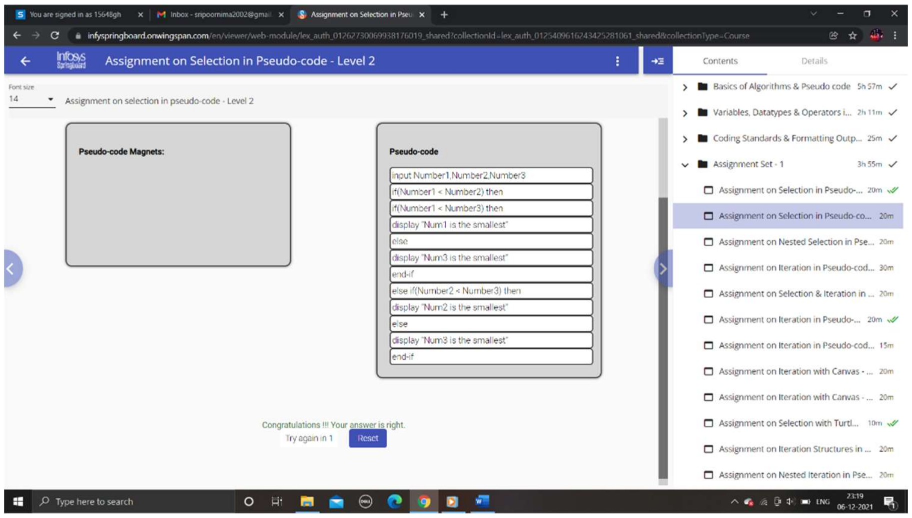
Name: T.Sripoornima
Class: III B.E-CSE A
Rollno: 19BCS027

# **FUNDAMENTALS OF PYTHON - PART 1**

ASSIGNMENT SET – 1:

## 1). Assignment on Selection in Pseudo Code:


Assignment on Selection in Pseudo-code - Level 2

Assignment on selection in pseudo-code - Level 2

Pseudo-code Magnets:

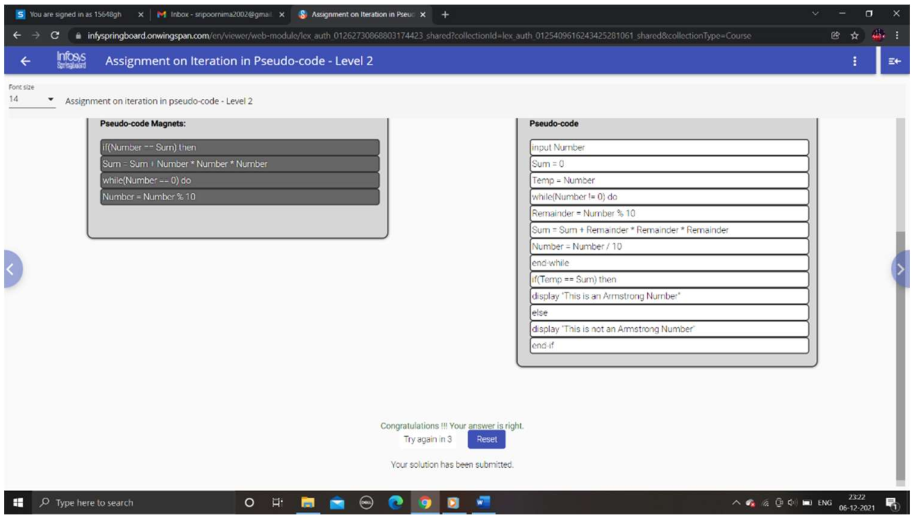
Pseudo-code

```
input Number1,Number2,Number3
if(Number1 < Number2) then
if(Number1 < Number3) then
display "Num1 is the smallest"
else
display "Num3 is the smallest"
end-if
else if(Number2 < Number3) then
display "Num2 is the smallest"
else
display "Num3 is the smallest"
end-if
```

