Task

=====

Time to complete : 3 days

2. Create a program to cater to the following functionalities

a.date time coversions between timezones ( Example UTC to US/Eastern etc)

b.date operations to add/subtract days from a given date

c.fetch number of days between two dates supplied

d.fetch number of days excluding weekends

e.fetch number of days since EPOCH

f.fetch number of business days between two days excluding holidays

(holiday calendar provided in a file in the following format)

holidays.dat

TIMEZONE,DATE,HOLIDAY

US/Eastern, 20211225, Christmas Day

Create a class called DateUtility

Define the following methods,

ALL These should return datetime objects

1.convert\_dt(from\_date, from\_date\_TZ , to\_date\_TZ)

2.add\_dt(from\_date, number\_of\_days)

3.sub\_dt(from\_date, number\_of\_days)

4.get\_days(from\_date, to\_date)

5.get\_days\_exclude\_we(from\_date, to\_date)

6.get\_days\_since\_epoch(from\_date)

7.get\_business\_days(from\_date, to\_date) --> This should look at holidays.dat to exclude

==========================================================

Coding Standard: Please follow PEP8 coding standards.

Doc strings for functions should be mentioned as below:

==========================================================

Example:

def send\_metric(

self,

metric\_name: str,

datapoint: Union[float, int],

tags: Optional[List[str]] = None,

type\_: Optional[str] = None,

interval: Optional[int] = None,

) -> Dict[str, Any]:

"""

Sends a single datapoint metric to DataDog

:param metric\_name: The name of the metric

:type metric\_name: str

:param datapoint: A single integer or float related to the metric

:type datapoint: int or float

:param tags: A list of tags associated with the metric

:type tags: list

:param type\_: Type of your metric: gauge, rate, or count

:type type\_: str

:param interval: If the type of the metric is rate or count, define the corresponding interval

:type interval: int

""”