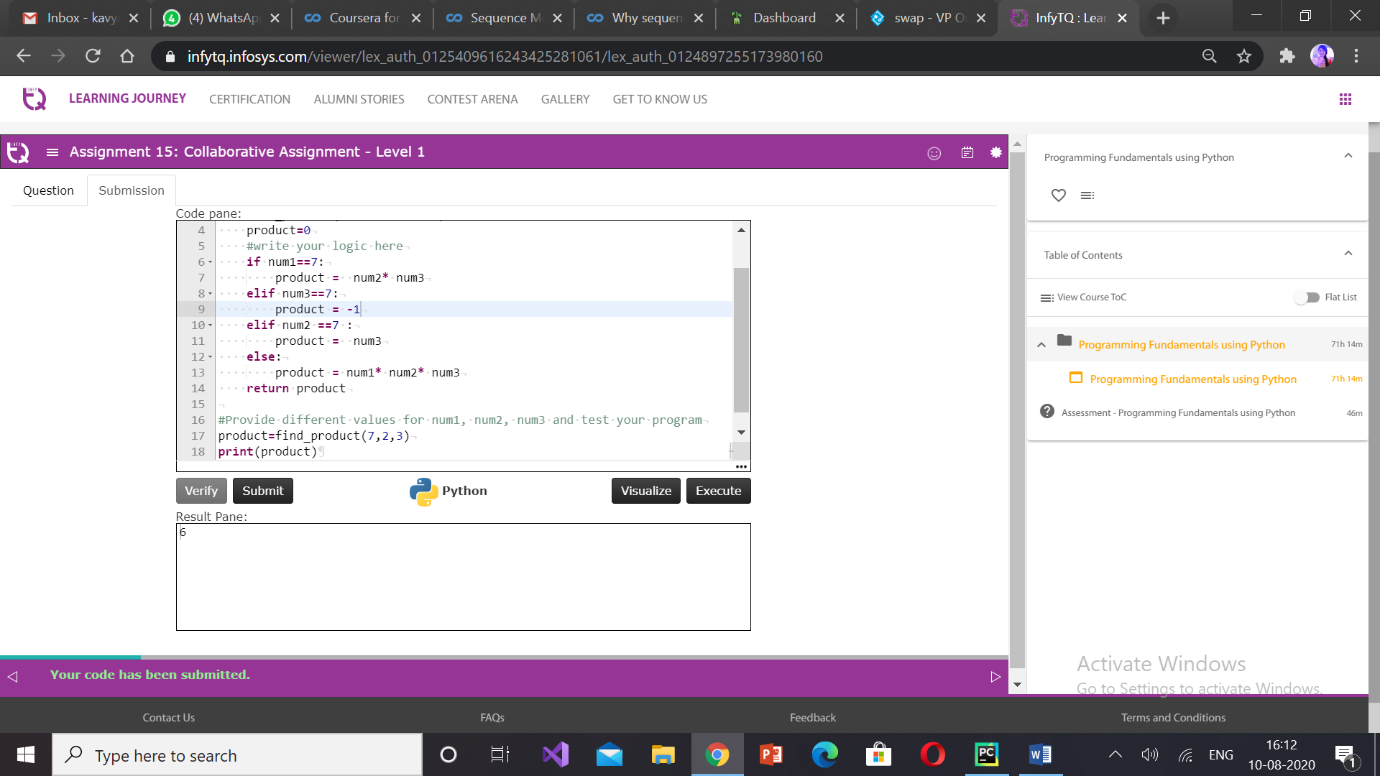
Question 15

Write a python program to find and display the product of three positive integer values based on the rule mentioned below:  
  
It should display the product of the three values except when one of the integer value is 7. In that case, 7 should not be included in the product and the values to its left also should not be included.  
If there is only one value to be considered, display that value itself. If no values can be included in the product, display -1.  
  
**Note**: Assume that if 7 is one of the positive integer values, then it will occur only once. Refer the sample I/O given below.

|  |  |
| --- | --- |
| **Sample Input** | **Expected Output** |
| 1, 5, 3 | 15 |
| 3, 7, 8 | 8 |
| 7, 4, 3 | 12 |
| 1, 5, 7 | -1 |

**Estimated Time: 20 minutes**

Question 16( temporary not executed)

You have x no. of 5 rupee coins and y no. of 1 rupee coins. You want to purchase an item for amount z. The shopkeeper wants you to provide exact change. You want to pay using minimum number of coins. How many 5 rupee coins and 1 rupee coins will you use? If exact change is not possible then display -1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sample Input** | | | **Expected Output** | |
| **Available Rs. 1 coins** | **Available Rs. 5 notes** | **Amount to be made** | **Rs. 1 coins needed** | **Rs. 5 notes needed** |
| 2 | 4 | 21 | 1 | 4 |
| 11 | 2 | 11 | 1 | 2 |
| 3 | 3 | 19 | -1 | |

