Maheshvaran S

DS205229119

Lab8. Python Regular Expressions

Question1: Using Email Collections file, mbox-short.txt, write a python program for the following queries

1.Search for lines that contain 'From' and print them

In [7]:
```
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if re.search('From:', line):
        print(line)
```

```
From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
From: gsilver@umich.edu
From: gsilver@umich.edu
From: zqian@umich.edu
From: gsilver@umich.edu
From: wagnermr@iupui.edu
From: zqian@umich.edu
From: antranig@caret.cam.ac.uk
From: gopal.ramasammycook@gmail.com
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: louis@media.berkeley.edu
From: ray@media.berkeley.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
```

2.Search for lines that start with 'From' and print them

In [8]:
```
import re
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if re.search('^From:', line):
        print(line)
```

```
From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
From: gsilver@umich.edu
```

```
From: gsilver@umich.edu
From: zqian@umich.edu
From: gsilver@umich.edu
From: wagnermr@iupui.edu
From: zqian@umich.edu
From: antranig@caret.cam.ac.uk
From: gopal.ramasammycook@gmail.com
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: louis@media.berkeley.edu
From: ray@media.berkeley.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
```

3.Search for lines that start with 'F', followed by 2 characters, followed by 'm:'

In [9]:

```
import re
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if re.search('^F..m:', line):
        print(line)
```

```
From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
From: gsilver@umich.edu
From: gsilver@umich.edu
From: zqian@umich.edu
From: gsilver@umich.edu
From: wagnermr@iupui.edu
From: zqian@umich.edu
From: antranig@caret.cam.ac.uk
From: gopal.ramasammycook@gmail.com
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: louis@media.berkeley.edu
From: ray@media.berkeley.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
```

4.Search for lines that start with From and have an at sign and print them

In [10]:

```
import re
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if re.search('^From:.+@', line):
        print(line)
```

```
From: stephen.marquard@uct.ac.za
```

```
From: louis@media.berkeley.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
From: gsilver@umich.edu
From: gsilver@umich.edu
From: zqian@umich.edu
From: gsilver@umich.edu
From: wagnermr@iupui.edu
From: zqian@umich.edu
From: antranig@caret.cam.ac.uk
From: gopal.ramasammycook@gmail.com
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: louis@media.berkeley.edu
From: ray@media.berkeley.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
```

5.Search for lines that have an at sign between characters and print them (Use findall())

In [12]:

```
import re
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    x = re.findall('\S+@\S+', line)
    if len(x) > 0:
        print(x)
```

```
['stephen.marquard@uct.ac.za']
['<postmaster@collab.sakaiproject.org>']
['<200801051412.m05ECIaH010327@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['stephen.marquard@uct.ac.za']
['source@collab.sakaiproject.org']
['stephen.marquard@uct.ac.za']
['stephen.marquard@uct.ac.za']
['louis@media.berkeley.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801042308.m04N8v6O008125@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['louis@media.berkeley.edu']
['source@collab.sakaiproject.org']
['louis@media.berkeley.edu']
['louis@media.berkeley.edu']
['zqian@umich.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801042109.m04L92hb007923@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
```

```
['apache@localhost)']
['source@collab.sakaiproject.org;']
['zqian@umich.edu']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['zqian@umich.edu']
['rjlowe@iupui.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801042044.m04Kiem3007881@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['rjlowe@iupui.edu']
['source@collab.sakaiproject.org']
['rjlowe@iupui.edu']
['rjlowe@iupui.edu']
['zqian@umich.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801042001.m04K1cO0007738@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['zqian@umich.edu']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['zqian@umich.edu']
['zqian@umich.edu']
['rjlowe@iupui.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801041948.m04JmdwO007705@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['rjlowe@iupui.edu']
['source@collab.sakaiproject.org']
['rjlowe@iupui.edu']
['rjlowe@iupui.edu']
['cwen@iupui.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801041635.m04GZQGZ007313@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['cwen@iupui.edu']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['cwen@iupui.edu']
['hu2@iupui.edu']
['cwen@iupui.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801041633.m04GX6eG007292@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['cwen@iupui.edu']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['cwen@iupui.edu']
['hu2@iupui.edu']
```

```
['gsilver@umich.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801041611.m04GB1Lb007221@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['gsilver@umich.edu']
['source@collab.sakaiproject.org']
['gsilver@umich.edu']
['gsilver@umich.edu']
['gsilver@umich.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801041610.m04GA5KP007209@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['gsilver@umich.edu']
['source@collab.sakaiproject.org']
['gsilver@umich.edu']
['gsilver@umich.edu']
['zqian@umich.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801041609.m04G9EuX007197@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['zqian@umich.edu']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['zqian@umich.edu']
['gsilver@umich.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801041608.m04G8d7w007184@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['gsilver@umich.edu']
['source@collab.sakaiproject.org']
['gsilver@umich.edu']
['gsilver@umich.edu']
['wagnermr@iupui.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801041537.m04Fb6Ci007092@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['wagnermr@iupui.edu']
['source@collab.sakaiproject.org']
['wagnermr@iupui.edu']
['wagnermr@iupui.edu']
['zqian@umich.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801041515.m04FFv42007050@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['zqian@umich.edu']
```

```
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['zqian@umich.edu']
['antranig@caret.cam.ac.uk']
['<postmaster@collab.sakaiproject.org>']
['<200801041502.m04F21Jo007031@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['antranig@caret.cam.ac.uk']
['source@collab.sakaiproject.org']
['antranig@caret.cam.ac.uk']
['antranig@caret.cam.ac.uk']
['gopal.ramasammycook@gmail.com']
['<postmaster@collab.sakaiproject.org>']
['<200801041403.m04E3psW006926@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['gopal.ramasammycook@gmail.com']
['source@collab.sakaiproject.org']
['gopal.ramasammycook@gmail.com']
['gopal.ramasammycook@gmail.com']
['david.horwitz@uct.ac.za']
['<postmaster@collab.sakaiproject.org>']
['<200801041200.m04C0gfK006793@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['david.horwitz@uct.ac.za']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['dhorwitz@david-horwitz-6:~/branchManagemnt/sakai_2-5-x>']
['dhorwitz@david-horwitz-6:~/branchManagemnt/sakai_2-5-x>']
['david.horwitz@uct.ac.za']
['<postmaster@collab.sakaiproject.org>']
['<200801041106.m04B6lK3006677@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['david.horwitz@uct.ac.za']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['<postmaster@collab.sakaiproject.org>']
['<200801040947.m049lUxo006517@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['david.horwitz@uct.ac.za']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['josrodri@iupui.edu']
['dhorwitz@david-horwitz-6:~/branchManagemnt/sakai_2-5-x>']
['david.horwitz@uct.ac.za']
```

```
['<postmaster@collab.sakaiproject.org>']
['<200801040932.m049W2i5006493@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['david.horwitz@uct.ac.za']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['josrodri@iupui.edu']
['dhorwitz@david-horwitz-6:~/branchManagemnt/sakai_2-5-x>']
['stephen.marquard@uct.ac.za']
['<postmaster@collab.sakaiproject.org>']
['<200801040905.m0495rWB006420@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['stephen.marquard@uct.ac.za']
['source@collab.sakaiproject.org']
['stephen.marquard@uct.ac.za']
['stephen.marquard@uct.ac.za']
['louis@media.berkeley.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801040023.m040NpCc005473@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['louis@media.berkeley.edu']
['source@collab.sakaiproject.org']
['louis@media.berkeley.edu']
['louis@media.berkeley.edu']
['louis@media.berkeley.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801032216.m03MGhDa005292@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['louis@media.berkeley.edu']
['source@collab.sakaiproject.org']
['louis@media.berkeley.edu']
['louis@media.berkeley.edu']
['ray@media.berkeley.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801032205.m03M5Ea7005273@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['ray@media.berkeley.edu']
['source@collab.sakaiproject.org']
['ray@media.berkeley.edu']
['ray@media.berkeley.edu']
['cwen@iupui.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801032133.m03LX3gG005191@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
```

```
['cwen@iupui.edu']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['cwen@iupui.edu']
['cwen@iupui.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801032127.m03LRUqH005177@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['cwen@iupui.edu']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['cwen@iupui.edu']
['wagnermr@iupui.edu']
['cwen@iupui.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801032122.m03LMFo4005148@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['cwen@iupui.edu']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['cwen@iupui.edu']
['wagnermr@iupui.edu']
```

