COMP 125, Fall 2021
Homework Assignment 1

Due: 23:59, November 8 (Monday), 2021

In this assignment, you will implement a program that takes the current date and the user's date of birth from the keyboard, calculates how many days the user has lived until the current date, and displays this number of days on the screen.

We will ask you to implement the entire program in Part E. However, as this is your first homework, we will guide you in the design of this entire program. For that, through Part A to Part D, we will ask you to implement specific parts of that program. In Part E, you will partially or entirely use these parts and come up with a solution.

This program can be better implemented if functions are used. However, you will learn functions later. Thus, in this homework, **you are NOT allowed to use any functions**, except the built-in `input` and `print` functions.

Additionally, as specified below, you will save the solution of each part in a separate py file. You MUST conform to the following naming convention to give a name to each of these py files. Otherwise, you will lose 5 points each time you are using another file name.

- For Part A, the file name should be **`hw1_part_a.py`**
- For Part B, the file name should be **`hw1_part_b.py`**
- For Part C, the file name should be **`hw1_part_c.py`**
- For Part D, the file name should be **`hw1_part_d.py`**
- For Part E, the file name should be **`hw1_part_e.py`**
- (optional) For Bonus Part, the file name should be **`hw1_part_bonus.py`**

Please put all these py files in a compressed file (zip, rar, etc) and submit it as a single file via Blackboard.

Please note that, the correctness of your program will, of course, affect your grade. However, in addition to this, the following are important to get the full credit from each part.

- You MUST use meaningful names for the variables.
- You MUST write explanatory and clearly understandable comments.
- At the beginning of each file, you MUST write your name, your id, and your section as comments. After these, you MUST provide a very brief description about what the program does as additional comments.

For this assignment, you are allowed to use the codes given in the recommended textbooks and/or our lecture slides. However, you ARE NOT ALLOWED to use any codes from other sources (including the codes given in other textbooks, found on the Internet, belonging to your classmates, etc.). Furthermore, you ARE NOT ALLOWED to use any Python built-in functions except the `input` and `print` functions. **Do not forget that plagiarism and cheating will be heavily punished. Please do the homework yourself.**

Before continuing further, please be sure that you carefully read all explanations given above and understood them very well.

PART A: (10 points)

A leap year is a calendar year that has 366 days, with the extra day of the 29th of February. To be a leap year, the year number should be divisible by 4. The only exception is the end-of-century years. An end-of-century year is a leap year only if it is divisible by 400. For example, 1996, 2000, and 2004 are leap years but 2015, 2003, 1900, and 2100 are not.

Write a program that takes a positive integer from the keyboard as a year number and determines whether or not this number corresponds to a leap year. This program displays "Leap year" on the screen if it is a leap year, and "Not leap year" otherwise. For the output format, check the examples below.

You may assume that the input is always valid and a positive integer.

Implement the program in a file called **hw1_part_a.py**. The file name should be exactly this one. Otherwise, you may receive at most 5 points from this part (i.e., minus 5 points for other file names).

After implementing the program, test it with different inputs. Below are example outputs to show how the program should work for two different inputs. You may also use these inputs to test your program but they are not complete. It means, we will test your program also with other inputs. Thus, we strongly recommend you to use other year numbers to test your program.

```
In [1]: runfile('/Users/hw1_part_a.py', wdir='/Users')

Enter a year: 2021
Not leap year

In [2]: runfile('/Users/hw1_part_a.py', wdir='/Users')

Enter a year: 2024
Leap year
```

PART B: (20 points)

Write a program that takes two integers from the keyboard as a month number and a year number, respectively, and calculates how many days the given month has in the given calendar year. This program displays this day number on the screen. For the output format, check the examples below.

You may assume that the inputs are always valid. That is, the month number is an integer and always in between 1 and 12, and the year number is always a positive integer.

