# 4: Input parsing, more functions

**Due** 28 Nov by 23:59 **Points** 20 **Submitting** an external tool

Here is a zip file with all the files you need to fill in for this module, and the example input files.

The first couple of assignments 4 assignments require you to read from the standard input. Do not read directly from the example files, as the automated testing system expects you to read from the standard input. Instead, process the lines as follows:

```
import sys
for l in sys.stdin.read().splitlines():
   pass # do stuff with each line
```

To run your program with an example file as input do the following:

- In PyCharm, go to Run at the top
- Select ``Run..."
- Select `Edit configurations"
- Check ``Redirect input from" and select a file
- Press run

## 1. Geography Grades 1

In the input file, grades are listed for the geography tests of group 2b. There have been three tests of which the grades will be included in the half-yearly report that is given to the students before the Christmas break.

On each line of the input you can find the name of the student, followed by one or more under scores ('_'). These are succeeded by the grades for the tests, for example:

```
Anne Adema___________6.5 5.5 4.5
Bea de Bruin_________6.7 7.2 7.7
Chris Cohen__________6.8 7.8 7.3
Dirk Dirksen_________1.0 5.0 7.7
```

The lowest grade possible is a 1, the highest a 10. If somebody missed a test, the grade on the list is a 1.

Your assignment is to make the report for the geography course of group 2b, which should look like this:

```
Report for group 2b
Anne Adema has an average grade of 5.5
Bea de Bruin has an average grade of 7.2
Chris Cohen has an average grade of 7.3
Dirk Dirksen has an average grade of 4.6
End of report
```

Use functions to structure your code.

Put your code in geography_grades_1.py

This program is tested via input/output tests

## 2. Geography Grades 2

Make a copy of your program for the problem Geography Grades 1 and change the code in such a way that your program no longer prints the average grade, but the final grade.

The final grade is calculated by rounding the average grade to the nearest multiple of a half. So, for example, a 7.2 becomes a 7.0 and 7.3 becomes a 7.5. If this calculation results in a 5.5, the final grade becomes a 6.0.

Your assignment is to make the report for the geography course of group 2b, that, with the same example input as for the problem Geography Grades 1, should look like this:

```
Report for group 2b
Anne Adema has a final grade of 6.0
Bea de Bruin has a final grade of 7.0
Chris Cohen has a final grade of 7.5
Dirk Dirksen has a final grade of 4.5
End of report
```

Put your program in geography_grades_2.py

This program is tested via input/output tests

## 3 - Geography Grades 3

Make a copy of your program for the problem Geography Grades 2 and change the code in such a way that your program can process multiple groups.

These groups are on the input separated by '=\n'. Every group starts with a first line that contains the name of the group and the lines after contain the information about the students in the same way as is specified for the problem Geography Grades 1.

With the input

```
1b
Erik Eriksen__________4.3 4.9 6.7
Frans Franssen________5.8 6.9 8.0
```

```
=
2b
Anne Adema__________6.5 5.5 4.5
Bea de Bruin_____6.7 7.2 7.7
Chris Cohen__________6.8 7.8 7.3
Dirk Dirksen_____1.0 5.0 7.7
```

The output should be:

```
Report for group 1b
Erik Eriksen has a final grade of 6.0
Frans Franssen has a final grade of 7.0
End of report

Report for group 2b
Anne Adema has a final grade of 6.0
Bea de Bruin has a final grade of 7.0
Chris Cohen has a final grade of 7.5
Dirk Dirksen has a final grade of 4.5
End of report
```

Put your program in geography_grades_3.py

This program is tested via input/output tests

# 4 - Administration

For the end of year administration of Programming for History of Arts students you are to write a program that can do 2 things:

1. calculate a final grade
2. print a small graph of similarity scores and, if applicable, list the students under investigation

The input is structured as follows:

```
Piet van Gogh___5 6 7 4 5 6
5=20=22=10=2=0=0=1=0=1;Vincent Appel,Johannes Mondriaan
Karel van Rijn___7 8 6 6
2=30=15=8=4=3=2=0=0=0;
```

The first line should be interpreted as follows:

You have to calculate the final grade of the student. All grades have the same weight. The final grade is rounded as follows:

- a grade that is >= 5.5 AND < 6 should be noted as a "6-"
- otherwise, a grade will be rounded to the nearest half

The second line should be interpreted as follows:

```
<10 numbers separated by '='>;<zero or more names separated by ','>
```

The first 10 numbers are the similarity scores. These scores represent the number of programs matching a certain percentage of the current program in steps of 10%. This means the first numbers indicates the matches from 1%-10% and the last number indicates the matches from 91%-100%.

Since this is not very readable, the professor would like a simple graph according to these rules:

- if there are zero matches, display an underscore: _
- if there are less than 20 matches, display a minus sign: -
- if there are 20 or more matches, display a caret: ∧

The names of the students after the semicolon are the names of the students with matches in the final 3 categories. The names of these students should be printed under the graph. If there are no matches, the program should print "No matches found".

The output for the aforementioned input should be:

```
Piet van Gogh has an average of 6-
-^^--__-_-
Vincent Appel
Johannes Mondriaan

Karel van Rijn has an average of 7.0
-^-----___
No matches found
```

Put your program in administration.py

This program is tested via input/output tests

# 5 - Anagram

Make a method is_anagram_of(a,b) that tests if a is an anagram of b. A string a is an anagram of a string b, if it uses exactly the same letters, but the order can be different. Spaces are ignored, as well as capitalization.

Examples of anagrams:

- "eleven plus two" - "twelve plus one"
- "William Shakespeare" - "I am a weakish speller"
- "Tom Marvolo Riddle" - "I am Lord Voldemort"
- "Anagrams" - "Ars manga"
- "television ads" - "enslave idiots"

Counter examples:

- "bla" - "aalb"
- "cat" - "tact"

Hint: Make a dictionary that holds how often a letter occurs in a word.

Put your program in anagram.py .

This program is tested via unit tests.

## 6 - Merge sorted arrays

Program a method merge_sorted(a,b) that when given two sorted arrays a and b, returns a new sorted array c that has the elements from array a and array b. For example when given
a = [1,3,5,6,10]
b = [1,4,6,8]

the resulting array should be:

c = [1,1,3,4,5,6,6,8,10]

This method should not call a sorting method. Instead, the resulting array should be produced by "zipping" the two input arrays together: we repeatedly select the least element that we did not consider before from a and b and include this in c.

For example:

```
a = [1,3,5,6,10]
       ^
b = [1,4,6,8]
     ^
c = [1,1,3,...]
```

the arrows (^) point to the lowest element we did not consider before. Of these, element 4 from b is less than element 5 from a. For this reason, we select 4 as the next element and advance the arrow ^ for b to point to 6.

Put your code in merge.py

This program is tested via unit tests.

## 7 - Sieve of Eratosthenes

The sieve of Eratosthenes is a way of computing all the prime numbers below a certain number. (A prime number is a number that is only divisible by itself and 1). This algorithm is explained excellently **in this video (https://www.youtube.com/watch?v=klclklsWzrY)** , or you can read about this ancient algorithm **on Wikipedia (https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes#Algorithm_and_variants)** .

Implement this algorithm:

- Implement a function cross_out_multiples that takes as arguments a list of boolean values (true/false) called is_prime and a number n. The function sets the boolean values at all multiples of n (2*n, 3*n, 4*n ...) that are in the list to false.
- Implement a function sieve(n)  which gives back a list of all primes below n.

Put your code in sieve.py  This program is tested via unit tests.

This tool needs to be loaded in a new browser window

Load 4: Input parsing, more functions in a new window