6.Search for lines that have an at sign between characters. The characters must be a letter or number and print them

In [13]:

```
import re
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    x = re.findall('[a-zA-Z0-9]\S+@\S+[a-zA-Z]', line)
    if len(x) > 0:
        print(x)
```

```
['stephen.marquard@uct.ac.za']
['postmaster@collab.sakaiproject.org']
['200801051412.m05ECIaH010327@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['stephen.marquard@uct.ac.za']
['source@collab.sakaiproject.org']
['stephen.marquard@uct.ac.za']
['stephen.marquard@uct.ac.za']
['louis@media.berkeley.edu']
['postmaster@collab.sakaiproject.org']
['200801042308.m04N8v6O008125@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['louis@media.berkeley.edu']
['source@collab.sakaiproject.org']
['louis@media.berkeley.edu']
['louis@media.berkeley.edu']
['zqian@umich.edu']
['postmaster@collab.sakaiproject.org']
```

```
['200801042109.m04L92hb007923@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['zqian@umich.edu']
['rjlowe@iupui.edu']
['postmaster@collab.sakaiproject.org']
['200801042044.m04Kiem3007881@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['rjlowe@iupui.edu']
['source@collab.sakaiproject.org']
['rjlowe@iupui.edu']
['rjlowe@iupui.edu']
['zqian@umich.edu']
['postmaster@collab.sakaiproject.org']
['200801042001.m04K1cO0007738@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['zqian@umich.edu']
['zqian@umich.edu']
['rjlowe@iupui.edu']
['postmaster@collab.sakaiproject.org']
['200801041948.m04JmdwO007705@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['rjlowe@iupui.edu']
['source@collab.sakaiproject.org']
['rjlowe@iupui.edu']
['rjlowe@iupui.edu']
['cwen@iupui.edu']
['postmaster@collab.sakaiproject.org']
['200801041635.m04GZQGZ007313@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['cwen@iupui.edu']
['hu2@iupui.edu']
['cwen@iupui.edu']
['postmaster@collab.sakaiproject.org']
['200801041633.m04GX6eG007292@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
```

```
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['cwen@iupui.edu']
['hu2@iupui.edu']
['gsilver@umich.edu']
['postmaster@collab.sakaiproject.org']
['200801041611.m04GB1Lb007221@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['gsilver@umich.edu']
['source@collab.sakaiproject.org']
['gsilver@umich.edu']
['gsilver@umich.edu']
['gsilver@umich.edu']
['postmaster@collab.sakaiproject.org']
['200801041610.m04GA5KP007209@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['gsilver@umich.edu']
['source@collab.sakaiproject.org']
['gsilver@umich.edu']
['gsilver@umich.edu']
['zqian@umich.edu']
['postmaster@collab.sakaiproject.org']
['200801041609.m04G9EuX007197@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['zqian@umich.edu']
['gsilver@umich.edu']
['postmaster@collab.sakaiproject.org']
['200801041608.m04G8d7w007184@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['gsilver@umich.edu']
['source@collab.sakaiproject.org']
['gsilver@umich.edu']
['gsilver@umich.edu']
['wagnermr@iupui.edu']
['postmaster@collab.sakaiproject.org']
['200801041537.m04Fb6Ci007092@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['wagnermr@iupui.edu']
['source@collab.sakaiproject.org']
['wagnermr@iupui.edu']
['wagnermr@iupui.edu']
['zqian@umich.edu']
['postmaster@collab.sakaiproject.org']
['200801041515.m04FFv42007050@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
```

```
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['zqian@umich.edu']
['antranig@caret.cam.ac.uk']
['postmaster@collab.sakaiproject.org']
['200801041502.m04F21Jo007031@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['antranig@caret.cam.ac.uk']
['source@collab.sakaiproject.org']
['antranig@caret.cam.ac.uk']
['antranig@caret.cam.ac.uk']
['gopal.ramasammycook@gmail.com']
['postmaster@collab.sakaiproject.org']
['200801041403.m04E3psW006926@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['gopal.ramasammycook@gmail.com']
['source@collab.sakaiproject.org']
['gopal.ramasammycook@gmail.com']
['gopal.ramasammycook@gmail.com']
['david.horwitz@uct.ac.za']
['postmaster@collab.sakaiproject.org']
['200801041200.m04C0gfK006793@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['dhorwitz@david-horwitz-6:~/branchManagemnt/sakai_2-5-x']
['dhorwitz@david-horwitz-6:~/branchManagemnt/sakai_2-5-x']
['david.horwitz@uct.ac.za']
['postmaster@collab.sakaiproject.org']
['200801041106.m04B6lK3006677@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['postmaster@collab.sakaiproject.org']
['200801040947.m049lUxo006517@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
```

```
['david.horwitz@uct.ac.za']
['josrodri@iupui.edu']
['dhorwitz@david-horwitz-6:~/branchManagemnt/sakai_2-5-x']
['david.horwitz@uct.ac.za']
