Homework 5: Regular Expressions

SI206 - Winter 2024

Introduction:

In this assignment, you will work with a set of real-world data retrieved from Wikipedia. The data is stored in the file `"hw5_world_leaders.txt"`. For each function, your task is to use regular expressions to extract useful information.

Sample data:

The data from Wikipedia is in plain text with the following format:

```
George Washington (February 22, 1732 – December 14, 1799) was an
American Founding Father, military officer, politician and statesman
who served as the first president of the United States from 1789 to
1797. Appointed by the Second Continental Congress as commander of the
Continental Army in June 1775, Washington led Patriot forces to
victory in the American Revolutionary War and then served as president
of the Constitutional Convention in 1787, which drafted and ratified
the Constitution of the United States and established the American
federal government. Washington has thus been called the "Father of the
Nation". On January 6, 1759, Washington, at age 26, married Martha
Dandridge Custis(June 2, 1731 O.S. — May 22, 1802), the 27-year-old
widow of wealthy plantation owner Daniel Parke Custis. The marriage
took place at Martha's estate; she was intelligent, gracious, and
experienced in managing a planter's estate, and the couple had a happy
marriage.[45] They moved to Mount Vernon, near Alexandria, where he
lived as a planter of tobacco and wheat and emerged as a political
figure.[46]
```

Your Tasks:

Please implement the following functions in **`HW5.py`**:

- **get_leader_info(file_name)**
  - This function reads the file with the file name and returns a list of strings. Each string should contain all the information for one leader. You need to use a regular expression to do this.
  - The expected output should be in the format:

```
Leader data:
['George Washington (February 22, 1732 – December 14, 1799) was an American Founding Father, military officer, politician and statesman who served as the first president of the United States from 1789 to 1797. Appointed by the Second Continental Congress as commander of the Continental Army in June 1775, Washington led Patriot forces to victory in the American Revolutionary War and then served as president of the Constitutional Convention in 1787, which drafted and ratified the Constitution of the United States and established the American federal government. Washington has thus been called the "Father of the Nation". On January 6, 1759, Washington, at age 26, married Martha Dandridge Custis(June 2, 1731 O.S. — May 22, 1802), the 27-year-old widow of wealthy plantation owner Daniel Parke Custis. The marriage took place at Martha\'s estate; she was intelligent, gracious, and experienced in managing a planter\'s estate, and the couple had a happy marriage.[45] They moved to Mount Vernon, near Alexandria, where he lived as a planter of tobacco and wheat and emerged as a political figure.[46]','Nelson Mandela (/mænˈdɛlə/ man-DEH-lə;[1] Xhosa: [xolíɬaɬa mandɛˆːla]; born Rolihlahla Mandela; July 18, 1918 – December 5, 2013) was a South African anti-apartheid activist and politician who served as the first president of South Africa from 1994 to 1999. He was the country's first black head of state and the first elected in a fully representative democratic election. His government focused on dismantling the legacy of apartheid by fostering racial reconciliation. Ideologically an African nationalist and socialist, he served as the president of the African National Congress (ANC) party from 1991 to 1997. A Xhosa, Mandela was born into the Thembu royal family in Mvezo, South Africa. He studied law at the University of Fort Hare and the University of Witwatersrand before working as a lawyer in Johannesburg. There he became involved in anti-colonial and African nationalist
```

```
politics, joining the ANC in 1943 and co-founding its Youth
League in 1944.',...]
```

- **create_leader_dict(*leader_data*)**
  - This function returns a dictionary with the keys being numbers **(1 - 10)** and the values being a tuple made up of **each leader's first and last name**. You can assume that all names are made up of two words (first and last name). This function should use a regular expression to add each name to a dictionary with the keys representing that name's position in the list of biographies and the values being the tuples of names.
  - The expected output should be in the format:
    ```
    Names dictionary: {1: ("George", "Washington"), 2: ("Nelson",
    "Mandela", ...}
    ```

