Homework 12

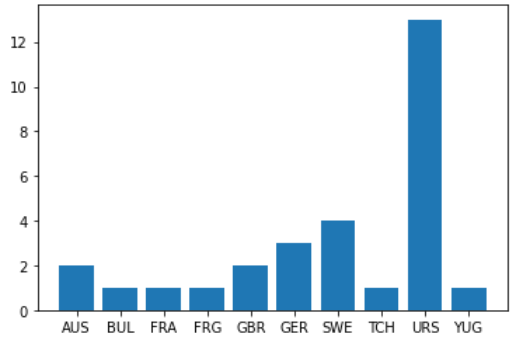
Due 4PM Nov 30, 2020

Problem 1: Women's 800 Meter

Which countries have done best at the Women's 800 Meter?

Gather the data from the World Records CSV, use a Dictionary to count the records, and create a bar chart showing the relative number of records per country.Sort the countries alphabetically, and make sure we can read the country names.

In [23]:
```
import pandas as pd
import matplotlib.pyplot as plt
filename = "WorldRecords.csv"

# read in data
df = pd.read_csv(filename)

# filter to women's 800m
df2 = df[(df['Event'] =='Womens 800m')]

# create dict to count records
nats = {}
for index, row in df2.iterrows():
    if row['Nationality'] not in nats:
        nats[row['Nationality']] = 1
    else:
        nats[row['Nationality']] += 1

# create & draw chart
nats = sorted(nats.items())
x = [k[0] for k in nats]
y = [k[1] for k in nats]

plt.bar(x,y)
plt.show()
```



Problem 2: Regular Expressions

We have used Beautiful Soup to scrape a website.Let's see what we can do with just urlib and Regular Expressions Take the DCE website, and find all the links. (Be sure to compare notes with Beautiful Soup)

In [24]:
```
import urllib.request
import string
import re

def find_links(url):
    """Returns the first URL and link txt on page"""

    # read in url text
    with urllib.request.urlopen(url) as f:
        text = f.read().decode('utf-8')

    re_links = re.findall(r'<a\s+.*/a>', text)
    return re_links
```

Unit Test

In [25]:
```
website = 'https://www.extension.harvard.edu'
```

In [26]:
```
results = find_links(website)
print("Number of links:",len(results),"\n")
for link in results:
   print(link)
```

```
Number of links: 59

<a href="#main-menu" class="skip">Jump to navigation</a>
<a href="#main-content" class="skip">Skip to Main Content</a>
<a class="topbar__link" href="https://www.harvard.edu">Harvard.edu</a>
<a href="https://www.extension.harvard.edu">Harvard Extension School</a>
<a href="https://www.summer.harvard.edu">Harvard Summer School</a>
<a
              href="https://www.extension.harvard.edu/professional-development">Professional Development
</a>
<a href="https://www.extension.harvard.edu/hilr">Learning In Retirement</a>
<a href="https://alumni.extension.harvard.edu/">Extension Alumni Association</a>
<a class="header__mobile-menu ir i-hamburger" data-grunticon-embed href="">Menu</a>
<a href="/academics">Academics</a>
<a href="/registration-admissions" title="Registration &amp; Admissions">Registration &amp; Admissions</
a>
<a href="/resources-policies">Resources &amp; Policies</a>
<a href="https://blog.dce.harvard.edu/extension" title="">Blog</a>
<a href="/request-information" title="">Get Info</a>
<a href="/about-us" title="">About</a>
<a href="/academic-calendar" title="">Calendar</a>
<a href="/completing-your-degree" title="">For Degree Candidates</a>
<a href="/academics/online-campus-courses" title="Link to courses">Courses</a>
<a href="/faculty-directory" title="">Faculty Directory</a>
<a href="https://www.extension.harvard.edu/login" title="">LOGIN</a>
<a id="main-content" tabindex="-1"></a>
<a href=https://www.extension.harvard.edu/covid-19-updates>latest COVID-19 news from Harvard Extension S
chool</a>
<a class="i-right-arrow" href="/academics/graduate-certificates">Graduate Certificates</a>
<a class="i-right-arrow" href="/academics/graduate-degrees">Master&rsquo;s Degrees</a>
<a class="i-right-arrow" href="/academics/academic-gap-year">Academic Gap Year</a>
<a class="i-right-arrow" href="/academics/bachelor-liberal-arts-degree">Bachelor&rsquo;s Degree</a>
<a class="i-right-arrow" href="/academics/undergraduate-certificates">Undergraduate Certificates</a>
<a class="i-right-arrow" href="/joint-undergraduate-graduate-program">Joint Undergraduate &amp; Graduate
Programs</a>
<a class="i-right-arrow" href="/course-catalog">Course Catalog</a>
<a class="i-right-arrow" href="/course-catalog/courses?subjects=Medical%20Sciences">Medical Sciences Cou
rses</a>
<a class="i-right-arrow" href="/academics/premedical-program">Premedical Program</a>
<a class="i-right-arrow" href="https://www.extension.harvard.edu/professional-development">Noncredit Pro
fessional Development Programs</a>
<a class="i-right-arrow" href="https://www.extension.harvard.edu/hilr">Learn about the program</a>
<a class="button-link" href="/about-us/why-hes">Find out why</a>
<a class="student-name h3" href="/about-us/peter-thielen">Peter Thielen</a>
<a class="student-name h3" href="/about-us/renee-m-greene">Renee M. Greene</a>
<a class="student-name h3" href="/about-us/diane-smith">Diane Smith</a>
<a class="button btn-outline-primary" href="/about-us/student-stories">Meet Other Alumni &amp; Students
<i class="far fa-long-arrow-alt-right"></i></a>
<a class="h3" href="https://harvardmagazine.com/2020/10/calling-the-2020-election" target="_blank">Calli
ng the 2020 Election</a>
<a class="h3" href="https://www.thecrimson.com/article/2020/9/18/extension-school-new-programs/" target
="_blank">Harvard Extension School Unveils New Academic Gap Year, Undergraduate Certificate Programs</a>
<a class="h3" href="https://www.educationdive.com/news/how-colleges-with-hybrid-instruction-this-fall-ca
n-support-online-students/582141/" target="_blank">How colleges with hybrid instruction this fall can su
pport online students</a>
<a class="h3" href="/about-us/press-announcements/michael-fabiano-joins-haa-board-directors" >Michael Fa
biano Joins HAA Board of Directors</a>
<a class="h3" href="https://blog.dce.harvard.edu/extension/announcing-new-graduate-and-undergraduate-cer
tificates-online-courses-for-2020-21" target="_blank">What&#039;s New for 2020-21</a>
<a class="h3" href="https://www.fas.harvard.edu/news/new-dean-division-continuing-education" target="_bl
ank">New Dean of the Division of Continuing Education</a>
<a class="button-link" href="https://blog.dce.harvard.edu/extension/6-strategies-for-staying-productive-
during-the-covid-19-crisis">Read the blog post</a>
<a class="button-link" href="https://www.extension.harvard.edu/course-catalog">Course Catalog</a>
<a class="button-link" href="https://www.extension.harvard.edu/professional-development/programs/buildin
g-organizational-cultures-framework-leaders-online">Learn More</a>
<a href="/contact-us" title="" class="menu__link">Contact Us</a>
<a href="/forms" title="" class="menu__link">Forms</a>
<a href="/website-archives" title="" class="menu__link">Archives</a>
<a href="https://twitter.com/HarvardEXT" title="" class="menu__link i-social-twitter">Twitter</a>
<a href="https://www.facebook.com/HarvardExtension" title="" class="menu__link i-social-facebook">Facebo
ok</a>
<a href="https://www.youtube.com/user/HarvardExtension" title="" class="menu__link i-social-youtube">You
Tube</a>
<a href="https://www.instagram.com/harvardextension/" title="" class="menu__link i-social-instagram">Ins
tagram</a>
<a class="menu__link" href="/privacy-policy" title="">Privacy</a>
<a class="menu__link" href="/resources-policies/accessibility-services-office-aso" title="">Accessibilit
y</a>
<a class="menu__link" href="/resources-policies/resources/rights-regulations" title="">Rights &amp; Regu
```

lations
<a class="menu__link" href="https://accessibility.huit.harvard.edu/digital-accessibility-policy" title
="">Digital Accessibility Policy
Cookie Settings

