

Contents

1	Project description	2
1.1	General Description	2
1.1.1	Use	2
1.1.2	Features	3
1.2	Plan	3
1.2.1	Ebay jobs quick scrap	3
1.2.2	Blogging	3
1.2.3	Courses	3
1.2.4	Jobs seeker	4
1.3	Implementation	4
1.3.1	Start a clean project	4
2	Explorer	4
2.1	Imports	4
2.1.1	ipython	4
2.1.2	pandas	5
2.2	Data load	5
2.2.1	load everything	5
2.2.2	rename	7
2.2.3	python example TEST	7
2.2.4	org doc elisp example TEST	7
2.2.5	python PYTHON	7
2.3	Manipulation	8
2.3.1	Pioneer	8
2.3.2	cleansing CLEAN	9
2.3.3	filtering	11
2.4	Stats	13
2.4.1	overview	13
2.4.2	days ago	14
2.4.3	companies	18
2.5	Words	24
2.5.1	most used word	24
2.6	Printing	24
2.6.1	quick overview	24
2.6.2	html pages	25
2.6.3	server	28
2.6.4	org table (python) PYTHON	28
2.6.5	org results: html TEST	29

2.6.6	soupprint	29
2.7	Queries	30
2.7.1	get queries metadata	30
2.7.2	launch scraper with the list	37
3	Documentation	37
3.1	doc : look for matching patern DOC	37
3.2	pandas	38
4	Tests	38
4.1	ob-ipython	38
4.1.1	hands-on tryout	38
4.1.2	doc tutorial	38
4.1.3	other tryouts	39
4.2	nlk	40
4.2.1	text selection	40
4.2.2	search	41
4.2.3	generation TEST	42
4.2.4	normalizing	42
4.2.5	vocabulary	42
4.2.6	TODO Build a corpus !	51

1 Project description

1.1 General Description

Find jobs

1.1.1 Use

1. target opportunities
 - (a) sheets of wanted words
 - (b) query matching algorithms
2. data exploration
3. cluster
 - (a) nlp
4. find jobs I didn't know about

5. get warned if new opportunities
6. use it as a model for finding my perfect match in the world / exploring the economy
7. make it open source and useable by anyone

1.1.2 Features

1. Update
2. Clustering
3. Visualization

1.2 Plan

1.2.1 Ebay jobs quick scrap

1. Think about it while normal digging
2. Build a simple tool to access the info offline and stay up to date
3. List the wanted features and their learning prerequisites

1.2.2 Blogging

1. Org babel
2. Website

1.2.3 Courses

1. Databases
2. Visualization
3. Machine learning
4. NLP
5. Hash tables / numpy computation
6. Proba / stats

1.2.4 Jobs seeker

1.3 Implementation

1.3.1 Start a clean project

1. **TODO** git
 - (a) a branch per functionality
2. **TODO** projectile
3. file system
 - (a) /
 - i. org
 - ii. scraper
 - iii. database
 - iv. explorer
4. database
 - (a) sql ?
 - (b) csv ?
5. org babel file / emacs env
 - (a) snippets C-c & ... Tables C-c C-t is snippet mode for test
 - (b) **TODO** track time
 - (c) track habits
 - (d) decide what goes public and what does not at expansion

2 Explorer

2.1 Imports

2.1.1 ipython

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
```

2.1.2 pandas

```
import pandas as pd
```

2.2 Data load

2.2.1 load everything

1. **TODO** time range selection

```
['bücherei.csv',  
'anfänger.csv',  
'digital art.csv',  
'graphql.log',  
'google trends.log',  
'jenkins.log',  
'cuisine.csv',  
'blumen.csv',  
'computer vision.csv',  
'küchenhilfe.csv',  
'scrapping.csv',  
'pilzen.csv',  
'virtual reality.csv',  
'google trends.csv',  
'vr.csv',  
'computer vision.log',  
'mushrooms.csv',  
'docker.log',  
'advertisement.csv',  
'buchhandel.csv',  
'flowers.csv',  
'digital artist.csv',  
'graphql.csv',  
'yoga.csv',  
'jenkins.csv',  
'museum.csv',  
'advertisement.csv',  
'küche.csv',  
'fintech.csv',  
'flower.csv',  
'movie.csv',
```

```

'restaurant.csv',
'crackers.csv',
'docker.csv',
'bio.csv',
'crackers.log',
'garden.csv',
'short movie.csv',
'gardening.csv',
'schneiderei.csv',
'heroku.csv',
'hammam.csv',
'advertisement.log',
'kunst und medien.csv',
'spa.csv']

```

2. file list with path

```

import os
csv_files = []
for dirpath, dirs, files in os.walk("../data/raw"):
    for filename in files:
        fname = os.path.join(dirpath, filename)
        if fname.endswith('.csv'):
            csv_files.append(fname)

```

3. dataframe creation

```

jobs = pd.DataFrame()

for fl in csv_files:
    print(fl+(30-len(fl)//2)*" *")
    try:
        jobs_set = pd.read_csv(fl)
        jobs_set.dropna(axis=0, how='any', subset=["desc"], inplace=True)
        jobs_set.drop_duplicates(subset="desc", inplace=True)
        try:
            jobs.iloc[0,0]
            jobs = jobs.append(jobs_set)
        except IndexError:
            jobs = jobs_set
    
```

```
except pd.errors.EmptyDataError:
    pass
```

2.2.2 rename

use to quickly reset original df

```
df = jobs
```

2.2.3 python example

TEST

```
x = 12
return x

return int(x)+1
```

2.2.4 org doc elisp example

TEST

```
1
2
3
4
```

(length table)

2.2.5 python

PYTHON

```
"~/data/projects/jobseeker/data/raw/18-09-07/dsp.csv"

None

"~/data/projects/jobseeker/data/raw/18-09-07/dsp.csv"