Congratulations !!! Your answer is right.
Try again in 1

Reset
Contents
Details
Basics of Algorithms & Pseudo code
Variables, Datatypes & Operators i...
Coding Standards & Formatting Outp...
Assignment Set - 1
Assignment on Selection in Pseudo-...
Assignment on Selection in Pseudo-co...
Assignment on Nested Selection in Pse...
Assignment on Iteration in Pseudo-cod...
Assignment on Selection & Iteration in ...
Assignment on Iteration in Pseudo-...
Assignment on Iteration in Pseudo-cod...
Assignment on Iteration with Canvas - ...
Assignment on Iteration with Canvas - ...
Assignment on Selection with Turtl...
Assignment on Iteration Structures in ...
Assignment on Nested Iteration in Pse...
Type here to search

## 2). Assignment on Iteration in Pseudo Code-Level2:


Assignment on Iteration in Pseudo-code - Level 2

Assignment on iteration in pseudo-code - Level 2

Pseudo-code Magnets:

```
if(Number == Sum) then
Sum = Sum + Number * Number * Number
while(Number == 0) do
Number = Number % 10
```

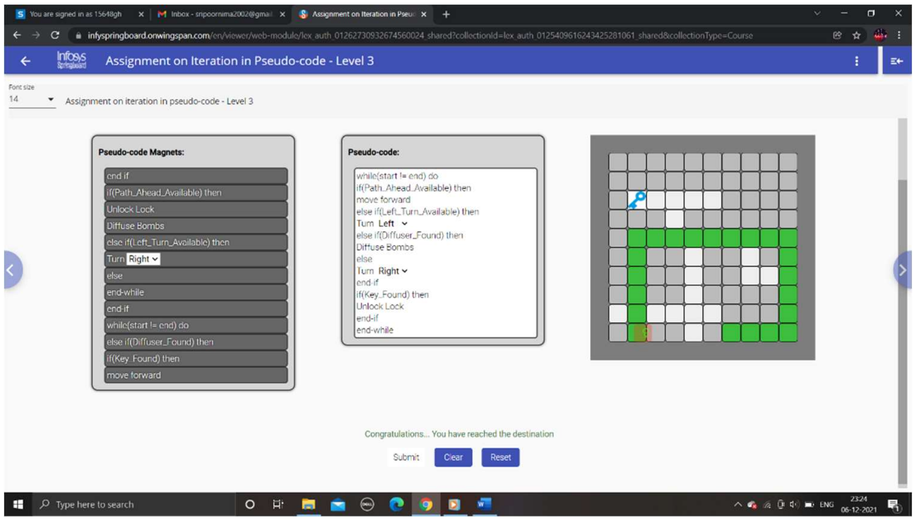
Pseudo-code

```
input Number
Sum = 0
Temp = Number
while(Number != 0) do
Remainder = Number % 10
Sum = Sum + Remainder * Remainder * Remainder
Number = Number / 10
end-while
if(Temp == Sum) then
display "This is an Armstrong Number"
else
display "This is not an Armstrong Number"
end-if
```

Congratulations !!! Your answer is right.
Try again in 3

Your solution has been submitted.

## 3). Assignment on Iteration in Pseudo Code-Level3 :


Assignment on Iteration in Pseudo-code - Level 3

Assignment on iteration in pseudo-code - Level 3

Pseudo-code Magnets:

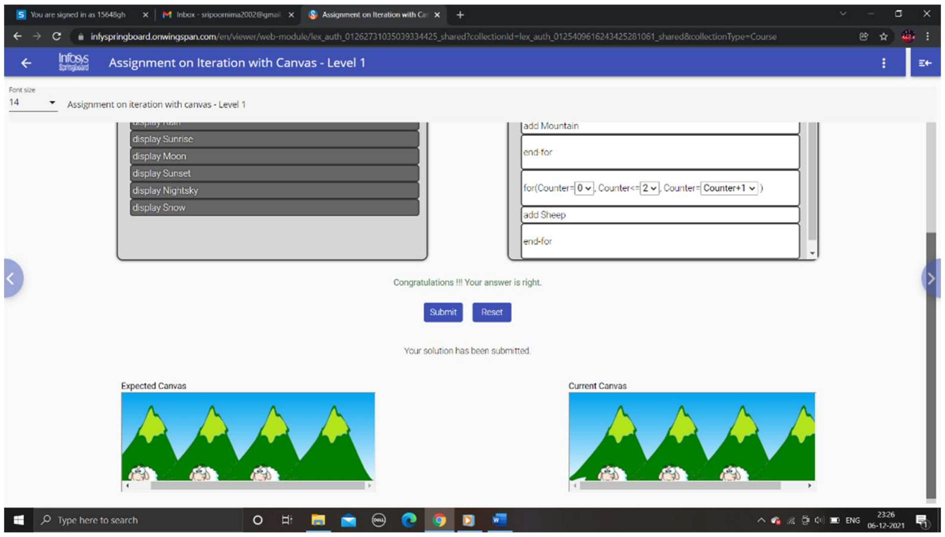
```
end if
if(Path_Ahead_Available) then
Unlock Lock
Diffuse Bombs
else if(Left_Turn_Available) then
Turn Right
else
end-while
end-if
while(start != end) do
else if(Diffuser_Found) then
if(Key_Found) then
move forward
```

