Once again, an additional PDF was considered necessary to properly describe everything going on in this week's assignment. Names were matched a total of 61 times and for all of those cases except two a date of birth, date of death and profession were also found. If a name was not matched, no attempts were made to match dates or professions since there was nobody to attribute them to. The following steps were taken to arrive at the final result.

1. After a cursory review of the file it became clear that sentence splitting by most means would be a nightmare. Periods appeared left and right in cases such as etc., St., a.k.a., Dr., Mrs. and many others. Moreover there was a great number of proper nouns that did **not** refer to people, such as "Russian Revolution", "Zagreb", "Three Spinsters" and "Feldmarschall". As a result it was decided that name matching would occur only at the very beginning of each article segment ("line"), thus ignoring all sentences after the first one each time. It really simplified the approach while still maintaining a healthy name sample size, given that the total number of article segments was 101. It also means that alternate forms for names in different languages have not been included in the solution. The exact process followed is described in the Python comment section.

2. While parentheses were a good indicator of where a person's important dates were to be found, the exact location of the temporal expression was inconsistent: the place of birth or death (if known) would be included as well, and a number of people had additional information regarding their name's pronunciation or its form in other languages. However, one thing was consistent across all 60 true positive results: the first two dates to appear in the text always referred to the person's bith and death respectively. Thus, a regex was simply used to match a date (with a specific format) once, and then reused on the remaining text to find the immediately next result.

3. Finally each person's profession had to be defined. The pattern noticed here was that a person's profession(s) would show up after "was" followed by an article, whether definite or indefinite. It should be noted that the person's nationality is more often than not mentioned right before their profession(s), but that information was found to be significant given that a lot of the professions listed had to do with politics, royalty, diplomacy and religion. That said, out of all the mathces this is readily the most relaxed one and thus the one with the most false positives and false negatives.

Some specific cases needed to be mentioned in this section. The numbering corresponds to the system used above: 1. for names, 2. for dates and 3. for professions.

1.
<u>Fixed False Negatives</u>
It's a fairly long list so instead of quoting the specific cases I will mention some of the methods used. Optional dashes (-) were included for names that have them. Connectors "von" and "de" are also looked for optionally, even though the former is sometimes part of a title instead of a name – not in the cases matched here however because of the "beginning of file" clause. Thanks to that same clause there was no need to specify that the first letter is uppercase, so the program now matches cases like "Štefan Moyses" (Š was not matched in [A-Z]) and "heodor Leschetizky". Finally "Charles I" was a special case where it was necessary to match a single uppercase character for the second part of the name.

Remaining False Negatives

Besides names that appear in places other than the beginning of a line, there are some names that are still not matched – or only matched partially.

Leopold Anton Johann Sigismund Josef Korsinus Ferdinand Graf Berchtold von und zu Ungarschitz, Frättling und Püllütz: it is only matched until (and including) “von”. This was a very difficult case that was abandoned fairly early on.

Johann (Yona) Kremenezky also spelled Kremenetski: only Johann is matched because the regex stops at the open parenthesis.

József baron Eötvös de Vásárosnamény: this name is not matched at all because of the ordering. If “baron” was either in the beginning or the end it would’ve been matched fine.

József, Karel, Ratzenhofer: single names followed by words starting with a lowercase letter were not matched, so this is intented behaviour for these three names.

Fixed False Positives

Titles were initially included in names until an addition was made; it is expained in the Python comment section. The only other false positive that was fixed was a case where “One” was considered a name because “of” followed it. This was arbitrarily fixed by requiring this category of names to be at least 4 characters long.

Remaining False Positives

There is only one false positive that was not fixed because it follows the pattern of the regex **too** well. It is actually the very last article segment because it starts with “In Vienna of the day”, and “In Vienna” is both a combination where the second word starts with a capital letter **and** it is followed by an “of”.

2.

As it was mentioned before, the method used to match the dates was very consistent. There were no false positives or false negatives found for sentences where a name was matched first. If one tries to use my date regex on sentences where **no** name is found then the dates matched are simply the first two in order.

3.

