
                print('Input can only contains letters')


def read_names():
    while True:
        name1=input_value('Unput your name: ')
        name2=input_value('Input name of you sympathy: ')
        if name1!=name2:
            return {'name1':name1, 'name2':name2}
        else:
            print('Names are the same!')
```

AGENCIJA ZA MOBILNOST I PROGRAME EU

Erasmus+

Programming

2. All characters will be arranged in a new unique sequence in which the characters will be arranged alphabetically



```python
def sort_letters(names):
    list_of_letters = []
    for letter in names['name1'].lower():
        list_of_letters.append(letter)
    for letter in names['name2'].lower():
        list_of_letters.append(letter)
    list_of_letters.sort()
    return list_of_letters
```

Programming

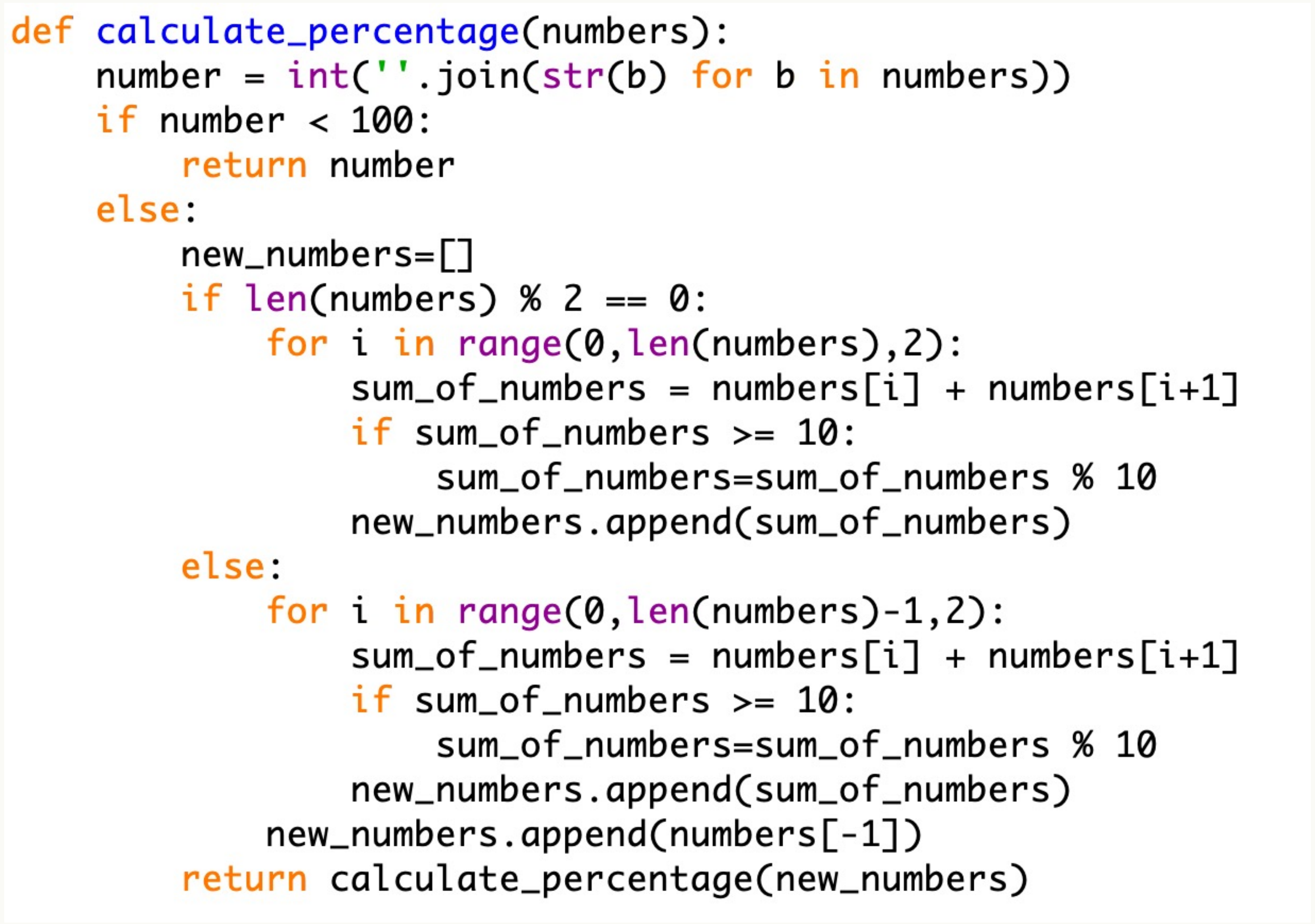
3. The letters from the created and sorted sequence will be counted and a series of numbers will be created. If a letter appears more than 9 times, only the unit part of the number will be taken

