Exercise set 10 – File management in Python

1. Download the file ”**artists.txt**” from Moodle and place it in your PyCharm project (same folder as your Python-file). Read the content of the file into a variable, **and print the content in a loop, one line at a time**. Each line should also have an arrow in the beginning, by using the characters "->".

   Remember to copy the file into the same folder as your Python-file! (you can use PyCharm to copy the file)

   Example of the application running:

```
Some of the best-selling music albums in history:

-> Whitney Houston, The Bodyguard, 1992
-> Eagles, Hotel California, 1976
-> Pink Floyd, The Dark Side of the Moon, 1973
-> AC/DC, Back in Black, 1980
-> Fleetwood Mac, Rumours, 1977
-> Celine Dion, Falling Into You, 1996
-> The Beatles, Sgt. Pepper's Lonely Hearts Club Band, 1967
-> ABBA, Gold: Greatest Hits, 1992
-> Dire Straits, Brothers in Arms, 1985
-> Santana, Supernatural, 1999
```

   Filename of the exercise = ***exercise10_1.py***
   Typical code amount : ***5-12 lines*** (including own function code, empty lines/comments not included)

   Note! You don't have to upload the **artists.txt** -file into Mimir, it's already there!

   Small extra task: Create also a separate function for processing the file!

2. Create a guestbook application. First, ask the user whether they want to read or write into the guestbook (r/w). If they want to read the guestbook, get all the lines from the file, and print them in a loop. If the user wants to write into the guestbook instead, add the new message in the end of the file (append, control character ***a***).

   Use the name ***guestbook.txt*** for the file**.** Create an empty file first in your PyCharm project, in the same folder where your Python-file is.

   Remember to return the **guestbook.txt** –file also into codePost!

   Extra task: Use the JSON-format in your guestbook (guestbook.json). For each message, save the current date + time as well (datetime –module!). **Note:** the easiest way is to create a list of dictionaries both in JSON and Python! You can return this version into the same return box in codePost!

   Examples of the application running:

```
Would you like to read or write into the guestbook? (r/w)
w
Write a new message:
Nice application you've got here!
```

```
Would you like to read or write into the guestbook? (r/w)
r
Greetings from Finland!
Nice application you've got here!
```

   Filename of the exercise = ***exercise10_2.py***
   Typical code amount : ***12-24 lines*** (including own function code, empty lines/comments not included)

3. Create an application that gives the user a random proverb (also known as **a traditional saying**) from a file! Download the file "**wisewords.txt**" from Moodle, and use them in your application.

   Tip: Read the **wisewords.txt** –file into a variable, and convert the proverbs into a separate list. The idea is to generate a random list index that can be used to get a random proverb. For example, if a list has 20 proverbs, then a random index would be **random.randint(0, 19)**

   Note: wisewords.txt is already in codePost, you don't have to upload it!

   The amount of all proverbs is the size of the list, but remember **-1**, because the first index is always 0. for example:
   `max_index = len(proverbs_list) - 1`

   Remember:
   `import random`

   Examples of the application running:

```
Today's proverb:
Rain before seven, fine before eleven
```

```
Today's proverb:
Honesty is the best policy
```

```
Today's proverb:
It's no use crying over spilt milk
```

   Filename of the exercise = ***exercise10_3.py***
   Typical code amount : ***7-14 lines*** (including own function code, empty lines/comments not included)

4. Download the file called ”**countries.json**” from Moodle, and add it to the same folder where your Python-code is. Read the contents of the file into a variable, and print out the contents in a loop, one country in a row (see the example below).

   If you open the JSON-file, you'll recognize it's a list containing dictionaries. The keys of the dictionary are "country" and "capital".

   Note: countries.json is already in codePost, you don't have to upload it!

   Example of the application running:

```
All countries and capitals:
Afghanistan: Kabul
Albania: Tirana
Algeria: Alger
American Samoa: Fagatogo
Andorra: Andorra la Vella
Angola: Luanda
```

   Filename of the exercise = ***exercise10_4.py***
   Typical code amount : ***7-12 lines*** (including own function code, empty lines/comments not included).
   If doing the extra task, there can be ***20-30*** code lines.

   Extra task:

   After printing the list of countries/capitals, ask the user to provide a letter (A- Z), and only print those countries/capitals that start with the given letter.

   Note: If you do this extra task, remember to also print all countries/capitals in the beginning of the application. An example:

```
Filter a country/capital with a starting letter:
S
Saint Helena: Jamestown
Saint Kitts and Nevis: Basseterre
Saint Lucia: Castries
Saint Pierre and Miquelon: Saint-Pierre
Saint Vincent and the Grenadines: Kingstown
Samoa: Apia
San Marino: San Marino
Sao Tome and Principe: S
Saudi Arabia: Riyadh
Scotland: Edinburgh
```

Extra execises!

