1. Retrieve phone numbers

Task: write a small program/script/function that is able to extract phone numbers from a given text.

Whenever the text contains "phone number: " the number next to it should be retrieved and stored for each cell multiple numbers can occur, in that case all numbers should be retrieved as a list.

The expected results are listed with each text sample. Please note tho, that arbitrary text could be used to evaluate your code - so your code should work for generic texts as well.

id	text	result
1	This is a sample text provided for training purposes. In this text the goal is to retrieve the following phone number: 123456. Its interesting because the phone number can be retrieved using various different techniques and approaches. Very curious also if the retrieved phone number will be valid.	123456
2	Another sample text with a phone number. Phone number: 432562143	432562143
3	phone number: 654321	654321
4	not a phone number: 23456211	23456211
5	just a number: 123432	-
6	text without a phone number	-
7	""	-
8	phone number: 654321, and another one, phone number: 43256123	654321, 43256123

You can use any programming language and/or framework as long as you can explain the logic behind it (please dont just copy and paste some random code without understanding it).

2. Frontend CSV Export

As a user, I want to be able to download the data I am seeing in my Analytics section.

Acceptance criteria:

- add a new button with label "Download CSV"
- clicking the button flattens the provided data structure and present the user a file dialog to select a download destination (everything happening in the frontend/in javascript)
    - The data flattening is based on the answers and objections lists inside each item
    - Every object inside each of the lists is mapped to a row in the CSV file
    - The mapping is as follows:
        - item_name - based on the item, stays the same for multiple answers/objections
        - rel_answer - based on the item, stays the same for multiple answers/objections
        - rel_objection - similar to rel_answer,
        - text - based on either the answer (for answers) or objection (for objections), does change in every row, multiple occurrences are possible if the same text is present for different items
        - count - based on the answer/objection, describes how often the answer/objection occurred
        - rel_count - relative count of the answer/objection
        - type - either ‘Answer’ or ‘Objection’ depending on the object type (note: can be “hardcoded”)
- confirming the dialog "downloads" the data as a CSV file

Outline (suggestion):

- Flatten data based on answers and objections in every single item
- create CSV with flattened data
- open download dialog using common JS API

Raw Data structure: see JSON file / below

CSV data structure:
item_name, rel_answer, rel_objection, text, count, rel_count, type

Examples:

```
'Vorbereitung: Taktik',0.1657754010695187,0.0,'Direkte Durchwahl / Handy',22,0.1176,'Answer'

'Vorbereitung: Taktik',0.1657754010695187,0.0,'Schicken Sie Unterlagen',1,0.0053,'Objection'
```

Note:

- the type of a row is dependent on the original data structure its based on (either answers or objections)
- some values repeat for two rows, that is because the data is based on the item but the rows are based on nested objects, this is intended.

JSON:


```
{
    "name": "Sales: Cold Call (05/2021) (ungerade Tage)",
    "items": [
        {
            "name": "Vorbereitung: Taktik",
            "answers": [
                {
                    "text": "Direkte Durchwahl / Handy",
                    "count": 22,
                    "totalImportance": 110,
                    "rel_count": 0.1176
                },
                {
                    "text": "Zentrale",
                    "count": 9,
                    "totalImportance": 45,
                    "rel_count": 0.0481
                },
                {
                    "text": "Unbekannt",
```

```
                "count": 1,
                "totalImportance": 5,
                "rel_count": 0.0053
            }
        ],
        "total_answers": 32,
        "rel_answer": 0.1657754010695187,
        "objections": [
            {
                "text": "Schicken Sie Unterlagen",
                "count": 1,
                "rel_count": 0.0053
            }
        ],
        "total_objections": 1,
        "rel_objection": 0.0
    },
    {
        "name": "Zentrale",
        "answers": [],
        "total_answers": 0,
        "rel_answer": 0.0,
        "objections": [
            {
                "text": "Schicken Sie Unterlagen",
                "count": 2,
                "rel_count": 0.0107
            },
            {
                "text": "Haben schon eine Lösung",
                "count": 1,
                "rel_count": 0.0053
            },
            {
                "text": "Kein Bedarf / kein Interesse",
                "count": 1,
                "rel_count": 0.0053
            }
        ],
```

```
            "total_objections": 4,
            "rel_objection": 0.0
        },
        {
            "name": "Pains",
            "answers": [
                {
                    "text": "Dokumentation",
                    "count": 8,
                    "totalImportance": 40,
                    "rel_count": 0.0428
                },
                {
                    "text": "Vor- und Einwände",
                    "count": 6,
                    "totalImportance": 30,
                    "rel_count": 0.0321
                },
                {
                    "text": "Zielerreichung",
                    "count": 6,
                    "totalImportance": 30,
                    "rel_count": 0.0321
                }
            ],
            "total_answers": 20,
            "rel_answer": 0.0427807486631016,
            "objections": [
                {
                    "text": "Haben schon eine Lösung",
                    "count": 4,
                    "rel_count": 0.0214
                },
                {
                    "text": "Schicken Sie Unterlagen",
                    "count": 3,
                    "rel_count": 0.016
                },
                {
```

```
                    "text": "Individuelle Gespräche",
                    "count": 2,
                    "rel_count": 0.0107
                },
                {
                    "text": "Kein Bedarf / kein Interesse",
                    "count": 2,
                    "rel_count": 0.0107
                }
            ],
            "total_objections": 11,
            "rel_objection": 0.0
        }
    ]
}
```

3. Code Review

```
# Python 3 code

def udb(*args, **kargs):
      pass

def asd(*args, **kargs):
      pass

def validate(parameter):
      if not parameter:
            raise Exception("Invalid input")

def myAwesomeFunction(self, arg1, *kwargs, args, key_word=None):
      if key_word.test():
            return True
      else:
            print(arg1)
            validate(kwargs.get('arg1'), None)
            try:
                  udb(args.get('arg1'))
                  asd(arg1)
            except:
                  print('There is a problem!')
            for f in list(arg1):
                  success = myAwesomeFunction(f, arg1)
            return success
```