# Beautiful Python Refactoring

Conor Hoekstra

code_report

NVIDIA RAPIDS

#include

# About Me

- I'm a Senior Library Software Engineer for NVIDIA
  - Working on the RAPIDS AI team (http://rapids.ai)
- I am a programming language enthusiast
- My team uses C++14 and Python 3
- I love algorithms and beautiful code
- I have a YouTube channel
  youtube.com/codereport
- My online alias is `code_report`

# Why am I giving this talk?

# Back in October 2019
# I joined …


# Back in October 2019
# I joined …

NVIDIA®

RAPIDS

# cuDF

NVIDIA®

RAPIDS

cuDF cuML cuGraph

CUDA®

DataFrame

NVIDIA®

RAPIDS

cuDF cuML cuGraph

CUDA®

DataFrame

# DataFrame Libraries

RAPIDS cuDF

pandas

# DataFrame Libraries

RAPIDS

## cuDF

cuDF provides a pandas-like API that will be familiar to data engineers & data scientists, so they can use it to easily accelerate their workflows without going into the details of CUDA programming.

pandas

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

# So what data did I want to manipulate or analyse?

CODEFORCES

CODEFORCES

# What are the most popular programming languages used on sites like CodeForces?

# How do we do that?

90%

1. Scrape HTML with pandas
2. Analyse with pandas
3. Done

# How do we do that?

1. Scrape HTML with pandas
2. Analyse with pandas
3. Done

Google

scrape html with pandas

Google Search

I'm Feeling Lucky

Google

scrape HTML with pandas

All | Images | Videos | Shopping | News | More | Settings | Tools

About 640,000 results (0.56 seconds)

pythonprogramminglanguage.com › web-scraping-with-pandas-and-b...

## Web Scraping with Pandas and Beautifulsoup - Learn Python

Web **Scraping** with **Pandas** and Beautifulsoup. APIs are not always available. Sometimes you have to **scrape** data from a webpage yourself. Luckily the modules ...

towardsdatascience.com › web-scraping-html-tables-with-python-c9ba...

## Web Scraping HTML Tables with Python - Towards Data ...

Jul 25, 2018 - Finally, we will store the data on a **Pandas** Dataframe. import requests import lxml .**html** as lh import **pandas** as pd. **Scrape** Table Cells. The code ...

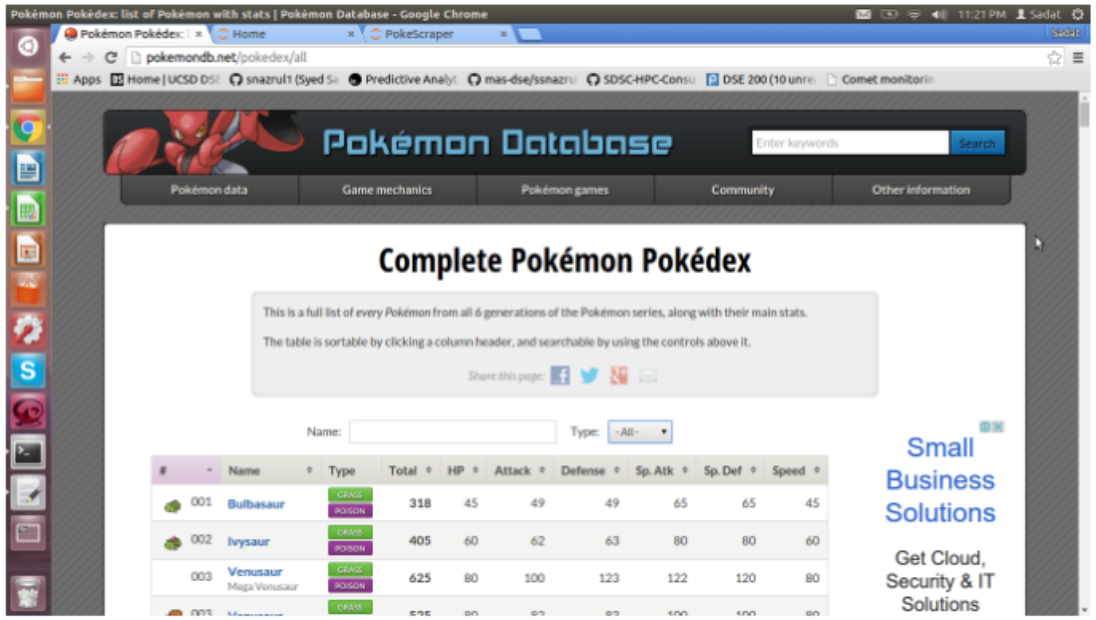
pandas.pydata.org › pandas-docs › version › generated › pandas.read...

## pandas.read_html — pandas 0.23.4 documentation

Read **HTML** tables into a list of DataFrame objects. Parameters: io : str or file-like. A URL, a file-like object, or a raw string containing **HTML**. Note that lxml only ...

