EEE 111 Software Project 1 Milestone 2

Problem	Submissions	Leaderboard	Discussions

Submitted 21 minutes ago • Score: 75.00   Status: Accepted

✔ Test Case #0	✔ Test Case #1	✔ Test Case #2
✔ Test Case #3	✔ Test Case #4	✔ Test Case #5
✔ Test Case #6	✔ Test Case #7	✔ Test Case #8
✔ Test Case #9	✔ Test Case #10	✔ Test Case #11
✔ Test Case #12	✔ Test Case #13	✔ Test Case #14
✔ Test Case #15	✔ Test Case #16	✔ Test Case #17
✔ Test Case #18	✔ Test Case #19	✔ Test Case #20
✔ Test Case #21	✔ Test Case #22	✔ Test Case #23
✔ Test Case #24	✔ Test Case #25	✔ Test Case #26
✔ Test Case #27	✔ Test Case #28	✔ Test Case #29
✔ Test Case #30	✔ Test Case #31	✔ Test Case #32
✔ Test Case #33	✔ Test Case #34	✔ Test Case #35
✔ Test Case #36	✔ Test Case #37	✔ Test Case #38
✔ Test Case #39	✔ Test Case #40	✔ Test Case #41
✔ Test Case #42	✔ Test Case #43	

Submitted Code

Language: Python 3   ⑂ Open in editor

```python
#!/bin/python

# Rick's Great Plan for Success
#  1) Validate date range
#     [/] Within date limit
#     [/] All inputs are integers
#     [/] Valid dates
#  2) Number crunch
#     [/] Compute total days
#     [/] Compute weekdays
#        [/] Compute weekends
#     [/] Compute leap years
#     [/] Compute holidays falling on weekdays
#        [/] New Year
#        [/] Labor Day
#        [/] All Saint's Day
#        [/] Christmas
#     [/] Compute work days

# To abstract calendar math
```

```python
21  import datetime
22  # To memoize commonly used functions
23  from functools import lru_cache
24  # For syntactic sugar and to make the date range immutable
25  from collections import namedtuple
26  DateRange = namedtuple("DateRange", ["start", "end"])
27
28
29  def get_user_input():
30      """Gets date range as user input
31
32      Raises:
33          Exception: If input dates are out of date limit (January 1, 1971 to December 31, 2020)
34          ValueError:
35            If at least one of the following occurs:
36              - At least one of the inputs is not an integer
37              - At least one of the input dates is not a valid date
38
39      Returns:
40          DateRange: The start and end dates as datetime objects (contained in a namedtuple)
41      """
42      # Gets user input in M\nD\nYYYY format for the start date
43      start_instrings = ["Enter start month: ",
44                         "Enter start day: ", "Enter start year: "]
45      raw_start_date = tuple(input(s) for s in start_instrings)
46      # Gets user input in M\nD\nYYYY format for the end date
47      end_instrings = ["Enter end month: ",
48                       "Enter end day: ", "Enter end year: "]
49      raw_end_date = tuple(input(s) for s in end_instrings)
50
51      # Uses map to convert string input to integers and stores the values in a tuple
52      start_date = tuple(map(int, raw_start_date))
53      end_date = tuple(map(int, raw_end_date))
54
55      # Checks if each year is within the date limit
56      if not(1971 <= start_date[2] <= 2020 and 1971 <= end_date[2] <= 2020):
57          raise Exception("Input date/s outside date limit.")
58
59      # Cyclic rotation of elements (because I really really **really** want to unpack)
60      # Source: https://www.geeksforgeeks.org/python-shift-last-element-to-first-position-in-list/
61      start_date, end_date = start_date[-1:] + \
62          start_date[:-1], end_date[-1:] + end_date[:-1]
63
64      # As you can see unpacking makes the line smaller and more readable
65      # return DateRange(datetime.date(start_date[2], start_date[0], start_date[1]),
66    datetime.date(end_date[2], end_date[0], end_date[1]))
66      return DateRange(datetime.date(*start_date), datetime.date(*end_date))
67
68
69  @lru_cache(maxsize=None)
70  def compute_total_days(start, end):
71      """Computes the total number of days between the start and end dates
72
73      Args:
74          start (datetime.date): The start date
75          end (datetime.date): The end date
76
77      Returns:
78          int: The total number of days between the start and end dates
79      """
80      # Use the datetime module to subtract the dates (+1 if inclusive)
81      return (end - start).days + 1
82
83
84  @lru_cache(maxsize=None)
85  def compute_weekends(start, end):
86      """Computes the total number of weekend days between the start and end dates
87
88      Args:
89          start (datetime.date): The start date
90          end (datetime.date): The end date
91
92      Returns:
93          int: The total number of weekend days between the start and end dates
94      """
```

```python
 95        # Initialize the weekends counter
 96        weekends = 0
 97
 98        # Do-while loop (to check if the start date falls on a weekend too)
 99        while True:
100            # Check if the day falls on a weekend
101            if start.weekday() == 5 or start.weekday() == 6:
102                weekends += 1
103
104            # The loop checks the days between the start date (inclusive) and
105            #  the next occurence of the end date's day of the week
106            if start.weekday() == end.weekday():
107                break
108
109            # Increment the start date by one day
110            start += datetime.timedelta(days=1)
111
112        # Once the start date and the end date fall on the same day of the week,
113        #  we can just find the number of weeks between them and multiply
114        #  by two
115        weekends += ((end - start).days // 7) * 2
116        return weekends
117
118
119 @lru_cache(maxsize=None)
120 def compute_weekdays(start, end):
121     """Computes the total number of weekdays between the start and end dates
122
123     Args:
124         start (datetime.date): The start date
125         end (datetime.date): The end date
126
127     Returns:
128         int: The total number of weekdays between the start and end dates