['postmaster@collab.sakaiproject.org']
['200801040932.m049W2i5006493@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['josrodri@iupui.edu']
['dhorwitz@david-horwitz-6:~/branchManagemnt/sakai_2-5-x']
['stephen.marquard@uct.ac.za']
['postmaster@collab.sakaiproject.org']
['200801040905.m0495rWB006420@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['stephen.marquard@uct.ac.za']
['source@collab.sakaiproject.org']
['stephen.marquard@uct.ac.za']
['stephen.marquard@uct.ac.za']
['louis@media.berkeley.edu']
['postmaster@collab.sakaiproject.org']
['200801040023.m040NpCc005473@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['louis@media.berkeley.edu']
['source@collab.sakaiproject.org']
['louis@media.berkeley.edu']
['louis@media.berkeley.edu']
['louis@media.berkeley.edu']
['postmaster@collab.sakaiproject.org']
['200801032216.m03MGhDa005292@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['louis@media.berkeley.edu']
['source@collab.sakaiproject.org']
['louis@media.berkeley.edu']
['louis@media.berkeley.edu']
['ray@media.berkeley.edu']
['postmaster@collab.sakaiproject.org']
['200801032205.m03M5Ea7005273@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['ray@media.berkeley.edu']
['source@collab.sakaiproject.org']
['ray@media.berkeley.edu']
['ray@media.berkeley.edu']
['cwen@iupui.edu']
['postmaster@collab.sakaiproject.org']
['200801032133.m03LX3gG005191@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
```

```
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['cwen@iupui.edu']
['cwen@iupui.edu']
['postmaster@collab.sakaiproject.org']
['200801032127.m03LRUqH005177@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['cwen@iupui.edu']
['wagnermr@iupui.edu']
['cwen@iupui.edu']
['postmaster@collab.sakaiproject.org']
['200801032122.m03LMFo4005148@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['cwen@iupui.edu']
['wagnermr@iupui.edu']
```

7.Search for lines that start with 'X' followed by any non white space characters and ':', followed by a space and any number. The number can include a decimal.

In [14]:

```
import re
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if re.search('^X\S*: [0-9.]+', line):
        print(line)
```

```
X-DSPAM-Confidence: 0.8475
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.6178
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.6961
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7565
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7626
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7556
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7002
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7615
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7601
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7605
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.6959
X-DSPAM-Probability: 0.0000
```

```
X-DSPAM-Confidence: 0.7606
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7559
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7605
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.6932
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7558
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.6526
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.6948
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.6528
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7002
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7554
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.6956
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.6959
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7556
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.9846
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.8509
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.9907
X-DSPAM-Probability: 0.0000
```

8.Search for lines that start with 'X' followed by any non whitespace characters and ':' followed by a space and any number. The number can include a decimal. Then print the number if it is greater than zero.

In [15]:

```
import re
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    x = re.findall('^X\S*: ([0-9.]+)', line)
    if len(x) > 0:
        print(x)
```

```
['0.8475']
['0.0000']
['0.6178']
['0.0000']
['0.6961']
['0.0000']
['0.7565']
['0.0000']
['0.7626']
['0.0000']
['0.7556']
['0.0000']
['0.7002']
['0.0000']
['0.7615']
['0.0000']
['0.7601']
['0.0000']
['0.7605']
['0.0000']
['0.6959']
['0.0000']
['0.7606']
```

```
['0.0000']
['0.7559']
['0.0000']
['0.7605']
['0.0000']
['0.6932']
['0.0000']
['0.7558']
['0.0000']
['0.6526']
['0.0000']
['0.6948']
['0.0000']
['0.6528']
['0.0000']
['0.7002']
['0.0000']
['0.7554']
['0.0000']
['0.6956']
['0.0000']
['0.6959']
['0.0000']
['0.7556']
['0.0000']
['0.9846']
['0.0000']
['0.8509']
['0.0000']
['0.9907']
['0.0000']
```

