

The one where numbers need to be sorted

Due 20 Oct by 23:59 **Points** 5 **Submitting** an external tool
Available until 20 Oct at 23:59

Successfully solving this problem will award the student **5 point** towards Examination 1

Requirements

- The script must pass all the automatic tests
- The script must pass a manual inspection
- **The student must write the script on his or her own**, any suspicion about cheating, cooperation, and/or plagiarism will be reported to the Discipline Board (see the [Study Guide](https://hkr.instructure.com/courses/5190/files/1027789/download?download_frd=1) ↓ (https://hkr.instructure.com/courses/5190/files/1027789/download?download_frd=1) for more information). The students script will be checked both manually and using anti-plagiarism software

Instructions

1. Analyse the problem below using [Polya's Problem solving Technique](https://math.berkeley.edu/~gmelvin/polya.pdf) ↗ (<https://math.berkeley.edu/~gmelvin/polya.pdf>)
2. Create a script named **Lab2_5.py** that solves the problem according to the given instructions
3. Pay special attention when formatting the output as it must match the specified format **exactly** (character by character) to pass the tests
4. Upload the script using the link below

Task description

A subsystem responsible for delivering priority numbers to an automated irrigation system has

stopped working and you need to deliver a quick fix that will work until the actual subsystem is fixed by a senior developer.

As you are the newest addition to the development team you have not fully grasped the complete picture of how all the systems work yet but you are confident that you can solve this as you got a few hints from a senior developer.

Here is what the senior developer told you as he was running out the door.

- There are two files with numbers in them
- Each line in the files can contain one or more numbers, if there are more than one number on a line, they are separated by spaces
- Both files will contain both odd and even numbers
- You need to create a filter that can pick all odd numbers from one file and all even numbers from the other file
- You then need to combine all the filtered out odd and even numbers (from the two files), and sort them in a reversed order using something called Bubble Sort.

He did not say it explicitly but you also understood that when you are done with the above steps, you need to display the sorted list of numbers on the screen.

When it comes to the design of the script it is pretty simple, it needs to contain the following functions.


read_file(filename)

Reads all the numbers in the specified file and adds them to a list as integers. The list is returned from the function.

filter_odd_or_even(numbers, odd)

The first parameter is a list of numbers and the second parameter is a Boolean value specifying if the filter function shall keep the odd numbers (True) or the even numbers (False). The function shall create a new list that is filled with either the odd or even numbers from the parameter list depending on the odd parameter. The new, filtered, list shall be returned from the function.

reversed_bubble_sort(numbers)

Takes a list of integer numbers as parameter and sorts it in place. Sorting it in place means there is no need to return anything from the function, the calling function will already have access to the sorted list. The sorting shall be done using [Bubble Sort](https://en.wikipedia.org/wiki/Bubble_sort) , (https://en.wikipedia.org/wiki/Bubble_sort), but in reverse order. Reverse order means that the biggest number shall be first and the smallest last. The implementation of Bubble Sort shall not be optimized using the optimization described in the link above.

main()

This function parses two command line arguments to file paths. From the first file (first argument) all odd numbers are read, and from the second file (second argument) all even numbers are read. The two lists of numbers are then combined and reverse sorted. The






result from the sort shall be displayed on screen using the format below. The main function shall use the other functions to accomplish its tasks.

```
[100, 65, 55, 46, 45, 40, 32, 23, 19, 17, 14, 9, 9, 8, 6, 4, 1]
```

To avoid problems with the testing framework, and to build a good habit, call the main function using the following construct:

```
if __name__ == '__main__':  
    main()
```

Comments and additional resources

- Students are highly recommended to first solve all the tasks from the corresponding lectures and exercises (including the extra tasks) before attempting to solve this problem. Having a solid plan before starting to code will also help
- The student can upload as many times as he or she likes to before the deadline
- These two files ([first](https://gist.github.com/andreas-hkr/489da1c0654d91d8b92d344cf65a1d56)  <https://gist.github.com/andreas-hkr/489da1c0654d91d8b92d344cf65a1d56>) & [second](https://gist.github.com/andreas-hkr/6e89dd6bf6fc5e984e430954f7956de8)  <https://gist.github.com/andreas-hkr/6e89dd6bf6fc5e984e430954f7956de8>.) can be used as an example of how the files with numbers might look like
- Watch [this](https://youtu.be/IOeIDvyRUQs)  <https://youtu.be/IOeIDvyRUQs> video if you want to learn more about why we call main the way we do
- This [video](https://youtu.be/o1Rxrtiqml8)  <https://youtu.be/o1Rxrtiqml8> covering how to use command line arguments might be useful
- This [video](https://youtu.be/zhl2ZwozxVc)  <https://youtu.be/zhl2ZwozxVc> recorded while running the finished script showcases how the script shall look and work when finished.

This tool needs to be loaded in a new browser window

The session for this tool has expired. Please reload the page to access the tool again