Note: You don't necessarily have to do each extra exercise to get a good grade. It's more important to find an alternative that interests you!

5. Create a simple maintenance report application, that asks the following information from the user:

- The name of the maintenance person
- Situation code
- Description

Add the current date automatically into the data as well (date-module).

Create a dictionary of the given information, and save it in JSON-format into a file: **maintenance.json** before the application ends. When you start the application, and if the file **maintenance.json** already exists, show the contents of the file before asking the user for new information. Example:

```
Latest maintenance report...
Date: 9.11.2021
By: Jess Banning
Situation code: 36
Message: Corner room window has a severe draft - insulation needs repairing, most likely.
```

Filename of the exercise = ***exercise10_5.py***

Extra challenge:
Instead of writing on top of the earlier maintenance report, append (add) all new maintenance reports after the previous one. When starting the application, always show only the latest maintenance report. JSON is very useful in this challenge (list of dictionaries).

6. Excel/CSV:

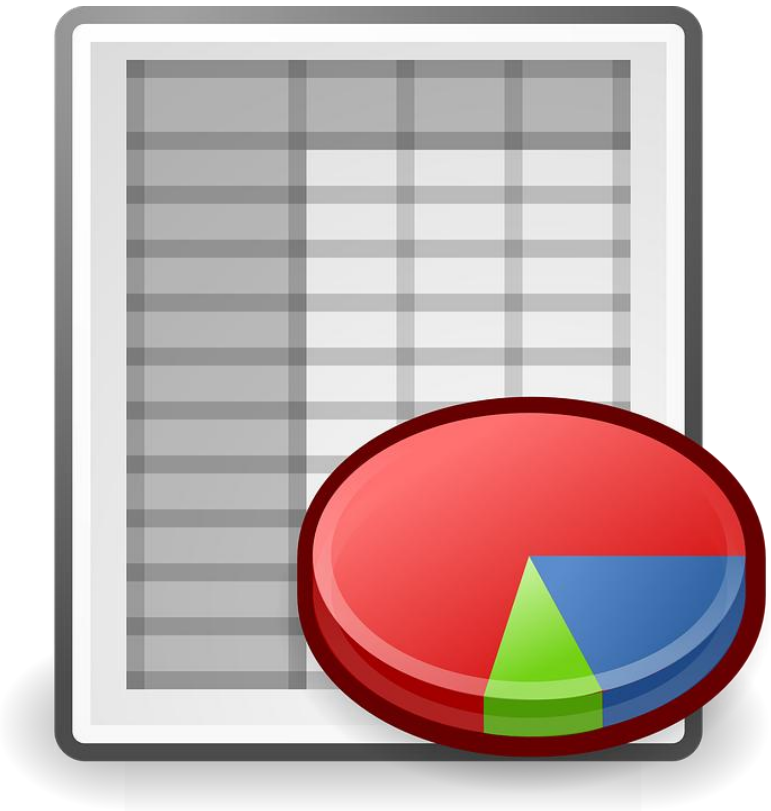
Download the ***serialtest.xlsx*** –file from Moodle, and open it in Excel. Save the file in CSV-format, and copy the CSV-file into the same folder where your Python-files are.

Note: In CSV, the values are separated by a comma (sometimes semicolon) and rows are separated by a new line (\n).

Use the CSV-module in Python, and read the contents of the file into a collection. Print out the contents by using a loop.

Filename of the exercise = ***exercise10_6.py***

Extra challenge:
Modify the original data by using Python, and increase everyone's salary by 100 €, and save the new version of the file into another CSV –file, and open it in Excel to check the results.



7. Data management (CRUD = Create, Read, Update, Delete):

Create an application that gets the names of products of a company from a file. You can decide what kind of products this company sells. Show a numbered list of products to the user in your application, and give the user the possibility to choose one product which they want to modify.

After asking the new value from the user, change the original list and allow the user to browse the products as long as they want. When the user decides to quit the application, save the modified list back into the file on top of the old one.

Filename of the exercise = ***exercise10_7.py***

Extra challenge 1:
Modify the list so that it contains dictionaries of each product. One product should have the following data: ***name, category, price***

Allow the user to modify any value of any product.

Extra challenge 2:
Allow the user to browse the products of only a single category at a time.

Extra challenge 3:
Allow te user only to add new products and delete existing ones.