Pseudo-code:

```
while(start != end) do
if(Path_Ahead_Available) then
move forward
else if(Left_Turn_Available) then
Turn Left
else if(Diffuser_Found) then
Diffuse Bombs
else
Turn Right
end-if
if(Key_Found) then
Unlock Lock
end-if
end-while
```

Congratulations... You have reached the destination

## 4). Assignment on Iteration with Canvas-level1:


Assignment on Iteration with Canvas - Level 1

Font size 14

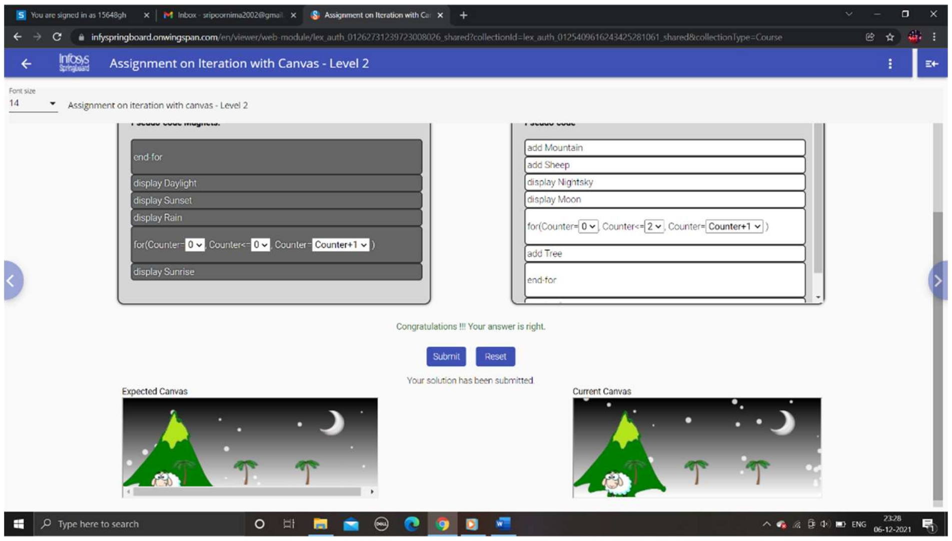
Assignment on iteration with canvas - Level 1

```
add Mountain
end-for
for(Counter=0, Counter<=2, Counter=Counter+1)
add Sheep
end-for
```

Congratulations !!! Your answer is right.

Your solution has been submitted.

## 5). Assignment on Iteration with Canvas-Level2:


Assignment on Iteration with Canvas - Level 2

Font size 14

Assignment on iteration with canvas - Level 2

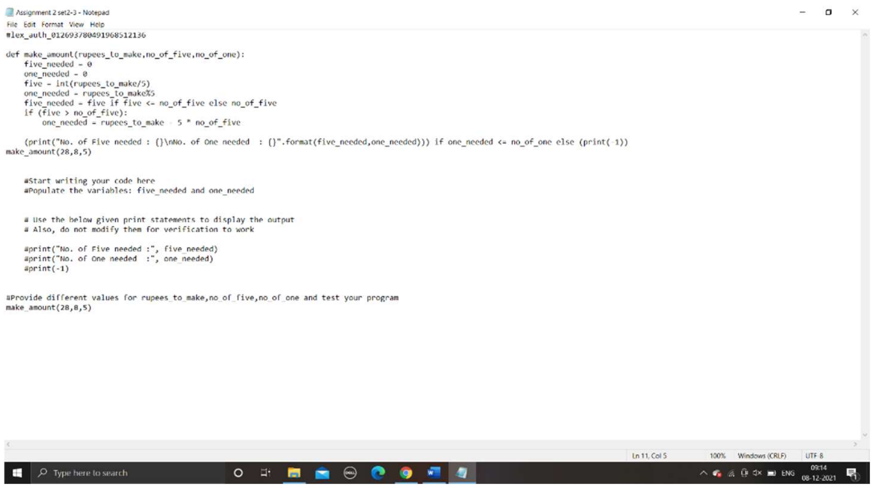
```
add Mountain
add Sheep
display Nightsky
display Moon
for(Counter=0, Counter<=2, Counter=Counter+1)
add Tree
end-for
```