#PF-Assgn-16

def make\_amount(rupees\_to\_make,no\_of\_five,no\_of\_one):

five\_needed=0

one\_needed=0

remaining\_fives = no\_of\_five

remaining\_ones = no\_of\_one

#Start writing your code here

#Populate the variables: five\_needed and one\_needed

five\_needed = rupees\_to\_make // 5

one\_needed = rupees\_to\_make % 5

if five\_needed> remaining\_fives:

one\_needed += 5 \* (five\_needed - remaining\_fives)

remaining\_fives = 0

else:

remaining\_fives -= five\_needed

if one\_needed > remaining\_ones:

return -1

else:

remaining\_ones -= one\_needed

# Use the below given print statements to display the output

# Also, do not modify them for verification to work

print("No. of Five needed :", no\_of\_five - remaining\_fives)

print("No. of One needed :", no\_of\_one - remaining\_ones)

#print(-1)

#Provide different values for rupees\_to\_make,no\_of\_five,no\_of\_one and test your program

make\_amount(28,8,5)

Question 18

A traveler on a visit to India is in need of some Indian Rupees (INR) but he has money belonging to another currency. He wants to know how much money he should provide in the currency he has, to get the specified amount in INR.

Write a python program to implement a currency calculator which accepts the amount needed in INR and the name of the currency which the traveler has. The program should identify and display the amount the traveler should provide in the currency he has, to get the specified amount in INR.

Note: Use the forex information provided in the table below for the calculation. Consider that only the currency names mentioned in the table are valid. For any invalid currency name, display -1.

| **Currency** | **Equivalent of 1.00 INR** |
| --- | --- |
| Euro | 0.01417 |
| British Pound | 0.0100 |
| Australian Dollar | 0.02140 |
| Canadian Dollar | 0.02027 |

Also identify the test data and use it to test the program.

#PF-Tryout

def convert\_currency(amount\_needed\_inr,current\_currency\_name):

current\_currency\_amount=0

#write your logic here

if current\_currency\_name =="Euro":

current\_currency\_amount = amount\_needed\_inr\*0.01417

elif current\_currency\_name == "British Pound":

current\_currency\_amount = amount\_needed\_inr\*0.0100

elif current\_currency\_name == "Australian Dollar":

current\_currency\_amount = amount\_needed\_inr\*0.02140

elif current\_currency\_name == "Canadian Dollar":

current\_currency\_amount = amount\_needed\_inr\*0.02027

else:

current\_currency\_amount=-1

return current\_currency\_amount

#Provide different values for amount\_needed\_inr,current\_currency\_name and test your program

currency\_needed=convert\_currency(3500,"British Pound")

if(currency\_needed!= -1):

print(currency\_needed )

else:

print("Invalid currency name")

Question 19:

The below information must be used to check the validity of the data provided by the customer:

Type of A vegetarian combo costs Rs.120 per plate and a non-vegetarian combo costs Rs.150 per plate. Their non-veg combo is really famous that they get more orders for their non-vegetarian combo than the vegetarian combo.  
Apart from the cost per plate of food, customers are also charged for home delivery based on the distance in kms from the restaurant to the delivery point. The delivery charges are as mentioned below:

| **Distance in kms** | **Delivery charge in Rs per km** |
| --- | --- |
| For first 3kms | 0 |
| For next 3kms | 3 |
| For the remaining | 6 |

* Given the type of food, quantity (no. of plates) and the distance in kms from the restaurant to the delivery point, write a python program to calculate the final bill amount to be paid by a customer  
  food must be ‘V’ for vegetarian and ‘N’ for non-vegetarian.
* Distance in kms must be greater than 0.
* Quantity ordered should be minimum 1.

If any of the input is invalid, the bill amount should be considered as -1.

#PF-Assgn-19

def calculate\_bill\_amount(food\_type,quantity\_ordered,distance\_in\_kms):

bill\_amount=0

food\_amt=0

dis=0

#write your logic here

if distance\_in\_kms <=0:

return -1

if quantity\_ordered<=0:

return -1

if food\_type=="N":

food\_amt=150

elif food\_type=="V":

food\_amt=120

else:

return -1

if distance\_in\_kms<=3:

dis=0

elif distance\_in\_kms>3 and distance\_in\_kms<=6:

dis=3\*(distance\_in\_kms-3)

else:

dis=(distance\_in\_kms-6)\*6+9

bill\_amount=food\_amt\*quantity\_ordered+dis

return bill\_amount

#Provide different values for food\_type,quantity\_ordered,distance\_in\_kms and test your program

bill\_amount=calculate\_bill\_amount("V",3,5)

print(bill\_amount)