129     """
130     # Subtracts the total number of weekend days from the total number of days
131     return compute_total_days(start, end) - compute_weekends(start, end)
132
133
134 @lru_cache(maxsize=None)
135 def compute_leap_years(start, end):
136     """Computes the total number of leap days between the start and end dates
137
138     Args:
139         start (datetime.date): The start date
140         end (datetime.date): The end date
141
142     Returns:
143         int: The total number of leap days between the start and end dates
144     """
145     # Generate the leap years between 1971 and 2020 inclusive
146     leap_years = tuple(1972 + 4*x for x in range(13))
147
148     # Looks for the closest leap year greater than or equal to the start year
149     min_leap_year = 0
150     for leap_year in leap_years:
151         if leap_year >= start.year:
152             min_leap_year = leap_year
153             break
154
155     # Looks for the closest leap year less than or equal to the end year
156     max_leap_year = 0
157     for leap_year in reversed(leap_years):
158         if leap_year <= end.year:
159             max_leap_year = leap_year
160             break
161
162     # Gets the number of leap years between the start and end year
163     # Note that if the leap year in between is just the same year it will zero out, thus the +1
164     leap_days_between = ((max_leap_year - min_leap_year) // 4) + 1
165
166     # If the start date occurs after Feb 29th of that year, we don't consider
167     if (start - datetime.date(min_leap_year, 2, 29)).days > 0:
168         leap_days_between -= 1
169
```

```python
170        # If the end date occurs before Feb 29th of that year, we don't consider
171        if (datetime.date(max_leap_year, 2, 29) - end).days > 0:
172            leap_days_between -= 1
173
174        return leap_days_between
175
176
177 @lru_cache(maxsize=None)
178 def compute_holidays(start, end):
179     """Computes the total number of holidays between the start and end dates
180
181     Args:
182         start (datetime.date): The start date
183         end (datetime.date): The end date
184
185     Returns:
186         dict: The number of relevant occurrences per holiday and the total number of holidays between
    the start and end dates
187     """
188     # The list of holidays and their given dates every year
189     holiday_dates = {
190         "new year holiday:": (1, 1),
191         "labor day holiday:": (5, 1),
192         "all saints day holiday:": (11, 1),
193         "christmas holiday:": (12, 25)
194     }
195
196     # Initialize the count of occurrences per holiday
197     holiday_counts = {holiday: 0 for holiday in holiday_dates.keys()}
198     # For loop to go through each holiday
199     for holiday in holiday_dates.keys():
200         # Sets the year for when counting the occurrences start
201         count_start = start.year
202         # If the holiday occurs before the start date, we disregard it
203         if (start - datetime.date(start.year, *holiday_dates[holiday])).days > 0:
204             count_start += 1
205         # Sets the year for when counting the occurrences end
206         count_end = end.year
207         # If the holiday occurs after the end date, we disregard it
208         if (datetime.date(end.year, *holiday_dates[holiday]) - end).days > 0:
209             count_end -= 1
210         # For loop to go through each year in the counting range
211         for year in range(count_start, count_end + 1):
212             # If the holiday falls on a weekday, we increment the occurrence count
213             if datetime.date(year, *holiday_dates[holiday]).weekday() < 5:
214                 holiday_counts[holiday] += 1
215
216     # The total number of holidays is the sum of the counts of each holiday
217     holiday_counts["total holidays:"] = sum(holiday_counts.values())
218
219     # Returns the dictionary with complete counts
220     return holiday_counts
221
222
223 @lru_cache(maxsize=None)
224 def compute_workdays(start, end):
225     """Computes the total number of workdays between the start and end dates
226
227     Args:
228         start (datetime.date): The start date
229         end (datetime.date): The end date
230
231     Returns:
232         int: The total number of workdays between the start and end dates
233     """
234     # Subtracts the total number of holidays from the total number of weekdays
235     return compute_weekdays(start, end) - compute_holidays(start, end)["total holidays:"]
236
237
238 if __name__ == "__main__":
239     # Getting user input and dealing with errors caused by invalid output
240     try:
241         dr = get_user_input()
242     except:
243         print("\nInvalid Input. Exiting Program.")
```

```
244        exit()
245
246    # Computing the total number of days between start and end date
247    print("\ntotal days from start date to end date:",
248          compute_total_days(dr.start, dr.end))
249
250    # Computing the total additional days from leap years
251    print("\ntotal additional days from leap years:",
252          compute_leap_years(dr.start, dr.end))
253
254    # Computing the total number of weekend days
255    print("\ntotal weekends:",
256          compute_weekends(dr.start, dr.end))
257
258    # Computing the total number of weekdays
259    print("\ntotal days without weekends:",
260          compute_weekdays(dr.start, dr.end))
261
262    # Extra line in format
263    print()
264
265    # Computing the total number of holidays
266    for holiday, count in compute_holidays(dr.start, dr.end).items():
267        print(holiday, count)
268
269    # Computing the total number of working days
270    print("\ntotal working days:",
271          compute_workdays(dr.start, dr.end))
272
```