9.Search for lines that start with 'Details: rev=', followed by numbers and '.'. Then print the number if it is greater than zero

In [16]:

```
import re
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    x = re.findall('^Details:.*rev=([0-9.]+)', line)
    if len(x) > 0:
        print(x)
```

```
['39772']
['39771']
['39770']
['39769']
['39766']
['39765']
['39764']
['39763']
['39762']
['39761']
['39760']
['39759']
['39758']
['39757']
['39756']
['39755']
['39754']
['39753']
['39752']
['39751']
['39750']
['39749']
['39746']
['39745']
['39744']
```

```
['39743']
['39742']
```

10.Search for lines that start with From and a character followed by a two digit number between 00 and 99 followed by ':'. Then print the number if it is greater than zero

In [17]:

```
import re
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    x = re.findall('^From .* ([0-9][0-9]):', line)
    if len(x) > 0: print(x)
```

```
['09']
['18']
['16']
['15']
['15']
['14']
['11']
['11']
['11']
['11']
['11']
['11']
['10']
['10']
['10']
['09']
['07']
['06']
['04']
['04']
['04']
['19']
['17']
['17']
['16']
['16']
['16']
```

Question2. Baby Names Popularity Analysis

Reference: https://developers.google.com/edu/python/exercises/baby-names

The Social Security administration has this neat data by year of what names are most popular for babies born that year in the USA. The files baby1990.html baby1992.html ... contain raw html pages. Take a look at the html and think about how you might scrape the data out of it.

In the babynames.py file, implement the extract_names(filename) function which takes the filename of a baby1990.html file and returns the data from the file as a single list -- the year string at the start of the list followed by the name-rank strings in alphabetical order. ['2006', 'Aaliyah 91', 'Abagail 895', 'Aaron 57', ...].

Modify main() so it calls your extract_names() function and prints what it returns (main already has the code for the command line argument parsing). If you get stuck working out the regular expressions for the year and each name, solution regular expression patterns are shown at the end of this document. Note that for parsing webpages in general, regular expressions don't do a good job, but these webpages have a simple and consistent format.

Rather than treat the boy and girl names separately, we'll just lump them all together. In some years, a name appears more than once in the html, but we'll just use one number per name. Optional: make the algorithm smart about this case and choose whichever number is smaller.

Build the program as a series of small milestones, getting each step to run/print something before trying the next step. This is the pattern used by experienced programmers -- build a series of incremental milestones, each with some output to check, rather than building the whole program in one huge step.

Printing the data you have at the end of one milestone helps you think about how to re-structure that data for the next milestone. Python is well suited to this style of incremental development. For example, first get it to the point where it extracts and prints the year and calls sys.exit(0). Here are some suggested milestones:

- Extract all the text from the file and print it
- Find and extract the year and print it
- Extract the names and rank numbers and print them
- Get the names data into a dict and print it
- Build the [year, 'name rank', ...] list and print it
- Fix main() to use the ExtractNames list

Earlier we have had functions just print to standard out. It's more re-usable to have the function *return* the extracted data, so then the caller has the choice to print it or do something else with it. (You can still print directly from inside your functions for your little experiments during development.)

Have main() call extract_names() for each command line arg and print a text summary. To make the list into a reasonable looking summary text, here's a clever use of join: text = '\n'.join(mylist) + '\n'

The summary text should look like this for each file:

2006

Aaliyah 91

Aaron 57

Abagail 895

Abbey 695

Abbie 650

...

In [1]:

```
import re
import sys

def extract_names(filename):
  """
  Given a file name for baby.html, returns a list starting with the year string
  followed by the name-rank strings in alphabetical order.
  ['2006', 'Aaliyah 91', Aaron 57', 'Abagail 895', ' ...]
  """

  names = []
  f = open(filename, 'r')
```

```
  text = f.read()
  year_match = re.search(r'Popularity\sin\s(\d\d\d\d)', text)
  if not year_match:
    sys.stderr.write('Couldn\'t find the year!\n')
    sys.exit(1)
  year = year_match.group(1)
  names.append(year)

  tuples = re.findall(r'<td>(\d+)</td><td>(\w+)</td>\<td>(\w+)</td>', text)

  names_to_rank =  {}
  for rank_tuple in tuples:
    (rank, boyname, girlname) = rank_tuple
    if boyname not in names_to_rank:
      names_to_rank[boyname] = rank
    if girlname not in names_to_rank:
      names_to_rank[girlname] = rank

  sorted_names = sorted(names_to_rank.keys())

  for name in sorted_names:
    names.append(name + " " + names_to_rank[name])

  return names
```

