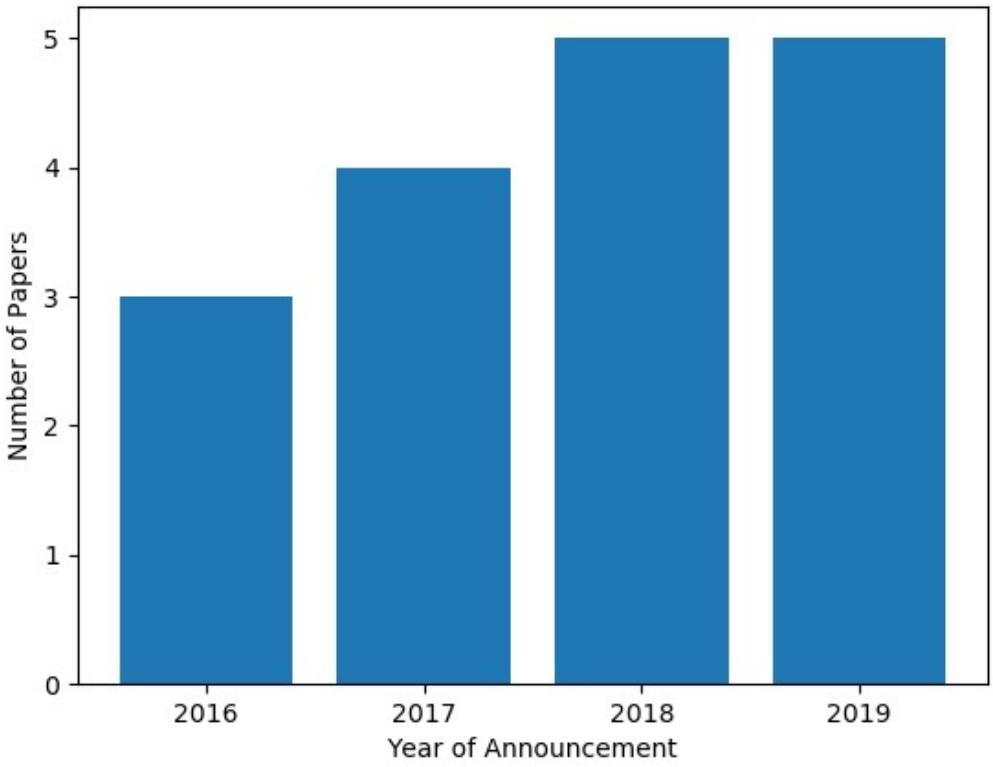
Programming Language Homework2 Report

張財實
資訊三乙
F74055047

Execution： $ python plhw2.py
[Out] Input Author :
[In] Nicholas Carlini
[Out: Couple of seconds]
[Author: Nicholas Carlini]
Abhibhav Garg : 1 times
Aleksander Madry : 1 times
Alexander Matyasko : 1 times
Alexander Ratner : 1 times
Alexandros Dimakis : 1 times
Alexey Kurakin : 2 times
Anish Athalye : 3 times
Aurko Roy : 1 times
Bill Dally : 1 times
Bryan Catanzaro : 1 times
Catherine Olsson : 1 times
Chang Liu : 1 times
Charles Elkan : 1 times
Chiyuan Zhang : 1 times
Cihang Xie : 1 times
Clark Barrett : 1 times
Colin Raffel : 1 times
Dan Alistarh : 1 times
David Berthelot : 1 times
David L. Dill : 1 times
David Wagner : 6 times
Dawn Song : 2 times
Dimitris Tsipras : 1 times
Eric Chung : 1 times
Fartash Faghri : 1 times
Florian Tramèr : 1 times
Furong Huang : 1 times
Garrison Cottrell : 1 times
Garth A. Gibson : 1 times
Gregory R. Ganger : 1 times
Grigori Fursin : 1 times
Gustavo Alonso : 1 times
Guy Katz : 1 times
Ian Goodfellow : 4 times
Inderjit S. Dhillon : 1 times
James Wei : 1 times
Jeff Dean : 1 times
Jens Behrmannn : 1 times
Jernej Kos : 1 times
Jonas Rauber : 2 times
Jonathan Uesato : 1 times

