```
soup =
BeautifulSoup(open('Kandidatenlijsten_TK2017_Amsterdam.eml.xml').rea
d())
parties= soup('affiliation')
partynames= [p.text  for p in soup.findAll('registeredname')]
genderlist= {p.find('registeredname').text : [g.text for g in
p.findAll('gender')] for p in parties}

AantalVrouwenPerPartijPerAantalGewonnenZetels={p:{N:
Counter(genderlist[p][:N])['female']  for N in range(50)}
                                                    for p in genderlist}
df1 =
pd.DataFrame.from_dict(AantalVrouwenPerPartijPerAantalGewonnenZete
ls, orient='index')
df1[49].plot(kind='barh')
#PVV= AantalVrouwenPerPartijPerAantalGewonnenZetels['PVV (Partij
voor de Vrijheid)']
for p in genderlist:
   titel= p+ ': aantal vrouwen per aantal gewonnen zetels'

pd.DataFrame.from_dict(AantalVrouwenPerPartijPerAantalGewonnenZete
ls[p],
                orient='index').plot(kind='bar', title=titel);

# translate the partynames so that we can join
translate={u'50PLUS':u'50PLUS',
      u'CDA':u'CDA',
      u'CU':u'ChristenUnie',
      u'D66':u'Democraten 66 (D66)',
      u'Denk': u'DENK',
      u'FvD':u'Forum voor Democratie',
      u'GL': u'GROENLINKS',
      u'PVV':u'PVV (Partij voor de Vrijheid)',
    u'PvdA':u'Partij van de Arbeid (P.v.d.A.)',
      u'PvdD':u'Partij voor de Dieren',
      u'SGP':u'Staatkundig Gereformeerde Partij (SGP)',
      u'SP':u'SP (Socialistische Partij)',
      u'VNL':u'VNL (VoorNederland)',
      u'VVD':u'VVD'}

# Voor de eerste N mensen op de lijst zet N =80 (is maximum) voor de hele
lijst
N=50
gendercount={p:Counter(genderlist[p][:N]) for p in genderlist}
#print gendercount
gdf=pd.DataFrame.from_dict(gendercount,
orient='index').fillna(0).astype(int)
#min_lijst_lengte= (gdf.sum(axis=1)).min()
#print min_lijst_lengte
# Voeg de ratio toe
gdf['Fratio']=  ((gdf.female/ gdf.sum(axis=1))*100).round()
gdf.sort_values('Fratio',ascending=False)
ef=pd.DataFrame(translate.values())
ef.index=ef[0]
print "Aantal vrouwen in  de top (met max 50) van elke lijst"
ef.join(gdf ).sort_values('Fratio' )[['male','female','Fratio']]

pvda= genderlist['Partij van de Arbeid (P.v.d.A.)']
sum([ (1/np.log2(2+i)) for i in range(len(pvda)) if pvda[i]=='female'])

# Peilingwijzer
laatstepeiling=Cijfers_Peilingwijzer.xlsx'
allepeilingen= Results_Longitudinal.xlsx'
laatstedf= pd.read_excel(laatstepeiling)
alledf= pd.read_excel(allepeilingen)
print alledf.shape
alledf.tail()
laatstedf.index= laatstedf.Partij
laatstedf.rename(translate, inplace=True)

# Dit is een handige file met de zetel aantallen
alleZdf=pd.read_csv('https://d1bjgq97if6urz.cloudfront.net/Public/Peiling
wijzer/Last/Results_DyGraphs_Seats.csv',index_col='Date')
print alleZdf.shape
print (alleZdf.sum(axis=1) == 150).mean()  # check if all add to 150
alleZdf =alleZdf.applymap(lambda s: int(s.split(';')[1])) # want (ci low;
mean; ci high)
# now translate to our names
alleZdf=alleZdf.T
alleZdf.rename(translate, inplace=True)
alleZdf=alleZdf.T
alleZdf.tail()

alleZdf.rename(columns = {'PP':'Piratenpartij'}, inplace=True)
for p in alleZdf.columns:
   alleZdf[p]= alleZdf[p].apply(lambda z:
AantalVrouwenPerPartijPerAantalGewonnenZetels[p][z])

fem= alleZdf.plot(figsize=(18,7),
            title='Aantal vrouwen in de Tweede Kamer door de tijd, per
partij.\n Gebaseerd op de Peilingwijzer.');

som= alleZdf.T.sum()
print  som.describe()
fem= alleZdf.T.sum().plot(figsize=(18,7),
            title='Aantal  vrouwen in de Tweede Kamer door de tijd.\n
Gebaseerd op de Peilingwijzer.');

df=laatstedf.join(gdf)
df
# Turn percentages into number of zetels
def Perc2Zetels(p):
   return round(p * (150/100))
def Ratio2Zetels(p):
   return round(p * 150)
print df.Percentage.apply(Perc2Zetels).sum()
ef= gdf.join(alledf.T.rename(translate)).dropna().drop([     u'male',
u'female',    u'Fratio'], axis =1)#.apply(Perc2Zetels).head()
ZetelsDoorDeTijd= ef.applymap(Ratio2Zetels )
#ZetelsDoorDeTijd.columns= pd.to_datetime(ZetelsDoorDeTijd.columns)
ZetelsDoorDeTijd.head()

ef= ZetelsDoorDeTijd.T
for p in ef.columns:
   ef[p]= ef[p].apply(lambda z:
AantalVrouwenPerPartijPerAantalGewonnenZetels[p][z])
fem= (ef.T.sum()/ ZetelsDoorDeTijd.sum()).plot(figsize=(18,7),
                        title='Percentage vrouwen in de Tweede Kamer
door de tijd.\n Gebaseerd op de Peilingwijzer.');

len([1 for mp in  genderlist['VVD'][:df.ZetelsLaag['VVD']]  if mp=='female' ])

df.rename(index = {'PP':'Piratenpartij'}, inplace=True)
def aantalvrouwen(p,column,df):
   zetels = df[column][p]
   aantalvrouwen= len([1 for mp in  genderlist[p][:zetels ]  if mp=='female'
])
   return aantalvrouwen
   for c in  [c for c in df.columns if c.startswith('Z')]:
   Fc= 'Fem'+c
   # compute aantal vrouwen
   Fem=pd.DataFrame.from_dict({p:aantalvrouwen(p,c,df) for p in
df.index},orient='index')
   Fem.columns=[Fc]
   # voeg toe aan df
   df=df.join(Fem)
   # bereken de ratio
   df['FratioPeiling'+c]= (df[Fc] /df[c]).fillna(0)

# Aantal vrouwen in de kamer
VrouwenPeilingwijzer=df.sort_values('Zetels',ascending=False)[[
u'Zetels',u'FemZetels',u'FratioPeilingZetels']]
print "Op basis van deze peiling komen er %s vrouwen in de kamer. Dat is
%s procent" % (VrouwenPeilingwijzer.FemZetels.sum(),
```