In [27]:
```
import requests
from bs4 import BeautifulSoup

"prettify print the html of a given url"

url = "https://www.extension.harvard.edu"
html_content = requests.get(url).text
soup = BeautifulSoup(html_content, 'html.parser')
pretty_soup = soup.prettify()

links = soup.find_all("a")


print("Number of links:",len(links),"\n")
for x in links:
    print(x)
```

```
Number of links: 62

<a class="skip" href="#main-menu">Jump to navigation</a>
<a class="skip" href="#main-content">Skip to Main Content</a>
<a class="topbar__logo i-harvard-logo ir" href="https://dce.harvard.edu" target="_blank">
        Harvard Division of Continuing Education
      </a>
<a class="topbar__link" href="https://www.harvard.edu">Harvard.edu</a>
<a href="https://www.extension.harvard.edu">Harvard Extension School</a>
<a href="https://www.summer.harvard.edu">Harvard Summer School</a>
<a href="https://www.extension.harvard.edu/professional-development">Professional Development</a>
<a href="https://www.extension.harvard.edu/hilr">Learning In Retirement</a>
<a href="https://alumni.extension.harvard.edu/">Extension Alumni Association</a>
<a class="header__mobile-menu ir i-hamburger" data-grunticon-embed="" href="">Menu</a>
<a class="header__logo i-hes-logo" href="/" id="logo" rel="home" title="Home">
<noscript><img alt="Home" class="header__logo-image" src="https://www.extension.harvard.edu/sites/extens
ion.harvard.edu/themes/extension/logo.png"/></noscript>
</a>
<a class="header__site-link" href="/" rel="home" title="Home"><span>Harvard Extension School</span></a>
<a href="/academics">Academics</a>
<a href="/registration-admissions" title="Registration &amp;amp; Admissions">Registration &amp;amp; Admissions</
a>
<a href="/resources-policies">Resources &amp;amp; Policies</a>
<a href="https://blog.dce.harvard.edu/extension" title="">Blog</a>
<a href="/request-information" title="">Get Info</a>
<a href="/about-us" title="">About</a>
<a href="/academic-calendar" title="">Calendar</a>
<a href="/completing-your-degree" title="">For Degree Candidates</a>
<a href="/academics/online-campus-courses" title="Link to courses">Courses</a>
<a href="/faculty-directory" title="">Faculty Directory</a>
<a href="https://www.extension.harvard.edu/login" title="">LOGIN</a>
<a id="main-content" tabindex="-1"></a>
<a href="https://www.extension.harvard.edu/covid-19-updates">latest COVID-19 news from Harvard Extension
School</a>
<a class="i-right-arrow" href="/academics/graduate-certificates">Graduate Certificates</a>
<a class="i-right-arrow" href="/academics/graduate-degrees">Master’s Degrees</a>
<a class="i-right-arrow" href="/academics/academic-gap-year">Academic Gap Year</a>
<a class="i-right-arrow" href="/academics/bachelor-liberal-arts-degree">Bachelor’s Degree</a>
<a class="i-right-arrow" href="/academics/undergraduate-certificates">Undergraduate Certificates</a>
<a class="i-right-arrow" href="/joint-undergraduate-graduate-program">Joint Undergraduate &amp;amp; Graduate
Programs</a>
<a class="i-right-arrow" href="/course-catalog">Course Catalog</a>
<a class="i-right-arrow" href="/course-catalog/courses?subjects=Medical%20Sciences">Medical Sciences Cou
rses</a>
<a class="i-right-arrow" href="/academics/premedical-program">Premedical Program</a>
<a class="i-right-arrow" href="https://www.extension.harvard.edu/professional-development">Noncredit Pro
fessional Development Programs</a>
<a class="i-right-arrow" href="https://www.extension.harvard.edu/hilr">Learn about the program</a>
<a class="button-link" href="/about-us/why-hes">Find out why</a>
<a class="student-name h3" href="/about-us/peter-thielen">Peter Thielen</a>
<a class="student-name h3" href="/about-us/renee-m-greene">Renee M. Greene</a>
<a class="student-name h3" href="/about-us/diane-smith">Diane Smith</a>
<a class="button btn-outline-primary" href="/about-us/student-stories">Meet Other Alumni &amp;amp; Students
<i class="far fa-long-arrow-alt-right"></i></a>
<a class="h3" href="https://harvardmagazine.com/2020/10/calling-the-2020-election" target="_blank">Calli
ng the 2020 Election</a>
<a class="h3" href="https://www.thecrimson.com/article/2020/9/18/extension-school-new-programs/" target
="_blank">Harvard Extension School Unveils New Academic Gap Year, Undergraduate Certificate Programs</a>
<a class="h3" href="https://www.educationdive.com/news/how-colleges-with-hybrid-instruction-this-fall-ca
n-support-online-students/582141/" target="_blank">How colleges with hybrid instruction this fall can su
pport online students</a>
<a class="h3" href="/about-us/press-announcements/michael-fabiano-joins-haa-board-directors">Michael Fab
iano Joins HAA Board of Directors</a>
<a class="h3" href="https://blog.dce.harvard.edu/extension/announcing-new-graduate-and-undergraduate-cer
tificates-online-courses-for-2020-21" target="_blank">What's New for 2020-21</a>
<a class="h3" href="https://www.fas.harvard.edu/news/new-dean-division-continuing-education" target="_bl
ank">New Dean of the Division of Continuing Education</a>
<a class="button-link" href="https://blog.dce.harvard.edu/extension/6-strategies-for-staying-productive-
during-the-covid-19-crisis">Read the blog post</a>
```

```
<a class="button-link" href="https://www.extension.harvard.edu/course-catalog">Course Catalog</a>
<a class="button-link" href="https://www.extension.harvard.edu/professional-development/programs/buildin
g-organizational-cultures-framework-leaders-online">Learn More</a>
<a class="menu__link" href="/contact-us" title="">Contact Us</a>
<a class="menu__link" href="/forms" title="">Forms</a>
<a class="menu__link" href="/website-archives" title="">Archives</a>
<a class="menu__link i-social-twitter" href="https://twitter.com/HarvardEXT" title="">Twitter</a>
<a class="menu__link i-social-facebook" href="https://www.facebook.com/HarvardExtension" title="">Facebo
ok</a>
<a class="menu__link i-social-youtube" href="https://www.youtube.com/user/HarvardExtension" title="">You
Tube</a>
<a class="menu__link i-social-instagram" href="https://www.instagram.com/harvardextension/" title="">Ins
tagram</a>
<a class="menu__link" href="/privacy-policy" title="">Privacy</a>
<a class="menu__link" href="/resources-policies/accessibility-services-office-aso" title="">Accessibilit
y</a>
<a class="menu__link" href="/resources-policies/resources/rights-regulations" title="">Rights &amp;amp; Regu
lations</a>
<a class="menu__link" href="https://accessibility.huit.harvard.edu/digital-accessibility-policy" title
="">Digital Accessibility Policy</a>
<a class="menu__link ot-sdk-show-settings" href="#" title="">Cookie Settings</a>
```

Compare your program with the results from Beautiful Soup

Do you get the same number of links? If not:

1) How many do you miss?

2) Can you explain why you miss them?

3) Can you fix it?

In [28]:
```
# I do not get the same number of links. I miss three using RE: the Harvard Division of Continuing Educat
# link, the second Harvard Extension School link, and the HES logo that links back to the home page.

# Could not quite figure out a reliable way to use RE to get the links I first miss,
# without then missing other links I didn't miss the first time.

# I surmise that the moral of the story is: don't use RE for URLs, use libraries that are built
# specifically for that sort of thing - such as Beautiful Soup
```

Problem 3: File Name Generator

Write a Generator that takes a directory, a file extension, and, optionally, a file size, and then yields a stream of tuples, (path, filename) so that path/filename is a legal path to a file that meets the conditions.

Use os.walk(dir) to create a generator that gives all files and directories below dir. Call this generator, and yield files (not directories) with the right extension and a file size greater than the given size.

We have three unit tests: demonstrate that you can walk recursivly through two or more directories, and that you can filter by file extension and filter by extension and by size.

