IDSN 542: Machine Intelligence

Homework 3: Calendar — Due: 10/1/2025 @ 11:59PM

Goal

In this homework, you will create a calendar, so I can remember birthdays and holidays.

Setup

- Create a Python file called **hw03.py**.
- Your **hw03.py** file must begin with comments in the following format (replace the name and email with your actual information):
```
'''
Name
IDSN 542, Fall 2025
USC email
Homework 3
'''
```

Requirements

The assignment is broken into several parts – and it is recommended that you proceed in the order given. It is also a GoodIdea™ to test each part before moving on to the next

Part 1: Months dictionary

- Create a dictionary that maps a month name (like “January”) to the number of days in that month. The following table should help

Month name	Days in the month
January	31
February	28
March	31
April	30
May	31
June	30
July	31
August	31
September	30
October	31
November	30
December	31

Part 2: Calendar dictionary

- Create an empty dictionary that will eventually hold your calendar. The keys will be the month’s name, the values will be a list of strings.
- Create a list of empty strings appropriate for each month. For example, for January there should be 31 strings in the list, for September there should only be 30. You should be able to do this in a loop

- After you've created the list, insert it into the dictionary. The list will be the value, the month name will be the key.
- You might want to try printing out the dictionary to make sure it's filled correctly.

Part 3: User input

- Ask the user for a month and day. The user should be able to type the month and the day separated by a space. For example, "April 1" is valid input. You should handle a variety of invalid input. If the user enters in a bad month or a bad day, indicate the user's mistake. Any other invalid input should cause the program to re-prompt the user. Stop prompting for input when the user gives the empty string (or just hits enter with no input).
- Prompt the user for an event to happen on the entered day. Add that to the appropriate place in the dictionary.
- **HINT**: The user will enter "1" for the first day in the month, but that item lives at index 0.
- **HINT**: You can use **split** to separate the user's string into the text before the space and the text after the space
- Sample output should look like this:

```
Enter a date for a holiday (for example "July 1"): April 1
What happens on April, 1? April Fool's

Enter a date for a holiday (for example "July 1"): Movember 2
I don't know about the month "Movember"

Enter a date for a holiday (for example "July 1"): February 31
That month only has 28 days!

Enter a date for a holiday (for example "July 1"): December 12 1979
I don't see good input in there!
```

Part 4: Display results

- Display all the dates with events. If the date has no event, don't display it.
- Sample output should look like this:

```
Enter a date for a holiday (for example "July 1"): April 1
What happens on April, 1? April Fool's Day

Enter a date for a holiday (for example "July 1"): December 24
What happens on December, 24? Christmas Eve

Enter a date for a holiday (for example "July 1"): July 12
What happens on July, 12? Bastille Day

Enter a date for a holiday (for example "July 1"):

April 1 : April Fool's Day
July 12 : Bastille Day
December 24 : Christmas Eve
```

Part 5: Write results to a text file

- Write the contents of your dictionary to a text file after you display the results to the user. You might want to use commas as a delimiter. Prompt the user for the name of the text file to create.

Part 6: Read prior saved results from a text file

- When the program starts up, and before you prompt the user to enter any new events, prompt the user for the file name to read from. Display the results to the user after reading from the file. Then prompt the user for new events. If the user just hits the Enter key, then don't read from a file. It is not an error.

A Note on Style

Be sure to comment your code. Also, you will lose points if your variable names are not meaningful. Make sure you use variable names that correspond to what you are storing in the variables.

Full Sample Output

Below is sample output for a full run-through of the program. As usual, you are not required to be as descriptive as the text below, so long as you perform and display the required calculations. User input is in **red**.

```
Enter the file name to read your events:

Enter a date for a holiday (for example "July 1"): January 1
What happens on January, 1? New Year's Day

Enter a date for a holiday (for example "July 1"): July 4
What happens on July, 4? Independence Day

Enter a date for a holiday (for example "July 1"): July 24
What happens on July, 24? Nathan's Birthday

Enter a date for a holiday (for example "July 1"): December 31
What happens on December, 31? New Year's Eve

Enter a date for a holiday (for example "July 1"):

Enter the file name to save your events: events.txt

January 1 : New Year's Day
July 4 : Independence Day
July 24 : Nathan's Birthday
December 31 : New Year's Eve

Goodbye!
```

Deliverables

1. A compressed folder containing **hw03.py**, named **hw03.zip**.

Grading

Item	**Points**
Part 1: Months dictionary	5
Part 2: Calendar dictionary	5
Part 3: User input	5
Part 4: Display results	5
Part 5: Write to file	5
Part 6: Read from file	5
Total	**30**