None

"~/data/projects/jobseeker/data/raw/18-09-07/python.csv"

None

"~/data/projects/jobseeker/data/raw/18-09-07/data scientist.csv"

None

"~/data/projects/jobseeker/data/raw/18-09-07/software engineer.csv"

None
```

2.3 Manipulation

2.3.1 Pioneer

1. get data from path as org variable

```
import pandas as pd
df = pd.read_csv(data)
```

2. infos about data

```
df.count()
```

3. show short data insight

- (a) raw pandas output

```
df.head()
```

4. browse offers

- (a) add custom function to prettyfy

```
from bs4 import BeautifulSoup
```

```
def souper(html):
    soup = BeautifulSoup(html, 'html.parser')
    print(soup.get_text())
```

```
def soupprint(df, begin, end):
    for i in range(begin, end):
        print(i, df.title.iloc[i])
        print("\n")
        print(df.company.iloc[i])
        print("\n")
        souper(df.desc.iloc[i])
        print("\n"*3)
        print("-"*100)
        print("\n"*3)
```

- (b) print it !

```
soupprint(head,0,3)
```


2.3.2 cleansing

CLEAN

1. duplicates

(a) `drop_duplicates`

```
df.drop_duplicates(subset="desc", inplace=True)
```

(b) `count`

```
df.title.count()
```

```
11636
```

2. olders

(a) `map lambda`

TEST

```
df = df[df.days_ago.str.contains("30+").map(lambda x: not x)]
```

(b) `~`

TEST

```
df = ~df[df.days_ago.str.contains("30+")]
```

(c) `==False`

```
df = df[df.days_ago.str.contains("30+")==False]
```

(d) `count`

```
len(df)
```

```
5578
```

3. string numbers to integers

(a) `sol`

```
df["days_ago"] = df.days_ago.apply(lambda x: int(x))
```

(b) `test`

```
df.days_ago.iloc[12]
```

```
3
```

4. drop erratic values

(a) `run`

```
df = df[df.days_ago.lt(30)]
```

(b) tests

```
df.days_ago.lt(30)
```

3	True
12	True
14	True
15	True
19	True
23	True
27	True
28	True
35	True
38	True
45	True
48	True
55	True
57	True
59	True
62	True
63	True
64	True
65	True
66	True
75	True
79	True
82	True
87	True
91	True
92	True
93	True
94	True
96	True
100	True
...	
44	True
46	True
49	True
54	True
55	True
65	True

```

68      True
69      True
70      True
74      True
77      True
82      True
84      True
87      True
89      True
90      True
93      True
95      True
96      True
97      True
102     True
105     True
109     True
115     True
116     True
119     True
121     True
124     True
126     True
2       True
Name: days_ago, Length: 1625, dtype: bool

```

2.3.3 filtering

1. look for keywords

(a) keyword definition

i. org variable

```
"kunst und medien"
```

```
kunst und medien
```

(b) look in title

i. boolean serie construction

```
df.title.str.contains(k, case=False)
```

```
3      False
```

```
12     False
```

TEST

14	False
15	False
19	False
23	False
27	False
28	False
35	False
38	False
45	False
48	False
55	False
57	False
59	False
62	False
63	False
64	False
65	False
66	False
75	False
79	False
82	False
87	False
91	False
92	False
93	False
94	False
96	False
100	False
...	
44	False
46	False
49	False
54	False
55	False
65	False
68	False
69	False
70	False
74	False
77	False

```

82      False
84      False
87      False
89      False
90      False
93      False
95      False
96      False
97      False
102     False
105     False
109     False
115     False
116     False
119     False
121     False
124     False
126     False
2       False
Name: title, Length: 1623, dtype: bool

```

ii. reduction of our dataset

```
df = df[df.title.str.contains(k, case=False, na=False)]
```

(c) look in description

```
df = df[df.desc.str.contains(k, case=False, na=False)]
```

(d) **TODO** test goto Johnny Kitchen

```

k
"# Out[91]:\n: 'database'"

```

2. companies

```
df = df[df.company.str.contains("berlin", case=False, na=False)]
```

2.4 Stats

2.4.1 overview

1. head

```
df.head()
```

```
Empty DataFrame
Columns: [location, related, title, url, company, days_ago, contract, desc]
Index: []
```

2. count

```
len(df)
```

```
0
```

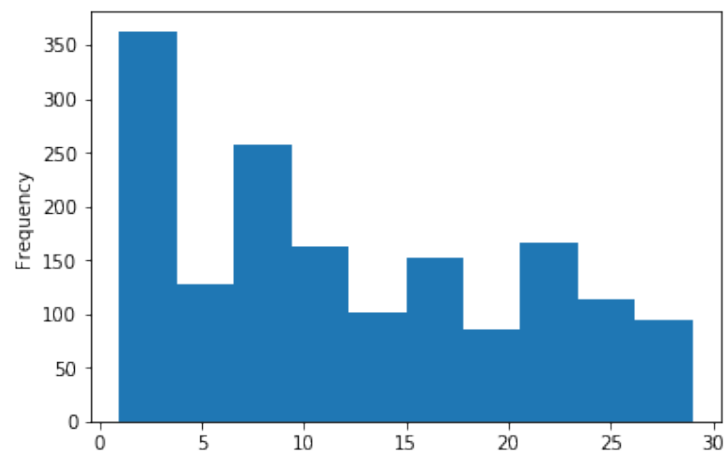
2.4.2 days ago

1. histogram

(a) pd plot

```
df.days_ago.plot.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1136869c18>
```



2. value count

```
df.days_ago.value_counts()
```

```
3      136
1      125
9      115
```

2	109
8	80
7	74
4	71
10	70
23	68
11	68
14	68
24	57
17	56
16	55
21	55
18	52
15	48
22	47
25	46
28	40
6	39
29	38
13	35
12	28
5	23
27	20
20	19
19	16
26	13
46	1
56	1

Name: days_ago, dtype: int64

3. groupby

(a) basic output