# Web Scraping HTML Tables with Python

Syed Sadat Nazrul Follow
Jul 25, 2018 · 3 min read




Pokemon Database Website

Starting off, we will try scraping the online Pokemon Database (http://pokemondb.net/pokedex/all).

```python
import requests
import lxml.html as lh
import pandas as pd
```

```python
#Create a handle, page, to handle the contents of the website
page = requests.get(url)

#Store the contents of the website under doc
doc = lh.fromstring(page.content)

#Parse data that are stored between <tr>..</tr> of HTML
tr_elements = doc.xpath('//tr')
```

```python
#Create empty list
col = []
i = 0

#For each row, store each first element (header) and an empty list
for t in tr_elements[0]:
    i += 1
    name = t.text_content()
    print('%d:"%s"'%(i, name))
    col.append((name, []))
```

```python
#Since out first row is the header, data is stored on the second row onwards
for j in range(1,len(tr_elements)):
    #T is our j'th row
    T = tr_elements[j]

    #If row is not of size 10, the //tr data is not from our table
    if len(T) != 10:
        break

    #i is the index of our column
    i = 0

    #Iterate through each element of the row
    for t in T.iterchildren():
        data = t.text_content()
        #Check if row is empty
        if i > 0:
        #Convert any numerical value to integers
            try:
                data = int(data)
            except:
                pass
        #Append the data to the empty list of the i'th column
        col[i][1].append(data)
        #Increment i for the next column
        i+=1
```

```python
Dict = {title:column for (title,column) in col}
df   = pd.DataFrame(Dict)

print(df.head())
```

# Time to refactor!

```python
#Create empty list
col = []
i = 0

#For each row, store each first element (header) and an empty list
for t in tr_elements[0]:
    i += 1
    name = t.text_content()
    print('%d:"%s"'%(i, name))
    col.append((name, []))
```

# Change 1 enumerate

```
col = []
i = 0

for t in tr_elements[0]:
    i += 1
    name = t.text_content()
    print('%d:"%s"'%(i, name))
    col.append((name, []))
```

# Change 1
# enumerate

```
col = []

for i, t in enumerate(tr_elements[0]):
    name = t.text_content()
    print('%d:"%s"'%(i, name))
    col.append((name, []))
```

| | | | |
|---|---|---|---|
| Rust | **enumerate** | trait.Iterator | Doc |
| Python | **enumerate** | - | Doc |
| Racket | **enumerate** | list-utils | Doc |
| D | **enumerate** | range | Doc |
| Ruby | **with_index** | Enumerable | Doc |
| Kotlin | **withIndex** | collections | Doc |
| Elixir | **with_index** | Enum | Doc |
| Racket | **indexed** | data/collection | Doc |
| Haskell | **indexed** | Data.List.Index | Doc |
| Clojure | **map-indexed*** | core | Doc |
| C# | **Select** | Enumerable | Doc |
| Scala | **zipWithIndex** | various | Doc |

# Change 1
# enumerate

```
col = []

for i, t in enumerate(tr_elements[0]):
    name = t.text_content()
    print('%d:"%s"'%(i, name))
    col.append((name, []))
```

# Change 2

## Delete print & enumerate

```
col = []

for i, t in enumerate(tr_elements[0]):
    name = t.text_content()
    print('%d:"%s"'%(i, name))
    col.append((name, []))
```

# Change 2

## Delete print & enumerate

```
col = []

for t in tr_elements[0]:
    name = t.text_content()
    col.append((name, []))
```

# Change 2

## Delete print & enumerate

```
col = []

for t in tr_elements[0]:
    col.append((t.text_content(), []))
```

# Change 3

## List comprehension

```
col = []

for t in tr_elements[0]:
    col.append((t.text_content(), []))
```

# Change 3

## List comprehension

```
col = [(t.text_content(), []) for t in tr_elements[0]]
```

# ITM

# Initialize
# Then
# Modify

```python
col = []
i = 0

for t in tr_elements[0]:
    i += 1
    name = t.text_content()
    print('%d:"%s"'%(i, name))
    col.append((name, []))
```

```python
col = []

for t in tr_elements[0]:
    name = t.text_content()
    col.append((name, []))
```

```python
#Since out first row is the header, data is stored on the second row onwards
for j in range(1,len(tr_elements)):
    #T is our j'th row
    T = tr_elements[j]

    #If row is not of size 10, the //tr data is not from our table
    if len(T) != 10:
        break
```

# Change 4

```python
#If row is not of size 10, the //tr data is not from our table
if len(T) != 10:
    break
```

**Delete if**

```python
        #check if row is empty
        if i > 0:
        #Convert any numerical value to integers
            try:
                data = int(data)
            except:
                pass
        #Append the data to the empty list of the i'th column
        col[i][1].append(data)
        #Increment i for the next column
        i+=1
```

```python
#Since out first row is the header, data is stored on the second row onwards
for j in range(1,len(tr_elements)):
    #T is our j'th row
    T = tr_elements[j]

    #i is the index of our column
    i = 0

    #Iterate through each element of the row
    for t in T.iterchildren():
        data = t.text_content()
        #Check if row is empty
        if i > 0:
        #Convert any numerical value to integers
            try:
                data = int(data)
            except:
                pass
        #Append the data to the empty list of the i'th column
        col[i][1].append(data)
        #Increment i for the next column
        i+=1
```