In [29]:
```
import os

def find_files_gen(path, filename, filesize=0):
    matches = []
    for root,dir, files in os.walk(path):
        for f in files:
            path = os.path.join(root, f)
            size = os.stat(path).st_size
            if filename in f and size > filesize:
                yield(root, f)


gen = find_files_gen('..', 'py')
print(next(gen))
print(next(gen))
print(next(gen))
print(next(gen))
print(next(gen))
```

```
('../Day08', 'koch_snowflake.py')
('../Day08', 'dna_complement.py')
('../Day08', 'Koch.py')
('../Day08', 'Day8.ipynb')
('../Day08', 'Time1.py')
```

Unit Test

In [30]:
```
# Show recursive search.  Make sure we can see at least two directories of files
gen = find_files_gen('..', 'py',35000)

for path, filename in gen:
    print(path, filename)
```

```
../Day08 Day8.ipynb
../Day08 CopyBox.jpg
../Day08/.ipynb_checkpoints Day8-checkpoint.ipynb
../Day07 Day7.ipynb
../Day11 Homework11_SUrista.ipynb
../Day11/.ipynb_checkpoints Homework11-checkpoint.ipynb
../Day11/.ipynb_checkpoints Homework11_original-checkpoint.ipynb
../Day11/.ipynb_checkpoints Homework11_SUrista-checkpoint.ipynb
../Day11/.ipynb_checkpoints Homework11SUrista-checkpoint.ipynb
../Day02 Day2.ipynb
../.ipynb_checkpoints Homework12_Surista-checkpoint.ipynb
../Day06 pywiki.txt
../Day06 pytext.txt
../Day06 Day6.ipynb
../Day12 Homework12_Surista.ipynb
../Day12/.ipynb_checkpoints Homework12_Surista-checkpoint.ipynb
../Day10 Homework10_S_Urista.ipynb
../Day10 BeautifulSoup-Lena.ipynb
../Day10 Homework10.ipynb
../Day10/.ipynb_checkpoints Homework10-checkpoint.ipynb
../Day10/.ipynb_checkpoints Homework10 (4)-checkpoint.ipynb
../Day10/.ipynb_checkpoints 2016_US_County_Level_Presidential_Results-checkpoint.ipynb
../Day04 Day4 (1).ipynb
../Day04/.ipynb_checkpoints Day4-checkpoint.ipynb
../Day04/.ipynb_checkpoints Day4 (1)-checkpoint.ipynb
../Day09 Iterator (1).ipynb
../Day09/.ipynb_checkpoints Day10-checkpoint.ipynb
../Day09/.ipynb_checkpoints Iterator-checkpoint.ipynb
../Day05 Day5.ipynb
```

In [31]:
```
# Show all notebooks in this directory
gen = find_files_gen('..', '.ipynb')

for path, filename in gen:
    print(path, filename)
```

```
../Day08 Day8.ipynb
../Day08 Homework8FirstDraft.ipynb
../Day08 Homework8 (1).ipynb
../Day08 While.ipynb
../Day08 Homework8.ipynb
../Day08/.ipynb_checkpoints Homework8 (1)-checkpoint.ipynb
../Day08/.ipynb_checkpoints Homework8FirstDraft-checkpoint.ipynb
../Day08/.ipynb_checkpoints Homework8-checkpoint.ipynb
../Day08/.ipynb_checkpoints Day8-checkpoint.ipynb
../Day07 Homework7_SUrista.ipynb
../Day07 Day7.ipynb
../Day07 Solution6.ipynb
../Day07/.ipynb_checkpoints Solution6-checkpoint.ipynb
../Day07/.ipynb_checkpoints Homework7_SUrista-checkpoint.ipynb
../Day11 Homework11_SUrista.ipynb
../Day11 Homework11 _original.ipynb
../Day11/.ipynb_checkpoints Homework11-checkpoint.ipynb
../Day11/.ipynb_checkpoints Homework11_original-checkpoint.ipynb
../Day11/.ipynb_checkpoints Homework11_SUrista-checkpoint.ipynb
../Day11/.ipynb_checkpoints Homework11SUrista-checkpoint.ipynb
../Day02 HW2Share.ipynb
../Day02 Homework2.ipynb
../Day02 Day2.ipynb
../Day03 HW2Share_sol.ipynb
../Day03 Debug.ipynb
../Day03 Day3.ipynb
../Day03 HW3Share.ipynb
../Day03/.ipynb_checkpoints HW2Share_sol-checkpoint.ipynb
../.ipynb_checkpoints Homework12_Surista-checkpoint.ipynb
../Day06 Homework6.ipynb
../Day06 Day6.ipynb
../Day06 HW5Share (2).ipynb
../Day06/.ipynb_checkpoints Homework6_SUrista-checkpoint.ipynb
../Day06/.ipynb_checkpoints Homework6_SUrista_new-checkpoint.ipynb
../Day01 Homework1.ipynb
../Day01 E7 Day1.ipynb
../Day01 Homework_01.ipynb
../Day01 HW1Share_solutions.ipynb
../Day01 Day1.ipynb
../Day01 HW1Share.ipynb
../Day12 Homework12_Surista.ipynb
../Day12 Pickle.ipynb
../Day12 Anagrams.ipynb
../Day12 Repeats.ipynb
../Day12 Homework12_original.ipynb
../Day12/.ipynb_checkpoints Homework12 (5)-checkpoint.ipynb
../Day12/.ipynb_checkpoints Repeats-checkpoint.ipynb
../Day12/.ipynb_checkpoints Homework12 (1)-checkpoint.ipynb
../Day12/.ipynb_checkpoints Homework12 (6)-checkpoint.ipynb
../Day12/.ipynb_checkpoints Homework12_original-checkpoint.ipynb
../Day12/.ipynb_checkpoints Homework12_Surista-checkpoint.ipynb
../Day10 Homework10_S_Urista.ipynb
../Day10 BeautifulSoup-Lena.ipynb
../Day10 Scipy Tutorial.ipynb
../Day10 Homework10_original.ipynb
../Day10 Homework10.ipynb
../Day10/.ipynb_checkpoints Homework10-checkpoint.ipynb
../Day10/.ipynb_checkpoints Homework10 (3)-checkpoint.ipynb
../Day10/.ipynb_checkpoints Homework10 (4)-checkpoint.ipynb
../Day10/.ipynb_checkpoints 2016_US_County_Level_Presidential_Results-checkpoint.ipynb
../Day10/.ipynb_checkpoints Homework10_original-checkpoint.ipynb
../Day04 PyCharmVEnv.ipynb
../Day04 Day4 (1).ipynb
../Day04 Homework4.ipynb
../Day04 HW4Share.ipynb
../Day04/.ipynb_checkpoints Day4-checkpoint.ipynb
../Day04/.ipynb_checkpoints Day4 (1)-checkpoint.ipynb
../Day09 Homework8Solutions.ipynb
../Day09 Iterator (1).ipynb
../Day09 Assignment_9.ipynb
../Day09 HW9Share.ipynb
../Day09 Assignment_9_original.ipynb
../Day09/.ipynb_checkpoints Day10-checkpoint.ipynb
../Day09/.ipynb_checkpoints HW9Share-checkpoint.ipynb
../Day09/.ipynb_checkpoints SourceControl-checkpoint.ipynb
../Day09/.ipynb_checkpoints Assignment_9 (1)-checkpoint.ipynb
../Day09/.ipynb_checkpoints Homework8Solutions-checkpoint.ipynb
../Day09/.ipynb_checkpoints Iterator-checkpoint.ipynb
../Day09/.ipynb_checkpoints Assignment_9 (2)-checkpoint.ipynb
../Day09/.ipynb_checkpoints Assignment_9-checkpoint.ipynb
../Day05 HW4Share (1).ipynb
```

```
../Day05 Day5.ipynb
../Day05 homework5.ipynb
../Day05 HW5Share.ipynb
../Day05 Solutions4 (1).ipynb
../Day05/.ipynb_checkpoints Day5-checkpoint.ipynb
```