MAX AANTAL VROUWEN P. PARTIJ

AANTAL M/V P. PARTIJ

Hoeveel vrouwen bij ZetelsLaag CDA

df.iloc[0,:] voor rij, df[col] voor kolom, df.sort_values(col1) of ([col1],[col2])

DF beperken tot selectie peilingen = pd.read_excel('') peilingen2 = peilingen[peilingen.Partij == 'VVD']

kolom toevoegen peilingen2['Testen'] = peilingen2.ZetelsLaag + peilingen2.ZetelsHoog

Round(VrouwenPeilingwijzer.FemZetels.sum()/VrouwenPeilingwijzer.Zetels.sum()*100))

print VrouwenPeilingwijzer

df_peilingwijzer.sort_values(by='Zetels', ascending=False).plot('Partij','Zetels', kind='BAR', title='verdeling zetels peilingwijzer')

```
import pandas as pd, import numpy as np, import seaborn as sn, %matplotlib inline, from lxml import etree
from collections import Counter, import matplotlib.pyplot as plt, from zipfile import ZipFile
from urllib import urlretrieve, from tempfile import mktemp
#Download Data and open zip
filename = 'Kandidatenlijsten.zip'
theurl =
'https://www.kiesraad.nl/binaries/kiesraad/documenten/rapporten/2017/2/definitieve-kandidatenlijsten-tk-2017/definiti
eve-kandidatenlijsten-tk-2017/Kandidatenlijsten_EML_TK2017-20170213.zip'
name, hdrs = urlretrieve(theurl, filename)
thefile=ZipFile(filename)
thefile.extractall()
thefile.close()
df_peilingwijzer =
pd.read_excel('https://d1bjgq97if6urz.cloudfront.net/Public/Peilingwijzer/Last/Cijfers_Peilingwijzer.xlsx')
xml_data = 'Kandidatenlijsten_EML_TK2017-20170213/Kandidatenlijsten_TK2017_Amsterdam.eml.xml'
#Parse XML
def xml2df(xml_data):
    s1 = '{urn:oasis:names:tc:evs:schema:eml}'
    s2 = '{urn:oasis:names:tc:ciq:xsdschema:xNL:2.0}'
    s3 = '{urn:oasis:names:tc:ciq:xsdschema:xAL:2.0}'
    columns = ['partij', 'nummer', 'naam kandidaat', 'voorletters', 'voornaam', 'gender', 'woonplaats']
    L = []
    doc = etree.parse(xml_data)
    root = doc.getroot().find(s1 + 'CandidateList/'+s1+'Election/'+s1+'Contest')
    for child in root.iter(s1+'Affiliation'):
        partij = child.find(s1+'AffiliationIdentifier/'+s1+'RegisteredName').text
        for child_2 in child:
            for candidate in child_2.iter(s1+'Candidate'):
                ID = candidate.find(s1+'CandidateIdentifier').get('Id')
                nameinfo = candidate.find(s1+'CandidateFullName/'+s2+'PersonName')
                initial = nameinfo.find(s2+'NameLine').text
                firstname = nameinfo.find(s2+'FirstName').text
                try:
                    prefix = nameinfo.find(s2+'NamePrefix').text + ' '
                except:
                    prefix = ''
                lastname = prefix + nameinfo.find(s2+'LastName').text
                gender = candidate.find(s1+'Gender').text[0]
                try:
                    living = candidate.find(s1+'QualifyingAddress/'+s3+'Locality/'+s3+'LocalityName').text
                except:
                    country = candidate.find(s1+'QualifyingAddress/'+s3+'Country/'+s3+'CountryNameCode').text
                    living = candidate.find(s1+'QualifyingAddress/'+s3+'Country/'+ s3+
'Locality/'+s3+'LocalityName').text+'('+country+')'
                L.append([partij, ID, lastname, initial, firstname, gender, living])
    df_kandidatenlijst = pd.DataFrame(L)
    df_kandidatenlijst.columns = columns
    return df_kandidatenlijst

df_kandidatenlijst_adam.partij.replace(dic, inplace=True)
df_join = df_kandidatenlijst_adam.join(df_peilingwijzer.set_index('Partij'), on = 'partij')
df_join.update(df_join[['Percentage', u'PercentageLaag', u'PercentageHoog', u'Zetels',  u'ZetelsLaag',
u'ZetelsHoog']].fillna(0))
namesamb.loc[namesamb.F > namesamb.M, 'winning gender'] = 'F'

df_peilingwijzer.sort_values(by='Zetels', ascending = False).plot('Partij', 'Zetels', kind = 'bar', title =
'Verdeling Zetels volgens Peilingwijzer')

df=df_kandidatenlijst_adam.pivot_table(index=["partij","gender"],aggfunc='count').dropna()
df=df.drop(['naam kandidaat','voorletters','voornaam','woonplaats'],axis=1).rename(columns={'nummer':'aantal'})
df.unstack('gender').fillna(0).plot(kind='barh',title= 'verhouding man/vrouw' ,stacked = True, figsize=(20,10))

cross_gender_woonplaats = pd.crosstab(df_kandidatenlijst_adam.woonplaats, df_kandidatenlijst_adam.gender, margins =
True)

for f in os.listdir('Directory')

df_peilingwijzer.shape() df_peilingwijzer.describe() alist.abs() alist.sorted()

for partij in df_peilingwijzer.Partij:
    frame = df_peilingwijzer_long[['Datum', partij, partij+'.low', partij+'.high']] #df_peilingwijzer_long['Datum'] =
df_peilingwijzer_long.index
    frame.columns = ['Datum', 'zetels', 'zetels_laag', 'zetels_hoog']
    frame['Partij'] = partij # Add the year column with constant value
    pieces.append(frame)
df_peilingwijzer_long_nieuw = pd.concat(pieces, ignore_index=True)
```