In [2]:

```
extract_names('baby2006.html')
```

Out[2]:

```
['2006',
 'Aaliyah 91',
 'Aaron 57',
 'Abagail 895',
 'Abbey 695',
 'Abbie 650',
 'Abbigail 490',
 'Abby 205',
 'Abdullah 888',
 'Abel 338',
 'Abigail 6',
 'Abigale 889',
 'Abigayle 777',
 'Abraham 183',
 'Abram 586',
 'Abril 845',
 'Ace 838',
 'Ada 715',
 'Adam 64',
 'Adamaris 920',
 'Adan 302',
 'Addison 27',
 'Addisyn 612',
 'Addyson 370',
 'Adelaide 921',
 'Adeline 467',
 'Aden 244',
 'Adison 767',
 'Aditya 810',
 'Adolfo 581',
 'Adonis 822',
 'Adrian 63',
 'Adriana 106',
 'Adrianna 179',
 'Adriel 707',
 'Adrien 754',
 'Adrienne 736',
 'Adyson 877',
```

```
'Aedan 760',
'Agustin 657',
'Ahmad 524',
'Ahmed 559',
'Aidan 44',
'Aiden 30',
'Aidyn 910',
'Aileen 537',
'Aimee 763',
'Ainsley 456',
'Aisha 683',
'Aiyana 739',
'Akeelah 711',
'Akira 965',
'Alaina 248',
'Alan 122',
'Alana 166',
'Alani 933',
'Alanna 473',
'Alayna 268',
'Albert 354',
'Alberto 315',
'Alden 811',
'Aldo 587',
'Aleah 667',
'Alec 328',
'Aleena 830',
'Alejandra 232',
'Alejandro 96',
'Alena 665',
'Alessandra 361',
'Alessandro 608',
'Alex 67',
'Alexa 39',
'Alexander 12',
'Alexandra 40',
'Alexandria 147',
'Alexandro 861',
'Alexia 164',
'Alexis 14',
'Alexus 545',
'Alexzander 615',
'Alfonso 552',
'Alfred 755',
'Alfredo 334',
'Ali 360',
'Alia 928',
'Alice 383',
'Alicia 167',
'Alijah 584',
'Alina 355',
'Alisa 890',
'Alisha 584',
'Alison 271',
'Alissa 377',
'Alivia 254',
'Aliya 699',
'Aliyah 236',
'Aliza 856',
'Alize 959',
'Allan 507',
'Allen 283',
'Allie 229',
'Allison 46',
'Ally 630',
'Allyson 259',
'Alma 616',
'Alondra 170',
'Alonso 643',
```

```
 'Alonzo 558',
 'Alvaro 570',
 'Alvin 513',
 'Alyson 471',
 'Alyssa 19',
 'Amanda 102',
 'Amani 687',
 'Amara 573',
 'Amare 778',
 'Amari 421',
 'Amaris 909',
 'Amaya 215',
 'Amber 136',
 'Amelia 82',
 'Amelie 784',
 'America 458',
 'Amina 978',
 'Amir 324',
 'Amira 571',
 'Amirah 971',
 'Amiya 742',
 'Amiyah 635',
 'Amy 128',
 'Amya 476',
 'Ana 143',
 'Anabelle 757',
 'Anahi 287',
 'Anais 869',
 'Anastasia 288',
 'Anaya 486',
 'Anders 987',
 'Anderson 399',
 'Andre 207',
 'Andrea 59',
 'Andreas 880',
 'Andres 161',
 'Andrew 8',
 'Andy 204',
 'Angel 31',
 'Angela 114',
 'Angelica 208',
 'Angelina 48',
 'Angeline 863',
 'Angelique 693',
 'Angelo 262',
 'Angie 389',
 'Anika 485',
 'Aniya 262',
 'Aniyah 220',
 'Ann 731',
 'Anna 23',
 'Annabel 848',
 'Annabella 556',
 'Annabelle 206',
 'Annalise 589',
 'Anne 460',
 'Annette 881',
 'Annie 398',
 'Annika 335',
 'Ansley 704',
 'Anthony 9',
 'Antoine 631',
 'Anton 725',
 'Antonio 93',
 'Antony 815',
 'Antwan 828',
 'Anya 405',
 'April 319',
 'Arabella 653',
```

```
 'Araceli 570',
 'Aracely 722',
 'Areli 979',
 'Arely 576',
 'Ari 634',
 'Aria 661',
 'Ariana 78',
 'Arianna 77',
 'Ariel 202',
 'Arielle 587',
 'Arjun 720',
 'Armando 264',
 'Armani 698',
 'Arnav 932',
 'Aron 640',
 'Arthur 377',
 'Arturo 323',
 'Aryan 684',
 'Aryana 823',
 'Aryanna 652',
 'Asa 623',
 'Ashanti 882',
 'Asher 252',
 'Ashlee 338',
 'Ashleigh 688',
 'Ashley 12',
 'Ashly 780',
 'Ashlyn 140',
 'Ashlynn 293',
 'Ashton 121',
 'Asia 332',
 'Aspen 572',
 'Athena 504',
 'Atticus 767',
 'Aubree 426',
 'Aubrey 92',
 'Aubrie 604',
 'Audrey 68',
 'August 618',
 'Augustus 831',
 'Aurora 312',
 'Austen 924',
 'Austin 41',
 'Austyn 974',
 'Autumn 95',
 'Ava 5',
 'Averie 899',
 'Avery 52',
 'Axel 295',
 'Ayana 700',
 'Ayanna 527',
 'Aydan 576',
 'Ayden 119',
 'Aydin 731',
 'Ayla 264',
 'Aylin 776',
 'Bailee 651',
 'Bailey 112',
 'Barbara 561',
 'Barrett 762',
 'Baylee 469',
 'Beatrice 966',
 'Beau 438',
 'Beckett 758',
 'Belen 914',
 'Belinda 809',
 'Bella 181',
 'Ben 555',
 'Benjamin 24',
```

```
 'Bennett 369',
 'Benny 962',
 'Bernard 945',