```
def count_letters(list_of_letters):
    dictionary = dict((i, list_of_letters.count(i)) for i in list_of_letters)
    numbers=[]
    for letter in dictionary:
        numbers.append(dictionary[letter])
    return numbers
```

Programming

4. The two adjacent numbers will be added together and entered in the row below. If the sum is greater than 9, only the unit part of the number will be taken.

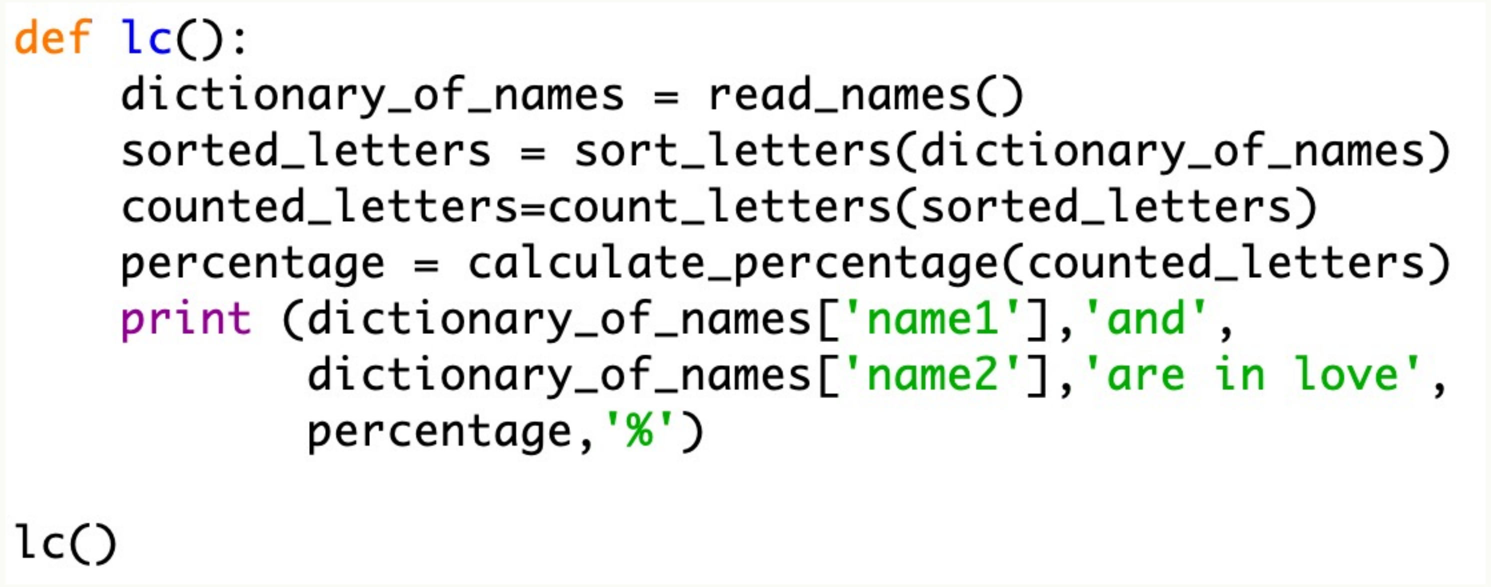
- This procedure will be repeated until we get a number that is less than 100.



```python
def calculate_percentage(numbers):
    number = int(''.join(str(b) for b in numbers))
    if number < 100:
        return number
    else:
        new_numbers=[]
        if len(numbers) % 2 == 0:
            for i in range(0,len(numbers),2):
                sum_of_numbers = numbers[i] + numbers[i+1]
                if sum_of_numbers >= 10:
                    sum_of_numbers=sum_of_numbers % 10
                new_numbers.append(sum_of_numbers)
        else:
            for i in range(0,len(numbers)-1,2):
                sum_of_numbers = numbers[i] + numbers[i+1]
                if sum_of_numbers >= 10:
                    sum_of_numbers=sum_of_numbers % 10
                new_numbers.append(sum_of_numbers)
            new_numbers.append(numbers[-1])
        return calculate_percentage(new_numbers)
```

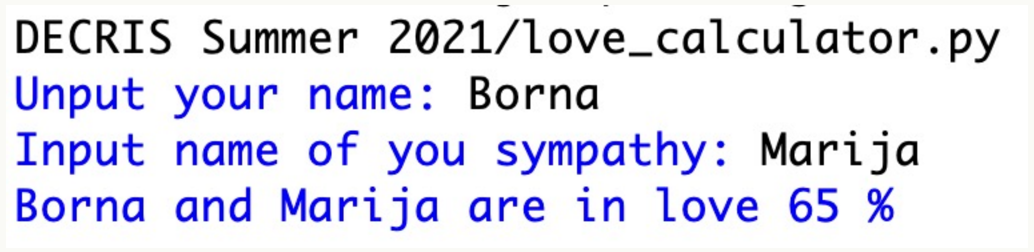
5. The result is a percentage of how much these two love each other

Programming



```
def lc():
    dictionary_of_names = read_names()
    sorted_letters = sort_letters(dictionary_of_names)
    counted_letters=count_letters(sorted_letters)
    percentage = calculate_percentage(counted_letters)
    print (dictionary_of_names['name1'],'and',
           dictionary_of_names['name2'],'are in love',
           percentage,'%')

lc()
```

Program execution



```
DECRIS Summer 2021/love_calculator.py
Unput your name: Borna
Input name of you sympathy: Marija
Borna and Marija are in love 65 %
```

The practical part

https://github.com/tjakopec/DESS2021

Tasks

1. Correct the text typo in the code
2. Extract the repeated part of the code twice in the def - test the success of the code change
3. Adjust the code to always print 99% for two names (your name and the name of your sympathy)
4. Implement Algorithm 1 from slide 4

AGENCIJA ZA MOBILNOST I PROGRAME EU
Erasmus+