There is exactly one case where no profession was matched for an actual name and that is “is known as a champion for Majorca's wildlife”. It was considered to be a vague reference to a profession and because lookbehind is used (which needs to have a fixed width) no special case was included for it. Measures were taken to ensure that the profession is not too long; namely there’s an optional lookahead for “who” meaning that if one is found by the end then the expression stops matching more characters. There are of course many cases where multiple professions are separated by commas, but those were omitted in order to avoid other cases where extremely long lines would be printed out (basically false positives). Some false positives still remain in the form of “son” or “wife”. For the most part however professions are under-reported instead of over-reported.

```python
import re

try:
    file_handle = open("text_about_people.txt", encoding = "utf8")
except IOError:
    print("An error occured while opening the file for reading")

'''
Since most of the big picture analysis for this program can be found in the
beginning of this PDF, here comments will be used to describe the expressions
themselves. The numbering system is consistent with the one mentioned earlier.
'''

'''
The very first regex used is simply a comprehensive list of titles found at the
very beginning of the lines of the txt file. "PhDr." was a special case because
of the period at the end (which had to be escaped) and because it was missed
during the first pass. No matter which title is matched, the whitespace right
after it should not be neglected, otherwise future matching would be
problematic.

A tactic was then reused from exercise 2, week 10 where after a search() is
completed everything that comes after it is grouped together so that *it* may
be searched anew. This is used again when matching dates.
'''

#1.
title = re.compile("^(Archduke|Archbishop|Marshal|Baron|Prince|Count|PhDr\.)\s"
                   + "(?P<the_rest>.*)")

'''
Short regex that was used to make the compile() that follows it look a little
nicer. It simply covers cases where "de" or "von" (followed by a space)
appears, but of course it's matched optionally.
'''

#1.
de_von = "((de|von)\s)?"

'''
Three separate regexes are used to match names - all of them looking
exclusively at the beginning of the line, as mentioned before - but most of the
work is done by the first one.

i. Alphanumeric characters are matched until a space is found and then that is
matched as well. If "de" or "von" follows it is matched. Then there's a loop of
at least one instance of an uppercase letter followed optionally by
alphanumberic characters and dashes, with optional "de"s or "von"s in between.

ii. Positive lookahead was used for single names that are followed by a comma.
This includes "Sophie,", "Karl," and "Archduke Friedrich,", since "Archduke"
gets filtered out before this stage.

iii. Positive lookahead is used again, this time for single names followed by
"of" or "(". These are "Johann (Yona)" and "Archduke John of Austria".
'''

#1.
person = re.compile("(" +

        "^\w+\s" + de_von + "([A-Z](\w-?)*\s?" + de_von + ")+" + "|" +

        "^\w{4,}(?=,)" + "|" + "^\w{4,}(?=(\sof|\s\())" +

        ")")

'''
Code was again partially reused, namely the regex that covers month. From there
it was only necessary to add an expression for numbers 1-31 and finally one for
years - in the file there are no years before 1000 or after 1999. There are
three patterns used for dates: "MONTH NUMBER, YEAR", "NUMBER MONTH(,) YEAR" and
finally "YEAR". In the first one the comma *always* appears while in the second
```

```
it only appears a few times, so it had to be made optional.
'''

#2.
months = "(January|February|March|April|May|June|July|August|September|" \
"October|November|December)"
month_number = "(3[01]|[12]\d|[1-9])"
year = "1\d{3}"

dates = re.compile("(" +

        months + "\s" + month_number + ",\s" + year + "|" +

        month_number + "\s" + months + ",?\s" + year + "|" +

        year +

        ")(?P<the_rest>.*)")

'''
This segment had to be of this length because of the three variations for the
lookbehind: professions are matched after "was a", "was an" or "was the" is
found and those three had to be separate. The main regex itself is simple
enough: it keeps matching until the first comma or period is found.
To make some outputs shorter, a lookahead was added to exclude text that
