Lab

Working with textual and numeric data

Dictionaries in real world (`*.json` data format)

(!) Solve the following problem and submit your results in MS Teams project within your source file with code and outputs (you can make this task in groups)

Problem

Suppose you have the data sample from HR platform HeadHunter. You need to process it and answer the following questions:

1. Which salary in Rubles is suggested in different cities? Use NET salary, round with 1 decimal place.
2. Which professions are provided in TOP-10 cities (with the largest mean salary)?
3. What is the share of vacancies, where salary is not omitted?
4. What is the share of Full-day schedule vacancies?
5. Which key skills are required in managerial vacancies?

Hints for Questions

Use the following link to obtain the data in `*.json` -format: https://api.hh.ru/vacancies?per_page=100&locale=EN. This is a dictionary with the last 100 published vacancies in HeadHunter platform. The structure of fields you can also find in official HeadHunter API.

You can SAVE AS the file in `*.json` -format (e.g. `your_file.json` and `load` into Python as Dictionary in `data` variable from your local directory (using `json` library):

```
import json

with open("your_file.json", "r") as read_file:
    data = json.load(read_file)
```

Or, you can directly download the Web-page with data (using `requests` library).

If you did not install a particular library once on your machine you can install it directly from Jupyter Notebook or form command line (terminal, prompt):

```
!pip install requests
```

Then, recognize it as dictionary with `json.loads` :

```
import json
import requests

url = 'https://api.hh.ru/vacancies?per_page=100&locale=EN'

# Load the WebPage
your_file = requests.get(url)

# Store the data (text) into Dictionary
data = json.loads(your_file.text)
```

Then, find out the structure of particular vacancies. The data about vacancies is stored with items key (according to HeadHunter API):

```
vacancies = data['items']
```

So, `vacancies` is the list of sub-Dictionaries. Check, that you've obtained 100 vacancies with `len` .

Some companies do not disclose the salary that they can suggest. Thus, instead of `salary` field it could be `None` . You need skip all `salary` fields with `None` content by introducing, e.g. the following check:

```
[i['salary'] for i in vacancies if i['salary'] is not None]
```

Then, extract only pairs (the city [`area.name`], and sub-dictionary with salaries) into new list, it is suggested to use the Tuple structure, e.g.:

```
[(i['area']['name'], i['salary']) for i in vacancies if i['salary'] is not None]
```

Now, you've obtained the list of pairs (_city_, _dictionary with salary information_). Then, you need to filter and use only vacancies with salary in `RUR` currency (`currency` field of the `salary` Dictionary). Use the previous logic.

After that, you can notify that not all companies provide the salary inside the range `from` and `to` fields. So, if the `salary` interval is open-ended, please duplicate the

known `salary`. The other moment is that the salary could be stated as GROSS or NET (`gross` field can be `True` or `False`). Thus, please calculate the NET salary in each case (assuming 13% deduction if the salary is GROSS [`gross` field equals to `True`]). However, again we can face the problem with None content along with the condition check, so, it is recommended to introduce your own function which will calculate an appropriate values. For example, you can calculate the mean NET salary for particular vacancy with the following one (the list with `from` and `to` fields from `salary` is used as input):

- Mean Salary:

```
def mymean(x):
    x = [i for i in x if i is not None]
    return sum(x) / len(x)
```

- Mean Salary with GROSS check:

```
def mymean(x, gross = False):
    x = [i for i in x if i is not None]
    if gross:
        return round((sum(x) / len(x)) * 0.87, 1)
    else:
        return round((sum(x) / len(x)), 1)
```

Calculate the mean NET salary for each vacancy and provide the list of Tuples in the following format: (_city_, _mean NET salary_).

After that, calculate mean, min, max NET salary for each city (use hints from Seminar 2 & 3 to calculate values for unique categories) and sort them in decreasing order by mean NET salary. It is suggested to organize data as list of Tuples of the following format: (_unique city_, *mean salary*, *min salary*, _max salary_).

Return to the initial vacancies' structure and extract for each unique city the set of professions' names (`name` field in `vacancies`).

Optional hint: as the most of vacancies are presented with Cyrillic letters, you can use transliteration of them with the following library (`cyrtranslit`) and example code (again, you can install this library with `pip install`):

```
# Transliteration of Cyrillic text (optional)
import cyrtranslit

cyrtranslit.to_latin('Привет мир!', 'ru')
```

If you are needed to extract particular skills that are required in particular vacancies, you can use the field `snippet` inside `vacancies`, and then, sub-filed `requirement`. Then, extract vacancies (using `name` filed) with at least one of words: `['менеджер', 'manager', 'menedzher']`. **Hint**: use `lower()` or `upper()` in order to extract such words in appropriate way.

Then, try to separate words with spaces in obtained list of vacancies (e.g. `join` at first all texts in one line, **avoid** `None`). And calculate the frequency table (more details in previous Seminar).