This document provides a series of Python tasks. Here are a couple of pointers:

- Each following task is an improvement of the Python script made in the previous task
- You should not create separate Python scripts for each task, but rather implement all tasks in one script
- All our prepared JSON input files will be sent in attachment
- Use Python 3.6 or newer
- You can use any standard Python libraries available in clean installation of Python

1. Parsing JSON file with Python

Task:

- Input: JSON file in attachment named data_task1.json
- Write a Python script that will parse a given JSON file and find out info of all the people that drive a car from manufacturer Honda and have more than 5000 $ on their bank account.
- Info of found persons that suffice the criteria, should in the end be written in human readable format in text (.txt) file.

For example, if the input file contains these entries:


```
[
 {
   "id": 148,
   "first_name": "Ruthann",
   "last_name": "Hodjetts",
   "email": " rhodjetts43@wordpress.com ",
   "gender": "Female",
   "ip_address": "139.53.52.94",
   "bank": "$43180.67",
   "car_make": "GMC",
   "car_model": "Canyon"
 },
 {
   "id": 151,
   "first_name": "Lorianna",
   "last_name": "Joubert",
   "email": " ljoubert46@bluehost.com ",
   "gender": "Female",
   "ip_address": "177.172.128.49",
   "bank": "$12518.65",
   "car_make": "Honda",
   "car_model": "Element"
 },
 {
   "id": 153,
   "first_name": "Zora",
   "last_name": "Edmonston",
   "email": " zedmonston48@bbb.org ",
   "gender": "Female",
   "ip_address": "165.207.126.168",
   "bank": "$5906.00",
   "car_make": "Honda",
   "car_model": "Taurus"
 }
]
```

Output file should contain:

```
====================================
ID: 151
Name: Lorianna
Surname: Joubert
Email: ljoubert46@bluehost.com
Gender: Female
Bank state in USD: 11.402,99
====================================
ID: 153
Name: Zora
Surname: Edmonston
Email: zedmonston48@bbb.org
Gender: Female
Bank state in USD: 5.906,00
====================================
```

2. Robust input handling (Python Exceptions)

Make script more robust, take into account various scenarios like:

- Script should gracefully, with clear message for end user, fail when:
  - Input file doesn't exist
  - JSON structure is corrupt
- Script should print to console warning message, but not fail, when:
  - Record is corrupt (doesn't contain one of the attributes that should be printed out)
  - Attribute value is incomplete ($ sign missing from bank property)
- Script should automatically handle lowercase/uppercase attribute values (HOnda is same as hONDa)

In the end task, script should either gracefully fail or successfully parse (with warnings):

- data_task2_1.json
- data_task2_2.json
- data_task2_3.json

And no we did not forget to send you any files :)

3. Command Line Interface 1.0

Extend script to ask the user to provide a file which should be parsed via terminal/console. Be careful to:

- Provide intuitive UX (clear questions for user, clear answers when script is done)
- If algorithm fails, provide descriptive error (task 2 from before) and ask user to provide another file or fix provided one before running script (based on what went wrong)
- Handle relative and absolute paths

Example:

```
> python.exe script_name.py
Path to JSON input file: <wait for user to input path to JSON file>
Parsing...
OK
Output: <path to txt file script generated>
```

4. Command Line Interface 2.0

If script is run with file path as parameter to script, script takes provided file via parameter as input and doesn't explicitly ask the user to provide file to parse in script.

If script is still started without parameters or file provided as parameter does not exist, handling should revert back to Command Line Interface 1.0

Example 1:

```
> python.exe script_name.py myfile.json
Parsing...
OK
Output: <path to txt file script generated>
```

Example 2:

```
> python.exe script_name.py
Path to JSON input file: <wait for user to input path to JSON file>
Parsing...
OK
Output: <path to txt file script generated>
```

Example 2:

```
> python.exe script_name.py myNonExistentfile.json
ERROR: File provided as script parameter does not exist!
Path to JSON input file: <wait for user to input path to JSON file>
Parsing...
OK
Output: <path to txt file script generated>
```