# Change 4

## Delete if

```python
#Since out first row is the header, data is stored on the second row onwards
for j in range(1, len(tr_elements)):
    #T is our j'th row
    T = tr_elements[j]

    #i is the index of our column
    i = 0
```

# Change 5

Use slicing

```python
#Since out first row is the header, data is stored on the second row onwards
for j in range(1, len(tr_elements)):
    #T is our j'th row
    T = tr_elements[j]
```

```python
            try:
                data = int(data)
            except:
                pass
        #Append the data to the empty list of the i'th column
        col[i][1].append(data)
        #Increment i for the next column
        i += 1
```

```python
for T in tr_elements[1:]:
    #i is the index of our column
    i = 0

    #Iterate through each element of the row
    for t in T.iterchildren():
        data = t.text_content()
        #Check if row is empty
        if i > 0:
        #Convert any numerical value to integers
            try:
                data = int(data)
            except:
                pass
        #Append the data to the empty list of the i'th column
        col[i][1].append(data)
        #Increment i for the next column
        i += 1
```

# Change 6

Use enumerate

# Change 5

Use slicing

# Change 6

## Use enumerate

```
for T in tr_elements[1:]:
    #Iterate through each element of the row
    for i, t in enumerate(T.iterchildren()):
        data = t.text_content()
        #Check if row is empty
        if i > 0:
        #Convert any numerical value to integers
            try:
                data = int(data)
            except:
                pass
        #Append the data to the empty list of the i'th column
        col[i][1].append(data)
```

# Change 7

## Delete if

```
for T in tr_elements[1:]:
    #Iterate through each element of the row
    for i, t in enumerate(T.iterchildren()):
        data = t.text_content()
        #Check if row is empty
        if i > 0:
        #Convert any numerical value to integers
            try:
                data = int(data)
            except:
                pass
        #Append the data to the empty list of the i'th column
        col[i][1].append(data)
```

# Change 7

## Delete if

```
for T in tr_elements[1:]:
    #Iterate through each element of the row
    for i, t in enumerate(T.iterchildren()):
        data = t.text_content()
        #Convert any numerical value to integers
        try:
            data = int(data)
        except:
            pass
        #Append the data to the empty list of the i'th column
        col[i][1].append(data)
```

# Change 8

## Use conditional expression

```
for T in tr_elements[1:]:
    #Iterate through each element of the row
    for i, t in enumerate(T.iterchildren()):
        data = t.text_content()
        #Convert any numerical value to integers
        try:
            data = int(data)
        except:
            pass
        #Append the data to the empty list of the i'th column
        col[i][1].append(data)
```

# Change 8

## Use conditional expression

```
for T in tr_elements[1:]:
    #Iterate through each element of the row
    for i, t in enumerate(T.iterchildren()):
        data = t.text_content()
        data = int(data) if data.isnumeric() else data
        #Append the data to the empty list of the i'th column
        col[i][1].append(data)
```

# Change 9

## Remove redundant comments

```
for T in tr_elements[1:]:
    #Iterate through each element of the row
    for i, t in enumerate(T.iterchildren()):
        data = t.text_content()
        data = int(data) if data.isnumeric() else data
        #Append the data to the empty list of the i'th column
        col[i][1].append(data)
```

