For exercise 10_2 an additional PDF had to be created to properly analyse the entirety of the program’s functionality. First off, since it wasn’t found to be clear what exactly constitutes a temporal expression, all sorts of relative expressions of time and duration were included in the output which ended up matching 256 words and phrases. That said, a number of words accompanying temporal expressions have been omitted on purpose.

“about”/”around”: There are three instances of “about” and one of “around” where temporal expressions are concerned. They were considered redundant since they simply added uncertainty when it came to the expressions they preceded.

“within”: It shows up twice in the phrase “within 5 minutes”, and it was decided that “5 minutes” can stand alone as a temporal expression with no need for “within”.

“late”/”the latest”: They were considered too vague to include despite them occurring relatively frequently.

“between”: This was a problem case. It appears five times and its behavior follows a pattern of “between X and Y”, where X and Y share a type – whether it be a date or the time – in all cases except one. Because of how the regexes have been implemented for this problem it was decided that it was more trouble than it was worth, and the temporal expressions it bridges are printed out anyway. Nevertheless, the code to match two dates with “between” has been added in comment form on line 177.

There are some special cases that need to be brought up as well.

<u>Fixed false positives</u>
Is there any cinema discount at 181 Broad Street in Birmingham?
==> at 1

What is name of the movie starring Adam Campbell Mary Castro Jayma Mays showing on 15th of October?
==> May

Can you tell me the date of the last performance of nights at the circus at Warwick Arts Centre?
==> nights

Is Nights at the Circus on in the West Midlands this year?
==> Nights

All four of these matches are no longer part of the output thanks to the introduction of negative lookahead. In the first case, negative lookahead didn’t allow for any more than 2 digits when it comes to describing time, while in the second months were no longer matched if followed by an “s”. The last two saw the addition of negative lookahead equal to “at the circus”, with an optional “s” added right before that for good measure to account for the fact that the “s” in “nights” is also optional.
These will be mentioned again in the Python PDF in regex #1, #8, #9 and #11.

Remaining false negatives

Are there any programmes for the Birmingham Hippodrome from the 21st to the 26th of May 2007?
==> the 26th of May 2007

One could say that “from the 21st to” should also be part of this temporal expression, but a similar problem that of “between” was found and the expression was left “incomplete”.

What is the name of the event that is on at the Grand Theatre on the 26th of May till the 30th of May 2007?
==> the 26th of May
till the 30th of May 2007?
==> the 30th of May 2007

Again, it would be better if this was merged into a single temporal expression but it was an extremely specific occurrence, so the loss was considered minimal.

I'd like to know whether there is a movie in Dudley for about 1 half 1 and half hours.
==> 1 and half hours

This notation was found to be too bizarre to accommodate for, therefore only the latter part of the expression was included.

Are there any cinemas near Wolverhampton that do weekday daytime discounts?
==> weekday

Clearly, “daytime” could be included as a more specific indicator for this temporal expression. However it was a unique instance that felt like it was better left alone.

Do any Birmingham cinemas do evening discounts during the week?
==> evening

“During the week” was simply not included because “during” was only combined with seasons at a very late stage of this solution.

Are there any local cinemas showing German films over the next few weeks?

“Over the next few weeks” was considered to be both too specific a case (one instance) and also too vague to include.

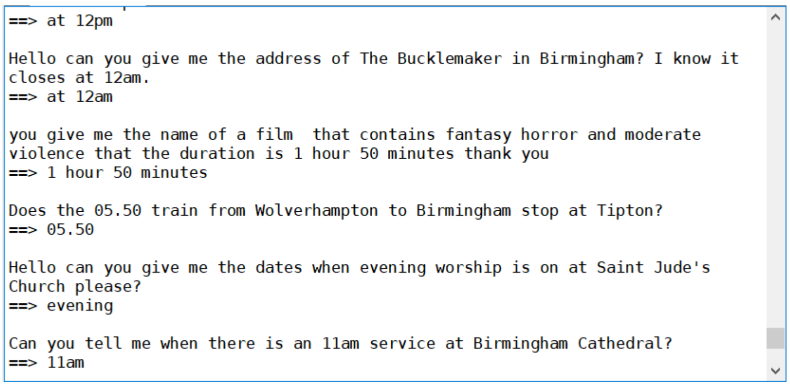
I would like to go to a place which reminds me of the nineteenth century.

