General Guidance

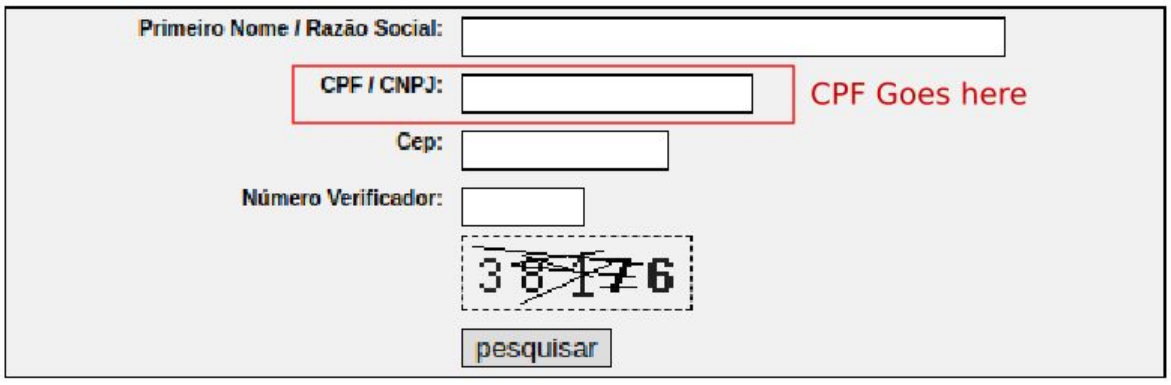
This is a guide with the information you need to build the solution. Here we present you the workflow we intent to execute in the web scraping and the expected outcome

Name: Totalexpress
Host: http://tracking.totalexpress.com.br/tracking/0?cpf_cnpj

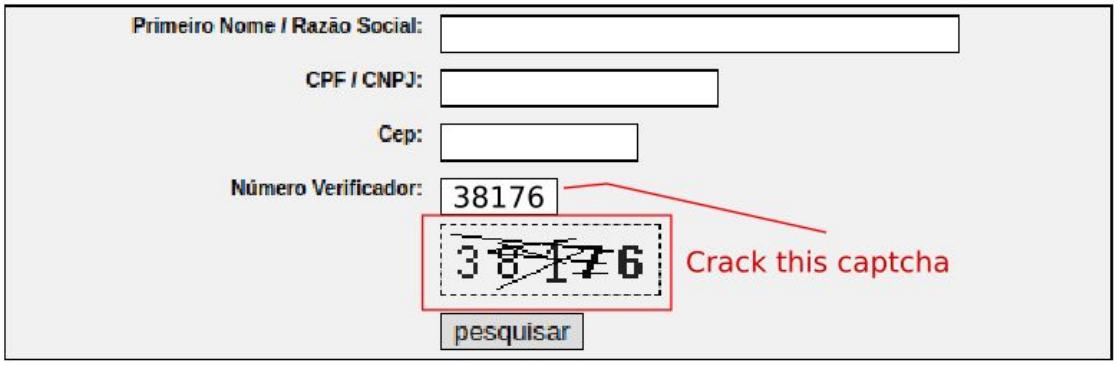
Workflow

1- Access http://tracking.totalexpress.com.br/tracking/0?cpf_cnpj

2- Insert the given input in the “CPF”[“cpf”], “Primeiro Nome”[“name”] and “Cep”[“cep”] fields. The input will be a python dictionary containing the “cpf”, “cep” and “name” keys



3- Crack the captcha challenge and insert the result in the “Número Verificador” field



4- Click the “Pesquisar” buttom

5- Collect the resulting Data

Collecting the resulting data

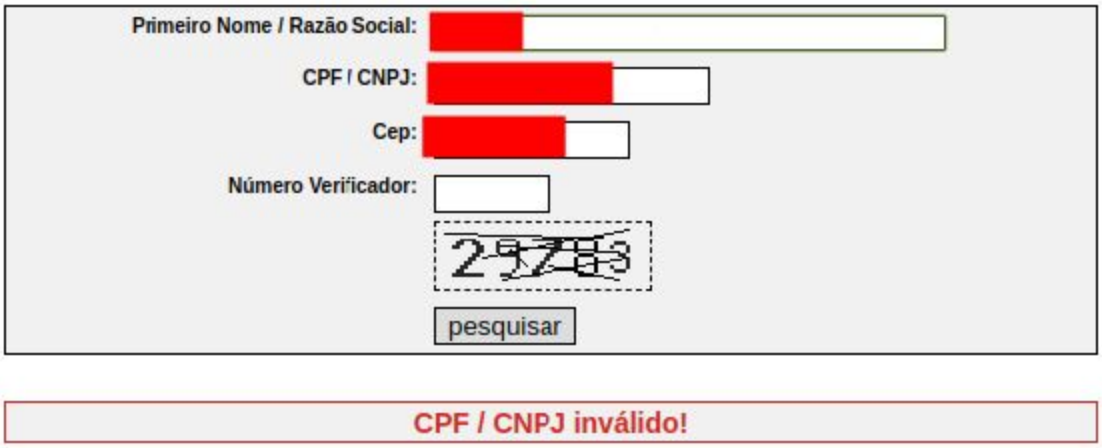
Once you have successfully entered the captcha challenge, you may encounter 3 possible outcomes:

- **Not found**
- **Information Found**

For each type of result, you will produce a JSON result. Your code must **always** return a python dictionary.