In [32]:
```
# Show all notebooks in this directory with at least **25K** bytes
gen = find_files_gen('..', '.ipynb', 25000)

for path, filename in gen:
    print(path, filename)
```

```
../Day08 Day8.ipynb
../Day08 While.ipynb
../Day08 Homework8.ipynb
../Day08/.ipynb_checkpoints Day8-checkpoint.ipynb
../Day07 Day7.ipynb
../Day11 Homework11_SUrista.ipynb
../Day11/.ipynb_checkpoints Homework11-checkpoint.ipynb
../Day11/.ipynb_checkpoints Homework11_original-checkpoint.ipynb
../Day11/.ipynb_checkpoints Homework11_SUrista-checkpoint.ipynb
../Day11/.ipynb_checkpoints Homework11SUrista-checkpoint.ipynb
../Day02 Day2.ipynb
../Day03 Day3.ipynb
../.ipynb_checkpoints Homework12_Surista-checkpoint.ipynb
../Day06 Homework6.ipynb
../Day06 Day6.ipynb
../Day12 Homework12_Surista.ipynb
../Day12/.ipynb_checkpoints Homework12_Surista-checkpoint.ipynb
../Day10 Homework10_S_Urista.ipynb
../Day10 BeautifulSoup-Lena.ipynb
../Day10 Homework10.ipynb
../Day10/.ipynb_checkpoints Homework10-checkpoint.ipynb
../Day10/.ipynb_checkpoints Homework10 (3)-checkpoint.ipynb
../Day10/.ipynb_checkpoints Homework10 (4)-checkpoint.ipynb
../Day10/.ipynb_checkpoints 2016_US_County_Level_Presidential_Results-checkpoint.ipynb
../Day04 Day4 (1).ipynb
../Day04/.ipynb_checkpoints Day4-checkpoint.ipynb
../Day04/.ipynb_checkpoints Day4 (1)-checkpoint.ipynb
../Day09 Iterator (1).ipynb
../Day09 Assignment_9.ipynb
../Day09/.ipynb_checkpoints Day10-checkpoint.ipynb
../Day09/.ipynb_checkpoints Iterator-checkpoint.ipynb
../Day09/.ipynb_checkpoints Assignment_9-checkpoint.ipynb
../Day05 Day5.ipynb
../Day05/.ipynb_checkpoints Day5-checkpoint.ipynb
```

Problem 4: Sorting Employees

We wish to take an unordered list of Employees, and get a list sorted by Company and Id.

Everyone who works at 'Springfield Department of Motor Vehicles' should be in one group. Everyone who works at 'Springfield Nuclear Power' would be in another group, later in the list, and everyone who works from the Mafia would be in a group earlier in the list. Within each group, we want to see the low ID numbers before this high ones.

For this problem, we do not want you to write a sorting program. You will use Python's sort.
You just need to define the magic method dunder lt(), less than, for the class Employee.

Once you have defined dunder lt(), calling Python's sorted() on a list of Employees will return a sorted list.

Add to the cell below

In [33]:
```
class Person:

    def __init__(self, first, last):
        self.firstname = first.capitalize()
        self.lastname = last.capitalize()

    def __str__(self):
        return self.firstname + " " + self.lastname


class Employee(Person):

    def __init__(self, first, last, company, id):
        # Call Superclass to set common information
        super().__init__(first, last)
        self.id = id
        self.company = company

    def __str__(self):
        # Call Superclass to dispaly common information
        return super().__str__() + ", " + str(self.id) + ' at ' + self.company

    def __lt__(self, other):
        "Is self less than other?"

        if not isinstance(other, Employee):
            return False
        return (self.company, self.id) < (other.company, other.id)
```

Unit Test

In [34]:
```
lst = [
    Employee('Homer', 'Simpson', 'Springfield Nuclear Power', 1005),
    Employee('Barney', 'Gumble', 'Plow King', 1),
    Employee('Clancy', 'Wiggum', 'Police Department', 1),
    Employee('Edna', 'Krabapple', 'Springfield Elementary School', 39),
    Employee('Seymour', 'Skinner', 'Springfield Elementary School', 1),
    Employee('Charles', 'Burns', 'Springfield Nuclear Power', 1),
    Employee('Waylon', 'Smithers', 'Springfield Nuclear Power', 2),
    Employee('Patty', 'Bouvier', 'Springfield Department of Motor Vehicles', 39),
    Employee('Selma', 'Bouvier', 'Springfield Department of Motor Vehicles', 38),
    Employee('Robert', 'Terwilliger', 'Channel 6', 31),
    Employee('Herschel', 'Krustofsky', 'Channel 6', 2),
    Employee('Lois', 'Pennycandy', 'Channel 6', 46),
    Employee('Johnny', 'Cevasco', 'Mafia', 2),
    Employee('Fat', 'Tony', 'Mafia', 1),
    Employee('Max', 'Legman', 'Mafia', 3 ),
    Employee('Louie', 'Walters', 'Mafia', 4)
    ]

for emp in lst:
    print(emp)

print('==========================')

# Sort the people
lst = sorted(lst)

# Now check that the list is sorted
for first, second in zip(lst[:-1], lst[1:]):
    assert (first.company, first.id) < (second.company, second.id)

for emp in lst:
    print(emp)

print("\n\tSuccess!")
```

```
Homer Simpson, 1005 at Springfield Nuclear Power
Barney Gumble, 1 at Plow King
Clancy Wiggum, 1 at Police Department
Edna Krabapple, 39 at Springfield Elementary School
Seymour Skinner, 1 at Springfield Elementary School
Charles Burns, 1 at Springfield Nuclear Power
Waylon Smithers, 2 at Springfield Nuclear Power
Patty Bouvier, 39 at Springfield Department of Motor Vehicles
Selma Bouvier, 38 at Springfield Department of Motor Vehicles
Robert Terwilliger, 31 at Channel 6
Herschel Krustofsky, 2 at Channel 6
Lois Pennycandy, 46 at Channel 6
Johnny Cevasco, 2 at Mafia
Fat Tony, 1 at Mafia
Max Legman, 3 at Mafia
Louie Walters, 4 at Mafia
==========================
Herschel Krustofsky, 2 at Channel 6
Robert Terwilliger, 31 at Channel 6
Lois Pennycandy, 46 at Channel 6
Fat Tony, 1 at Mafia
Johnny Cevasco, 2 at Mafia
Max Legman, 3 at Mafia
Louie Walters, 4 at Mafia
Barney Gumble, 1 at Plow King
Clancy Wiggum, 1 at Police Department
Selma Bouvier, 38 at Springfield Department of Motor Vehicles
Patty Bouvier, 39 at Springfield Department of Motor Vehicles
Seymour Skinner, 1 at Springfield Elementary School
Edna Krabapple, 39 at Springfield Elementary School
Charles Burns, 1 at Springfield Nuclear Power
Waylon Smithers, 2 at Springfield Nuclear Power
Homer Simpson, 1005 at Springfield Nuclear Power

        Success!
```

Problem 5: Finding Repeats

DNA has a great deal of structure. DNA often contains repeats: this is a fascinating area that we are not going to explore. Investigate 'transposons'.

Write a program that finds the longest repeat in a sequence of DNA stored in a FASTA file.

There will be a single string of DNA in the file. The first line has a description of the contents, while the remainder is a string of A, C, G, and T with line breaks. Be sure to remove the line breaks.