Spiekbrief Datascience hertentamen

```
import pandas as pd
import os
from lxml import etree
import xml.etree.ElementTree as ET
import seaborn as sns
%matplotlib inline
import pandas as pd
import re
import numpy as np
from lxml import etree
#from bz2file import BZ2File
import codecs
import nltk
from collections import defaultdict
from  itertools import combinations
# ideal for creating all possible
pairs that out can make out of a set
from __future__ import division
from math import sqrt
import collections
import matplotlib.pyplot as plt
import os
from bs4 import BeautifulSoup
import requests
import matplotlib.pyplot as plt
```

Bestanden openen:

```
pd.read_excel('peilingwijzer.xlsx')
```

Select:

```
df[df['column'] == 'value']
```

Sorteren:

```
df.sort_values('column')
value.counts() -> bij een kolom
aantal waarden tellen
```

loc: gebruik .loc[(voorwaarden) & / | (voorwaarden)] vb:

```
df.loc[(df['partij'] == 'VVD')]) en
df.loc[(df['partij'] == 'VVD') &
(df['geslacht'] == 'man')]
```

Print out the top 20 most ambiguous names. Take those names with a "log-value" between −0.1–0.1 and 0.10.1 and sort them reversely on the total number of babies with that name.

```
ambiguous = ef[(ef.logratio < 0.1) &
(ef.logratio > -
0.1)].sort_values('All',
ascending=False)
ambiguous.head(20)
OF
names = total_baby.query('log_2
<= 0.1 & -0.1 <= log_2')[:20]
```

maten + wiskundige dingen df = .shape & .describe

Unieke tags xml:

```
tags = []
for event, elem in -
ET.iterparse('Kandidatenlijsten_TK2
017_Amsterdam.eml.xml'):
   tag = elem.tag
   tags.append(tag)
   elem.clear()
set(tags) #als je set weghaalt heb
je alle tags uit het file
```

hoeveel mannen/vrouwen in alle files:

```
path =
'Kandidatenlijsten_EML_TK2017-20
170213/'
for _file in os.listdir(path):
   if not -
_file.endswith('xml'):continue
  malecount = 0
   femalecount = 0
   with open(path+_file): -
      for event, elem in
ET.iterparse(path +_file):
         tag = elem.tag
         value = elem.text
         if tag ==
"{urn:oasis:names:tc:evs:schema:e
ml}Gender" :
            if value == 'male':
               malecount += 1
               totalmalecount += 1
            else:
               femalecount += 1
         elem.clear() # discard the
element
   print _file.strip('.eml.xml'),
malecount, femalecount,
(malecount+femalecount)
```