```
df.groupby(["days_ago"]).groups
```

```
{1: Int64Index([20, 25, 49, 136], dtype='int64'),
 2: Int64Index([2, 4, 10, 30, 71, 77, 116, 125, 139], dtype='int64'),
 3: Int64Index([27, 54, 73, 98, 106, 128], dtype='int64'),
 4: Int64Index([29, 32, 60, 97, 114, 119, 143], dtype='int64'),
 5: Int64Index([50, 135], dtype='int64'),
```

```

6: Int64Index([129], dtype='int64'),
7: Int64Index([127], dtype='int64'),
8: Int64Index([104, 112, 113, 121, 138], dtype='int64'),
9: Int64Index([142], dtype='int64'),
10: Int64Index([3, 96], dtype='int64'),
11: Int64Index([86, 132], dtype='int64'),
12: Int64Index([109], dtype='int64'),
13: Int64Index([31], dtype='int64'),
14: Int64Index([22, 24, 95], dtype='int64'),
16: Int64Index([47], dtype='int64'),
17: Int64Index([6, 37, 41], dtype='int64'),
18: Int64Index([80], dtype='int64'),
20: Int64Index([79], dtype='int64'),
21: Int64Index([55], dtype='int64'),
22: Int64Index([1, 144], dtype='int64'),
23: Int64Index([21, 52, 75, 110], dtype='int64'),
24: Int64Index([66, 67], dtype='int64'),
25: Int64Index([14], dtype='int64'),
26: Int64Index([91], dtype='int64'),
27: Int64Index([48], dtype='int64'),
29: Int64Index([145], dtype='int64')}]

```

(b) loop print

```

grouped = df.groupby("days_ago")

for name,group in grouped:
    print(name)
    print(group)

```

(c) documentation

DOC

i. pandas doc

```

help(df.groupby(["days_ago"]))
Help on DataFrameGroupBy in module pandas.core.groupby
object:
class DataFrameGroupBy(NDFrameGroupBy)

```


Class for grouping and aggregating relational data. See `aggregate`, `transform`, and `apply` functions on this object.

It's easiest to use `obj.groupby(...)` to use `GroupBy`, but you can also do:

```
::
```

```
grouped = groupby(obj, ...)
```

Parameters

`obj` : pandas object
`axis` : int, default 0
`level` : int, default None
Level of MultiIndex
`groupings` : list of Grouping objects
Most users should ignore this
`exclusions` : array-like, optional
List of columns to exclude
`name` : string
Most users should ignore this

Notes

After grouping, see `aggregate`, `apply`, and `transform` functions. Here are some other brief notes about usage. When grouping by multiple groups, the result index will be a `MultiIndex` (hierarchical) by default.

Iteration produces (key, group) tuples, i.e. chunking the data by group. So you can write code like:

```
::
```

```
grouped = obj.groupby(keys, axis=axis)
for key, group in grouped:
    # do something with the data
```

Function calls on `GroupBy`, if not specially implemented, "dispatch" to the grouped data. So if you group a `DataFrame` and wish to invoke the `std()` method on each group, you can simply do:

```
::
```

17

```
df.groupby mapper).std()
```

rather than

```
::
```

ii. tutorial https://www.tutorialspoint.com/python_pandas/python_pandas_groupby.htm

(d) use

```
grouped = df.groupby(["days_ago"])
grouped.title.count().sort_values(ascending=False)
```

days_ago

2	9
4	7
3	6
8	5
1	4
23	4
17	3
14	3
10	2
24	2
22	2
5	2
11	2
6	1
7	1
9	1
29	1
12	1
27	1
16	1
18	1
20	1
21	1
25	1
26	1
13	1

Name: title, dtype: int64

2.4.3 companies

1. groupby

(a) define group

```

comp_group = df.groupby(["company"])

(b) print groups

comp_group.groups

{'All My Homes': Int64Index([104], dtype='int64'),
'Ares Tech GmbH': Int64Index([80], dtype='int64'),
'Arweave': Int64Index([113], dtype='int64'),
'Asana Rebel': Int64Index([47], dtype='int64'),
'Atfarm': Int64Index([22], dtype='int64'),
'Atos': Int64Index([14], dtype='int64'),
'Avabis GmbH': Int64Index([135], dtype='int64'),
'BankenScore.de': Int64Index([132], dtype='int64'),
'BigchainDB': Int64Index([143], dtype='int64'),
'Bosch Software Innovations': Int64Index([139], dtype='int64'),
'CGI': Int64Index([121], dtype='int64'),
'Carmaq GmbH': Int64Index([2, 4, 10, 125], dtype='int64'),
'Conrad Electronic': Int64Index([50], dtype='int64'),
'Detecon': Int64Index([60], dtype='int64'),
'Deutsche Telekom AG, VTI': Int64Index([79], dtype='int64'),
'Door2Door': Int64Index([30], dtype='int64'),
'Fraunhofer-Institut für Nachrichtentechnik, Heinrich-Hertz-Institut': Int64Index([118], dtype='int64'),
'Freie Universität': Int64Index([119], dtype='int64'),
'GIM - Gesellschaft für Innovative Marktforschung mbH': Int64Index([71], dtype='int64'),
'Get It Done': Int64Index([106], dtype='int64'),
'Goldland Media GmbH': Int64Index([116], dtype='int64'),
'Hays': Int64Index([20], dtype='int64'),
'HelloFresh': Int64Index([109], dtype='int64'),
'JLink connecting experts GmbH': Int64Index([29], dtype='int64'),
'Joblift GmbH': Int64Index([73], dtype='int64'),
'KLEO Connect': Int64Index([95], dtype='int64'),
'Klarna': Int64Index([145], dtype='int64'),
'Lesara GmbH': Int64Index([41], dtype='int64'),
'Menzel IT GmbH': Int64Index([112], dtype='int64'),
'Modis GmbH': Int64Index([24, 37], dtype='int64'),
'NVIDIA': Int64Index([142], dtype='int64'),
'Novate IT Ltd': Int64Index([27], dtype='int64'),
'Planet Expat': Int64Index([97], dtype='int64'),
'Project A Ventures': Int64Index([32, 49, 86], dtype='int64'),
'Publicis Pixelpark': Int64Index([77], dtype='int64'),

```

