Objectives of this assignment

• Learn or review basic features of the Python language.
• Use the Python 3 programming language to write a less resource-restricted implementation of a function named `icsout` (but without using regular expressions or user-defined classes).
• Test your code against the provided test cases

`icsout2.py:` Returning to the problem

For this assignment please use the description of the problem as provided at the end of this document. Some of the limits that were placed on certain values are no longer needed (e.g., maximum number of events, maximum line length, etc.). That is, you will indicate the range of dates to be used to generate the schedule by providing "`--start`" and "`--end`" arguments. However, the executable will be named "`icsout2.py`"

```
./icsout2.py   --start=2021/2/14  --end=2021/2/14  --file=one.ics
```

All output is to *stdout.* However, four different kinds of constraints have been placed on your program.

1. For this assignment do not use regular expressions.
2. You must not write your own classes
3. You must not use global variables. Note that this is not a hardship as it is easier in Python to pass mutable data structures as function parameters
4. You must make good use of functional decomposition. Phrased another way, your submitted work must not contain one or two giant functions where all of your program logic is concentrated.

What you must submit:

• A single Python source file named "`icsout2.py`" within your `git` repository containing a solution.
• No regular-expressions, global variables, or user-defined classes are to be used
• The program will be a Python program. Its name must be "icsout2.py"

Input specification:

1. All input is from ASCII test files.
2. Data lines for an "event" begin with a line "BEGIN:VEVENT" and end with a line "END:VEVENT".
3. Starting time: An event's starting date and time is contained on a line of the format "DTSTART: <icalendardate>" where the characters following the colon comprise the date/ time in icalendar format.
4. Ending time: An event's ending date and time is contained on a line of the format "DTEND: <icalendardate>" where the characters following the colon comprise the date/ time in icalendar format.

5. Event location: An event’s location is contained on a line of the format “LOCATION: <string>” where the characters following the colon comprise the string describing the event location. These strings will never contain the “:” character.
6. Event description: An event’s description is contained on a line of format “SUMMARY: <string>” where the characters following the colon comprise the string describing the event’s nature. These strings will never contain the “:” character.
7. Repeat specification: If an event repeats, this will be indicated by a line of the format “RRULE:FREQ= <frequency>; UNTIL=<icalendardate>”. The only frequencies you must account for are weekly frequencies. The date indicated by UNTIL is the last date on which the event will occur (i.e., is inclusive). Note that this line contains a colon (“:”) and semicolon (“;”) and equal signs (“=”).
8. Events within the input stream are not necessarily in chronological order.
9. Events may overlap in time.
10. No event will ever cross a day boundary.
11. All times are local time (i.e., no timezones will appear in a date/time string).

Output specification:

1. All output is to *stdout.*
2. All events which occur from 12:00 am on the --start date and to 11:59 pm on the --end date must appear in chronological order based on the event’s starting time that day.
3. If events occur on a particular date, then that date must be printed only once in the following format: <month text> <day>, <year> (<day of week>)------------------------------
Note that the line of dashes below the date must match the length of the date. You may use Python’s date time module in order to create the calendar-date line.
4. Days are separated by a single blank line. There is no blank line at the start or at the end of the program’s output.
5. Starting and ending times given in 12-hour format with “am” and “pm” as appropriate. For example, five minutes after midnight is represented as “12:05 am”.
6. A colon is used to separate the start/end times from the event description
7. The event SUMMARY text appears on the same line as the even time. (This text may include parentheses.)
8. The event LOCATION text appears on after the SUMMARY text and is surrounded by square brackets. Events from the same day are printed on successive lines in chronological order by starting time. Do not use blank lines to separate the event lines within the same day. In the case of tests provided by the instructor, the Unix “diff” utility will be used to compare your program’s output with what is expected for that test. Significant differences reported by “diff” may result in grade reductions.

Sample Testcase 1 (Without any rule):

Suppose the ics file is like this:

```
BEGIN:VCALENDAR
BEGIN:VEVENT
DTSTART:20210214T180000
DTEND:20210214T210000
LOCATION:Burger King
SUMMARY:Romantic dinner with Chris
END:VEVENT
END:VCALENDAR
```

Then if we pass this input:

```
./icsout --start=2021/2/14 --end=2021/2/14 --file=one.ics
```

We get Output as follows:

```
February 14, 2021 (Sun)
-----------------------
 6:00 PM to  9:00 PM: Romantic dinner with Chris {{Burger King}}
```

Sample Testcase 2 (With rules):

Suppose the file is ics is like this:

```
ABEGIN:VCALENDAR
VERSION:A
BEGIN:VEVENT
DTSTART:20210102T111500
DTEND:20210102T123000
RRULE:FREQ=WEEKLY;WKST=MO;UNTIL=20210301T235959;BYDAY=SA
LOCATION:The Bumptious Barista
SUMMARY:Coffee with Pat
END:VEVENT
END:VCALENDAR
```

Then if we pass this input:

```
./icsout --start=2021/2/1 --end=2021/3/1 --file=many.ics
```

We get the following output:

```
February 06, 2021 (Sat)
-----------------------
11:15 AM to 12:30 PM: Coffee with Pat {{The Bumptious Barista}}

February 13, 2021 (Sat)
-----------------------
11:15 AM to 12:30 PM: Coffee with Pat {{The Bumptious Barista}}

February 20, 2021 (Sat)
-----------------------
11:15 AM to 12:30 PM: Coffee with Pat {{The Bumptious Barista}}

February 27, 2021 (Sat)
-----------------------
11:15 AM to 12:30 PM: Coffee with Pat {{The Bumptious Barista}}
```