**ESCRIPTION**

GET HELP

SUBMISSIONS

DISCUSS

List of dates

You are given two dates D1 and D2.

Write a program to print the list of dates between two dates D1 and D2(including D1 and D2) in the increasing order of the dates.

The date in string format is like "Feb 20 2021".Input

The first line of input will be a string, denoting the D1 in the string format.

The second line of input will be a string, denoting the D2 in the string format.Output

The output should be multiple lines containing the dates from D1 to D2 each in a new line.Explanation

For example, if the given D1 and D2 is

Jul 11 2014

Jul 19 2014

Your code should print the dates from 11th July to 19th July of 2014. So the output should be

2014-07-11 00:00:00

2014-07-12 00:00:00

2014-07-13 00:00:00

2014-07-14 00:00:00

2014-07-15 00:00:00

2014-07-16 00:00:00

2014-07-17 00:00:00

2014-07-18 00:00:00

2014-07-19 00:00:00Constraints

Date D1 comes before Date D2

Sample Input 1

Jul 11 2014

Jul 19 2014

Sample Output 1

2014-07-11 00:00:00

2014-07-12 00:00:00

2014-07-13 00:00:00

2014-07-14 00:00:00

2014-07-15 00:00:00

2014-07-16 00:00:00

2014-07-17 00:00:00

2014-07-18 00:00:00

2014-07-19 00:00:00

Sample Input 2

Feb 18 2014

Feb 24 2014

Sample Output 2

2014-02-18 00:00:00

2014-02-19 00:00:00

2014-02-20 00:00:00

2014-02-21 00:00:00

2014-02-22 00:00:00

2014-02-23 00:00:00

2014-02-24 00:00:00

CODE

from datetime import datetime,timedelta

given\_date\_string\_1=input()

given\_date\_string\_2=input()

date\_format\_1=datetime.strptime(given\_date\_string\_1, '%b %d %Y')

date\_format\_2=datetime.strptime(given\_date\_string\_2, '%b %d %Y')

days\_in\_middle=(date\_format\_2-date\_format\_1).days

for i in range(days\_in\_middle+1):

print(date\_format\_1+timedelta(days=i))

**DESCRIPTION**

GET HELP

SUBMISSIONS

DISCUSS

Name of the Weekday

Write a program to print the name of the weekday of a given date.

The date in string format is like "8 Feb 2021".Input

The input will be a single line containing the date in the string format.Output

The output should be a single line containing the name of the weekday.Explanation

For example, if the given date is "7 Sept 2020", your code should print the name of the weekday on 7th September 2020. So the output should be "Sunday".

Sample Input 1

2 Jul 2000

Sample Output 1

Sunday

Sample Input 2

14 Oct 1999

Sample Output 2

Thursday

CODE:

from datetime import datetime

date=input()

date\_string="%d %b %Y"

current\_date=datetime.strptime(date,date\_string)

weekday=datetime.strftime(current\_date,"%A")

print(weekday)

**ESCRIPTION**

GET HELP

SUBMISSIONS

DISCUSS

Count of Mondays

You are given two years A and B.

Write a program to count the number of Mondays M, which is the 1st day of the month from year A to year B(including B).Input

The input will be a single line containing the two space-separated integers, denoting the years A and B.Output

For example, if the given years are 2015 and 2017, the dates which contain Mondays as their month starting weekday from 2015 to 2017 are

2015-06-01

2016-02-01

2016-08-01

2017-05-01

So the output should be 4.

Sample Input 1

2015 2017

Sample Output 1

4

Sample Input 2

2021 2021

Sample Output 2

3

CODE:  
from datetime import datetime

year\_a, year\_b = input().split()

mondays = 0

months = range(1, 13)

for year in range(int(year\_a), int(year\_b)+1):

for month in months:

date\_time\_object = datetime(year, month, 1)

name\_of\_weekday = datetime.strftime(date\_time\_object, "%A")

if name\_of\_weekday == "Monday":

mondays += 1

print(mondays)

**DESCRIPTION**

GET HELP

SUBMISSIONS

DISCUSS

UNIX Timestamp to DateTime

The Unix timestamp is a way to track time as a running total of seconds.

This count starts at the Unix Epoch on January 1st, 1970, at UTC.

Therefore, the Unix timestamp is merely the number of seconds between a particular date and the Unix Epoch.

You are given UNIX timestamp U.

Write a program to convert U to a readable date time(in UTC) and print it.Input

The input will be a single line containing a UNIX timestamp U.Output

The output should be a single line containing the timestamp in UTC DateTime format.

DateTime format: YYYY-MM-DD HH:MM:SS.

Time is in 24-hour format.

Explanation

For example, if the given timestamp is 1284105682 states that, that many number of seconds have passed from the reference time mentioned in the above question.

Your code should print the datetime corresponds to the timestamp in UTC after the given number of seconds have passed from the reference time. So the output should be

2010-09-10 08:01:22

Sample Input 1

1284105682

Sample Output 1

2010-09-10 08:01:22

Sample Input 2

979293600

Sample Output 2

2001-01-12 10:00:00

CODE:  
import datetime

given\_data=datetime.datetime(1970,1,1)

seconds=datetime.timedelta(seconds=int(input()))

date\_format="%Y-%m-%d %H:%M:%S"

date\_result=given\_data+seconds

print(date\_result.strftime(date\_format))

Sample Input 1

5 0

Sample Output 1

Denominator can't be 0

Sample Input 2

12 a

Sample Output 2

Input should be an integer

Sample Input 3

10 2

Sample Output 3

5.0

CODE:

try:

a, b = input().split()

a = int(a)

b = int(b)

c = a/b

print(c)

except ZeroDivisionError:

print("Denominator can't be 0")

except ValueError:

print("Input should be an integer")

**ESCRIPTION**

GET HELP

SUBMISSIONS

DISCUSS

Item

You are given an incomplete Item class.

Go through the comments in the prefilled code to implement the Item class with the described attributes and methods.

Points to Note

* The output of the testcase Checking Default Tests is given by the default\_test function in the prefilled code.
* This Coding question does not have the usual input/output testcases. The class defined by you will be tested internally whether the attributes are present or not. So in testcases results you will be shown the rough description of the tests that will be verified.

Sample Input

Checking Default Tests

Sample Output

Oreo Biscuits

30

Food

Oreo Biscuits@30-Food

CODE:

# Item class should have the following attributes & methods

#

# Attributes:

# name, price, category

#

# Methods:

# get\_detail: Print the details of the item in the format '{name}@{price}-{category}'

#

# If the value of price is less than 1, raise ValueError exception like

# "ValueError: Invalid value for price, got {price}"

# Implement the Item class appropriately

class Item:

def \_\_init\_\_(self, name, price, category):

if price <=0:

raise ValueError("Invalid value for price, got {}".format(price))

self.name=name

self.price=price

self.category=category

def get\_detail(self):

return "{}@{}-{}".format(self.name,self.price,self.category)

# You need not change any code below.

# Do not call this function anywhere. It will automatically be called internally during tests.

def default\_test():

item = Item(name="Oreo Biscuits", price=30, category="Food")

print(item.name) # prints "Oreo Biscuits"

print(item.price) # prints '30'

print(item.category) # prints "Food"

print(item.get\_detail()) # prints "Oreo Biscuits@30-Food"