Question 20:

The Metro Bank provides various types of loans such as car loans, business loans and house loans to its account holders. Write a python program to implement the following requirements:

* Initialize the following variables with appropriate input values:account\_number, account\_balance, salary, loan\_type, loan\_amount\_expected and customer\_emi\_expected.
* The account number should be of 4 digits and its first digit should be 1.
* The customer should have a minimum balance of Rupees 1 Lakh in the account.
* If the above rules are valid, determine the eligible loan amount and the EMI that the bank can provide to its customers based on their salary and the loan type they expect to avail.
* The bank would provide the loan, only if the loan amount and the number of EMI’s requested by the customer is less than or equal to the loan amount and the number of EMI’s decided by the bank respectively.

Display appropriate error messages for all invalid data. If all the business rules are satisfied ,then display account number, eligible and requested loan amount and EMI’s.  
Test your code by providing different values for the input variables.

| **Salary** | **Loan type** | **Eligible loan amount** | **No. of EMI’s required to repay** |
| --- | --- | --- | --- |
| > 25000 | Car | 500000 | 36 |
| > 50000 | House | 6000000 | 60 |
| > 75000 | Business | 7500000 | 84 |

def calculate\_loan(account\_number,salary,account\_balance,loan\_type,loan\_amount\_expected,customer\_emi\_expected):

   eligible\_loan\_amount=0

   bank\_emi\_expected=0

   if account\_number>999 and account\_number<2000:

       if account\_balance>=100000:

           if salary>25000 and loan\_type=="Car":

               eligible\_loan\_amount=500000

               bank\_emi\_expected=36

               if loan\_amount\_expected<=eligible\_loan\_amount and customer\_emi\_expected<=bank\_emi\_expected:

                   print("Account number:", account\_number)

                   print("The customer can avail the amount of Rs.", eligible\_loan\_amount)

                   print("Eligible EMIs :", bank\_emi\_expected)

                   print("Requested loan amount:", loan\_amount\_expected)

                   print("Requested EMI's:",customer\_emi\_expected)

               else:

                   print("The customer is not eligible for the loan")

           elif salary>50000 and loan\_type=="House":

               eligible\_loan\_amount = 6000000

               bank\_emi\_expected = 60

               if loan\_amount\_expected<=eligible\_loan\_amount and customer\_emi\_expected<=bank\_emi\_expected:

                   print("Account number:", account\_number)

                   print("The customer can avail the amount of Rs.", eligible\_loan\_amount)

                   print("Eligible EMIs :", bank\_emi\_expected)

                   print("Requested loan amount:", loan\_amount\_expected)

                   print("Requested EMI's:", customer\_emi\_expected)

               else:

                   print("The customer is not eligible for the loan")

           elif salary>75000 and loan\_type=="Business":

               eligible\_loan\_amount = 7500000

               bank\_emi\_expected = 84

               if loan\_amount\_expected<=eligible\_loan\_amount and customer\_emi\_expected<=bank\_emi\_expected:

                   print("Account number:", account\_number)

                   print("The customer can avail the amount of Rs.", eligible\_loan\_amount)

                   print("Eligible EMIs :", bank\_emi\_expected)

                   print("Requested loan amount:", loan\_amount\_expected)

                   print("Requested EMI's:", customer\_emi\_expected)

               else:

                   print("The customer is not eligible for the loan")

           else:

               print("Invalid loan type or salary")

       else:

           print("Insufficient account balance")

   else:

       print("Invalid account number")

calculate\_loan(1001,40000,250000,"Car",300000,30)

**Question 21:**

Write a python program to generate and display the next date of a given date.  
  
Assume that

* Date is provided as day, month and year as shown in below table.
* The input provided is always valid. Output should be day-month-year.  
  **Hint:** print(day,"-",month,"-",year) will display day-month-year

|  |  |  |
| --- | --- | --- |
|  | **Sample Input** | **Expected Output** |
| Day | 1 | 2-9-2015 |
| Month | 9 |
| Year | 2015 |

Also identify the test data and use it to test the program.

def generate\_next\_date(day,month,year):

#Start writing your code here

if ((year % 4 == 0) and (year % 100!= 0)) or (year%400 == 0):

if month==2:

d1=29

elif month==1 or month==3 or month==5 or month==7 or month==8 or month==10 or month==12:

d1=31

else :

d1=30

else:

if month==2:

d1=28

elif month==1 or month==3 or month==5 or month==7 or month==8 or month==10 or month==12:

d1=31

else :

d1=30

if day==31 and month==12 :

day=1

month=1

year=year+1

elif d1-day==0 :

month=month+1

day=1

else:

day=day+1

print(day,"-",month,"-",year)

generate\_next\_date(30,6,2015)