Extremely specific fringe case that would require a lot of effort, since there would be need for a list of words that count order. If I’d gone down this path, I would also have to exclude “Twentieth Century Fox”.

What follows next is the Python PDF along with a tiny sample of the end of this program’s output. The lines are so many that the results are partially overwritten on Spyder (buffer left at default 500).

```python
#Exercise 2

import re

'''
This comment section will describe the following variables - and corresponding
regex, where necessary - in order of appearance.

As a side note, it should be mentioned that a lot of the variables include
superfluous parentheses that make formatting look better. It should also be
mentioned that when something like "(X)" appears in this comment section it
means that "X" is optional for the expression in question.


#1. Times of day are the highlight of this expression. There's an optional "s"
after each one in case it's found in plural form. Also optionally one can find
"tomorrow" or "___day" preceding the times of day, the latter standing for any
day of the week but also for words like "weekday".
Negative lookahead has been included for a very special case mentioned in the
first part of this PDF.

#2. This appears after #1. because "weekday" is matched by itself. The first
two lines represent clear-cut relative temporal expressions while the third one
introduces a group that allows statements in the form of "(A)B past (C)D",
where A, B, C and D are digits that follow specific rules found in regex #2.
"midday" only appears following "before" or "after" in this file and thus
they're included in order to make the temporal expression more complete.

#3. Description of upcoming events. "week" can optionally turn into "weekend".

#4. This category was named after the main use of its subject. The expression
always ends in "X minutes", where X is any number. Optionally preceding that is
either "(a/1) hundred (and)" or "(an/1) hour (and)".

#5. Again, ordering is important because regex #4. matches combinations of
hours and minutes while this one matches "(less than) X (and (a) half) hours",
where X is any number.

#6. #7. These both refer to dates in a style similar to the file that was
available on Canvas. The first instance is unique and it refers to the pattern
"DAY MONTH Xth", while the second regex symbolises "(DAY) (the) Xth of MONTH
(YEAR)" which appears fairly often. "DAY" is the name of a day, "MONTH" is the
name of a month, "YEAR" is any year between 2000 and 2009 - the only year in
this file is 2007 anyway - "th" symbolises a group of suffixes to show rank,
and finally "X" is any number. "X" should be between 1 and 31 but it's left to
its own devices because of everything else surrounding it; nobody would say
"the 42nd of August".

#8. #9. There are four regexes combined here: prefixes, suffixes and two forms
of expressing time. Prefixes and suffixes were included because there are a lot
of instances where the time is referred to with a plain number, but when that
happens either a) there's a temporal prefix, b) there's a temporal suffix, or
c) time is described in a way covered by "clock" (explained farther down).
Therefore, regexes #8. and #9. cover "(PRE) clock/the_time SUF" and
"PRE clock/the_time (SUF)" respectively. That means that prefixes are optional
for the former and suffixes are optional for the latter, but they're never
optional at the same time to avoid matching numbers unrelated to temporal
expressions. Ordering is once more important because otherwise "clock pm" cases
are missed.

"the_time" is "(half) X\D" where "half" is optional, "X" is a number between 1
and 12, and "\D" is negative lookahead to avoid one specific case mentioned
in the first part of this PDF.

"clock" is "(X)X.YY\D" where "(X)X" is either a non-zero digit or a number from
00 to 23 and "YY" is a number from 00 to 59 and "\D" is the same as above.

Both of the above regexes are necessary for different expressions. That's why