Here is a sample run on pKLMF-FX.fasta

```
10089
(5535, 5541, 15)
CACGGGCACGGGCAC
CACGGGCACGGGCAC
CPU times: user 191 ms, sys: 2.49 ms, total: 193 ms
Wall time: 193 ms
```

In [1]:
```
# Read contents of fasta file with a single sequence
# Skip the first line, and return a string holding the contents

def read_fasta_file(filename: str) -> str:
    with open(filename, 'r') as f:
        temp = [line.strip() for line in f]
        seq = ''.join(temp[1:])

    return seq
```

In [2]:
```
# Take a string and look for the longest repeat
# Return a tuple: (pos1, pos2, length) or None if there are no repeats
#    pos1 != pos2 and text[pos1:pos1+length)] == text[pos2:pos2+length]

from collections import defaultdict


def longest_repeat(text):

    # create the initial dictionary with all length-2 repeats

    cntr = 2  # size of initial substring length we look for
    d = defaultdict(list)
    for i in range(len(text)):
        d[text[i:i + cntr]].append(i)

    # find any item in dict that wasn't repeated at least once
    del_list = [(d[item]) for item in d if len(d[item]) > 1]
    sol = (0,0,0)

    # Keep looking as long as del_list isn't empty,
    while len(del_list) > 0:

        d = defaultdict(list)     # reset dictionary
        cntr += 1                 # increment search length
        for item in del_list:
            for i in item:
                d[text[i:i + cntr]].append(i)
        # filter as above
        del_list = [(d[item]) for item in d if len(d[item]) > 1]

        # if not empty, update solution
        if len(del_list) != 0:
            sol = (del_list[0][0], del_list[0][1], cntr)

    return sol
```

Unit tests

In [14]:
```
%%time
filename = 'pKLMF-FX.fasta'

text = read_fasta_file(filename)
print(len(text))
assert len(text) == 9988

tup = longest_repeat(text)

print(tup)
assert len(tup) == 3
assert isinstance(tup, tuple)

print(text[tup[0]:tup[0]+tup[2]])
print(text[tup[1]:tup[1]+tup[2]])

assert text[tup[0]:tup[0]+tup[2]] == text[tup[1]:tup[1]+tup[2]]
```

```
9988
(5434, 5440, 15)
CACGGGCACGGGCAC
CACGGGCACGGGCAC
CPU times: user 16.3 ms, sys: 0 ns, total: 16.3 ms
Wall time: 16.6 ms
```

In [15]:
```
%%time
filename = 'pACYC184.fasta'        # An EColi plasmid cloning vector

# See https://www.snapgene.com/resources/plasmid-files/?set=basic_cloning_vectors&plasmid=pACYC184


text = read_fasta_file(filename)
print(len(text))                   # DNA is 4289 Bytes long: remove first line and \n
assert len(text) == 4245

tup = longest_repeat(text)
print(tup)
assert len(tup) == 3
assert isinstance(tup, tuple)

print(text[tup[0]:tup[0]+tup[2]])
print(text[tup[1]:tup[1]+tup[2]])

assert tup[2] == 94
assert text[tup[0]:tup[0]+tup[2]] == text[tup[1]:tup[1]+tup[2]]
```

```
4245
(2180, 3274, 94)
AGCTCCTTCCGGTGGGCGCGGGGCATGACTATCGTCGCCGCACTTATGACTGTCTTCTTTATCATGCAACTCGTAGGACAGGTGCCGGCAGCGC
AGCTCCTTCCGGTGGGCGCGGGGCATGACTATCGTCGCCGCACTTATGACTGTCTTCTTTATCATGCAACTCGTAGGACAGGTGCCGGCAGCGC
CPU times: user 8.86 ms, sys: 139 µs, total: 9 ms
Wall time: 8.48 ms
```