Dataframe van XML:

```
partij = []
voornaam = []
initialen = []
achternaam = []
gender = []
woonplaats = []
nummer = []

for event, elem in
ET.iterparse('Kandidatenlijsten_TK2
017_Amsterdam.eml.xml'):
   tag = elem.tag
   value = elem.text
   if tag ==
"{urn:oasis:names:tc:evs:schema:e
ml}RegisteredName":
      partij_val = value
   if tag ==
'{urn:oasis:names:tc:ciq:xsdschem
a:xNL:2.0}NameLine':
      initialen.append(value)
   if tag ==
"{urn:oasis:names:tc:ciq:xsdschem
a:xNL.2.0}FirstName":
      partij.append(partij_val)
      voornaam.append(value)
   if tag ==
"{urn:oasis:names:tc:ciq:xsdschem
a:xNL:2.0}LastName":
      achternaam.append(value)
   if tag ==
"{urn:oasis:names:tc:evs:schema:e
ml}Gender":
      if value == 'male':
         gender_val = 'man'
      else:
         gender_val = 'vrouw'
      gender.append(gender_val)
   if tag ==
'{urn:oasis:names:tc:ciq:xsdschem
a:xAL:2.0}LocalityName':
      woonplaats.append(value)

df_amsterdam =
pd.DataFrame({'partij':partij,
'voornaam':voornaam,
'initialen':initialen,
'achternaam':achternaam,
'geslacht':gender,
'woonplaats':woonplaats})
df_amsterdam =
df_amsterdam[['partij', 'voornaam',
'initialen', 'achternaam', 'geslacht',
'woonplaats']]
```

stel dat de PVV 21 zetels haalt, hoeveel vrouwelijke PVVers komen er dan in de Kamer? .

```
#ONELINER
len(df.loc[(df['partij'] == 'PVV (Partij
voor de
Vrijheid)')].head(peilingwijzer['Zetel
s'].loc[(peilingwijzer['Partij'] ==
'PVV')].values.item()).loc[(df['geslac
ht'] == 'vrouw')])
#MAAK EEN DATAFRAME MET X
AANTAL NAMEN VAN DE
KANDIDATENLIJST VAN EEN
SPECIFIEKE PARTIJ
df_pvv = df.loc[(df['partij'] == 'PVV
(Partij voor de
Vrijheid)')].head(peilingwijzer['Zetel
s'].loc[(peilingwijzer['Partij'] ==
'PVV')].values.item())

#VIND ALLE KANDIDATEN DIE IN.
df_woonplaats =
df['woonplaats'].loc[df['woonplaats'
].isin(['Breda', 'Tilburg',
'Eindhoven'])]
```