- **finding_long_words*(leader_data)***
  - This function finds all words between 10 and 20 letters (both inclusive) used in the text file and returns a tuple. The long words should only be made up of letters and not contain any punctuation. The first value of the tuple should be a list of all long words found and the second value should be the average length of those found words. When computing the average, you should use floor division.
  - The expected output of "`Theodore Roosevelt Jr. (/ˈroʊzəvɛlt/ ROH-zə-velt;[b] October 27, 1858 – January 6, 1919), often referred to as Teddy or by his initials, T. R., was an American politician, statesman, conservationist, naturalist, and writer who served as the 26th president of the United States from 1901 to 1909.`"
    would be:
    ```
    Long words info: (['politician', 'conservationist',
    'naturalist'], 11)
    ```

- **compute_age_dict*(leader_data)***
  - This function gets a list of strings with each leader's information and returns a dictionary with the leaders' full names (including any suffixes such as Jr) as keys and their birth year, death year, and age as values in tuples.
  - NOTE: If the leader died before their birthday within the same year, the age should reflect this accurately by considering the exact birth and death dates.
  - Sample output:

```
Age info: {'George Washington': (1732, 1799, 67), 'Nelson
Mandela': (1918, 2013, 95), 'Thomas Jefferson': (1743,
1826, 83), 'Margaret Thatcher': (1925, 2013, 87),
'Theodore Roosevelt Jr.': (1858, 1919, 60), 'Mary
Teresa': (1910, 1997, 87), 'John Kennedy': (1917, 1963,
46), 'Jawaharlal Nehru': (1889, 1964, 74), 'Andrew
Jackson Jr': (1767, 1845, 78), 'Mao Zedong': (1893, 1976,
82), 'Mohandas Gandhi': (1869, 1948, 78), 'James
Madison': (1751, 1836, 85), 'Abraham Lincoln': (1809,
1865, 56), 'Swami Vivekananda': (1863, 1902, 39),
'Zachary Taylor': (1784, 1850, 65), 'Rosa Parks': (1913,
2005, 92), 'Franklin Pierce': (1804, 1869, 64), 'Thomas
Wilson': (1856, 1924, 67), 'Lee Yew': (1923, 2015, 91),
'Julius Nyerere': (1922, 1999, 77)}
```

- **Test Cases**
  - Make at least 2 test cases each for `get_leader_info`, `create_leader_dict`, `finding_long_words` and `compute_age_dict`. Make sure your tests cover two scenarios: confirming that things that should match indeed do, and ensuring that things that shouldn't match don't.
  - Eg: `compute_age_dict` should return `"George Washington" : (1732, 1799, 67)` as one of the dictionary items (key-value pair) but not `"Martha Dandridge Custis" : (1731, 1802, 70)`

Extra credit (6 points):

- **find_abbr_dict(*leader_data)***
  - This function finds all **abbreviations** and returns a dictionary with abbreviations as keys and full forms as values.
  - Abbreviations are made of consecutive uppercase letters inside a set of parentheses.
  - Full forms are consecutive words starting with an uppercase letter.
  - Sample format in data: `African National Congress (ANC)`
  - Sample output:

```
Abbreviation dictionary:
{'ANC': 'African National Congress',
 'ARW': 'American Revolutionary War',
 'CCP': 'Chinese Communist Party',
 'EPC': 'Equal Protection Clause'}
```

Grading Rubric (60 points + 6 points):

This rubric does not show all the ways you can lose points. For each of the functions, you can earn:

- points for correctly implementing each of the 4 functions (44 points)
  - `get_leader_info` (8 points)
  - `create_leader_dict` (10 points)
  - `finding_long_words` (12 points)
  - `compute_age_dict` (14 points)
- points for creating tests for each function (16 points)
  - 2 points each * 8 test cases
- points for extra credits and test cases (6 points)
  - correctly implementing `find_abbr_dict` (4 points)
  - creating test cases (2 points)

Submission:

Make at least 4 git commits and turn in your GitHub repo URL on Canvas by the due date to receive credit.