Extra credit: Find the longest repeat in EColi

In [26]:
```
%%time
filename = 'ecoli.fasta'

text = read_fasta_file(filename)
print(len(text))
assert len(text) == 4641652

tup = longest_repeat(text)
print(tup)

assert len(tup) == 3
assert isinstance(tup, tuple)

print(text[tup[0]:tup[0]+tup[2]])
print(text[tup[1]:tup[1]+tup[2]])

assert text[tup[0]:tup[0]+tup[2]] == text[tup[1]:tup[1]+tup[2]]
```

```
4641652
(4168618, 4210020, 2815)
AAGAAACATCTTCGGGTTGTGAGGTTAAGCGACTAAGCGTACACGGTGGATGCCCTGGCAGTCAGAGGCGATGAAGGACGTGCTAATCTGCGATAAGCGTCGGT
AAGGTGATATGAACCGTTATAACCGGCGATTTCCGAATGGGGAAACCCAGTGTGTTTCGACACACTATCATTAACTGAATCCATAGGTTAATGAGGCGAACCGG
GGGAACTGAAACATCTAAGTACCCCGAGGAAAAGAAATCAACCGAGATTCCCCCAGTAGCGGCGAGCGAACGGGGAGCAGCCCAGAGCCTGAATCAGTGTGTGT
GTTAGTGGAAGCGTCTGGAAAGGCGCGCGATACAGGGTGACAGCCCCGTACACAAAAATGCACATGCTGTGAGCTCGATGAGTAGGGCGGGACACGTGGTATCC
TGTCTGAATATGGGGGGACCATCCTCCAAGGCTAAATACTCCTGACTGACCGATAGTGAACCAGTACCGTGAGGGAAAGGCGAAAAGAACCCCGGCGAGGGGAG
TGAAAAAGAACCTGAAACCGTGTACGTACAAGCAGTGGGAGCACGCTTAGGCGTGTGACTGCGTACCTTTTGTATAATGGGTCAGCGACTTATATTCTGTAGCA
AGGTTAACCGAATAGGGGAGCCGAAGGGAAACCGAGTCTTAACTGGGCGTTAAGTTGCAGGGTATAGACCCGAAACCCGGTGATCTAGCCATGGGCAGGTTGAA
GGTTGGGTAACACTAACTGGAGGACCGAACCGACTAATGTTGAAAAATTAGCGGATGACTTGTGGCTGGGGGTGAAAGGCCAATCAAACCGGGAGATAGCTGGT
TCTCCCCGAAAGCTATTTAGGTAGCGCCTCGTGAATTCATCTCCGGGGGTAGAGCACTGTTTCGGCAAGGGGGTCATCCCGACTTACCAACCCGATGCAAACTG
CGAATACCGGAGAATGTTATCACGGGAGACACACGGCGGGTGCTAACGTCCGTCGTGAAGAGGGAAACAACCCAGACCGCCAGCTAAGGTCCCAAAGTCATGGT
TAAGTGGGAAACGATGTGGGAAGGCCCAGACAGCCAGGATGTTGGCTTAGAAGCAGCCATCATTTAAAGAAAGCGTAATAGCTCACTGGTCGAGTCGGCCTGCG
CGGAAGATGTAACGGGGCTAAACCATGCACCGAAGCTGCGGCAGCGACGCTTATGCGTTGTTGGGTAGGGGAGCGTTCTGTAAGCCTGCGAAGGTGTGCTGTGA
GGCATGCTGGAGGTATCAGAAGTGCGAATGCTGACATAAGTAACGATAAAGCGGGTGAAAAGCCCGCTCGCCGGAAGACCAAGGGTTCCTGTCCAACGTTAATC
GGGGCAGGGTGAGTCGACCCCTAAGGCGAGGCCGAAAGGCGTAGTCGATGGGAAACAGGTTAATATTCCTGTACTTGGTGTTACTGCGAAGGGGGACGGAGAA
GGCTATGTTGGCCGGGCGACGGTTGTCCCGGTTTAAGCGTGTAGGCTGGTTTTCCAGGCAAATCCGGAAAATCAAGGCTGAGGCGTGATGACGAGGCACTACGG
TGCTGAAGCAACAAATGCCCTGCTTCCAGGAAAAGCCTCTAAGCATCAGGTAACATCAAATCGTACCCCAAACCGACACAGGTGGTCAGGTAGAGAATACCAAG
GCGCTTGAGAGAACTCGGGTGAAGGAACTAGGCAAAATGGTGCCGTAACTTCGGGAGAAGGCACGCTGATATGTAGGTGAGGTCCCTCGCGGATGGAGCTGAAA
TCAGTCGAAGATACCAGCTGGCTGCAACTGTTTATTAAAAACACAGCACTGTGCAAACACGAAAGTGGACGTATACGGTGTGACGCCTGCCCGGTGCCGGAAGG
TTAATTGATGGGGTTAGCGCAAGCGAAGCTCTTGATCGAAGCCCCGGTAAACGGCGGCCGTAACTATAACGGTCCTAAGGTAGCGAAATTCCTTGTCGGGTAAG
TTCCGACCTGCACGAATGGCGTAATGATGGCCAGGCTGTCTCCACCCGAGACTCAGTGAAATTGAACTCGCTGTGAAGATGCAGTGTACCCGCGGCAAGACGGA
AAGACCCCGTGAACCTTTACTATAGCTTGACACTGAACATTGAGCCTTGATGTGTAGGATAGGTGGGAGGCTTTGAAGTGTGGACGCCAGTCTGCATGGAGCCG
ACCTTGAAATACCACCCTTTAATGTTTGATGTTCTAACGTTGACCCGTAATCCGGGTTGCGGACAGTGTCTGGTGGGTAGTTTGACTGGGGCGGTCTCCTCCTA
AAGAGTAACGGAGGAGCACGAAGGTTGGCTAATCCTGGTCGGACATCAGGAGGTTAGTGCAATGGCATAAGCCAGCTTGACTGCGAGCGTGACGGCGCGAGCAG
GTGCGAAAGCAGGTCATAGTGATCCGGTGGTTCTGAATGGAAGGGCCATCGCTCAACGGATAAAAGGTACTCCGGGGATAACAGGCTGATACCGCCCAAGAGTT
CATATCGACGGCGGTGTTTGGCACCTCGATGTCGGCTCATCACATCCTGGGGCTGAAGTAGGTCCCAAGGGTATGGCTGTTCGCCATTTAAAGTGGTACGCGAG
CTGGGTTTAGAACGTCGTGAGACAGTTCGGTCCCTATCTGCCGTGGGCGCTGGAGAACTGAGGGGGGCTGCTCCTAGTACGAGAGGACCGGAGTGGACGCATCA
CTGGTGTTCGGGTTGTCATGCCAATGGCACTGCCCGGTAGCTAAATGCGGAAGAGATAAGTGCTGAAAGCATCTAAGCACGAAACTTGCCCCGAGATGAGTTCT
CCCTGAC
AAGAAACATCTTCGGGTTGTGAGGTTAAGCGACTAAGCGTACACGGTGGATGCCCTGGCAGTCAGAGGCGATGAAGGACGTGCTAATCTGCGATAAGCGTCGGT
AAGGTGATATGAACCGTTATAACCGGCGATTTCCGAATGGGGAAACCCAGTGTGTTTCGACACACTATCATTAACTGAATCCATAGGTTAATGAGGCGAACCGG
GGGAACTGAAACATCTAAGTACCCCGAGGAAAAGAAATCAACCGAGATTCCCCCAGTAGCGGCGAGCGAACGGGGAGCAGCCCAGAGCCTGAATCAGTGTGTGT
GTTAGTGGAAGCGTCTGGAAAGGCGCGCGATACAGGGTGACAGCCCCGTACACAAAAATGCACATGCTGTGAGCTCGATGAGTAGGGCGGGACACGTGGTATCC
TGTCTGAATATGGGGGGACCATCCTCCAAGGCTAAATACTCCTGACTGACCGATAGTGAACCAGTACCGTGAGGGAAAGGCGAAAAGAACCCCGGCGAGGGGAG
TGAAAAAGAACCTGAAACCGTGTACGTACAAGCAGTGGGAGCACGCTTAGGCGTGTGACTGCGTACCTTTTGTATAATGGGTCAGCGACTTATATTCTGTAGCA
AGGTTAACCGAATAGGGGAGCCGAAGGGAAACCGAGTCTTAACTGGGCGTTAAGTTGCAGGGTATAGACCCGAAACCCGGTGATCTAGCCATGGGCAGGTTGAA
GGTTGGGTAACACTAACTGGAGGACCGAACCGACTAATGTTGAAAAATTAGCGGATGACTTGTGGCTGGGGGTGAAAGGCCAATCAAACCGGGAGATAGCTGGT
TCTCCCCGAAAGCTATTTAGGTAGCGCCTCGTGAATTCATCTCCGGGGGTAGAGCACTGTTTCGGCAAGGGGGTCATCCCGACTTACCAACCCGATGCAAACTG
CGAATACCGGAGAATGTTATCACGGGAGACACACGGCGGGTGCTAACGTCCGTCGTGAAGAGGGAAACAACCCAGACCGCCAGCTAAGGTCCCAAAGTCATGGT
TAAGTGGGAAACGATGTGGGAAGGCCCAGACAGCCAGGATGTTGGCTTAGAAGCAGCCATCATTTAAAGAAAGCGTAATAGCTCACTGGTCGAGTCGGCCTGCG
CGGAAGATGTAACGGGGCTAAACCATGCACCGAAGCTGCGGCAGCGACGCTTATGCGTTGTTGGGTAGGGGAGCGTTCTGTAAGCCTGCGAAGGTGTGCTGTGA
GGCATGCTGGAGGTATCAGAAGTGCGAATGCTGACATAAGTAACGATAAAGCGGGTGAAAAGCCCGCTCGCCGGAAGACCAAGGGTTCCTGTCCAACGTTAATC
GGGGCAGGGTGAGTCGACCCCTAAGGCGAGGCCGAAAGGCGTAGTCGATGGGAAACAGGTTAATATTCCTGTACTTGGTGTTACTGCGAAGGGGGACGGAGAA
GGCTATGTTGGCCGGGCGACGGTTGTCCCGGTTTAAGCGTGTAGGCTGGTTTTCCAGGCAAATCCGGAAAATCAAGGCTGAGGCGTGATGACGAGGCACTACGG
TGCTGAAGCAACAAATGCCCTGCTTCCAGGAAAAGCCTCTAAGCATCAGGTAACATCAAATCGTACCCCAAACCGACACAGGTGGTCAGGTAGAGAATACCAAG
GCGCTTGAGAGAACTCGGGTGAAGGAACTAGGCAAAATGGTGCCGTAACTTCGGGAGAAGGCACGCTGATATGTAGGTGAGGTCCCTCGCGGATGGAGCTGAAA
TCAGTCGAAGATACCAGCTGGCTGCAACTGTTTATTAAAAACACAGCACTGTGCAAACACGAAAGTGGACGTATACGGTGTGACGCCTGCCCGGTGCCGGAAGG
TTAATTGATGGGGTTAGCGCAAGCGAAGCTCTTGATCGAAGCCCCGGTAAACGGCGGCCGTAACTATAACGGTCCTAAGGTAGCGAAATTCCTTGTCGGGTAAG
TTCCGACCTGCACGAATGGCGTAATGATGGCCAGGCTGTCTCCACCCGAGACTCAGTGAAATTGAACTCGCTGTGAAGATGCAGTGTACCCGCGGCAAGACGGA
AAGACCCCGTGAACCTTTACTATAGCTTGACACTGAACATTGAGCCTTGATGTGTAGGATAGGTGGGAGGCTTTGAAGTGTGGACGCCAGTCTGCATGGAGCCG
ACCTTGAAATACCACCCTTTAATGTTTGATGTTCTAACGTTGACCCGTAATCCGGGTTGCGGACAGTGTCTGGTGGGTAGTTTGACTGGGGCGGTCTCCTCCTA
AAGAGTAACGGAGGAGCACGAAGGTTGGCTAATCCTGGTCGGACATCAGGAGGTTAGTGCAATGGCATAAGCCAGCTTGACTGCGAGCGTGACGGCGCGAGCAG
GTGCGAAAGCAGGTCATAGTGATCCGGTGGTTCTGAATGGAAGGGCCATCGCTCAACGGATAAAAGGTACTCCGGGGATAACAGGCTGATACCGCCCAAGAGTT
CATATCGACGGCGGTGTTTGGCACCTCGATGTCGGCTCATCACATCCTGGGGCTGAAGTAGGTCCCAAGGGTATGGCTGTTCGCCATTTAAAGTGGTACGCGAG
CTGGGTTTAGAACGTCGTGAGACAGTTCGGTCCCTATCTGCCGTGGGCGCTGGAGAACTGAGGGGGGCTGCTCCTAGTACGAGAGGACCGGAGTGGACGCATCA
CTGGTGTTCGGGTTGTCATGCCAATGGCACTGCCCGGTAGCTAAATGCGGAAGAGATAAGTGCTGAAAGCATCTAAGCACGAAACTTGCCCCGAGATGAGTTCT
CCCTGAC
CPU times: user 57.3 s, sys: 247 ms, total: 57.5 s
Wall time: 57.6 s
```