Random

```
crosstab (wat , tegenover wat,
margins = true) = kruistabel
axis = is kolommen
#PLOT HET DATAFRAME IN EEN
STACKED BARPLOT
ax = manvrouwdf.plot(x='partij',
y='mannen', kind = 'bar')
manvrouwdf.plot(x='partij',
y='vrouwen', kind = 'bar',
color='red', ax=ax)
```

```
# GENERAL ########################################################
# Is gelijk aan: ==      is niet gelijk aan: !=
round(2.05467, 2) # rond een getal af op 2 decimalen
float(2) # maak kommagetal, dus 2.00 ipv 2
list.append('item 4') # add item to list
list[:2] # selection of list
print 'Henk is %s jaar oud' % str(leeftijd)
if 'Henk' in mensen: # check of Henk voorkomt in mensen

# CREATE DATAFRAME ###############################################
# van string
df = pd.DataFrame(data, index=[1,2,3,4],
columns=['date','b','c','d'])

# van file (read_excel/table/csv etc.)
df = pd.read_excel('http://url.nl/file.xls')
df = pd.read_csv('./output.csv',
    sep=',', # of bijvoorbeeld \t voor tab
    skiprows=1, # sla de eerste rij van file over
    header=None, # neem niet header van file over
    names=['name','F','M'] # voeg kolomnamen toe
)

# SUMMARIZE DATAFRAME ############################################
df.head(10) # of df.tail(10)
df.describe() # algemene info over df
df.shape # aantal kolommen en aantal rijen
len(df) # number of rows in df
len(df.clmn_name.unique()) # no distinct values

df.clmn_name.value_counts() # count number of rows for each
unique value (voor een specifieke kolom)
df.apply(pd.value_counts) # voer de functie van hierboven uit
op alle kolommen van de df

# DOING STUFF WITH DATAFRAME #####################################
df.sort_values('clmn_name', ascending=False) # sorteer rijen
df.sort_index() # sort index of a df
df.reset_index() # reset index of df to row numbers, moving
the existing index to a column
new_df = df.set_index('date') # set column as index
df.columns=['name', 'F', 'M'] # set column names
df.groupby(by='clmn_name') # return groupby object, grouped
by values in column
df.clmn_name = df.clmn_name.astype(int) # change datatype of
values in a column
df.transpose() # draai df kwartslag/maak columns van rijen
df.clmn_name.min() df.clmn_name.max() # min of max van column
df.sum(axis=1) # som van waarden (kan ook axis=0)
df.count()

# DELETE DATA ####################################################
df = df.drop(['Length','Height'], axis=1) # delete columns
df.drop_duplicates() # remove duplicate rows
```

```
# STUFF WITH MISSING DATA ########################################
df = df.dropna() # drop rows with any column having NA/null
df = df.dropna(axis=1) # drop columns with any column having
NA/null
df.dropna(subset=['column_name']) # drop rows with NA/null in
specific column
df = df.fillna(value) # replace all NA/null with value

# SELECTING DATA FROM DATAFRAME ##################################
# select rows
df[(df.c >= 3) & (df.b < 5)] # extract rows with condition
df.loc[df.clmn_name == 'John'] # extract rows with condition
df[1:3] # select rows based on position
df.loc[df['a'] > 10, ['a','c']] # select rows with condition
and only in specified columns

# select columns
df['width'] # select single column
df[['width','length','species']] # select multiple columns
df.loc[:,'width':'length'] # select all columns between those
df.iloc[:,[1,2,5]] # select columns in positions 1, 2 and 5