Congratulations !!! Your answer is right.

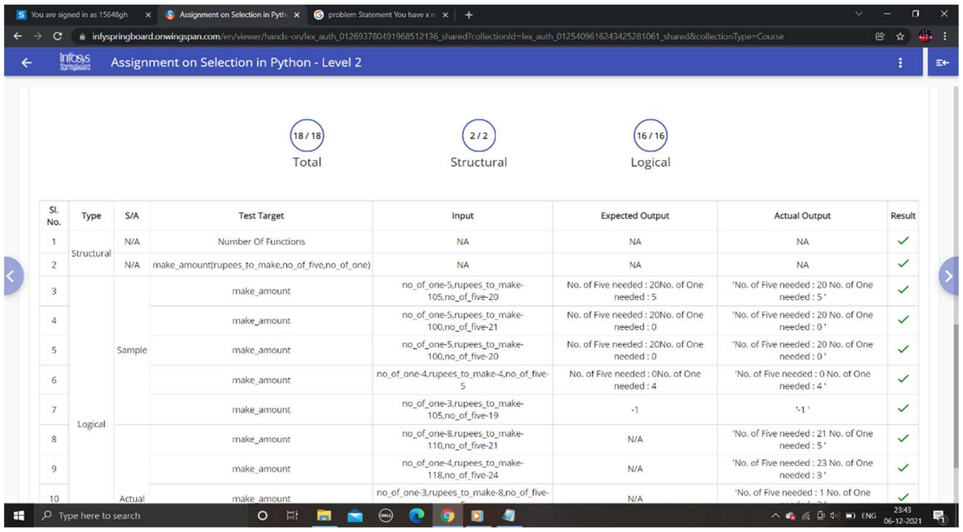
Your solution has been submitted

## ASSIGNMENT SET-2:

## 1). Assignment on Selection in Python Level2:



```
#lex_auth_012693780491968512136

def make_amount(rupees_to_make,no_of_five,no_of_one):
    five_needed = 0
    one_needed = 0
    five = int(rupees_to_make/5)
    one_needed = rupees_to_make%5
    five_needed = five if five <= no_of_five else no_of_five
    if (five > no_of_five):
        one_needed = rupees_to_make - 5 * no_of_five

    (print("No. of Five needed : {}\nNo. of One needed  : {}".format(five_needed,one_needed))) if one_needed <= no_of_one else (print(-1))
make_amount(28,8,5)


    #Start writing your code here
    #Populate the variables: five_needed and 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 :", five_needed)
    #print("No. of One needed  :", one_needed)
    #print(-1)


#Provide different values for rupees_to_make,no_of_five,no_of_one and test your program
make_amount(28,8,5)
```