 'Bernardo 975',
 'Bethany 244',
 'Bethzy 878',
 'Betsy 743',
 'Bianca 182',
 'Billy 473',
 'Blaine 572',
 'Blaise 985',
 'Blake 97',
 'Blanca 800',
 'Blaze 868',
 'Bo 771',
 'Bobby 480',
 'Bode 848',
 'Boston 626',
 'Brad 897',
 'Braden 141',
 'Bradley 188',
 'Brady 105',
 'Bradyn 675',
 'Braeden 331',
 'Braedon 744',
 'Braelyn 855',
 'Braiden 602',
 'Branden 472',
 'Brandi 967',
 'Brandon 27',
 'Brandy 801',
 'Branson 862',
 'Braulio 966',
 'Braxton 224',
 'Brayan 294',
 'Brayden 79',
 'Braydon 396',
 'Braylen 756',
 'Braylon 401',
 'Breana 844',
 'Breanna 126',
 'Bree 955',
 'Brenda 239',
 'Brendan 185',
 'Brenden 317',
 'Brendon 501',
 'Brenna 381',
 'Brennan 299',
 'Brennen 732',
 'Brent 481',
 'Brenton 942',
 'Brett 304',
 'Bria 870',
 'Brian 72',
 'Briana 121',
 'Brianna 20',
 'Brice 695',
 'Bridget 347',
 'Brielle 459',
 'Briley 960',
 'Brisa 605',
 'Britney 474',
 'Brittany 318',
 'Brittney 626',
 'Brock 261',
 'Broderick 884',
 'Brodie 442',
 'Brody 147',
 'Brogan 999',
```

```
'Brooke 44',
'Brooklyn 67',
'Brooklynn 237',
'Brooks 592',
'Bruce 482',
'Bruno 793',
'Bryan 66',
'Bryanna 424',
'Bryant 389',
'Bryce 109',
'Brycen 588',
'Brylee 885',
'Brynn 402',
'Bryson 176',
'Byron 523',
'Cade 288',
'Caden 91',
'Cadence 214',
'Cael 917',
'Caiden 312',
'Cailyn 872',
'Caitlin 207',
'Caitlyn 199',
'Cale 733',
'Caleb 34',
'Cali 602',
'Callie 303',
'Calvin 220',
'Camden 221',
'Cameron 52',
'Camila 180',
'Camilla 825',
'Camille 308',
'Campbell 659',
'Camren 579',
'Camron 345',
'Camryn 216',
'Cannon 796',
'Cara 559',
'Carina 864',
'Carissa 585',
'Carl 429',
'Carla 501',
'Carlee 654',
'Carley 550',
'Carlie 592',
'Carlo 937',
'Carlos 70',
'Carly 251',
'Carmelo 870',
'Carmen 258',
'Carmine 785',
'Carol 968',
'Carolina 281',
'Caroline 89',
'Carolyn 517',
'Carrie 859',
'Carson 87',
'Carter 75',
'Case 951',
'Casey 308',
'Cash 378',
'Cason 753',
'Cassandra 219',
'Cassidy 198',
'Cassie 732',
'Catalina 696',
'Catherine 122',
'Cayden 213',
```

```
'Cayla 886',
'Cecelia 669',
'Cecilia 265',
'Cedric 595',
'Celeste 327',
'Celia 707',
'Cesar 168',
'Chad 375',
'Chaim 946',
'Chana 990',
'Chance 273',
'Chandler 402',
'Chanel 917',
'Charity 673',
'Charlee 915',
'Charles 60',
'Charlie 337',
'Charlize 751',
'Charlotte 123',
'Chase 83',
'Chasity 744',
'Chaya 785',
'Chaz 905',
'Chelsea 192',
'Chelsey 790',
'Cherish 754',
'Cheyanne 657',
'Cheyenne 171',
'Chloe 18',
'Chris 358',
'Christian 21',
'Christiana 984',
'Christina 158',
'Christine 437',
'Christopher 7',
'Ciara 213',
'Cierra 563',
'Cindy 365',
'Citlali 948',
'Claire 86',
'Clara 233',
'Clare 663',
'Clarence 818',
'Clarissa 468',
'Clark 696',
'Claudia 339',
'Clay 708',
'Clayton 226',
'Clinton 844',
'Cloe 941',
'Coby 832',
'Cody 106',
'Coen 963',
'Cohen 415',
'Colby 271',
'Cole 84',
'Coleman 661',
'Colin 111',
'Colleen 902',
'Collin 181',
'Colt 906',
'Colten 510',
'Colton 133',
'Conner 144',
'Connor 53',
'Conor 496',
'Conrad 788',
'Cooper 113',
'Cora 384',
```

```
'Corbin 289',
'Corey 234',
'Corinne 826',
'Cornelius 939',
'Cortez 938',
'Cory 431',
'Courtney 190',
'Craig 548',
'Cristal 660',
'Cristian 132',
'Cristina 495',
'Cristobal 991',
'Cristofer 804',
'Cristopher 499',
'Cruz 500',
'Crystal 201',
'Cullen 790',
'Curtis 339',
'Cynthia 240',
'Cyrus 515',
'Dahlia 988',
'Daisy 149',
'Dakota 172',
'Dale 745',
'Dalia 849',
'Dallas 352',
'Dalton 199',
'Damarion 646',
'Damaris 609',
'Damian 136',
'Damien 196',
'Damion 497',
'Damon 386',
'Dana 395',
'Dandre 992',
'Dane 393',
'Dangelo 863',
'Dania 985',
'Danica 352',
'Daniel 6',
'Daniela 132',
'Daniella 307',
'Danielle 116',
'Danika 569',
'Danna 518',
'Danny 307',
'Dante 291',
'Daphne 606',
'Darian 590',
'Darien 830',
'Dario 875',
'Darion 869',