# Change 9

## Remove redundant comments

```
for T in tr_elements[1:]:
    for i, t in enumerate(T.iterchildren()):
        data = t.text_content()
        col[i][1].append(int(data) if data.isnumeric() else data)
```

```python
url = 'http://pokemondb.net/pokedex/all'

#Create a handle, page, to handle the contents of the website
page = requests.get(url)

#Store the contents of the website under doc
doc = lh.fromstring(page.content)

#Parse data that are stored between <tr>..</tr> of HTML
tr_elements = doc.xpath('//tr')

#Check the length of the first 12 rows
# [len(T) for T in tr_elements[:12]]

tr_elements = doc.xpath('//tr')

#Create empty list
col = []
i = 0

#For each row, store each first element (header) and an empty list
for t in tr_elements[0]:
    i += 1
    name = t.text_content()
    print('%d:"%s"'%(i, name))
    col.append((name, []))

#Since out first row is the header, data is stored on the second row onwards
for j in range(1, len(tr_elements)):
    #T is our j'th row
    T = tr_elements[j]

    #If row is not of size 10, the //tr data is not from our table
    if len(T) != 10:
        break

    #i is the index of our column
    i = 0

    #Iterate through each element of the row
    for t in T.iterchildren():
        data = t.text_content()
        #Check if row is empty
        if i > 0:
        #Convert any numerical value to integers
            try:
                data = int(data)
            except:
                pass
        #Append the data to the empty list of the i'th column
        col[i][1].append(data)
        #Increment i for the next column
        i += 1

# [len(C) for (title,C) in col]

Dict = {title:column for (title,column) in col}
df   = pd.DataFrame(Dict)

print(df.head())
```

```python
url = 'http://pokemondb.net/pokedex/all'

#Create a handle, page, to handle the contents of the website
page = requests.get(url)

#Store the contents of the website under doc
doc = lh.fromstring(page.content)

#Parse data that are stored between <tr>..</tr> of HTML
tr_elements = doc.xpath('//tr')

col = [(t.text_content(), []) for t in tr_elements[0]]

for T in tr_elements[1:]:
    for i, t in enumerate(T.iterchildren()):
        data = t.text_content()
        col[i][1].append(int(data) if data.isnumeric() else data)

Dict = {title:column for (title,column) in col}
df   = pd.DataFrame(Dict)

print(df.head())
```

```python
url = 'http://pokemondb.net/pokedex/all'

page = requests.get(url)                          # page handle
doc  = lh.fromstring(page.content)                # website contents
tr   = doc.xpath('//tr')                          # html <tr> data
col  = [(t.text_content(), []) for t in tr[0]]     # column titles

# scrape data
for T in tr[1:]:
    for i, t in enumerate(T.iterchildren()):
        data = t.text_content()
        col[i][1].append(int(data) if data.isnumeric() else data)

Dict = {title:column for (title,column) in col}
df   = pd.DataFrame(Dict)

print(df.head())
```

```python
url = 'http://pokemondb.net/pokedex/all'

page    = requests.get(url)                # page handle
doc     = lh.fromstring(page.content)      # website contents
tr      = doc.xpath('//tr')                # html <tr> data
titles  = [t.text_content() for t in tr[0]] # column titles

# scrape data
for T in tr[1:]:
    for i, t in enumerate(T.iterchildren()):
        data = t.text_content()
        col[i][1].append(int(data) if data.isnumeric() else data)

Dict = {title:column for (title,column) in col}
df   = pd.DataFrame(Dict)

print(df.head())
```

```python
url = 'http://pokemondb.net/pokedex/all'

page   = requests.get(url)                # page handle
doc    = lh.fromstring(page.content)      # website contents
tr     = doc.xpath('//tr')                # html <tr> data
titles = [t.text_content() for t in tr[0]] # column titles

# scrape data
cols = [] * len(titles)
for T in tr[1:]:
    for i, t in enumerate(T.iterchildren()):
        data = t.text_content()
        cols[i].append(int(data) if data.isnumeric() else data)

Dict = {title:column for (title,column) in col}
df   = pd.DataFrame(Dict)

print(df.head())
```

```python
url = 'http://pokemondb.net/pokedex/all'

page   = requests.get(url)                # page handle
doc    = lh.fromstring(page.content)      # website contents
tr     = doc.xpath('//tr')                # html <tr> data
titles = [t.text_content() for t in tr[0]] # column titles

# scrape data
fmt = lambda data : int(data) if data.isnumeric() else data

cols = [] * len(titles)
for T in tr[1:]:
    for i, t in enumerate(T.iterchildren()):
        cols[i].append(fmt(t.text_content()))

Dict = {title:column for (title,column) in col}
df   = pd.DataFrame(Dict)

print(df.head())
```

```python
url = 'http://pokemondb.net/pokedex/all'

page   = requests.get(url)                    # page handle
doc    = lh.fromstring(page.content)          # website contents
tr     = doc.xpath('//tr')                    # html <tr> data
titles = [t.text_content() for t in tr[0]] # column titles

fmt    = lambda data : int(data) if data.isnumeric() else data
cols   = zip(*[[fmt(t.text_content()) for t in T.iterchildren()]
                                      for T in tr[1:]])

Dict   = {title: column for title, column in zip(titles, cols)}
df     = pd.DataFrame(Dict)

print(df.head())
```

| | | | |
|---|---|---|---|
| RAPIDS (Python) | **transpose** | cuDF | Doc |
| pandas | **transpose** | DataFrame | Doc |
| Haskell | **transpose** | Data.List | Doc |
| Elixir | **transpose** | Matrix | Doc |
| Scala | **transpose** | various | Doc |
| J | **\|: (transpose)** | - | Doc |
| APL | **⍉ (transpose)** | - | Doc |
| Clojure | **transpose** | core.matrix | Doc |
| Ruby | **transpose** | Array | Doc |
| q | **flip** | - | Doc |
| D | **transposed** | range | Doc |
| Python | **zip(*)** | - | Doc |

```python
url = 'http://pokemondb.net/pokedex/all'

page   = requests.get(url)                    # page handle
doc    = lh.fromstring(page.content)          # website contents
tr     = doc.xpath('//tr')                    # html <tr> data
titles = [t.text_content() for t in tr[0]] # column titles

fmt    = lambda data : int(data) if data.isnumeric() else data
cols   = zip(*[[fmt(t.text_content()) for t in T.iterchildren()]
                                      for T in tr[1:]])

Dict   = {title: column for title, column in zip(titles, cols)}
df     = pd.DataFrame(Dict)

print(df.head())
```

# Can we do better?
# What is the mistake I made?

Google

scrape html with pandas

Google Search

I'm Feeling Lucky

Google

scrape HTML with pandas

All | Images | Videos | Shopping | News | More | Settings | Tools

About 640,000 results (0.56 seconds)

pythonprogramminglanguage.com › web-scraping-with-pandas-and-b...

## Web Scraping with Pandas and Beautifulsoup - Learn Python

Web **Scraping** with **Pandas** and Beautifulsoup. APIs are not always available. Sometimes you have to **scrape** data from a webpage yourself. Luckily the modules ...

towardsdatascience.com › web-scraping-html-tables-with-python-c9ba...