Assignment on Selection in Python - Level 2

18 / 18 Total

2 / 2 Structural

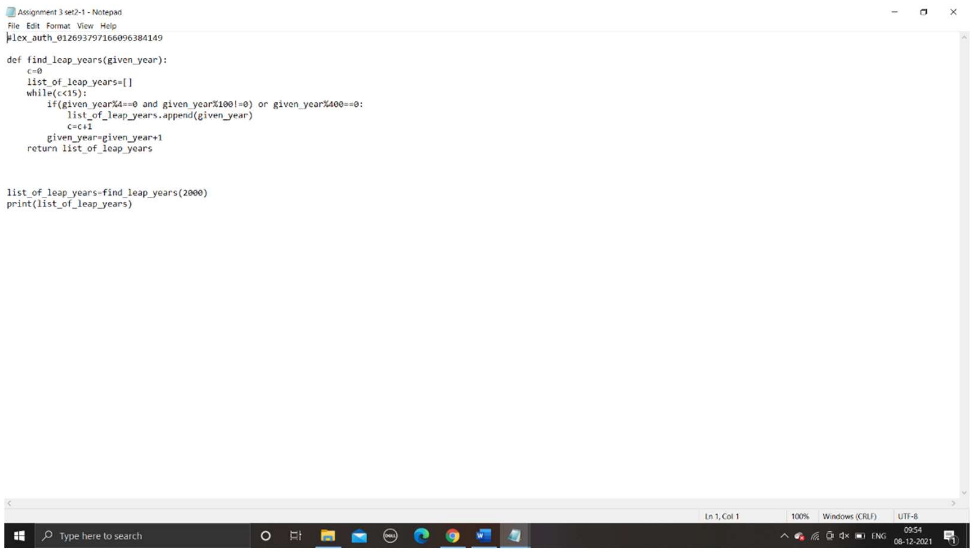
16 / 16 Logical

| Sl. No. | Type | S/A | Test Target | Input | Expected Output | Actual Output | Result |
|---|---|---|---|---|---|---|---|
| 1 | Structural | N/A | Number Of Functions | NA | NA | NA | ✓ |
| 2 | | N/A | make_amount(rupees_to_make,no_of_five,no_of_one) | NA | NA | NA | ✓ |
| 3 | Logical | Sample | make_amount | no_of_one-5,rupees_to_make-105,no_of_five-20 | No. of Five needed : 20No. of One needed : 5 | 'No. of Five needed : 20 No. of One needed : 5 ' | ✓ |
| 4 | | | make_amount | no_of_one-5,rupees_to_make-100,no_of_five-21 | No. of Five needed : 20No. of One needed : 0 | 'No. of Five needed : 20 No. of One needed : 0 ' | ✓ |
| 5 | | | make_amount | no_of_one-5,rupees_to_make-100,no_of_five-20 | No. of Five needed : 20No. of One needed : 0 | 'No. of Five needed : 20 No. of One needed : 0 ' | ✓ |
| 6 | | | make_amount | no_of_one-4,rupees_to_make-4,no_of_five-5 | No. of Five needed : 0No. of One needed : 4 | 'No. of Five needed : 0 No. of One needed : 4 ' | ✓ |
| 7 | | | make_amount | no_of_one-3,rupees_to_make-105,no_of_five-19 | -1 | '-1 ' | ✓ |
| 8 | | Actual | make_amount | no_of_one-8,rupees_to_make-110,no_of_five-21 | N/A | 'No. of Five needed : 21 No. of One needed : 5 ' | ✓ |
| 9 | | | make_amount | no_of_one-4,rupees_to_make-118,no_of_five-24 | N/A | 'No. of Five needed : 23 No. of One needed : 3 ' | ✓ |
| 10 | | | make_amount | no_of_one-3,rupees_to_make-8,no_of_five- | N/A | 'No. of Five needed : 1 No. of One | ✓ |