Joseph E. Gonzalez : 1 times
Justin Gottschlich : 1 times
Jörn-Henrik Jacobsen : 1 times
Karen Hambardzumyan : 1 times
Kim Hazelwood : 1 times
Lise Getoor : 1 times
Martin Jaggi : 1 times
Nicolas Papernot : 3 times
Paul Christiano : 1 times
Paul Hendricks : 1 times
Peter Bailis : 1 times
Phillip B. Gibbons : 1 times
Pradeep Dubey : 1 times
Reuben Feinman : 1 times
Rujun Long : 1 times
Ryan Sheatsley : 1 times
Sarah Bird : 1 times
Song Han : 1 times
Tom B. Brown : 1 times
Tom Brown : 1 times
Vahid Behzadan : 1 times
Warren He : 1 times
Wieland Brendel : 1 times
Willi Gierke : 1 times
Xinyun Chen : 1 times
Yao Qin : 1 times
Yash Sharma : 1 times
Yi-Lin Juang : 1 times
Yinpeng Dong : 1 times
Zhi Li : 1 times
Zhishuai Zhang : 1 times
Úlfar Erlingsson : 1 times



qt5ct: using qt5ct plugin
Only have 17 papers for author Nicholas Carlini //Only appear if the figure is closed

Figure – Need to close it by hand, not support for key

Code Explanation:

```
import urllib.request as urlreq
import re
import matplotlib.pyplot as plt
import math
```

Import needed packages for Homework2

- *Matplotlib* is a library to draw chart or graph in python
- *Math* (I used *ceil* in my implementation)

```
author = input("Input Author: ")
# author = "Aoki%2C+S"
# author = "Ian Goodfellow"
query_author = author.replace(" ", "+")
query_author = query_author.replace(".", "%2C")
# print(query_author)
# print(author)
```

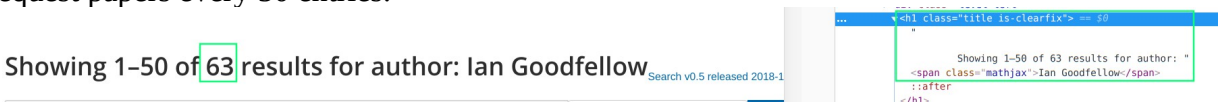
Input auther name, I did not use *stdin* all the time, for convenience hard code some author name for testing.
It is because of parameter query request, the URL doesn't support " " and ".". It has to be replaced by "+" and "%2C" respectively in order to get the correct response.

```
year_list = list()
co_author = list()

# First call to the author
url = "https://arxiv.org/search/?query=" + query_author + "&searchtype=author"
content = urlreq.urlopen(url)
html = content.read().decode("utf-8")
```

year_list is a list contain all the paper-year by query author
co_author is a list contain all the co-author by query author

```
# pattern for the entries
pattern_entry = "title is-clearfix[\s\S]*?Showing \d+&ndash;\d+ of \d+ results for author:
<span class=\"mathjax\">"
result_entry = re.findall(pattern_entry, html)

# Getting how many entries papers does this author has
entries = int(re.findall("\d+", result_entry[0].split("of")[1])[0])
print(entries)
```

By Default the page will show 50 paper entries. I have to record the entry number so I can request papers every 50 entries.



For *Ian Goodfellow*, the search result has 63 entries.
Even the entry is bellow 50, we still have to search the papers, so this step has to be done at least once.

```
# pattern of getting each arxiv
arxiv_pattern = "arxiv-result[\s\S]*?</li>"
arxiv_list = re.findall(arxiv_pattern, html)
```

We can start our page paper search on the first page.
The *DOM element* that encloses the entire paper is *<li class=”arvix-result”> ….. *.
Just get whatever inside the “li”.
arixiv_list contain the HTMLs that match the regex. (First page we have 50 entries, the *arxiv_list* will have 50 too.)

```
def processArxiv(arxivs, counter, year_list, co_author):
  for arxiv_html in arxivs:
    # authors
    author_pattern = "<a href=\"/search/[\s\S]*?\">[\s\S]*?</a>"
    author_list = re.findall(author_pattern, arxiv_html)
    authors_in_list = []
    this_paper_has_the_author = False
    # print(author_list)
    for a in author_list:
      a = re.findall(">[\s\S]*?<", a)[0]
      a = a.split(">")[1].split("<")[0]
      If a != author:
        if a[0] == " ":
          a = a.replace(" ", "")
         authors_in_list.append(a)
      elif a == author:
        this_paper_has_the_author = True
```

```
        if this_paper_has_the_author is True:
          counter = counter + 1
          co_author.extend(authors_in_list)
          # year
          year_pattern ="originally announced</span>[\s\S]*?</p>"
          year_html = re.findall(year_pattern, arxiv_html)[0]
          year = int(re.findall("\d+", year_html)[0])
          year_list.append(year)
          return counter, year_list, co_author
```

After all of this, we finally get list of HTMLs specifically include the info of a paper. We only interest in co-authors and year of announcement.
But it is a good practice to put parsing steps inside a function. Later on, this function can be reused by 51-63 entries(For author Ian Goodfellow at this instance) to generate our result.