## Web Scraping HTML Tables with Python - Towards Data ...

Jul 25, 2018 - Finally, we will store the data on a **Pandas** Dataframe. import requests import lxml .**html** as lh import **pandas** as pd. **Scrape** Table Cells. The code ...

pandas.pydata.org › pandas-docs › version › generated › pandas.read...

## pandas.read_html — pandas 0.23.4 documentation

Read **HTML** tables into a list of DataFrame objects. Parameters: io : str or file-like. A URL, a file-like object, or a raw string containing **HTML**. Note that lxml only ...

```python
url = 'http://pokemondb.net/pokedex/all'

page   = requests.get(url)                    # page handle
doc    = lh.fromstring(page.content)          # website contents
tr     = doc.xpath('//tr')                    # html <tr> data
titles = [t.text_content() for t in tr[0]] # column titles

fmt    = lambda data : int(data) if data.isnumeric() else data
cols   = zip(*[[fmt(t.text_content()) for t in T.iterchildren()]
                                      for T in tr[1:]])

Dict   = {title: column for title, column in zip(titles, cols)}
df     = pd.DataFrame(Dict)

print(df.head())
```

```python
url = 'http://pokemondb.net/pokedex/all'

header = { "User-Agent": # ...
r      = requests.get(url, headers=header)
df     = pd.read_html(r.text)[0]

print(df.head())
```

1. Know your algorithms
2. Know your collections
3. Know your libraries

| | | |
|---|---|---|
| Change 11: Separate title and columns / list comprehension<br>codereport committed 21 hours ago | | 02d1c48 |
| Change 10: Reformat / align<br>codereport committed 21 hours ago | | cf3a29c |
| Change 9: Remove redundant comments<br>codereport committed 21 hours ago | | bfe9d68 |
| Change 8: Use conditional expression<br>codereport committed 21 hours ago | | 958d0d8 |
| Change 7: Delete unnecessary if<br>codereport committed 22 hours ago | | 2f9b1a2 |
| Change 6: Use enumerate<br>codereport committed 22 hours ago | | 639a704 |
| Change 5: Use slicing<br>codereport committed 22 hours ago | | b4e9fe4 |
| Change 4: Delete unnecessary if<br>codereport committed 22 hours ago | Verified | 2aa6299 |
| Change 3: List comprehension<br>codereport committed 22 hours ago | Verified | 97812ba |
| Change 2: Delete print and enumerate<br>codereport committed 22 hours ago | Verified | e11b010 |
| Change 1: Use enumerate<br>codereport committed 22 hours ago | Verified | fe677c8 |
| Change 0: Remove unnecessary lines<br>codereport committed 22 hours ago | Verified | d3b21da |
| Create final.py<br>codereport committed 22 hours ago | Verified | 90aa300 |

# How do we do that?

1. Scrape HTML with pandas
2. Analyse with pandas
3. Done

HOME TOP CONTESTS GYM PROBLEMSET GROUPS RATING API CALENDAR HELP 10 YEARS!