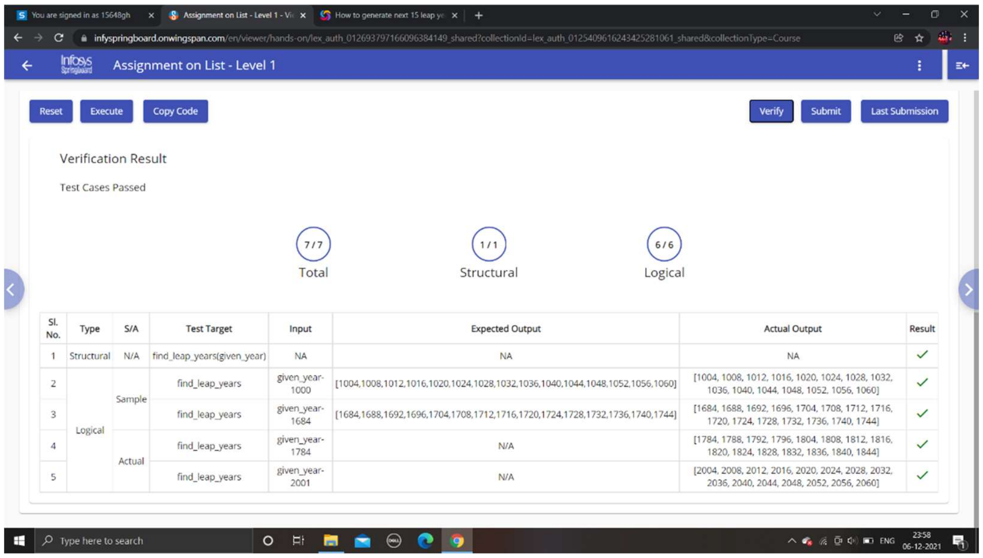
## Code:

```
#lex_auth_012693780491968512136

def make_amount(rupees_to_make,no_of_five,no_of_one):

    five_needed = 0

    one_needed = 0

    five = int(rupees_to_make/5)

    one_needed = rupees_to_make%5

    five_needed = five if five <= no_of_five else no_of_five

    if (five > no_of_five):

        one_needed = rupees_to_make - 5 * no_of_five


    (print("No. of Five needed : {}\nNo. of One needed  : {}".format(five_needed,one_needed))) if one_needed <= no_of_one else (print(-1))

make_amount(28,8,5)



    #Start writing your code here

    #Populate the variables: five_needed and 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 :", five_needed)

    #print("No. of One needed  :", one_needed)

    #print(-1)



#Provide different values for rupees_to_make,no_of_five,no_of_one and test your program

make_amount(28,8,5)
```

## 2). Assignment on list-level 1:



```
#lex_auth_012693797166096384149

def find_leap_years(given_year):
    c=0
    list_of_leap_years=[]
    while(c<15):
        if(given_year%4==0 and given_year%100!=0) or given_year%400==0:
            list_of_leap_years.append(given_year)
            c=c+1
        given_year=given_year+1
    return list_of_leap_years


list_of_leap_years=find_leap_years(2000)
print(list_of_leap_years)
```



Assignment on List - Level 1

Verification Result

Test Cases Passed

7 / 7 Total 1 / 1 Structural 6 / 6 Logical

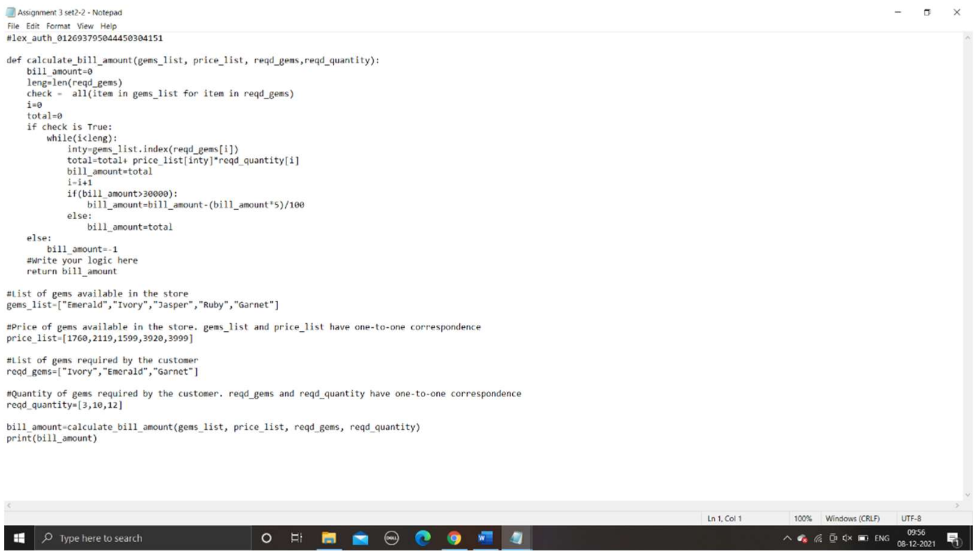
| Sl. No. | Type | S/A | Test Target | Input | Expected Output | Actual Output | Result |
|---|---|---|---|---|---|---|---|
| 1 | Structural | N/A | find_leap_years(given_year) | NA | NA | NA | ✓ |
| 2 | Logical | Sample | find_leap_years | given_year-1000 | [1004,1008,1012,1016,1020,1024,1028,1032,1036,1040,1044,1048,1052,1056,1060] | [1004, 1008, 1012, 1016, 1020, 1024, 1028, 1032, 1036, 1040, 1044, 1048, 1052, 1056, 1060] | ✓ |
| 3 | | | find_leap_years | given_year-1684 | [1684,1688,1692,1696,1704,1708,1712,1716,1720,1724,1728,1732,1736,1740,1744] | [1684, 1688, 1692, 1696, 1704, 1708, 1712, 1716, 1720, 1724, 1728, 1732, 1736, 1740, 1744] | ✓ |
| 4 | | Actual | find_leap_years | given_year-1784 | N/A | [1784, 1788, 1792, 1796, 1804, 1808, 1812, 1816, 1820, 1824, 1828, 1832, 1836, 1840, 1844] | ✓ |
| 5 | | | find_leap_years | given_year-2001 | N/A | [2004, 2008, 2012, 2016, 2020, 2024, 2028, 2032, 2036, 2040, 2044, 2048, 2052, 2056, 2060] | ✓ |