```

'Qtixx GmbH': Int64Index([1], dtype='int64'),
'Rakuten Deutschland GmbH': Int64Index([129], dtype='int64'),
'Relayr': Int64Index([128], dtype='int64'),
'ResearchGate GmbH': Int64Index([31], dtype='int64'),
'Retresco': Int64Index([114], dtype='int64'),
'Scout24': Int64Index([75], dtype='int64'),
'Sixt GmbH & Co. Autovermietung KG': Int64Index([6], dtype='int64'),
'Sparkassen-Finanzportal GmbH': Int64Index([52], dtype='int64'),
'Sparks42': Int64Index([48], dtype='int64'),
'Technische Universität Berlin': Int64Index([138], dtype='int64'),
'Tillhub GmbH': Int64Index([98], dtype='int64'),
'Twilio': Int64Index([21], dtype='int64'),
'Two Visions Consulting OHG': Int64Index([67], dtype='int64'),
'TÜV Rheinland Group': Int64Index([55], dtype='int64'),
'Upvest': Int64Index([96], dtype='int64'),
'Volkswagen AG': Int64Index([127], dtype='int64'),
'YEAY GmbH': Int64Index([66], dtype='int64'),
'car2go Group GmbH': Int64Index([110], dtype='int64'),
'eBay Inc.': Int64Index([3], dtype='int64'),
'mytaxi!': Int64Index([54], dtype='int64'),
'omni:us': Int64Index([25], dtype='int64'),
'scondoo GmbH': Int64Index([91], dtype='int64'),
'solvemate GmbH': Int64Index([136], dtype='int64')}]

```

(c) count groups

```
len(comp_group.groups)
```

58

(d) number of job per company

i. hack

A. loop

```

for company in comp_group.groups.keys():
    lenght = len(comp_group.groups[company])
    if lenght > 1:
        print(company, lenght)

```

B. single

```

key = list(comp_group.groups.keys())[0]
list(comp_group.groups[key])

```

[32, 49, 86]

C. test

```
len(comp_group.groups["Fraunhofer-Institut für Nachrichtentechnik, Hei.
```

1

ii. pandas

```
count = comp_group.title.count()
count.sort_values(ascending=False)
```

company

Carmeq GmbH

4

Project A Ventures

3

Modis GmbH

2

solvemate GmbH

1

Door2Door

1

KLEO Connect

1

Joblift GmbH

1

JLink connecting experts GmbH

1

HelloFresh

1

Hays

1

Goldland Media GmbH

1

Get It Done

1

GIM - Gesellschaft für Innovative Marktforschung mbH

1

Freie Universität

1

Fraunhofer-Institut für Nachrichtentechnik, Heinrich-Hertz-Institut

1

Deutsche Telekom AG, VTI

1

Lesara GmbH

1

Detecon

1

Conrad Electronic

1

CGI

1

Bosch Software Innovations

1

BigchainDB

1

BankenScore.de

1

Avabis GmbH

1

Atos

1

Atfarm

1

Asana Rebel

1

Arweave

1

Ares Tech GmbH

1

Klarna

1

Menzel IT GmbH

1

scondoo GmbH	1
Technische Universität Berlin	1
omni:us	1
mytaxi!	1
eBay Inc.	1
car2go Group GmbH	1
YEAY GmbH	1
Volkswagen AG	1
Upvest	1
TÜV Rheinland Group	1
Two Visions Consulting OHG	1
Twilio	1
Tillhub GmbH	1
Sparks42	1
NVIDIA	1
Sparkassen-Finanzportal GmbH	1
Sixt GmbH & Co. Autovermietung KG	1
Scout24	1
Retresco	1
ResearchGate GmbH	1
Relayr	1
Rakuten Deutschland GmbH	1
Qtixx GmbH	1
Publicis Pixelpark	1
Planet Expat	1
Novate IT Ltd	1
All My Homes	1
Name: title, dtype: int64	

2. value count

```
df.company.value_counts()
```

Status	7
ZipJet	7
MVP Factory	6
Zattoo	5
virtualQ GmbH	5
Hays	5
8fit	5

SumUp	4
Relayr	4
Opitz Personalberatung	4
Book a Street Artist	4
nteam GmbH	3
trecker.com	3
unu GmbH	3
medneo GmbH	3
Chatterbug	3
SmartRecruiters Inc	3
SAP	3
YARA Digital Farming Niederlassung YARA GmbH&Co KG	3
Computer Manufaktur	3
blogfoster	3
DATAGROUP Inshore Services GmbH	3
Planet	3
Oracle	3
flowkey GmbH	3
Cogs Agency	2
Bidmanagement	2
Ares Tech GmbH	2
media.net berlinbrandenburg	2
Mirantis	2
..	
Mason Bedford	1
YND Consult GmbH	1
DB	1
Translation Royale	1
perZukunft	1
Cornerstone Search Group, LLC	1
Bloomberg	1
data Artisans	1
CORE	1
Brandnew IO	1
OLX Group	1
Plexus Resource Solutions	1
Catapult	1
Transparency International Secretariat	1
GS-Company	1
Rekode	1

HubSpot	1
Shishi	1
Headmatch	1
White & Case	1
AUTO1	1
mmpo film- und medienproduktion GmbH	1
Native Instruments	1
Productsup	1
Imperva	1
Data Artisans	1
Groupon	1
Heaven Media Ltd	1
CANCOM SE	1
AirHelp	1

Name: company, Length: 194, dtype: int64

2.5 Words

2.5.1 most used word

1. category to look in

"desc"

- 2.

2.6 Printing

2.6.1 quick overview

1. head

```
df.head()
```

	location	related \
0	Berlin	https://de.indeed.com/Python-Developer-Jobs-in...
1	Berlin	NaN
2	Berlin	https://de.indeed.com/Senior-Software-Tester-J...
3	Berlin	https://de.indeed.com/Lead-Product-Analyst-Job...
4	Berlin	https://de.indeed.com/Softwareentwickler-Entwi...