df.columns # get the column names (df.column[12:50] voor een
selectie van een deel van de kolommen)
list(df.columns.values)

# CREATE A NEW COLUMN ############################################
df['Ratio'] = df.M + df.F

# MAKING PLOTS ###################################################
df.plot(kind=kind) # create plot (kind: bar, barh, etc.)
# HISTOGRAM - Maak een "horizontaal histogram" waarin je voor
elke partij aangeeft hoeveel van dit soort stukken die partij
heeft ingediend.
df.clmn_name.value_counts().sort_values(ascending=True).plot(
'barh')
# SCATTER PLOT
df.plot.scatter(x='column 1',y='column 2', title='')
# KRUISTABEL
pd.crosstab(df.proposaltype, df.votetype, margins=True)

# OPDRACHTEN #####################################################
# Maak een nieuwe df (im) van de selectie waarbij de partij
van indiener (kolom: authorparty) gelijk is aan die van
medeindiener(s) (kolom: supporterparties).
im = rutte2.loc[(rutte2.authorparty ==
rutte2.supporterparties) & (rutte2.authorparty != '[]')]

# Hoe vaak heeft Barry Madlener voor gestemd bij hoofdelijke
stemmingen? Kijk hiervoor in de pro kolom.
len([item[1] for item in df.pro.iteritems() if 'Barry
Madlener' in item[1]])

# Check of je resultaat gelijk is aan wat het moet zijn
print (string == 100) # dit print True of False
```

```
# PRETTIFY SOUP ##################################################
print soup.prettify()

# XML ############################################################
import urllib
import zipfile
import os
from collections import defaultdict
from bs4 import BeautifulSoup

urllib.urlretrieve
("https://data.openstate.eu/dataset/85aaef1d-9084-4d3f-a6d1-1
1566a9538b6/resource/dd639f9f-4035-4e17-b09b-33fc388e9780/dow
nload/kandidatenlijstenemltk2017-20170213.zip", "files.zip")
with zipfile.ZipFile("files.zip", "r") as zipf:
    zipf.extractall("")

directory = "Kandidatenlijsten_EML_TK2017-20170213/"
candidates_list = []

# loop through all the xml files in the directory
for filename in os.listdir(directory):
    if filename.endswith(".xml"):
        f = open(directory + filename, "r")
        soup = BeautifulSoup(f.read(), 'lxml')

        kieskring = soup.find('contestname').contents[0]

        # loop through all affiliations (partijen)
        for affiliation in soup.findAll('affiliation'):
            partij = affiliation.find('registeredname').contents[0]

          # loop through all candidates
          for c in affiliation.findAll('candidate'):
              c_id = c.find('candidateidentifier').attrs['id']
              c_initials = c.find('ns3:nameline').contents[0]
              c_wp = c.find('ns2:localityname').contents[0]

              try: c_lastname=c.find('ns3:nameprefix').contents[0] +
c.find('ns3:lastname').contents[0]
               except: c_lastname=c.find('ns3:lastname').contents[0]

              c_firstname = c.find('ns3:firstname')
              try: c_firstname = c_firstname.contents[0]
              except: print 'No firstname'

              c_gender = c.find('gender')
              try: c_gender = c_gender.contents[0]
              except: print 'No gender'

              candidates_list.append([kieskring, partij, c_id,
c_lastname, c_initials, c_firstname, c_gender, c_wp])

        f.close()

df = pd.DataFrame(candidates_list, columns=['kieskring',
'partij','nummer','achternaam','voorletters','voornaam','gend
er','woonplaats'])
```