Not Found

You may encounter the following screens when executing the scraping:

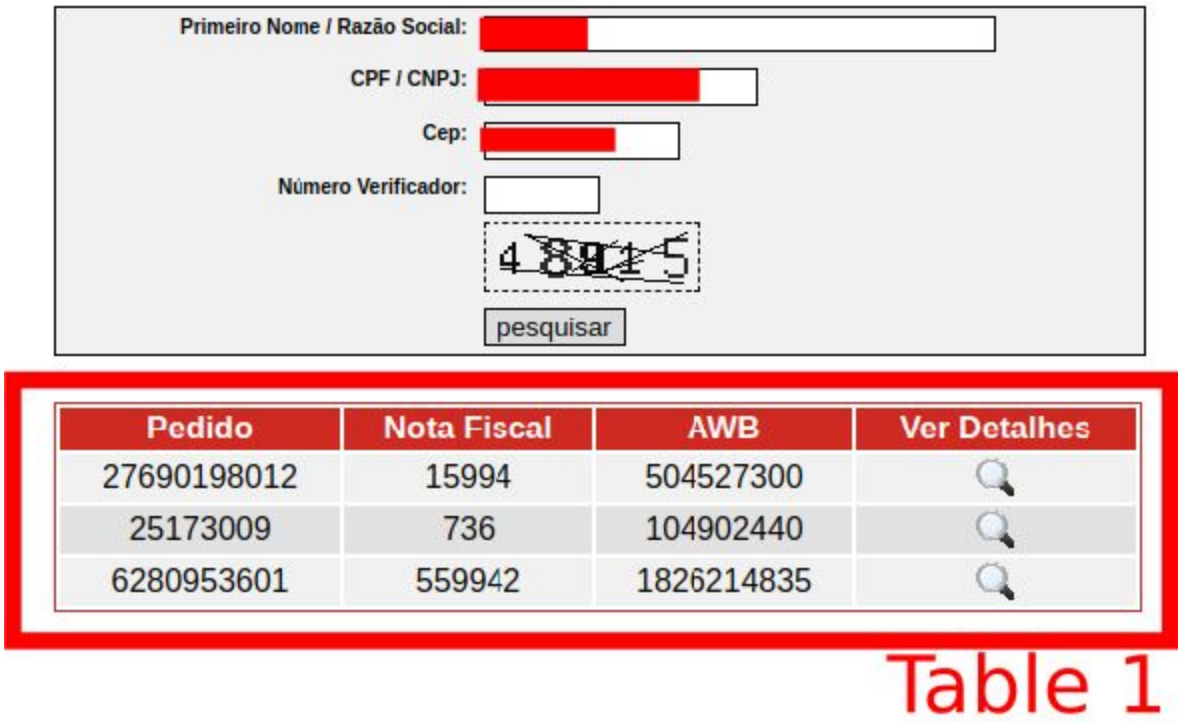


If you received any of those screens, you must return a dictionary following the template:

