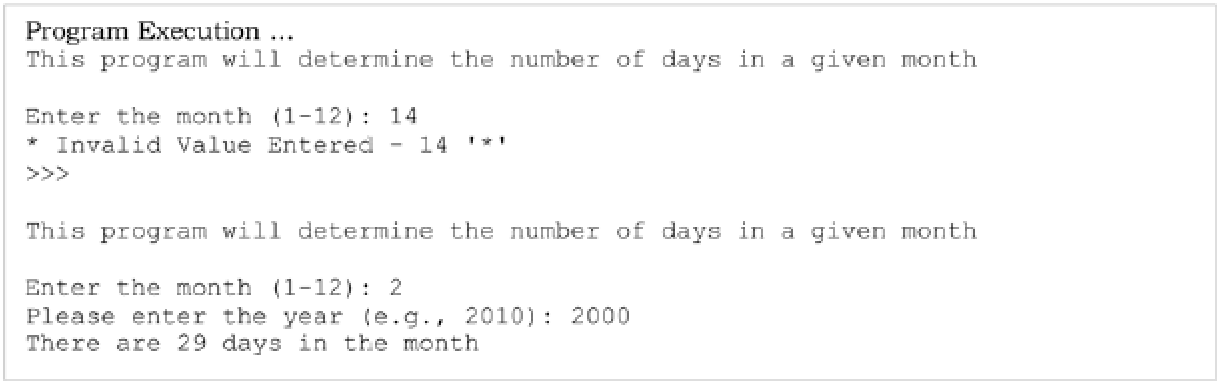
Number of Days in Month Program

The following Python program prompts the user for a given month (and year for February), and displays how many days are in the month. This program utilizes the following programming features:

- ➤ if statement
- ➤ elif header

Example execution of the program is given below:



Program Execution ...

```
This program will determine the number of days in a given month

Enter the month (1-12): 14
* Invalid Value Entered - 14 '*'
>>>

This program will determine the number of days in a given month

Enter the month (1-12): 2
Please enter the year (e.g., 2010): 2000
There are 29 days in the month
```

Task: In IDLE, open a new project and save at as DaysInMonth_yourLastName. Copy the code from the sample on the next page. Test and revise the program, as needed.

```
 1  # Number of Days in Month Program
 2
 3  # program greeting
 4  print('This program will display the number of days in a given month\n')
 5
 6  # init
 7  valid_input = True
 8
 9  # get user input
10  month = int(input('Enter the month (1-12): '))
11
12  # determine num of days in month
13
14  # february
15  if month == 2:
16      year = int(input('Please enter the year (e.g., 2010): '))
17
18      if (year % 4 == 0) and (not (year % 100 == 0) or (year % 400 == 0)):
19          num_days = 29
20      else:
21          num_days = 28
22
23  # january, march, may, july, august, october, december
24  elif month in (1, 3, 5, 7, 8, 10, 12):
25      num_days = 31
26
27  # april, june, september, november
28  elif month in (4,6,9,11):
29      num_days = 30
30
31  # invalid input
32  else:
33      print('* Invalid Value Entered - ', month, '*')
34      valid_input = False
35
36  # output result
37  if valid_input:
38      print('There are', num_days, 'days in the month')
```

Notes:

Lines 1–4 provide the program header and program greeting. On line 7, variable `valid_input` is initialized to `True` for the input error-checking performed. Line 10 prompts the user for the month, read as an integer value (1–12), and stores in variable `month`. On line 15 the month of February is checked for. February is the only month that may have a different number of days—28 for a regular year, and 29 for leap years. Thus, when February (2) is entered, the user is also prompted for the year (line 16). If the `year` is a leap year, then variable `num_days` is set to 29—otherwise, it is set to 28.

Generally, if a year is (evenly) divisible by 4, then it is a leap year. However, there are a couple of exceptions. If the year is divisible by 4 but is also divisible by 100, then it is *not* a leap year -

unless, it is also divisible by 400, then it is. For example, 1996 and 2000 were leap years, but 1900 was not. This condition is given below.

```
(year % 4 == 0) and (not (year % 100 == 0) or (year % 400 == 0))
```

Thus, the conditions for which this Boolean expression is true are,

```
(year % 4 == 0) and not (year % 100 == 0)
```

and

```
(year % 4 == 0) and (year % 400 == 0)
```

Line 24 checks if `month` is equal to 1, 3, 5, 7, 8, 10, or 12. If true, then `num_days` is assigned to 31. If not true, line 28 checks if `month` is equal to 4, 6, 9, or 11 (all the remaining months except February). If true, then `num_days` is assigned to 30. If not true, then an invalid month (number) was entered, and `valid_input` is set to `False`. Finally, the number of days in the month is displayed only if the input is valid (line 38).