In [16]:
```
%%time
filename = 'Human22.fasta'

text = read_fasta_file(filename)
print(len(text))
assert len(text) == 31264301

tup = longest_repeat(text)
print(tup)

assert len(tup) == 3
assert isinstance(tup, tuple)

print(text[tup[0]:tup[0]+tup[2]])
print(text[tup[1]:tup[1]+tup[2]])

assert text[tup[0]:tup[0]+tup[2]] == text[tup[1]:tup[1]+tup[2]]
```

```
31264301
(23413667, 23425819, 2681)
CCTCTGCCTCCCAGGTTCAAGCAATTCTCCTGCCTCAGCCTCCTGAGTAGCTGGGATTACAGGCACCCACCACCACGCCCAGCTAATTTTTTGTATTTTTAGTA
GAGACGGGGTTTCACCATGTTGGCCAGGCTGGTCTCGAACATCTGACCTCAGGTGATCCACCCATCTCGGCCTCCCAAAGTGCTGGGATTACAGGCGTGAGCCA
CCCTGCCCAGCCCAATTATAGATTTTTTAGGTTTAGGTGTTGACAGTAGCTCTCACCTCAGCCTGTTCTCTCTCCTTGTCATGCAGCCCACAGGGGAGATGGTC
AGGCCAGTGTGGGGGCTAATGAATAAATGCTACACTGTGCCCACTCAGGTGGGTAAGGGCTGGCACTCCTCTTCCCCTGGAGTGGGGCGGCTGTGCTGGCACCC
TTGGCAGACACAGTAAGGGGGACTGCACCTGGAAAGGATGGGCCAGTCGGGGCAGGACTACTCATCACTCATAGTGTGGGTGTCAGGGTTGTGTCACCCCTCCC
ACCTCCCTCTGCAGAGACGCAAAGTCAAGAGTAGGAAGAAGCCAACCTCTGAGGTAAGGCTTCCCCTGGAAGGCCCAGGGCTGGGGCTCTCTCCTTTCAGAGCT
CAGTTAGACCCAGACACACGGCAGGGAGTCCCAAGGGTAGTGGCAGGCCCCCTCCAGGAAACTCACAAGGTTACCACAGCTCAACTGAAAAGGAAGAACTTCCC
AGGACTGTGACACCCCAGTGTGAGAACAGGAGGATGAGGTGCTCTGAAGGCCTTTCTGCCCAGTCTGCCCTCTTATTCCTCCTGCAGGTCACGACCCCCAGGAG
ACCTGGAGGACTGAATGCTGCTGCCCCCAAGGAGGAGGCTGCCGTCTTATCCCAGGAGGGAGAGCAGGTGAAGTCCCCAGGGGAGGAAGCACCTAGCCCCATTC
CTGCTGAGCAGGAGGTGGCAGGTACCCCAGACTGGGAGGTAAGGACAGCCCGGGGCTTCGACTGAACGTCTCCAGCGTGGGTCCAACTGAGCAGCCATGGAGCA
CTGCAGAGTGGGAGGCAGCAGGGCAGGGAGGCAGTGCTGGAGGCTGGCTCAACCCCAAGACCAGCAGGCCAAGCTGCCATCCCAGGGGAGCGAGGACGTCTGTG
CAGAGCTGAGAGGCAGCAGCCATGTGTGAACAGACTGGGCCTCATCCTGGCCCCACCGACTTTGTGTGGACAGAGCCTGTTTCCCTGTCTGTGCAACACAGAAC
CTGCCTGATCTCACTGCTGGATCCCTCTTCTTCCTGCCAGGAAAATAAAAAGGTTCAAAAGGAAGTTGCTGCGTATCCATCTGGTAAGACCACTGACCCAGCGT
GCTGCAGGGGGCTGCTTCCACCCTGCTTCTCAGTGACTGCCAGGGTCACAGACACCCCAGCCCTTTCCCACCTTCCTGACCTGGGGAGGGGAGGGGAGGGAAGC
AGCCCAGGAGTCAGGTGCCTTGACCTTCCTGGGAGCCTCCTTGGGTGGGCAGGAACTCTGGGCCACTCCCCTGAGCTGGCTGCATCCCTACCTTTCACCACAGC
TGACCTGGCCCCGGGGCATCTCAGAGGGAGGGTTGGTTGCTCCCAGGAGGGGACTCACAAGGCTGCCTGTTTCTACTTTGCAGAGGCCTCTGAGGACAGCAAAG
AGCAAAGGCCCTGGGACCGGGTCTACGTGCCCATGACAGAGCTCTGGCTGGACTGGTTCTGAGCCTCTAACACCCCCAAGACTCAGAACCGTGAAGAAAATCTT
TCCAATAAATCCAAGAGTTGCTGCTGCTATAGGCCAGGCTGCCACCTTTCGGGGCCTCCGTCTTCAGACAAACCCAGCCTGGCTTCATCCACACTCCCTGTCCC
CACAGCTGCAGGAACAGCACTTCCTGCCACCGAGCCGTGTGACCACAGTGGATTGTCTCTGGAGGGGCCCAAGGGGGCCCTGGCCACCCTTCTGACTGACTCGG
TGCCAGGGGACAGACCAACGTCCCTCTCGTGCTGACAGCCGGGCCGCACCCTGGCATGAGGGCATTTACAGAAATGCTGGCGGAACTGCTGCCAGGGAGGCTGT
AGGGTCCTCTGGCAAAAGAGGCCTCAGGTGGCTCCTCAGAGTGTCTGTGGTTCTCTGTCCCAGGCTGTTCCCTAAGAAGGTCTGCCCAGGACTCAGGTAATCAT
ATGCTCATTAGAAACTCTTGGGCACTGCCTGTGTGCCCAGCCCAGCCCATTATGTCGGTGAGGACAGACGTGGAGGACAGCAGTCCCTGCCCTTGGTTGGGGCT
CCAGGCCAGCAAGGGCCACAGCCCCAGAAGGCAGAGCAGGAAGACAGGACTCGGGGCAGGTGAAGCAGCCTTCTCGTTGGCAGAAGGGAAACAGAAGCCCGGGG
TGGGGAAGGGTGGGGAAGGGTGGGGAAGGGTGGGCCCGGGGTCACACGGGGTAATGGCAGAGCAAGGACTAGGGTCAGGGTCTCTGGCTCTCAGCTGCCCATGC
CACCTCCTCCTTCTCTGCCCGCCCCAGTGCCTTATGGGTCCAAGGTTGACTCCTGTCCCTAGGGCAGGCCTGTGGGCCCTGCCTGATCCCTACTGGGAGGATGG
TACCTAGGGTTGGAGCCAAACAAGTGTCCTCCTCCAGCGCCAGCCTGGCCCTGAGTGCGAACTCGTCACTGGTCAGGGGTC
CCTCTGCCTCCCAGGTTCAAGCAATTCTCCTGCCTCAGCCTCCTGAGTAGCTGGGATTACAGGCACCCACCACCACGCCCAGCTAATTTTTTGTATTTTTAGTA
GAGACGGGGTTTCACCATGTTGGCCAGGCTGGTCTCGAACATCTGACCTCAGGTGATCCACCCATCTCGGCCTCCCAAAGTGCTGGGATTACAGGCGTGAGCCA
CCCTGCCCAGCCCAATTATAGATTTTTTAGGTTTAGGTGTTGACAGTAGCTCTCACCTCAGCCTGTTCTCTCTCCTTGTCATGCAGCCCACAGGGGAGATGGTC
AGGCCAGTGTGGGGGCTAATGAATAAATGCTACACTGTGCCCACTCAGGTGGGTAAGGGCTGGCACTCCTCTTCCCCTGGAGTGGGGCGGCTGTGCTGGCACCC
TTGGCAGACACAGTAAGGGGGACTGCACCTGGAAAGGATGGGCCAGTCGGGGCAGGACTACTCATCACTCATAGTGTGGGTGTCAGGGTTGTGTCACCCCTCCC
ACCTCCCTCTGCAGAGACGCAAAGTCAAGAGTAGGAAGAAGCCAACCTCTGAGGTAAGGCTTCCCCTGGAAGGCCCAGGGCTGGGGCTCTCTCCTTTCAGAGCT
CAGTTAGACCCAGACACACGGCAGGGAGTCCCAAGGGTAGTGGCAGGCCCCCTCCAGGAAACTCACAAGGTTACCACAGCTCAACTGAAAAGGAAGAACTTCCC
AGGACTGTGACACCCCAGTGTGAGAACAGGAGGATGAGGTGCTCTGAAGGCCTTTCTGCCCAGTCTGCCCTCTTATTCCTCCTGCAGGTCACGACCCCCAGGAG
ACCTGGAGGACTGAATGCTGCTGCCCCCAAGGAGGAGGCTGCCGTCTTATCCCAGGAGGGAGAGCAGGTGAAGTCCCCAGGGGAGGAAGCACCTAGCCCCATTC
CTGCTGAGCAGGAGGTGGCAGGTACCCCAGACTGGGAGGTAAGGACAGCCCGGGGCTTCGACTGAACGTCTCCAGCGTGGGTCCAACTGAGCAGCCATGGAGCA
CTGCAGAGTGGGAGGCAGCAGGGCAGGGAGGCAGTGCTGGAGGCTGGCTCAACCCCAAGACCAGCAGGCCAAGCTGCCATCCCAGGGGAGCGAGGACGTCTGTG
CAGAGCTGAGAGGCAGCAGCCATGTGTGAACAGACTGGGCCTCATCCTGGCCCCACCGACTTTGTGTGGACAGAGCCTGTTTCCCTGTCTGTGCAACACAGAAC
CTGCCTGATCTCACTGCTGGATCCCTCTTCTTCCTGCCAGGAAAATAAAAAGGTTCAAAAGGAAGTTGCTGCGTATCCATCTGGTAAGACCACTGACCCAGCGT
GCTGCAGGGGGCTGCTTCCACCCTGCTTCTCAGTGACTGCCAGGGTCACAGACACCCCAGCCCTTTCCCACCTTCCTGACCTGGGGAGGGGAGGGGAGGGAAGC
AGCCCAGGAGTCAGGTGCCTTGACCTTCCTGGGAGCCTCCTTGGGTGGGCAGGAACTCTGGGCCACTCCCCTGAGCTGGCTGCATCCCTACCTTTCACCACAGC
TGACCTGGCCCCGGGGCATCTCAGAGGGAGGGTTGGTTGCTCCCAGGAGGGGACTCACAAGGCTGCCTGTTTCTACTTTGCAGAGGCCTCTGAGGACAGCAAAG
AGCAAAGGCCCTGGGACCGGGTCTACGTGCCCATGACAGAGCTCTGGCTGGACTGGTTCTGAGCCTCTAACACCCCCAAGACTCAGAACCGTGAAGAAAATCTT
TCCAATAAATCCAAGAGTTGCTGCTGCTATAGGCCAGGCTGCCACCTTTCGGGGCCTCCGTCTTCAGACAAACCCAGCCTGGCTTCATCCACACTCCCTGTCCC
CACAGCTGCAGGAACAGCACTTCCTGCCACCGAGCCGTGTGACCACAGTGGATTGTCTCTGGAGGGGCCCAAGGGGGCCCTGGCCACCCTTCTGACTGACTCGG
TGCCAGGGGACAGACCAACGTCCCTCTCGTGCTGACAGCCGGGCCGCACCCTGGCATGAGGGCATTTACAGAAATGCTGGCGGAACTGCTGCCAGGGAGGCTGT
AGGGTCCTCTGGCAAAAGAGGCCTCAGGTGGCTCCTCAGAGTGTCTGTGGTTCTCTGTCCCAGGCTGTTCCCTAAGAAGGTCTGCCCAGGACTCAGGTAATCAT
ATGCTCATTAGAAACTCTTGGGCACTGCCTGTGTGCCCAGCCCAGCCCATTATGTCGGTGAGGACAGACGTGGAGGACAGCAGTCCCTGCCCTTGGTTGGGGCT
CCAGGCCAGCAAGGGCCACAGCCCCAGAAGGCAGAGCAGGAAGACAGGACTCGGGGCAGGTGAAGCAGCCTTCTCGTTGGCAGAAGGGAAACAGAAGCCCGGGG
TGGGGAAGGGTGGGGAAGGGTGGGGAAGGGTGGGCCCGGGGTCACACGGGGTAATGGCAGAGCAAGGACTAGGGTCAGGGTCTCTGGCTCTCAGCTGCCCATGC
CACCTCCTCCTTCTCTGCCCGCCCCAGTGCCTTATGGGTCCAAGGTTGACTCCTGTCCCTAGGGCAGGCCTGTGGGCCCTGCCTGATCCCTACTGGGAGGATGG
TACCTAGGGTTGGAGCCAAACAAGTGTCCTCCTCCAGCGCCAGCCTGGCCCTGAGTGCGAACTCGTCACTGGTCAGGGGTC
CPU times: user 6min 46s, sys: 2.08 s, total: 6min 49s
Wall time: 6min 48s
```