```

title \
0          Python Developer (m/w)
1          Software-Entwickler w/m
2          Senior Software-Tester (w/m)
3          Lead Product Analyst
4 Softwareentwickler (m/w) für Entwicklungsumgeb...

url          company days_ago \
0 https://de.indeed.com/viewjob?jk=05f2b8ca5157f... Bidmanagement 30+
1 https://de.indeed.com/cmp/Qtixx-GmbH/jobs/Soft... Qtixx GmbH 22
2 https://de.indeed.com/viewjob?jk=d9b44d35ab5be... Carmeq GmbH 2
3 https://de.indeed.com/viewjob?jk=9b572e61f1945... eBay Inc. 10
4 https://de.indeed.com/viewjob?jk=c181a1609f4bb... Carmeq GmbH 2

contract          desc
0 NaN <span id="job_summary" class="summary"><div><d...
1 NaN <span id="job_summary" class="summary"><p>Die ...
2 NaN <span id="job_summary" class="summary"><div><d...
3 NaN <span id="job_summary" class="summary"><div><p...
4 NaN <span id="job_summary" class="summary"><div><d...

```

2. count

```
df.title.count()
```

```
146
```

3. titles

```
df.title
```

2.6.2 html pages

1. hacked around solution

TEST

(a) function to save results to html

```

from datetime import datetime
from os import mkdir

def htmllexport(df, begin, end):

```

```

date = str(datetime.now())
path = "../reports/html/" + date + "/"
mkdir(path)
for i in range(begin, end):
    html = ""
    html = html + "\n"
    html = html + "Job number " + str(i)
    html = html + "\n"
    html = html + "-"*100
    html = html + "\n" + df.title.iloc[i]
    html = html + "\n"
    html = html + df.company.iloc[i]
    html = html + "\n"
    html = html + "-"*100
    html = html + "\n"
    html = html + df.desc.iloc[i]
    html = html + "\n"*3
    html = html + "-"*100
    html = html + "\n"*3
    filename = path + "job-" + str(i) + ".html"
    with open(filename, "a") as file:
        file.write(html)

```

(b) call function

```
htmlexport(dfk, 0, dfk.title.count())
```

(c) PB : impossible to add links because of some encoding pb

2. use xml.dom

TEST

(a) use

```

from xml.dom import minidom
minidom.parseString(dfk.desc.iloc[10])

```

(b) PB : some descs are separated by comas

- i. change spider
- ii. use regexp to parse again
- iii. test with proper html files : maybe it is just not working with
html ?

```

from xml.dom import minidom
minidom.parseString("~/code/web/plasma-city/application/static/front.html")

```

3. use yattag

(a) imports

```
from datetime import datetime
from os import mkdir
from yattag import Doc
```

(b) html page generation

i. functions definition

```
def linksgen(filename_base, pagenum, url):
    doc, tag, text = Doc().tagtext()

    with tag('a', href = "."):
        text('Home page ')
    with tag("a", href = filename_base + str(pagenum - 1) + ".html"):
        text("Previous page ")
    with tag("a", href = filename_base + str(pagenum + 1) + ".html"):
        text("Next page ")
    with tag("a", href = url):
        text("Original page ")
    return doc.getvalue()

def pagegen(filename_base, pagenum, title, desc, company, days, url):
    doc, tag, text = Doc().tagtext()

    doc.asis('<meta charset="UTF-8">')
    with tag("title"):
        text(title)
    with tag("body"):
        doc.asis(linksgen(filename_base, pagenum, url))
        with tag("h1"):
            text(title + " - " + company)
        with tag("p"):
            text(str(days))
        with tag("div"):
            doc.asis(desc)
        doc.asis(linksgen(filename_base, pagenum, url))

    return doc.getvalue()
```

```

ii. test pagegen
pagegen("nom", 0, "titre", "desc", "firm", "days", "www")
'<meta charset="UTF-8"><title>titre</title><body><a href=".">Home page </a>
iii. test linksgen
linksgen("file", 10, "wwwwww")
'<a href=".">Home page </a><a href="file9.html">Previous page </a><a href="

```

(c) htmlexport function

i. definition

```

def htmlexport(df, begin, end):
    date = str(datetime.now())
    path = "../reports/html/" + date + "/"
    mkdir(path)
    for i in range(begin, end):
        filename_base = "job-"
        html = pagegen(filename_base,
                        i,
                        df.title.iloc[i],
                        df.desc.iloc[i],
                        df.company.iloc[i],
                        df.days_ago.iloc[i],
                        df.url.iloc[i]
                        )
        filename = path + filename_base + str(i) + ".html"
        with open(filename, "a") as file:
            file.write(html)

```

ii. call

```

htmlexport(df, 0, len(df))

```

2.6.3 server

1. flask ? :D !!!

2.6.4 org table (python)

PYTHON

1. john kitchen example

TEST

```

import pandas as pd
test = pd.DataFrame({'A': [1000, 1000], 'B' : [60, 100]})

```

```

test2 = [list(test)] + [None] + test.values.tolist()
test3 = test.values.tolist()
return (test, test2, test3)

```

2. my program

SLOW