This function is call "processArxiv"
Parameters:

- *arxivs* – list of paper HTML
- *counter* – to count the valid entry (for debug purpose, not important)
- *year_list* – list paper announce year by an author
- *co_authors* – co-authors of that author

I used call by value to pass *year_list* and *co_authors* to this function, and update themselves every next 50 entries.

Detail:

1. **Check the paper authors** – We have to ensure the author we search is valid. In some cases, **Ian Goodfellow** and **Ian Not Goodfellow** is not the same author, but the search result will treat them as the same when an author name has "Ian Goodfellow" inside the string anyway. We have to avoid this happening.
2. **",et al.(88 additional authors not shown)"** is not happening here – this text is not a hyperlink, so my regex on it won't match it. Just ignore it.
3. ***authors_in_list* and *this_paper_has_the_author*** – First is a list storing current paper's all authors, second is a flag tells whether the current paper is a valid paper. By doing so, we will get all the authors of an paper (to be discarded or to be extended), and a flag telling us what to do next.
4. **If the flag is False** – It means the current paper we are working on and what we get inside *author_in_list* are all useless, we will discard this iteration and go to the next iteration instead.
5. **If the flag is True** – If this_paper_has_the_author is True, the paper has the author we asked, the authors_in_list will became the co-authors. So we have to accumulate these useful authors (they are *co_authors*) by doing *co_authors.extend(authors_in_list).*
6. **Announce Year** – We can get the paper year when the paper is legit, so we process parsing year if and only if the flag is true. Other than *extend* the list, we have to use *append* to accumulate (*append* is for element, *extend* is list-to-list concatenatation)

```
counter, year_list, co_author = processArxiv(arxiv_list, counter, year_list, co_author)
```

Calling the function that we created the first time, and in the first page.
We will get *year_list* and *co_author.*

```
if entries > 50:
  start = 0
  # do it once and we done
  pages = math.ceil(entries / 50)
  for i in range(pages - 1):
    # do it the rest of your live
    url = "https://arxiv.org/search/?query=" + query_author +
"&searchtype=author&size=50&start=" + str(50 * (i+1))
    content = urlreq.urlopen(url)
    html = content.read().decode("utf-8")

    # pattern of getting each arxiv
    arxiv_pattern = "arxiv-result[\s\S]*?</li>"
    arxiv_list = re.findall(arxiv_pattern, html)

    counter, year_list, co_author = processArxiv(arxiv_list, counter, year_list, co_author)
```

Now we get to use the entry number to determine how many times we have run a new request for the rest of pages (next 50s).
Like 63, we have to do 1 more time page request, 100 is also 1, 101 is 2.
We can notice that if we want to go for next pages, we just have to specify, size of query, *size*, and start of entry, *start* in request URL.

1-50 size 50, start 0
51-63 size 50, start 50

Here we go, we need to call the processArxiv function every new page. All the final accumulated *year_list* and *co_author* will be done by now.

```
co_author_dict = {}
for i in co_author:
  if i not in co_author_dict:
    co_author_dict[i] = 0
    co_author_dict[i] += 1

sortedCoauthor = sorted(co_author_dict.keys())

for i in sortedCoauthor:
  print("{:<25s}: {:d} times".format(i, co_author_dict[i]))
```

Q2 Because we use concatenation, some of the author name are repeated, this is useful, we have to count their occurrence.
Sort and print in desired output format.

```
result_year = {}
for i in year_list:
if i not in result_year:
result_year[i] = 0
result_year[i] += 1
plt.figure()
plt.bar(result_year.keys(),result_year.values())
plt.show()
```

Q1 Here we go, we do it the same way to *year_list*, but this time no sort needed. Plot the bar and the *xticks* will be sorted themselves!

Review :
It is not hard to do, but has certain degree of complexity.
Once I used the pandas library, it is because it provides method like groupby and count, much simpler.