In [17]:
```
# well, I spent a looooong time on this problem.
# My first attempt - I thought I was removing items that were not repeat candidates.
# But I wasn't; I was going through the entire string again, just with a longer
# search leght! So it was both really really slow and really really stupid (facepalm).
# I've included that doozy of a code horror below as 'parkers_double_table.


# My second attempt was to simply use the 'get all the suffixes, sort them, the compare adjacent suffixes
# This worked well and I think is easier to read. I use a helper function to do the
# adjacent comps. This works - really really fast - for the pACYC184 and pKLMF-FX files - like, 75% faste
# BUT - trying to run it on ecoli kills the kernal or I run out of memory after about 15 seconds.


# How can it be faster on the 10,000 character one but not run at all on the longer one??
# what's interesting is that it even works on strings of up to 10 million characters
# (see https://www.bioinformatics.org/sms2/random_dna.html) if the length of the longest substring is fai
# short.
# I've included that slightly better effort as parkers_revenge


# Re-reading one of the posts from Nick on Piazza had me go back and re-look at my code. That's when
# I realized I wasn't actually removing items on each pass (double facepalm)

# I personally think that instead of returning the length-3 tuple:
# (index of first instance, index of second index, length of repeated substring
# we should return a tuple:
# ([index list for where each repeated substring occurs], length of repeated substring)

# Because it's possible the longest repeated substring is repeated three or more times, not just twice.

# So my preferred solution is instead of returning: sol = (del_list[0][0], del_list[0][1], cntr)
# I'd go: sol = (del_list[0][0:], cntr)

# I also need to go back and look at how I handle cases where there are two substrings of the same length
```

In [2]:
```
#parkers_double_table.py

from collections import defaultdict

def read_fasta_file(filename: str) -> str:
    with open(filename, 'r') as f:
        temp = [line.strip() for line in f]
    seq = ''.join(temp[1:])
    return seq

def longest_repeat(text, cntr=2):
    sol = (0, 0, 0)
    del_list = ['01', '01', '01']

    while len(del_list) != 0:
        d = defaultdict(list)

        for i in range(len(text)):
            d[text[i:i + cntr]].append(i)

        del_list = [(item, d[item]) for item in d if len(d[item]) > 1]

        # if list is empty, we're done
        if len(del_list) == 0:
            return sol
        else:
            sol = (del_list[0][1][0], (del_list[0][1][1]), len(del_list[0][0]))
        cntr += 1

    return sol
```

In [28]:
```
# parkers_revenge
def read_fasta_file(filename: str) -> str:
    with open(filename, 'r') as f:
        temp = [line.strip() for line in f]
    seq = ''.join(temp[1:])
    return seq


def longest_prefix(suf1, suf2, mx=None):
    # compares adjacent suffixes to find longest prefix
    min_len = min(len(suf1), len(suf2))
    for i in range(min_len):
        if suf1[i] != suf2[i]:
            return (suf1[0:i], len(suf1[0:i]))
    return (suf1[0:i], len(suf1[0:i]))


def longest_repeat(txt):
    # create sorted suffix list
    lst = sorted([text[i:] for i in range(len(text))])

    mxLen = 0
    mx_string = ""
    for x in range(len(lst) - 1):
        temp = longest_prefix(lst[x], lst[x + 1])
        if temp[1] > mxLen:
            mxLen = temp[1]
            mx_string = temp[0]
    first = txt.find(mx_string)
    last = txt.rfind(mx_string)
    return (first, last, mxLen)
```

In [ ]:
```
```