Описание практик, используемых в курсе

Курс «Data Analysis in Python», относящийся к курсам «Data Culture», в первую очередь ориентирован на применение возможностей языка программирования Python (язык программирования как инструмент) для решения прикладных задач, связанных с анализом данных. Поскольку данных курс, прежде всего, рассчитан на студентов 2-го курса магистратуры образовательной программы «Прикладная экономика и математические методы», основное внимание было уделено элементам курса, позволяющим студентам применять и развивать знания, полученные в курсах «Data Culture».

В качестве вышеупомянутых элементов мной было разработано и внедрено нижеследующее:

- собственный сайт курса на платформе с открытым исходным кодом Gitbook, позволяющий не только структурировать информацию о курсе, заданиях, кейсах в одном месте, но и облегчить демонстрацию раздаточных материалов без дополнительной установки специализированного программного обеспечения (демонстрация блокнотов с кодом и результатами анализа на языке Python, а также поддержка интерактивных элементов и вставок компьютерных программ прямо на сайте в формате longread). Сайт доступен по ссылке: https://ternikov.gitbook.io/da-in-python/;
- семинарские задания и кейсы, оформленные в традиции мультимедийных longread (статья, конспект или раздаточный материал для сайта, наполненные интерактивными элементами, например, цитатами, изображениями, вставками кода на Python с подсветкой синтаксиса языка программирования). Примеры заданий доступны на сайте курса https://ternikov.gitbook.io/da-in-python/, а также в приложенных к заявке файлах;
- задания для самостоятельной работы с гибкими сроками сдачи, направленные на применение знаний по «Data Culture», которые базируются на открытых материалах вступительных испытаний и собеседований на позиции аналитика данных, специалиста по моделированию экономических процессов и data scientist. Примеры заданий доступны в приложенных к заявке файлах.

В качестве результатов, достигнутых благодаря вышеупомянутым элементам, можно выделить нижеследующие аспекты:

- достигнута гибкость в терминах подачи и восприятия материалов курса студентами, а именно: студенты в любой момент имеют возможность обратиться к материалам курса, вдумчиво повторить и воспроизвести материалы семинарских занятий, имеют возможность получить обратную связь как на семинаре, так и посредствам созданного канала в Telegram;
- учитывается специфика восприятия и воспроизведения полученной в рамках курса информации каждым конкретным студентом по всем элементам курса, включающим в себя прохождение онлайн-курса, материалы семинарских занятий, ознакомление с дополнительными раздаточными материалами для углублённого усвоения по ссылкам на сайте дисциплины, финальными заданиями на оценку;
- повышена эффективность обратной связи относительно применения гибких сроков сдачи работ на оценку, а также открытости всех элементов контроля и материалов заблаговременно: студенты концентрируют своё внимание на прикладных задачах и кейсах и задают в большей степени вопросы относительно сути заданий, а не вариантах их реализации в терминах сухого написания программ на языке программирования.

Таким образом, внедрение новых элементов в курс полностью обусловлено, что подтверждается отзывами студентов и качеством их работ по итогам прочтения курса (см. Приложение к заявке).

Day 2

Managing Files. Functions and Loops

Tuples and Dictionaries Notebook

Recommended Tasks with Solutions

1. About dictionaries (all tasks)
2. https://www.practicepython.org/ (ex. 30, 31, 33, 34)
3. Read about *.csv and *.json files here

Working with textual and numeric data

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

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 | Mean Salary with GROSS check

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

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.

⚠ Important Hint

Try to separate not the words itself but phrases, e.g. the text sequences separated by dot, comma, semi-colon, etc.

In addition, remove the rest of punctuation and excessive spacing.

In order to implement this you can use **regular expressions** (the matching, splitting, replacing within introduced patterns).

Obligatory: please read the following links about regular expressions

1. **Regular Expression HOWTO**
2. **Official documentation about `re` library in Python**

In this case, we first install (if not installed) `re` library, e.g. directly from Jupyter Notebook:

```
!pip install re
```

Then, we can implement several functions from this library in order to match several textual patterns. Please find out several examples below:

Split by several symbols | Remove excessive spacing

```
import re

# Splitting by several punctuation symbols
# NOTE: some symbols are already reserved for text manipulations (metacharacters)
# If we need to use such symbols directly, we need ESCAPE them (use backslash befor
# For example, [.]-dot symbol is used for matching any character except a newline

# First, provide the split pattern (we separate several variant with [|]-pipe symbo
# And write down this pattern as string variable
# Here we use the following symbols for split: . , ; ! ( )
pattern = ',|\.|;|!|\(|\)'

# Then, we introduce the sample text and apply split function from re library
text = "Regular expressions can be concatenated to form new regular expressions; if
re.split(pattern, text)

# The result will be the list
```

ⓘ **Additional material:**

1. https://www.w3resource.com/python-exercises/re/index.php (all)

2. https://developers.google.com/edu/python/regular-expressions (materials inside and the last exercise)

Previous
Day 1

Next
Day 3

Last modified 2mo ago

Imagine, you help the producer to choose proper shooting days. The producer asks you to choose two days in order to ensure that the temperature in such days will be not more than 5 degree different between each other (by the absolute value). You know the weather forecast for n few days (the sequence of integer numbers: $t_1, t_2, \ldots, t_n$, where i - the number of the days in the increasing order, t_i - the temperature in the particular day). For two choosen days should be true the following: $|t_i - t_j| \leq 5$. You know also that the producer do not want to have long pauses between these days, so, you need also to minimize the distance between two particular days.

Input: you have the list of numbers (temperature in days, ordered by the number of the day). **Output**: write the function `mindays(list)`, that returns the minimum days of leisure (pause) in between for the producer; if no such days you can choose - return -1.

---

n people from different companies sit at round table. People are enumerated clockwise from 1 to n. The i-th person is from to company $comp_i$. You need to allocate microphones among such people before meeting stats. Assume you have an infinite number of microphones of different colour. On the behalf of companies' ethics, the i-th person does not want to sit nearby the person from the other company, who has microphone of the same colour. Finally, you should allocate microphones in such the way in order to minimize the number of their colours.

Input: You are given with the list of people (in the proper order), in which representatives of the same company are decoded with the same integer number.

Output: You need to provide the minimum number of microphones' colours.

Hint: Write the function `min_mic(list)` in order to check your code below.

Let the annual fuel supply contract of the following configuration. The buyer cannot exceed the unity as a `total sum of daily contracts` quantity (during the year). In addition, the minimum sum of contracts' quantity (during the year) should not be less than 76.5% of `total sum of daily contracts` quantity. The buyer can vary daily the `daily contracts` quantity during the year in the range from `0.0009589` to `0.002884` (per `daily contract`).

Calculate and plot the area of acceptable cumulative trajectories of fuel withdrawal by the buyer. Assume 365 days a year.