appears after a "who" is matched, since that text usually doesn't have any
new professions to add. Since most of the cases did not have a "who", however,
it had to be an option that appears in the beginning of the regex so that the
regular version can be split with an "|" to cover all bases.
'''

#3.
profession = re.compile("(" +

        "(?<=was\sa\s)([^,\.]+)(?=who)" + "|" +

        "(?<=was\san\s)([^,\.]+)(?=who)" + "|" +

        "(?<=was\sthe\s)([^,\.]+)(?=who)" + "|" +

        "(?<=was\sa\s)([^,\.]+)" + "|" +

        "(?<=was\san\s)([^,\.]+)" + "|" +

        "(?<=was\sthe\s)([^,\.]+)" +

        ")")

'''
Last comment section for portfolio part 2!

The first thing this for loop does - after printing the input for reference is
it tries to filter out titles found at the very beginning; if one isn't found,
nothing happens.
Then it tries to match a person's name(s). If it fails the program goes on to
the file's next line. Otherwise it prints the name(s) found and goes on to
search for a date.
However it is not printed right away; a second date is searched for in the text
following that of the first date matched. Then if *that* is found both dates
are printed, the first one assumed as the date of birth and the second as the
date of death. There was no instance matched where one appeared without the
other so no information is lost here.
Finally, assuming a name was matched earlier, a search is conducted for
professions. If any are found, they are printed.
The last print() exists purely for formatting reasons. The two strip() methods
used are also used to avoid printing whitespace.
'''

for line in file_handle:
    print(line)

    match_title = title.search(line)
```

```
        if match_title:
            line = match_title.group("the_rest")

        match_name = person.search(line)
        if match_name:
            print("==> Name:", match_name.group(1).strip())

            match_dob = dates.search(line)
            if match_dob:
                print("==> Date of birth:", match_dob.group(1))
                line = match_dob.group("the_rest")

                match_dod = dates.search(line)
                if match_dod:
                    print("==> Date of death:", match_dod.group(1))

            match_prof = profession.search(line)
            if match_prof:
                print("==> Profession(s):", match_prof.group(1).strip())

        print()

    file_handle.close()
```

```
Count Ferdinánd Pálffy de Erdőd (1 February 1774 – 4 February 1840) was a mining
engineer and civil servant of the Austrian Empirewho is better remembered for his role
in managing the Theater an der Wien, Vienna, in pursuit of which he lost his not
inconsiderable fortune and retired from his creditors in Vienna.

==> Name: Ferdinánd Pálffy de Erdőd
==> Date of birth: 1 February 1774
==> Date of death: 4 February 1840
==> Profession(s): mining engineer and civil servant of the Austrian Empire

After graduating with a law doctorate from the University of Vienna, Emanuel Herrmann,
the son of the Bezirkshauptmann (district administrator) of Klagenfurt, entered the
civil service in the Austrian ministry of commerce and qualified for a university
career as a Privatdozent in the field of national economics.


Prince Franz de Paula of Liechtenstein(Franz de Paula Joachim Joseph) (Vienna, 25
February 1802 – Vienna, 31 March 1887) was a son of Johann I Joseph, Prince of
Liechtenstein (1760-1836) and wife Landgravine Josepha of Fürstenberg-Weitra, nephew
of Aloys I, brother of Aloys II and uncle of Johann II and Franz I.
```

```
Baron Eligius Franz Joseph von Münch-Bellinghausen (German: Eligius Franz Joseph
Freiherr von Münch-Bellinghausen) (2 April 1806 – 22 May 1871) was an Austrian
dramatist, poet and novella writer of the Austrian Biedermeierperiod and beyond, and
is more generally known under his pseudonym Friedrich Halm.

==> Name: Eligius Franz Joseph von Münch-Bellinghausen
==> Date of birth: 2 April 1806
==> Date of death: 22 May 1871
==> Profession(s): Austrian dramatist

Joseph Heicke, Josef Heicke (Josef Heike), Hungarian: Heicke József, Heike József (12
March, 1811, Vienna, Imp.-R. Austria – 6 November, 1861, Vienna, Imp.&R. Austria), was
an Austrian painter and lithographer, producing landscapes, portraits, natural history
water-colours and images of historic events.

==> Name: Joseph Heicke
==> Date of birth: 12 March, 1811
==> Date of death: 6 November, 1861
==> Profession(s): Austrian painter and lithographer
```