Remember that January, March, May, July, August, October, and December (whose month numbers are 1, 3, 5, 7, 8, 10, and 12, respectively) have 31 days and April, June, September, and November (whose month numbers are 4, 6, 9, and 11, respectively) have 30 days. February, whose month number is 2, has 29 days if the calendar year is a leap year, and 28 days if it is not a leap year.

Implement the program in a file called **hw1_part_b.py**. The file name should be exactly this one. Otherwise, you may receive at most 15 points from this part (i.e., minus 5 points for other file names).

Hint: You may reuse the code fragments that you write for Part A.

After implementing the program, test it with different inputs. Below are example outputs to show how the program should work for four different inputs. You may also use these inputs to test your program but they are not complete. It means, we will test your program also with other inputs. Thus, we strongly recommend you to use other month and year numbers to test your program.

```
In [1]: runfile('/Users/hw1_part_b.py', wdir='/Users')

Enter a month: 3

Enter a year: 2021
Number of days:  31

In [2]: runfile('/Users/hw1_part_b.py', wdir='/Users')

Enter a month: 2

Enter a year: 2021
Number of days:  28

In [3]: runfile('/Users/hw1_part_b.py', wdir='/Users')

Enter a month: 2

Enter a year: 2020
Number of days:  29

In [4]: runfile('/Users/hw1_part_b.py', wdir='/Users')

Enter a month: 6

Enter a year: 1992
Number of days:  30
```

PART C: (20 points)

Write a program that takes three integers from the keyboard as a day number, a month number, and a year number, respectively, and displays whether this corresponds to a valid date. This program displays "Valid" on the screen if these inputs correspond to a valid date, and "Invalid" otherwise. For the output format, check the examples below.

You may assume that the inputs are always integers. However, these inputs can be any positive or negative integers. You should perform the validity check on the day, month, and year. As mentioned in the previous parts, a valid year number should be positive, a valid month number should be in between 1 and 12, and a valid day number should be in between 1 and the number of days, which you calculated according to the month and year in Part B.

Implement the program in a file called **hw1_part_c.py**. The file name should be exactly this one. Otherwise, you may receive at most 15 points from this part (i.e., minus 5 points for other file names).

Hint: You may reuse the code fragments that you write for Part A and Part B.

After implementing the program, test it with different inputs. Below are example outputs to show how the program should work for seven different inputs. You may also use these inputs to test your program but they are not complete. It means, we will test your program also with other inputs. Thus, we strongly recommend you to use other day, month, and year numbers to test your program.

```
In [1]: runfile('/Users/hw1_part_c.py', wdir='/Users')

Enter a day: 12

Enter a month: 3

Enter a year: 2012
Valid date

In [2]: runfile('/Users/hw1_part_c.py', wdir='/Users')

Enter a day: 29

Enter a month: 2

Enter a year: 2012
Valid date

In [3]: runfile('/Users/hw1_part_c.py', wdir='/Users')

Enter a day: 29

Enter a month: 2

Enter a year: 2011
Invalid day

In [4]: runfile('/Users/hw1_part_c.py', wdir='/Users')

Enter a day: 10

Enter a month: 13

Enter a year: 2020
Invalid month
```

```
In [5]: runfile('/Users/hw1_part_c.py', wdir='/Users')

Enter a day: 31

Enter a month: 14

Enter a year: 2012
Invalid month

In [6]: runfile('/Users/hw1_part_c.py', wdir='/Users')

Enter a day: 35

Enter a month: -2

Enter a year: 2021
Invalid day
Invalid month

In [7]: runfile('/Users/hw1_part_c.py', wdir='/Users')

Enter a day: -6

Enter a month: 14

Enter a year: -1023
Invalid day
Invalid month
Invalid year
```

PART D: (30 points)

Write a program that takes three integers from the keyboard as a day number, a month number, and a year number, respectively, and calculates how many days have passed from the new year day (from the 1st of January of the given year). You need to include the current day into the number of days that have passed.

Afterwards, the program displays the number of days that have passed from the 1st of January of the given year as well as the number of days left to the 31st of December of the same year. Note that the number of days left is equal to (366 – number of days passed) if it is a leap year and (365 – number of days passed) otherwise. For the output format, check the examples below.