## Code:

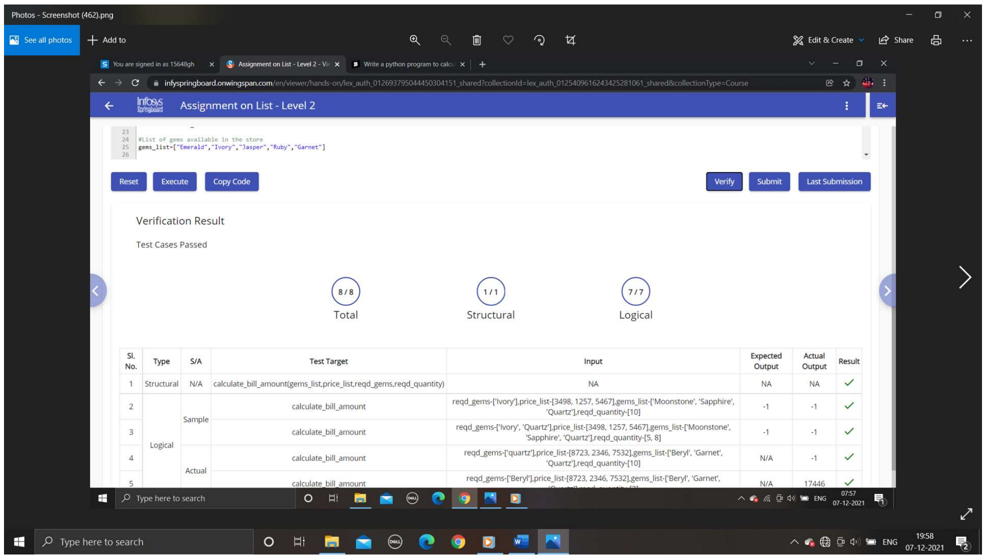
```
#lex_auth_012693797166096384149

def find_leap_years(given_year):

   c=0

   list_of_leap_years=[]

   while(c<15):

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

         list_of_leap_years.append(given_year)

         c=c+1

      given_year=given_year+1

   return list_of_leap_years

list_of_leap_years=find_leap_years(2000)

print(list_of_leap_years)
```

## ASSIGNMENT SET-3:

## 1). Assignment on list level-3:



```
#lex_auth_012693795044450304151

def calculate_bill_amount(gems_list, price_list, reqd_gems,reqd_quantity):
    bill_amount=0
    leng=len(reqd_gems)
    check =  all(item in gems_list for item in reqd_gems)
    i=0
    total=0
    if check is True:
        while(i<leng):
            inty=gems_list.index(reqd_gems[i])
            total=total+ price_list[inty]*reqd_quantity[i]
            bill_amount=total
            i=i+1
            if(bill_amount>30000):
                bill_amount=bill_amount-(bill_amount*5)/100
            else:
                bill_amount=total
    else:
        bill_amount=-1
    #Write your logic here
    return bill_amount

#List of gems available in the store
gems_list=["Emerald","Ivory","Jasper","Ruby","Garnet"]

#Price of gems available in the store. gems_list and price_list have one-to-one correspondence
price_list=[1760,2119,1599,3920,3999]

#List of gems required by the customer
reqd_gems=["Ivory","Emerald","Garnet"]

#Quantity of gems required by the customer. reqd_gems and reqd_quantity have one-to-one correspondence
reqd_quantity=[3,10,12]

bill_amount=calculate_bill_amount(gems_list, price_list, reqd_gems, reqd_quantity)
print(bill_amount)
```