'Darius 280',
'Darnell 711',
'Darrell 597',
'Darren 361',
'Darrius 930',
'Darryl 773',
'Darwin 772',
'Dashawn 690',
'Davian 682',
'David 13',
'Davin 601',
'Davion 478',
'Davis 405',
'Davon 603',
'Dawson 233',
'Dayana 553',
'Dayanara 435',
```

```
 'Dayton 540',
 'Deacon 687',
 'Dean 385',
 'Deandre 452',
 'Deangelo 794',
 'Deanna 532',
 'Deborah 676',
 'Declan 364',
 'Deja 753',
 'Delaney 193',
 'Delilah 548',
 'Demarcus 735',
 'Demarion 749',
 'Demetrius 437',
 'Denise 379',
 'Dennis 313',
 'Denzel 940',
 'Deon 849',
 'Derek 159',
 'Derick 736',
 'Derrick 256',
 'Deshaun 853',
 'Deshawn 518',
 'Desirae 883',
 'Desiree 300',
 'Desmond 464',
 'Destin 960',
 'Destinee 554',
 'Destiney 972',
 'Destini 929',
 'Destiny 37',
 'Devan 582',
 'Deven 606',
 'Devin 100',
 'Devon 197',
 'Devonte 841',
 'Devyn 748',
 'Dexter 913',
 'Diamond 316',
 'Diana 120',
 'Diego 56',
 'Dillan 774',
 'Dillon 223',
 'Dion 988',
 'Domenic 964',
 'Dominic 85',
 'Dominick 216',
 'Dominik 612',
 'Dominique 648',
 'Donald 303',
 'Donavan 812',
 'Donna 832',
 'Donovan 198',
 'Donte 722',
 'Dorian 469',
 'Douglas 365',
 'Drake 239',
 'Draven 685',
 'Drew 205',
 'Dulce 274',
 'Duncan 654',
 'Dustin 259',
 'Dwayne 610',
 'Dylan 26',
 'Ean 775',
 'Earl 993',
 'Easton 359',
 'Eddie 395',
 'Eddy 855',
```

```
'Eden 320',
'Edgar 171',
'Edison 947',
'Edith 638',
'Eduardo 126',
'Edward 143',
'Edwin 158',
'Efrain 663',
'Efren 968',
'Eileen 770',
'Elaina 448',
'Elaine 719',
'Eleanor 277',
'Elena 187',
'Eli 139',
'Eliana 282',
'Elias 186',
'Elijah 29',
'Elisa 543',
'Elisabeth 502',
'Elise 218',
'Elisha 629',
'Eliza 325',
'Elizabeth 11',
'Ella 21',
'Elle 480',
'Ellen 544',
'Elliana 857',
'Ellie 175',
'Elliot 372',
'Elliott 388',
'Ellis 783',
'Elmer 907',
'Elsa 792',
'Elsie 879',
'Elvis 761',
'Elyse 684',
'Emanuel 263',
'Emely 309',
'Emerson 305',
'Emery 778',
'Emilee 375',
'Emilia 420',
'Emiliano 332',
'Emilie 491',
'Emilio 298',
'Emily 1',
'Emma 2',
'Emmalee 685',
'Emmanuel 166',
'Emmett 569',
'Enrique 281',
'Enzo 737',
'Eric 77',
'Erica 222',
'Erick 174',
'Ericka 993',
'Erik 189',
'Erika 238',
'Erin 130',
'Ernest 723',
'Ernesto 379',
'Esmeralda 224',
'Esperanza 675',
'Essence 839',
'Esteban 333',
'Estefani 905',
'Estevan 846',
'Esther 298',
```

```
 'Estrella 311',
 'Ethan 4',
 'Ethen 952',
 'Eugene 647',
 'Eva 124',
 'Evan 42',
 'Evangeline 597',
 'Eve 590',
 'Evelin 674',
 'Evelyn 65',
 'Everett 451',
 'Ezekiel 269',
 'Ezequiel 541',
 'Ezra 340',
 'Fabian 272',
 'Fabiola 980',
 'Faith 64',
 'Fatima 228',
 'Felicity 617',
 'Felipe 457',
 'Felix 397',
 'Fernanda 431',
 'Fernando 151',
 'Finley 886',
 'Finn 456',
 'Finnegan 779',
 'Fiona 333',
 'Flor 1000',
 'Frances 779',
 'Francesca 428',
 'Francis 561',
 'Francisco 157',
 'Franco 918',
 'Frank 245',
 'Frankie 648',
 'Franklin 439',
 'Freddy 641',
 'Frederick 483',
 'Fredrick 903',
 'Frida 836',
 'Gabriel 28',
 'Gabriela 110',
 'Gabriella 50',
 'Gabrielle 62',
 'Gael 314',
 'Gage 156',
 'Gaige 919',
 'Galilea 840',
 'Gannon 889',
 'Garret 750',
 'Garrett 138',
 'Garrison 895',
 'Gary 350',
 'Gauge 881',
 'Gaven 948',
 'Gavin 38',
 'Gavyn 710',
 'Genesis 169',
 'Genevieve 368',
 'George 153',
 'Georgia 273',
 'Gerald 544',
 'Gerardo 268',
 'German 839',
 'Gia 643',
 'Giana 708',
 'Giancarlo 716',
 'Gianna 98',
 'Gianni 611',
```

```
 'Gideon 591',
 'Gilbert 658',
 'Gilberto 554',
 'Gillian 758',
 'Gina 712',
 'Giovani 791',
 'Giovanna 723',
 'Giovanni 146',
 'Giovanny 702',
 'Giselle 168',
 'Gisselle 677',
 'Glenn 850',
 'Gloria 453',
 'Gonzalo 925',
 'Gordon 900',
 'Grace 17',
 'Gracelyn 759',
 'Gracie 103',
 'Grady 475',
 'Graham 430',
 'Grant 155',
 'Grayson 218',
 'Gregory 208',
 'Greta 680',