```

import pandas as pd
df = pd.read_csv(data)
return [list(df)] + [None] + df.values.tolist()

```

2.6.5 org results: html

TEST

```
dfk.desc.iloc[0]
```

2.6.6 soupprint

1. session functions

(a) souper (using get text)

```
from bs4 import BeautifulSoup
```

```

def souper(html):
    "returns only the text from a html string"
    soup = BeautifulSoup(html, 'html.parser')
    return soup.get_text()

```

(b) soupprint

i. definition

```
from bs4 import BeautifulSoup
```

```

def souper(html):
    soup = BeautifulSoup(html, 'html.parser')
    print(soup.get_text())

```

```

def soupprint(df, begin, end):
    for i in range(begin, end):
        print(i, df.title.iloc[i])
        print("\n")
        print(df.company.iloc[i])
        print("\n")
        souper(df.desc.iloc[i])

```

```

        print("\n"*3)
        print("-"*100)
        print("\n"*3)
    ii. call
        soupprint(df, 0, 10)

```

2. soupprint as org function

(a) definition

```

from bs4 import BeautifulSoup

def souper(html):
    soup = BeautifulSoup(html, 'html.parser')
    print(soup.get_text())

def soupprint(df, begin, end):
    for i in range(begin, end):
        print(i, df.title.iloc[i])
        print("\n")
        print(df.company.iloc[i])
        print("\n")
        souper(df.desc.iloc[i])
        print("\n"*3)
        print("-"*100)
        print("\n"*3)

```

(b) call

```

soupprint(dfk, 0, dfk.title.count())

```

2.7 Queries

2.7.1 get queries metadata

1. dataframe using os results

```

import os
queries_name = []
queries_size = []
queries_path = []
queries_time = []
for dirpath, dirs, files in os.walk("../data/raw"):

```

```

for filename in files:
    if filename.endswith('.csv'):

        path = os.path.join(dirpath, filename)
        queries_path.append(path)

        size = os.path.getsize(path)
        queries_size.append(size)

        fname = filename.replace(".csv", "")
        queries_name.append(fname)

        time = os.path.getmtime(path)
        queries_time.append(time)

```

```

queries = pd.DataFrame({"name" : queries_name, "path" : queries_path, "size" : qu

```

2. remove oldests results

(a) datetime time format

```

queries["time"] = queries.time.apply(datetime.fromtimestamp)

```

(b) y-m-d format time

```

def format_time(x):
    y = x.strftime("%Y-%m-%d")
    return y

```

```

queries["time_formatted"] = queries.time.apply(format_time)

```

3. remove null size results

```

queries_null = queries[queries["size"] < 1]
queries = queries[queries["size"] > 1]

```

4. number of entries in csv file

(a) read as pandas dataframe

```

def entries_count(csv):
    return len(pd.read_csv(csv))

```

```

queries["entries"] = queries.path.apply(entries_count)

```

5. inspection

```
import humanize
queries["size_for_humans"] = queries["size"].apply(humanize.naturalsize)
queries.sort_values("size", ascending=False)[["name", "size_for_humans", "entries"]]
```

index		name	size_for_humans	entries
0	134	data	3.5 MB	1363
1	133	python	3.4 MB	1291
2	121	intelligence	3.2 MB	1108
3	126	pyjobs	3.2 MB	1260
4	122	python-jobs-berlin-21-05-18	3.2 MB	1260
5	119	software engineer	2.7 MB	774
6	130	aws	2.7 MB	1247
7	127	marketing	2.4 MB	956
8	136	database	2.3 MB	630
9	124	finance	2.3 MB	849
10	114	analyst	2.2 MB	873
11	131	developer	2.2 MB	1077
12	112	business_analyst	1.7 MB	632
13	113	oracle	921.4 kB	417
14	129	internship	789.4 kB	356
15	83	e business	601.2 kB	160
16	70	intelligence	581.0 kB	148
17	94	sql	549.3 kB	158
18	40	business intelligence	546.2 kB	138
19	102	startup	540.5 kB	143
20	51	system	536.3 kB	161
21	178	python	517.0 kB	146
22	74	finance	501.8 kB	156
23	64	engineer	492.5 kB	130
24	110	frontend	487.3 kB	128
25	75	html	459.5 kB	149
26	103	data	449.2 kB	125
27	177	software engineer	442.6 kB	129
28	89	aws	439.1 kB	100
29	141	c	434.8 kB	148
..
117	120	junior_python	21.1 kB	98
118	111	live coding	19.6 kB	5

119	168	camera	18.9 kB	6
120	140	keras	17.5 kB	5
121	22	museum	17.2 kB	7
122	57	system admin	17.1 kB	5
123	88	beginner	17.0 kB	5
124	23	advertisement	16.2 kB	5
125	1	anfänger	12.1 kB	4
126	175	dsp	11.3 kB	3
127	62	growth hacker	11.1 kB	3
128	32	garden	10.9 kB	4
129	27	movie	10.6 kB	3
130	60	clojure	10.2 kB	3
131	38	kunst und medien	9.6 kB	4
132	65	français	7.2 kB	2
133	42	growth hacking	6.9 kB	2
134	16	buchhandel	6.3 kB	2
135	158	assembly language	6.0 kB	1
136	18	digital artist	5.7 kB	3
137	50	audio unit	4.7 kB	2
138	157	bsd	4.5 kB	1
139	67	lean analyst	4.4 kB	1
140	165	punk	3.4 kB	1
141	72	flask	3.1 kB	1
142	142	midi	3.0 kB	3
143	84	webapp	2.6 kB	1
144	41	fin tech	2.5 kB	1
145	143	d3.js	1.7 kB	1
146	17	flowers	900 Bytes	1

[147 rows x 4 columns]

6. time evolution

7. return list for next scraper launch remove null size results before (or not)