they are separated by an "or" in #8 and #9.

#10. "clock" appears by itself to cover cases such as "05.50" and "2.20" where
no temporal prefix or suffix exists.
```

```python

#11. "months" appears by itself because some of the questions refer to a month
instead of a specific date. Negative lookahead was included for one special
case mentioned in the first part of this PDF.

#12. "week_days" appears by itself for the same reason as #11.

#13. This category is last simply because it was noticed at the end. It matches
patterns like "(during) (the) SEASON", where "SEASON" is the name of a season.
'''


#1.
time_of_day = "((morning|afternoon|evening|night|midnight)s?)"
tod_prefixes = r"(((\w+day|tomorrow)\s+)?)"

#2.
special_cases = r"(tonight|tomorrow|weekday|" \
"at\s+the\s+moment|at\s+the\s+weekend|((before|after)\s+)?midday|" \
"[1-5]?\d\s+past\s+(1[0-2]|[1-9]))"

#3.
upcoming = r"(this|next)\s+(year|month|week(end)?)"

#4.
movies = r"(((a\s+|1\s+)?hundred\s+(and\s+)?)|" \
"((an\s+|1\s+)?hour\s+(and\s+)?))?" \
"\d+\s+minutes"

#5.
hours = r"(less\s+than\s+)?\d+\s+(and\s+(a\s+)?half\s+)?hours?"

#6. #7. #11. #12.
week_days = "(Monday|Tuesday|Wednesday|Thursday|Friday|Saturday|Sunday)"
months = "((January|February|March|April|May|June|July|August|September|" \
"October|November|December)(?!s))"

#6. #7.
rank = "(st|nd|rd|th)"

#8. 9. 10.
clock = r"(((([01]\d)|(2[0-3])|[1-9])\.[0-5]\d)(?!\d))"
the_time = r"(((half\s+)?(1[0-2]|[1-9]))(?!\d))"

prefixes = "(at|by|after|before|from|till|until)"
suffixes = "(o'clock|am|pm)"

#13.
seasons = r"((during\s+)?(the\s+)?(winter|spring|summer|autumn))"

timex = re.compile(
        "(" +

        #1.
        tod_prefixes + time_of_day + r"(?!s?\s+at\s+the\s+circus)" + "|" +

        #2.
        special_cases + "|" +

        #3.
        upcoming + "|" +

        #4.
        movies + "|" +

        #5.
        hours + "|" +


        #6.
        week_days + r"\s+" + months + r"\s+\d+" + rank + "|" +

        #7
        "(" + week_days + r"\s+)?(the\s+)?" + r"\d+" + rank + r"\s+of\s+" +
        months + r"(\s+200\d)?" + "|" +


        #8.
        "(" + prefixes + r"\s+)?(" + clock + "|" + the_time + r")\s*" +
        suffixes + "|" +

        #9
        prefixes + r"\s+(" + clock + "|" + the_time + r")\s*" +
        suffixes + "?" + "|" +


        #10.
        clock + "|" +

        #11.
        months + "|" +

        #12.
        week_days + "s?|" +

        #13.
        seasons +

        ")(?P<the_rest>.*)", re.I)


'''
"Between" code mentioned on the first page of the PDF.

        "between\s+(" + week_days + r"\s+)?(the\s+)?" + r"\d+" + rank +
        r"\s+of\s+" + months + r"(\s+200\d)?\s+and\s+" + "(" + week_days +
        r"\s+)?(the\s+)?"n+ r"\d+" + rank + r"\s+of\s+" + months +
        r"(\s+200\d)?" + "|" +
'''



'''
The last part of this program is almost untouched from the original version
uploaded by you. One change is that the output goes through .strip() first.
'''

try:
    file_questions = open("list-of-questions.txt")
    for question in file_questions:
        while True:
            match = timex.search(question)
            if match:
                print(question.strip())
                print("==>", match.group(1).strip(), "\n")
                question = match.group("the_rest")
            else:
                break

except IOError:
    print("The file with questions cannot be read")
```



```
==> at 12pm

Hello can you give me the address of The Bucklemaker in Birmingham? I know it
closes at 12am.
==> at 12am

you give me the name of a film  that contains fantasy horror and moderate
violence that the duration is 1 hour 50 minutes thank you
==> 1 hour 50 minutes

Does the 05.50 train from Wolverhampton to Birmingham stop at Tipton?
==> 05.50

Hello can you give me the dates when evening worship is on at Saint Jude's
Church please?
==> evening

Can you tell me when there is an 11am service at Birmingham Cathedral?
==> 11am
```