PROBLEMS SUBMIT CODE MY SUBMISSIONS STATUS HACKS STANDINGS CUSTOM INVOCATION

Contest status

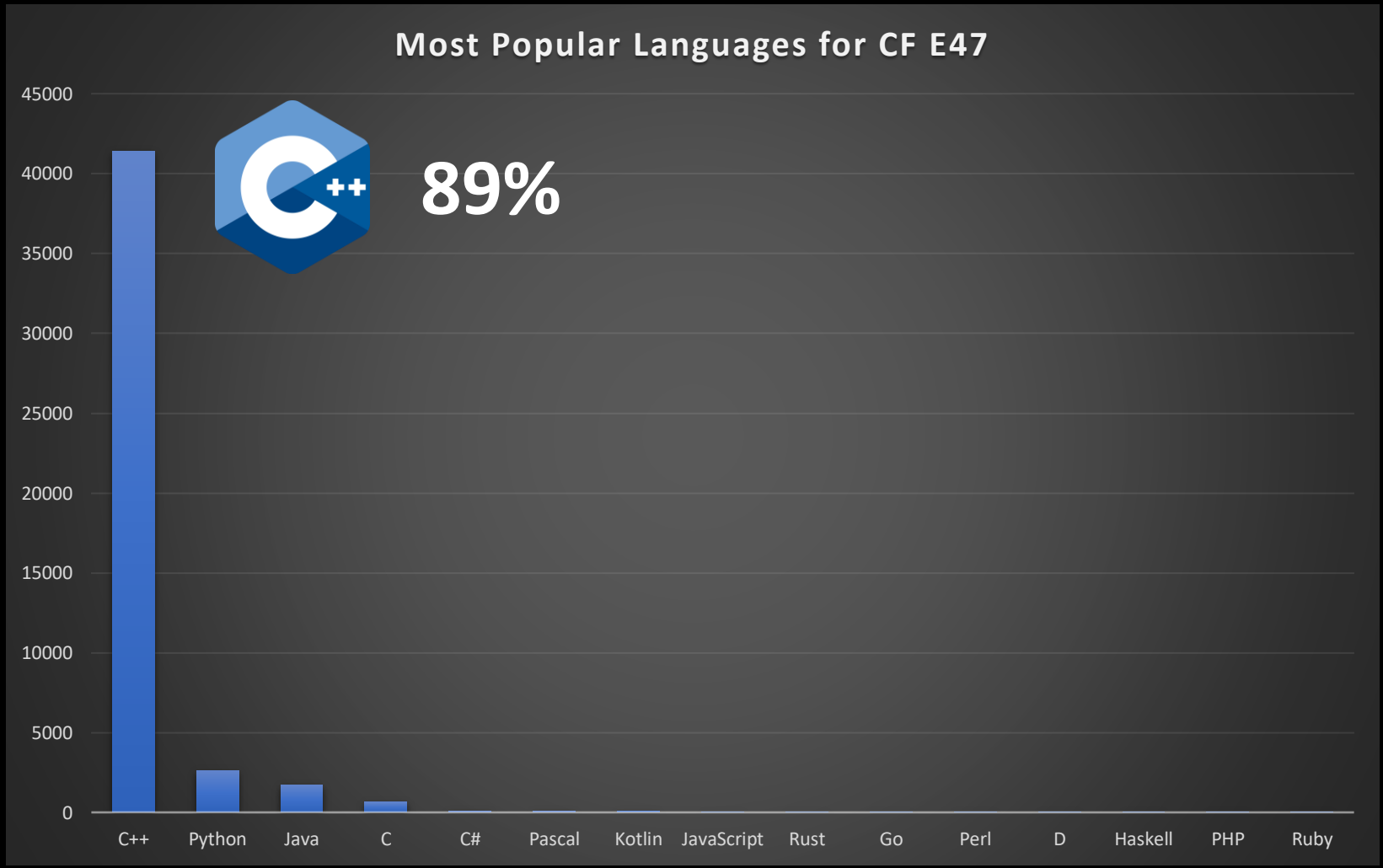
| # | When | Who | Problem | Lang | Verdict | Time | Memory |
|---|---|---|---|---|---|---|---|
| 75652062 | Apr/06/2020 07:46UTC-4 | nguyenhoangminh31082003 | D - AB-string | GNU C++11 | Accepted | 31 ms | 9900 KB |
| 75650841 | Apr/06/2020 07:32UTC-4 | RAVAL_KUSH_460 | A - Prime Subtraction | GNU C++17 | Accepted | 31 ms | 0 KB |
| 75650597 | Apr/06/2020 07:29UTC-4 | Invisible_Shadow | D - AB-string | GNU C++14 | Accepted | 31 ms | 1600 KB |
| 75650333 | Apr/06/2020 07:26UTC-4 | _chanchal | A - Prime Subtraction | Python 3 | Accepted | 109 ms | 0 KB |
| 75650010 | Apr/06/2020 07:22UTC-4 | Faiaz-1999 | A - Prime Subtraction | GNU C++17 | Wrong answer on test 2 | 31 ms | 0 KB |
| 75649904 | Apr/06/2020 07:20UTC-4 | Faiaz-1999 | A - Prime Subtraction | GNU C++17 | Compilation error | 0 ms | 0 KB |
| 75649815 | Apr/06/2020 07:19UTC-4 | sarafat_adir | A - Prime Subtraction | GNU C++11 | Accepted | 31 ms | 0 KB |
| 75649733 | Apr/06/2020 07:18UTC-4 | cwf123 | A - Prime Subtraction | GNU C++17 | Accepted | 31 ms | 0 KB |
| 75649436 | Apr/06/2020 07:15UTC-4 | Faiaz-1999 | A - Prime Subtraction | GNU C++17 | Wrong answer on test 2 | 31 ms | 0 KB |
| 75647309 | Apr/06/2020 06:46UTC-4 | noob_cyborg | A - Prime Subtraction | GNU C++14 | Accepted | 15 ms | 0 KB |
| 75646957 | Apr/06/2020 06:42UTC-4 | ganpat_98 | A - Prime Subtraction | GNU C11 | Accepted | 31 ms | 200 KB |