```
queries_list = list(set(queries.name))
```

```
['garten',
 'junior_python',
```

'aws',
'vr',
'startup',
'sensors',
'buchhandel',
'python internship',
'ic',
'keras',
'numpy',
'live coding',
'telearbeit',
'lean analytics',
'data',
'github',
'blockchain',
'c++',
'frontend',
'd3.js',
'electronics',
'restaurant',
'growth hacking',
'bio',
'blumen',
'javascript',
'kunst und medien',
'modeling',
'system',
'ruby',
'lean analyst',
'dsp',
'midi',
'docker',
'webapp',
'beginner',
'database',
'bsd',
'pyjobs',
'garden',
'c',
'unix',

'français',
'photography',
'embedded system',
'css',
'html',
'linux',
'küche',
'punk',
'investment',
'pandas',
'fin tech',
'camera',
'junior',
'show',
'movie',
'oracle',
'küchenhilfe',
'sql',
'python-jobs-berlin-21-05-18',
'jenkins',
'remote',
'intelligence',
'e business',
'finance',
'praktikum',
'teilzeit',
'spa',
'cuisine',
'flowers',
'yoga',
'anfänger',
'fintech',
'seo',
'online marketing',
'venture capital',
'3d',
'audio unit',
'advertisement',
'computer art',
'assembly language',

'computer vision',
'museum',
'virtual reality',
'data scientist',
'data visualization',
'certification_analysis',
'sem',
'google trends',
'django',
'real-time',
'scraping',
'french',
'backend',
'python_internship',
'python praktikum',
'facebook ads',
'kunst',
'music',
'digital art',
'zuhause',
'tableau',
'engineer',
'java',
'git',
'part-time',
'marketing',
'business intelligence',
'system admin',
'software engineer',
'clojure',
'developer',
'python',
'mapping',
'digital artist',
'heroku',
'graphql',
'sound',
'analyst',
'art',
'internship',

```
'architect',
'flask',
'growth hacker',
'business_analyst',
'rock']
```

2.7.2 launch scraper with the list

3 Documentation

3.1 doc : look for matching pattern

DOC

```
help(df.title.str.contains)
```

Help on method contains in module pandas.core.strings:

contains(pat, case=True, flags=0, na=nan, regex=True) method of pandas.core.strings.StringMethods
Return boolean Series/'array' whether given pattern/regex is contained in each string in the Series/Index.

Parameters

pat : string

Character sequence or regular expression

case : boolean, default True

If True, case sensitive

flags : int, default 0 (no flags)

re module flags, e.g. re.IGNORECASE

na : default NaN, fill value for missing values.

regex : bool, default True

If True use re.search, otherwise use Python in operator

Returns

contained : Series/array of boolean values

See Also

match : analogous, but stricter, relying on re.match instead of re.search

3.2 pandas

Pandas cheat sheet

4 Tests

4.1 ob-ipython

4.1.1 hands-on tryout

1. hello world

```
print 'hello world'
```

2. function definition

```
def fn():  
    print "I am in the session !"
```

3. function call

```
fn()
```

4.1.2 doc tutorial

1. imports

```
%matplotlib inline  
import matplotlib.pyplot as plt  
import numpy as np
```

2. ex2

```
def foo(x):  
    return x + 9  
  
[foo(x) + 7 for x in range(7)]
```

3. images

- (a) ex1

```
plt.hist(np.random.randn(20000), bins=200)
```

(b) ex2

```
plt.hist(np.random.randn(20000), bins=200)
```

(c) config

```
%config InlineBackend.figure_format = 'svg'
```

4. other kernel

(+ 1 2)

5. async

```
import time
time.sleep(3)
plt.hist(np.random.randn(20000), bins=200)
```

4.1.3 other tryouts

1. functions

(a) call

```
lol()
```

2. formater

(a) init

```
import IPython
from tabulate import tabulate

class OrgFormatter(IPython.core.formatters.BaseFormatter):
    def __call__(self, obj):
        try:
            return tabulate(obj, headers='keys',
                             tablefmt='orgtbl', showindex='always')
        except:
            return None
```

```
ip = get_ipython()
ip.display_formatter.formatters['text/org'] = OrgFormatter()
```

(b) arrays

3. kernel tests

- (a) session header arg after run console

```
print("hello")
```

- (b) kernel headerarg

```
print("hello")
```

4.2 nltk

4.2.1 text selection

1. sample text base

```
from nltk.book import *
```

2. access text as string

- (a) imports

```
import nltk, re, pprint
from nltk import word_tokenize
```

- (b) with one description

- i. definition

```
string = df.iloc[0].desc
```

- ii. formating

- A. html

```
string = souper(string)
```

- B. case

```
string = string.lower()
```

- C. punctiations

- D. definition

```
def multi_replace(string, *args, replace=" "):
    for target in args:
        string = string.replace(target, replace)
    return string
```

```
trash_car = (" ", "\'", "\"", "&", "#", "{", "}",
             "(", ")", "[", "]", "_", "\\ ", "~", "-",
```



```

",", ";", ":", ".", "?", "!", "+", "|",
"@", "/", "_", "*", "\"", ",", "%", " ",
"€")

```

E. call

```
string = multi_replace(string, *trash_car)
```

(c) to nltk text object

i. tokenizing