You may assume that the inputs are always valid. That is, the day, month, and year numbers entered as inputs are always valid. Thus, you do not need to make any validity check on these numbers.

In this question, you MUST use loops to make this calculation. Otherwise, you will receive 0 points from this part.

Implement the program in a file called **hw1_part_d.py**. The file name should be exactly this one. Otherwise, you may receive at most 25 points from this part (i.e., minus 5 points for other file names).

Hint: You may reuse the code fragments that you write for Part A, Part B, and Part C.

After implementing the program, test it with different inputs. Below are example outputs to show how the program should work for three different inputs. You may also use these inputs to test your program but they are not complete. It means, we will test your program also with other inputs. Thus, we strongly recommend you to use other day, month, and year numbers to test your program.

```
In [1]: runfile('/Users/hw1_part_d.py', wdir='/Users')

Enter a day: 14

Enter a month: 4

Enter a year: 2020
Days passed:  105
Days left:  261

In [2]: runfile('/Users/hw1_part_d.py', wdir='/Users')

Enter a day: 14

Enter a month: 4

Enter a year: 2021
Days passed:  104
Days left:  261

In [3]: runfile('/Users/hw1_part_c.py', wdir='/Users')

Enter a day: 29

Enter a month: 12

Enter a year: 1993
Days passed:  363
Days left:  2
```

PART E: (20 points)

Now you are ready!

Implement a program that takes the current date (first a day number, second a month number, and last a year number) and the user's birth of date (first a day number, second a month number, and last a year number) from the keyboard, calculates how many days the user has lived until the current date, and displays this number of days on the screen. For the output format, check the examples below.

You may assume that the inputs are always valid. That is, the day, month, and year numbers entered as inputs are always valid. Additionally, you may assume that the user does not enter a future date as the birth of date. Thus, you do not need to make any validity check on these inputs.

Furthermore, to make your coding easier, you may assume that the current year and the year of birth are not the same.

In this question, you MUST also use loops to make this calculation. Otherwise, you will receive 0 points from this part.

Implement the program in a file called **hw1_part_e.py**. The file name should be exactly this one. Otherwise, you may receive at most 15 points from this part (i.e., minus 5 points for other file names).

Hint: You may reuse the code fragments that you write for Part A, Part B, Part C, and Part D.

After implementing the program, test it with different inputs. Below are example outputs to show how the program should work for two different inputs. You may also use these inputs to test your program but they are not complete. It means, we will test your program also with other inputs. Thus, we strongly recommend you to use other current dates and the birth of dates to test your program.

```
In [1]: runfile('/Users/hw1_part_e.py', wdir='/Users')

Enter the current day: 21

Enter the current month: 10

Enter the current year: 2021

Enter the day for your date of birth: 12

Enter the month for your date of birth: 4

Enter the year for your date of birth: 1978
You have lived 15898 days

In [2]: runfile('/Users/hw1_part_e.py', wdir='/Users')

Enter the current day: 21

Enter the current month: 10

Enter the current year: 2021

Enter the day for your date of birth: 17

Enter the month for your date of birth: 12

Enter the year for your date of birth: 2020
You have lived 308 days
```

BONUS PART: (up to 20 points)

You may receive extra credits up to 20 points, according to what you will additionally implement to get a more improved version of the program that you write in Part E. Here are some possibilities:

- You may add validity checks on the current date and the birth of date. If either is invalid, you may give a warning message to the user.
- You may add validity check to understand whether the date of birth indicates a future date. If so, you may give a warning message to the user.
- You may get your program to work properly for the inputs where the current year and the year of birth are the same.

For this part, in addition to providing your code, please also add comment lines to the beginning of the program and clearly indicate what kind of improvement you make. We will test your program and grade it accordingly.

Implement the program in a file called **hw1_part_bonus.py**. The file name should be exactly this one. Otherwise, you may receive at most 15 points from this part (i.e., minus 5 points for other file names).