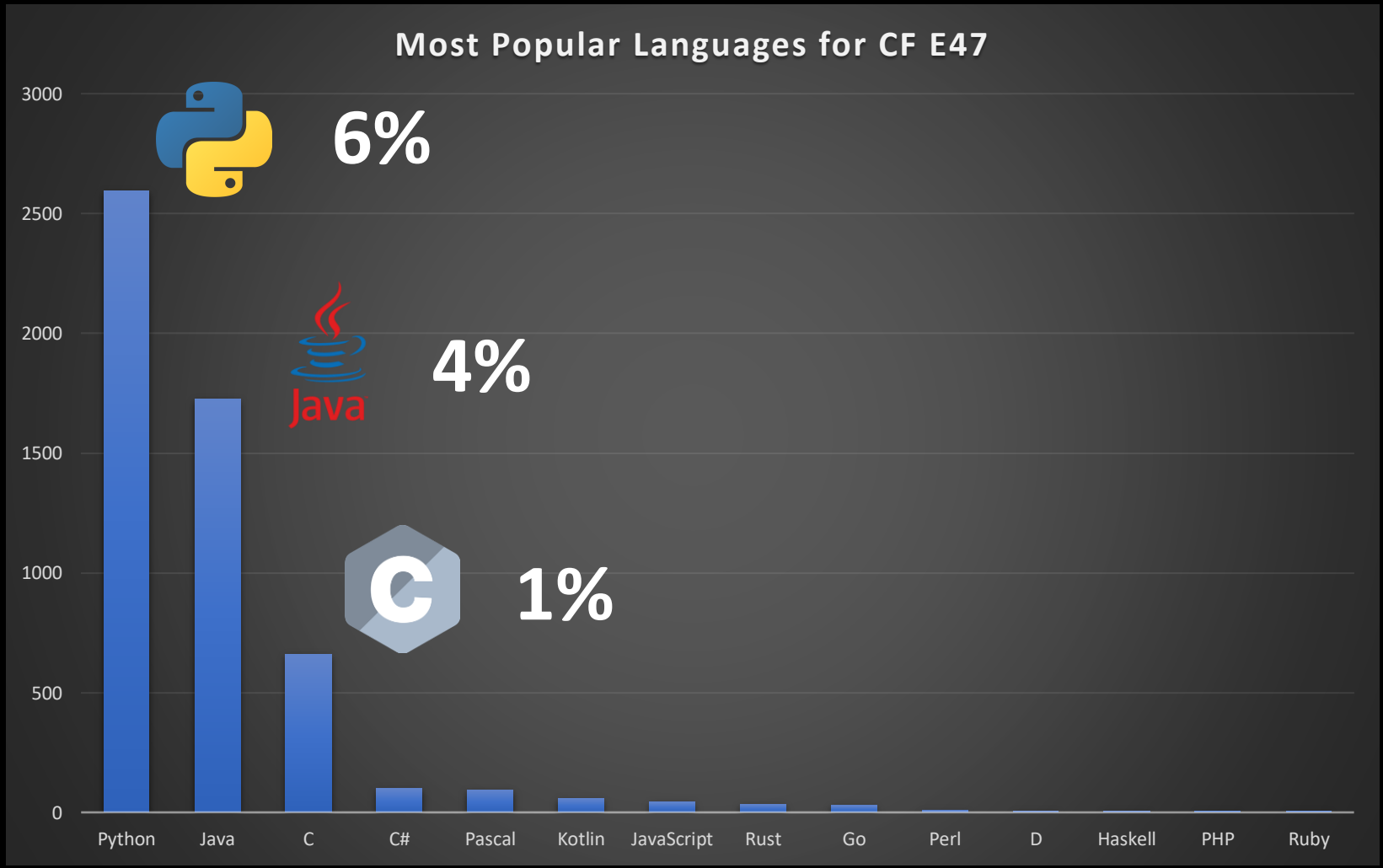
```python
mapping = {
    'GNU C++11'                 : 'C++',
    'GNU C++14'                 : 'C++',
    'GNU C++17'                 : 'C++',
    'MS C++'                    : 'C++',
    'MS C++ 2017'               : 'C++',
    'Clang++17 Diagnostics': 'C++',
    'Mono C#'                   : 'C#',
    'Python 2'                  : 'Python',
    'Python 3'                  : 'Python',
    # ...
```

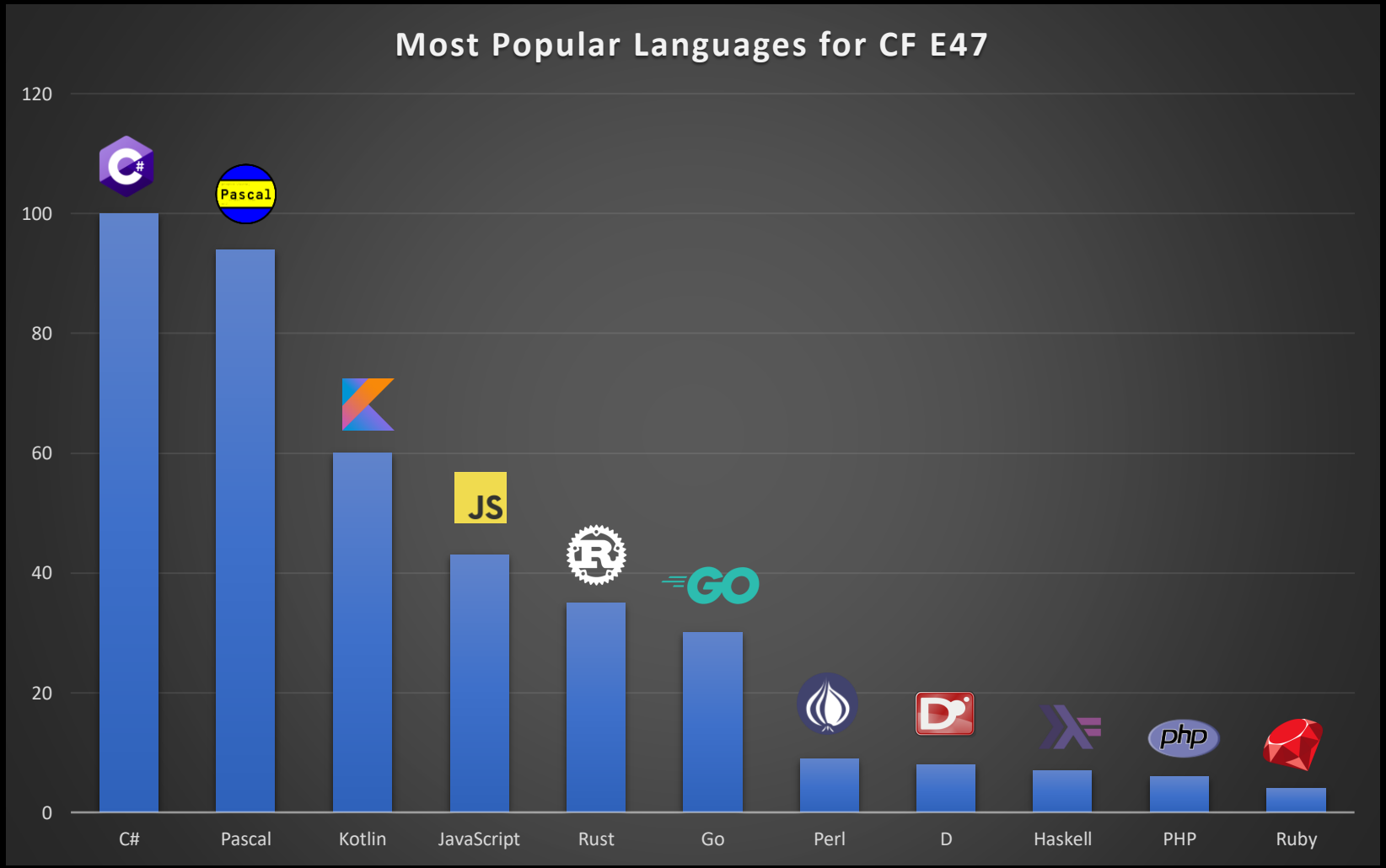
```python
import pandas as pd

mapping = { # ...

df = pd.read_csv('cf_ed_74.csv')

print(df.Lang
        .replace(mapping)
        .value_counts())
```