```
tokens = word_tokenize(string)
```

ii. use as nltk text

```
text = nltk.Text(tokens)
```

4.2.2 search

1. concordance

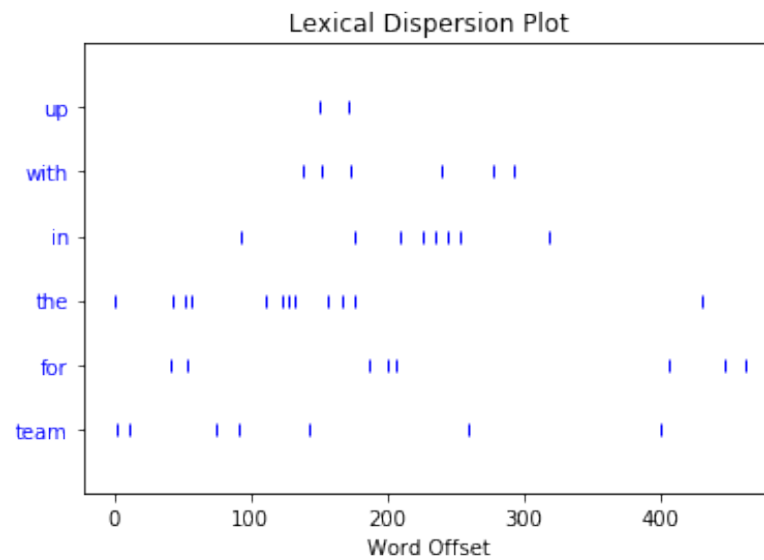
```
text.concordance("data")
```

2. similar word

```
text.similar("analyst")
```

3. dispersion

```
text.dispersion_plot(["up", "with", "in", "the", "for", "team"])
```



4.2.3 generation

TEST

```
text.generate(["The", "job", "is", "for", "data", "team"])
```

4.2.4 normalizing

1. steaming
2. lemmatization

4.2.5 vocabulary

1. sorted set

```
sorted(set(text))
```

```
['17',
 '2008',
 '23',
 '3',
 '40',
 '5',
 '<',
 'a',
```

'able',
'about',
'academic',
'across',
'active',
'adapt',
'additional',
'advertising',
'all',
'also',
'an',
'analysis',
'analysts',
'analytical',
'analytics',
'analyzing',
'and',
'anja',
'are',
'area',
'art',
'as',
'aspects',
'assistance',
'at',
'atmosphere',
'attitude',
'available',
'backgrounds',
'basis',
'behavior',
'beverages',
'bieten',
'brands',
'bringing',
'building',
'business',
'but',
'celebrate',
'centrally',

'challenges',
'change',
'changing',
'choose',
'closely',
'coaching',
'com',
'come',
'commerce',
'committed',
'communicating',
'comparable',
'competitive',
'computer',
'conflicts',
'connecting',
'context',
'contribute',
'crm',
'customer',
'customers',
'daily',
'data',
'databases',
'de',
'decided',
'decisions',
'deep',
'department',
'develop',
'developing',
'development',
'different',
'digital',
'direct',
'discount',
'discounts',
'diverse',
'diversity',
'drive',

'e',
'easily',
'economics',
'effectively',
'ein',
'einkaufserlebnis',
'employee',
'employment',
'empowerment',
'enable',
'entire',
'environment',
'equipment',
'equivalent',
'europas',
'europe',
'experience',
'expertise',
'experts',
'external',
'fashion',
'fast',
'feedback',
'field',
'flexible',
'focus',
'for',
'foundation',
'free',
'from',
'fruits',
'full',
'führende',
'g',
'getting',
'great',
'groups',
'have',
'head',
'health',

'help',
'holidays',
'https',
'ideally',
'impact',
'in',
'independently',
'information',
'innovations',
'insights',
'inspiring',
'intelligence',
'interests',
'internal',
'international',
'internationals',
'into',
'is',
'ist',
'it',
'its',
'junior',
'keep',
'knowledge',
'kunden',
'languages',
'lay',
'lead',
'leading',
'located',
'logistics',
'long',
'look',
'looking',
'lounge',
'mail',
'managing',
'many',
'markets',
'mathematics',

'means',
'members',
'mentoring',
'merit',
'methods',
'million',
'models',
'more',
'most',
'municipality',
'name',
'need',
'needed',
'new',
'not',
'of',
'off',
'offering',
'offerings',
'offices',
'on',
'online',
'only',
'opensource',
'opportunities',
'or',
'other',
'our',
'out',
'paced',
'partners',
'perks',
'personal',
'perspectives',
'platform',
'plattform',
'positive',
'potential',
'priorities',
'proactive',

'public',
'python',
'qualifications',
'questions',
'quick',
'r',
'radar',
're',
'recruiter',
'related',
'relevant',
'relocation',
'reports',
'represent',
'research',
'responsibility',
'run',
's',
'salary',
'science',
'segment',
'seit',
'senior',
'services',
'sets',
'share',
'shop',
'shopping',
'showcases',
'similar',
'size',
'skill',
'skills',
'solutions',
'solve',
'spierling',
'sports',
'sql',
'stakeholders',
'state',

'statistical',
'statistics',
'strategy',
'structures',
'subject',
'tailored',
'team',
'teams',
'tech',
'technical',
'term',
'than',
'that',
'the',
'their',
'them',
'things',
'thinking',
'through',
'time',
'times',
'to',
'toe',
'together',
'too',
'top',
'transforming',
'transport',
'trust',
'umfassendes',
'understanding',
'unseren',
'up',
'use',
'user',
'using',
'valuable',
'variety',
'volunteering',
'warehouse',

```
'ways',  
'we',  
'well',  
'what',  
'where',  
'which',  
'will',  
'wir',  
'with',  
'work',  
'working',  
'workplace',  
'x',  
'years',  
'you',  
'your',  
'zalando']
```

2. lexical richness

(a) tryout

```
len(text) / len(set(text))
```

(b) function

```
def lexical_diversity(text):  
    return len(text) / len(set(text))
```

```
1.6950819672131148
```

3. specific word

(a) tryout

```
100 * text.count('for') / len(text)
```

```
1.5473887814313345
```

(b) function

```
def word_percentage(word):  
    return 100 * text.count(word) / len(text)
```

4.2.6 TODO Build a corpus !

1. sklearn

```
docs = df['desc']
```

```
tfs = tfidf.fit_transform(docs)
```