Peilingwijzer namen gelijk zetten

```
peilingwijzer.Partij[2] = 'Partij van de Arbeid (P.v.d.A.)'
peilingwijzer.Partij[3] = 'PVV (Partij voor de Vrijheid)'
peilingwijzer.Partij[4] = 'SP (Socialistische Partij)'
```

XML lezen

```
from lxml import etree
import xml.etree.ElementTree as ET
tree = ET.parse(ams)
root = tree.getroot()
data = []

for item in tree.iter():
    if item.tag == '{urn:oasis:names:tc:evs:schema:eml}Affiliation':
        partij = item.getchildren()[0][0].text
        for candidate in item.findall("{urn:oasis:names:tc:evs:schema:eml}Candidate"):
            nummer = candidate.getchildren()[0].attrib["Id"]
            voornaam = candidate.getchildren()[1][0][1].text
            achternaam = candidate.getchildren()[1][0][2].text
            initialen = candidate.getchildren()[1][0][0].text
            gender = candidate.getchildren()[2].text
            place = candidate.getchildren()[3][0][0].text

            data.append({"initialen":initialen .....)
kandidaten = pd.DataFrame(data)
kandidaten.geslacht.replace({"male": "m", "female": "v"}, inplace=True)
```

Dataframes samenvoegen

```
genders = kandidaten.groupby("Partij")["geslacht"].sum().to_frame().reset_index()
compleet = peilingwijzer.merge(genders, on="Partij")
```

Min/Max/Aantal vrouwen/mannen

```
for item in compleet.iterrows():
vrouwenlaag.append(item[1]["geslacht"][:item[1]["ZetelsLaag"]].count("v"))
compleet["vrouwenlaag"] = vrouwenlaag etc…
```

Pivot table kandidatenlijst

```
pivot = kandidaten.pivot_table(index=["partij","gender"], aggfunc="count",values="woonplaats")
df1 = pd.DataFrame(pivot.unstack("gender"))
```

PLOT: `df1.plot(kind="barh",figsize=(20,15))`

RATIO + TOTAAL

```
df1["total"] =(df1["m"]+df1["v"])
df1["ratio"] =df1["m"]/(df1["m"]+df1["v"])
```

Groupby

```
g = amsterdam.groupby(["partij", "gender"])
g.size()
```

Meerdere files

```
for _file in os.listdir(path):
    if not _file.endswith('xml'):continue
       with open(path+_file):
              tree = ET.parse(path+_file)
```