| | | | |
|---|---|---|---|
| pandas | **value_counts** | Series | Doc |
| RAPIDS (Python) | **value_counts** | Series | Doc |
| Clojure | **frequencies** | core | Doc |
| Racket | **frequencies** | list-utils | Doc |
| Python | **Counter*** | collections | Doc |
| Haskell | **count** | Data.List.Unique | Doc |

```
conorhoekstra@LAPTOP-GBS6IJDC:
C++            41422
Python          2595
Java            1728
C                660
C#               100
Pascal            94
Kotlin            60
JavaScript        43
Rust              35
Go                30
Perl               9
D                  8
Haskell            7
PHP                6
Ruby               4
Name: Lang, dtype: int64
```


Most Popular Languages for CF E47
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
89%
C++
Python
Java
C
C#
Pascal
Kotlin
JavaScript
Rust
Go
Perl
D
Haskell
PHP
Ruby


Most Popular Languages for CF E47
3000
2500
2000
1500
1000
500
0
6%
4%
1%
Python
Java
C
C#
Pascal
Kotlin
JavaScript
Rust
Go
Perl
D
Haskell
PHP
Ruby

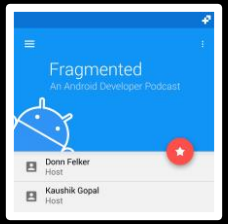

Most Popular Languages for CF E47
120
100
80
60
40
20
0
C#
Pascal
Kotlin
JavaScript
Rust
Go
Perl
D
Haskell
PHP
Ruby

```python
import pandas as pd

mapping = { # ...

df = pd.read_csv('cf_ed_74.csv')

print(df.Lang
        .replace(mapping)
        .value_counts())
```

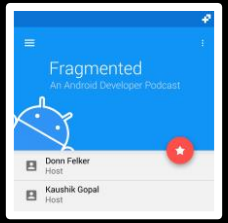
```python
import cudf

mapping = { # ...

df = cudf.read_csv('cf_ed_74.csv')

print(df.Lang
        .replace(mapping)
        .value_counts())
```


Special Episode: Books, Art, and Zombies: How to Survive in Today's World


Ep. 114: Credit Check - Capital One’s Kyle Nicholson on Modern Machine Learning in Finance




Ep. 193: Working from Home – Pandemic on hard mode




Ep. 194: Polyglot programmers


Ep. 87: Evolution, Intelligence, Simulation, and Memes

Ep. 114: Credit Check - Capital One’s Kyle Nicholson on Modern Machine Learning in Finance

~30x cheaper

100x faster

RAPIDS

https://rapids.ai

# 8 Coolest Python Programming Language Features

Jeremy Grifski Mar 3 *Originally published at therenegadecoder.com on Mar 03, 2020*

· 8 min read

#python #programming

https://dev.to/renegadecoder94/8-coolest-python-programming-language-features-58i9

list comprehensions
generator expressions
slice assignment
iterable unpacking
dictionary comprehensions

# Thank You

https://github.com/codereport/Talks/


Conor Hoekstra

code_report

codereport

# Questions?*

https://github.com/codereport/Talks/


Conor Hoekstra

code_report

codereport