 'Gretchen 771',
 'Greyson 503',
 'Griffin 254',
 'Guadalupe 255',
 'Guillermo 440',
 'Gunnar 502',
 'Gunner 578',
 'Gustavo 305',
 'Guy 989',
 'Gwendolyn 631',
 'Haden 858',
 'Hadley 494',
 'Haiden 872',
 'Hailee 454',
 'Hailey 25',
 'Hailie 760',
 'Haleigh 596',
 'Haley 75',
 'Halie 937',
 'Halle 493',
 'Hallie 450',
 'Hamza 738',
 'Hana 716',
 'Hanna 269',
 'Hannah 8',
 'Harley 388',
 'Harmony 342',
 'Harold 652',
 'Harper 510',
 'Harrison 232',
 'Harry 593',
 'Hassan 765',
 'Haven 610',
 'Hayden 73',
 'Haylee 247',
 'Hayley 306',
 'Haylie 427',
 'Hazel 465',
 'Heath 786',
 'Heather 341',
 'Heaven 253',
 'Hector 175',
 'Heidi 295',
 'Helen 343',
 'Helena 508',
```

```
 'Henry 95',
 'Hezekiah 885',
 'Hillary 982',
 'Holden 384',
 'Holly 346',
 'Hope 200',
 'Houston 837',
 'Howard 836',
 'Hudson 249',
 'Hugh 994',
 'Hugo 371',
 'Humberto 697',
 'Hunter 54',
 'Ian 81',
 'Ibrahim 594',
 'Ignacio 703',
 'Iliana 727',
 'Imani 409',
 'Imanol 969',
 'India 568',
 'Ingrid 619',
 'Irene 593',
 'Iris 369',
 'Irvin 670',
 'Isaac 48',
 'Isabel 87',
 'Isabela 470',
 'Isabell 724',
 'Isabella 4',
 'Isabelle 85',
 'Isai 706',
 'Isaiah 40',
 'Isaias 531',
 'Isiah 406',
 'Isis 566',
 'Ismael 327',
 'Israel 203',
 'Issac 409',
 'Itzel 378',
 'Ivan 127',
 'Ivy 334',
 'Iyana 880',
 'Iyanna 991',
 'Izabella 290',
 'Izabelle 998',
 'Izaiah 461',
 'Izayah 976',
 'Jabari 609',
 'Jace 187',
 'Jacey 747',
 'Jack 35',
 'Jackson 36',
 'Jaclyn 860',
 'Jacob 1',
 'Jacoby 909',
 'Jacqueline 118',
 'Jacquelyn 668',
 'Jada 93',
 'Jade 111',
 'Jaden 88',
 'Jadon 398',
 'Jadyn 286',
 'Jaeden 666',
 'Jaelyn 442',
 'Jaelynn 793',
 'Jaheim 977',
 'Jaida 560',
 'Jaiden 214',
 'Jaidyn 586',
```

```
 'Jaime 279',
 'Jair 726',
 'Jairo 564',
 'Jakayla 730',
 'Jake 107',
 'Jakob 293',
 'Jalen 228',
 'Jaliyah 762',
 'Jalyn 956',
 'Jalynn 949',
 'Jamal 504',
 'Jamar 649',
 'Jamarcus 914',
 'Jamari 444',
 'Jamarion 459',
 'Jamel 970',
 'James 16',
 'Jameson 424',
 'Jamie 242',
 'Jamir 768',
 'Jamison 534',
 'Jamya 737',
 'Jan 751',
 'Janae 679',
 'Jane 478',
 'Janelle 390',
 'Janessa 594',
 'Janet 562',
 'Janiah 824',
 'Janice 994',
 'Janiya 386',
 'Janiyah 461',
 'Jaquan 656',
 'Jaqueline 512',
 'Jared 137',
 'Jaron 819',
 'Jarrett 659',
 'Jarvis 1000',
 'Jase 565',
 'Jasiah 933',
 'Jasmin 183',
 'Jasmine 29',
 'Jasmyn 926',
 'Jason 55',
 'Jasper 568',
 'Javen 954',
 'Javier 162',
 'Javion 644',
 'Javon 410',
 'Jax 995',
 'Jaxon 211',
 'Jaxson 391',
 'Jay 351',
 'Jayce 427',
 'Jaycee 765',
 'Jayda 294',
 'Jayden 50',
 'Jaydin 971',
 'Jaydon 416',
 'Jayla 99',
 'Jaylah 903',
 'Jaylan 692',
 'Jaylee 818',
 'Jayleen 833',
 'Jaylen 191',
 'Jaylene 912',
 'Jaylin 441',
 'Jaylon 479',
 'Jaylyn 796',
```

```
 'Jaylynn 632',
 'Jayson 353',
 'Jazlyn 533',
 'Jazmin 156',
 'Jazmine 226',
 'Jazmyn 620',
 'Jean 721',
 'Jefferson 642',
 'Jeffery 432',
 'Jeffrey 180',
 'Jenna 88',
 'Jennifer 51',
 'Jenny 475',
 'Jeramiah 926',
 'Jeremiah 71',
 'Jeremy 123',
 'Jerimiah 931',
 'Jermaine 474',
 'Jerome 577',
 'Jerry 318',
 'Jesse 102',
 'Jessica 32',
 'Jessie 485',
 'Jesus 74',
 'Jett 535',
 'Jewel 861',
 'Jillian 174',
 'Jimena 472',
 'Jimmy 325',
 'Joan 978',
 'Joana 999',
 'Joanna 256',
 'Joaquin 286',
 'Jocelyn 73',
 'Joe 370',
 'Joel 124',
 'Joey 537',
 'Johan 528',
 'Johana 850',
 'Johanna 376',
 'John 20',
 'Johnathan 177',
 'Johnathon 542',
 'Johnny 237',
 'Johnpaul 979',
 'Jolie 614',
 'Jon 455',
 'Jonah 170',
 'Jonas 357',
 'Jonathan 22',
 'Jonathon 376',
 'Jordan 46',
 'Jorden 759',
 'Jordon 739',
 'Jordy 677',
 'Jordyn 178',
 'Jorge 120',
 'Jorja 969',
 'Jose 32',
 'Josef 972',
 'Joselin 892',
 'Joselyn 314',
 'Joseph 11',
 'Josephine 221',
 'Josh 714',
 ...]
```