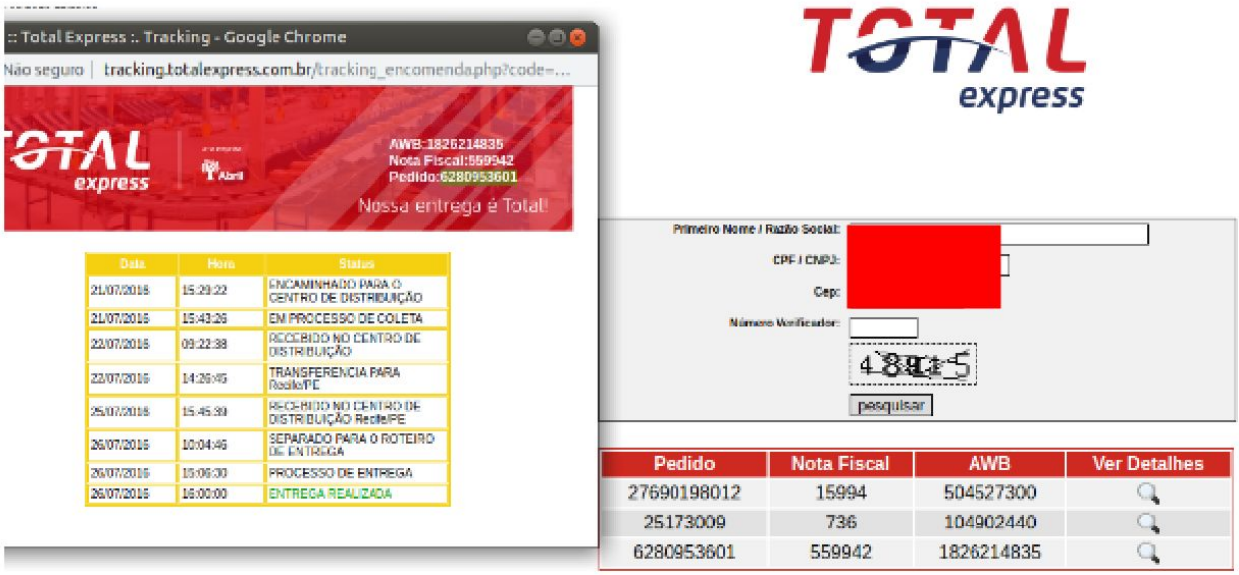
```
{
   "found_packages": False,
   "total_packages": 0,
   "packages": []
}
```

Information Found

The expected result you will find when executing the scraper is the following:



When you reach that screen, you must click in all the rows in the “table 1” section and collect the information that will open up.



This is the screen that will open up

You need to collect all the data and structure it in the JSON format. Each line in the “table 1” table will produce an item in the awway “packages”. (See the resulting JSON)

Data	Hora	Status
21/07/2016	15:29:22	ENCAMINHADO PARA O CENTRO DE DISTRIBUIÇÃO
21/07/2016	15:43:26	EM PROCESSO DE COLETA
22/07/2016	09:22:38	RECEBIDO NO CENTRO DE DISTRIBUIÇÃO
22/07/2016	14:26:45	TRANSFERENCIA PARA Recife/PE
25/07/2016	15:45:39	RECEBIDO NO CENTRO DE DISTRIBUIÇÃO Recife/PE
26/07/2016	10:04:46	SEPARADO PARA O ROTEIRO DE ENTREGA
26/07/2016	15:06:30	PROCESSO DE ENTREGA
26/07/2016	16:00:00	ENTREGA REALIZADA

For each line in the “Data” and “Status” columns, you will create a new entry in the status_list array in the resulting JSON.

If the “Status” line has the string “ENTREGA REALIZADA”, this is the “delivery_date” value.

Resulting JSON:

```
{
   "found_packages": true,
   "total_packages": 3,
```

```
   "packages": [
       {
           "delivery_date": "26/07/2016",
           "package_id": "6280953601",
           "status_list": [
               {
                   "date": "21/07/2016",
                   "status": "ENCAMINHADO PARA O CENTRO DE DISTRIBUIÇÃO"
               },
               {
                   "date": "21/07/2016",
                   "status": "EM PROCESSO DE COLETA"
               },
               {
                   "date": "22/07/2016",
                   "status": "RECEBIDO NO CENTRO DE DISTRIBUIÇÃO"
               },
               {
                   "date": "22/07/2016",
                   "status": "TRANSFERENCIA PARA Recife/PE"
               },
               {
                   "date": "25/07/2016",
                   "status": "RECEBIDO NO CENTRO DE DISTRIBUIÇÃO Recife/PE"
               },
               {
                   "date": "26/07/2016",
                   "status": "SEPARADO PARA O ROTEIRO DE ENTREGA"
               },
               {
                   "date": "26/07/2016",
                   "status": "PROCESSO DE ENTREGA"
               },
               {
                   "date": "26/07/2016",
                   "status": "ENTREGA REALIZADA"
               }
           ]
       }
   ]
}
```


NOTE:
For simplicity sake, we omitted the other lines from the resulting JSON. But keep in mind that you need to scrap the data from **all the lines** in the “Table 1” table and add them to the “packages” array. We have attached a “result_example” containing a full example of accepted result

Implementation Details

You will receive a zip file containing files that will help you build the scraper code in the desired patterns. You need to fill the PES014.py file with your code. The file basically contains a skeleton of what we expect from a scraper that will be executed in our architecture. You can notice that there is a “test_request” method implemented. This method contains a base case of success and it may serve you as a guide to understand the final result we need your code to return.

For further details about how to build an acceptable scraper, access:

https://hydra.neurolake.io/hydra_sdk/index.html
https://hydra.neurolake.io/hydra_sdk/usage/basic_concepts.html
https://hydra.neurolake.io/hydra_sdk/usage/quickstart.html