Assignment on List - Level 2

```
#List of gems available in the store
gems_list=["Emerald","Ivory","Jasper","Ruby","Garnet"]
```

Verification Result

Test Cases Passed

8 / 8 Total — 1 / 1 Structural — 7 / 7 Logical

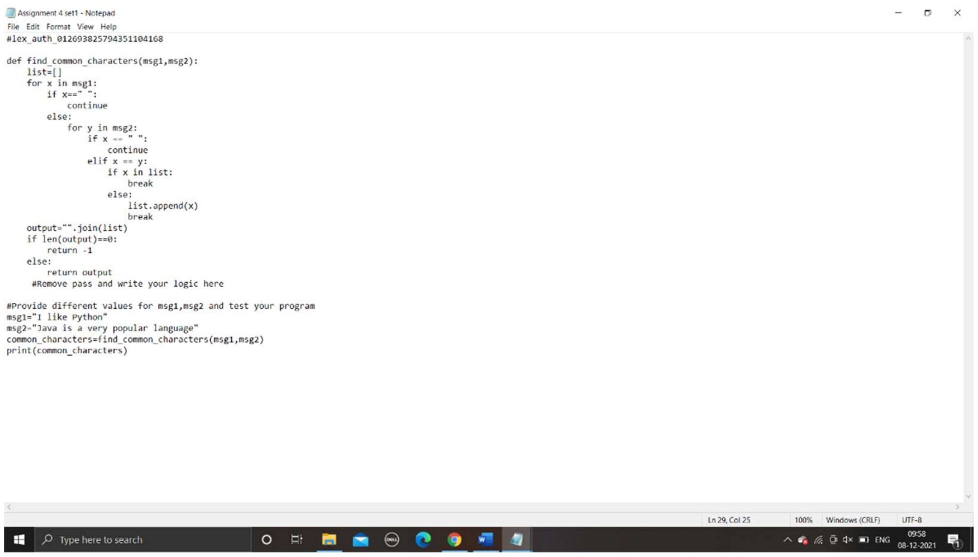
| Sl. No. | Type | S/A | Test Target | Input | Expected Output | Actual Output | Result |
|---|---|---|---|---|---|---|---|
| 1 | Structural | N/A | calculate_bill_amount(gems_list,price_list,reqd_gems,reqd_quantity) | NA | NA | NA | ✓ |
| 2 | Logical | Sample | calculate_bill_amount | reqd_gems-['Ivory'],price_list-[3498, 1257, 5467],gems_list-['Moonstone', 'Sapphire', 'Quartz'],reqd_quantity-[10] | -1 | -1 | ✓ |
| 3 | | | calculate_bill_amount | reqd_gems-['Ivory', 'Quartz'],price_list-[3498, 1257, 5467],gems_list-['Moonstone', 'Sapphire', 'Quartz'],reqd_quantity-[5, 8] | -1 | -1 | ✓ |
| 4 | | Actual | calculate_bill_amount | reqd_gems-['quartz'],price_list-[8723, 2346, 7532],gems_list-['Beryl', 'Garnet', 'Quartz'],reqd_quantity-[10] | N/A | -1 | ✓ |
| 5 | | | calculate_bill_amount | reqd_gems-['Beryl'],price_list-[8723, 2346, 7532],gems_list-['Beryl', 'Garnet', | N/A | 17446 | ✓ |

## Code:

#lex_auth_012693795044450304151

```
def calculate_bill_amount(gems_list, price_list, reqd_gems,reqd_quantity):

    bill_amount=0

    leng=len(reqd_gems)

    check =  all(item in gems_list for item in reqd_gems)

    i=0

    total=0

    if check is True:

        while(i<leng):

            inty=gems_list.index(reqd_gems[i])

            total=total+ price_list[inty]*reqd_quantity[i]

            bill_amount=total

            i=i+1

            if(bill_amount>30000):

                bill_amount=bill_amount-(bill_amount*5)/100

            else:

                bill_amount=total
    else:
        bill_amount=-1
    #Write your logic here
    return bill_amount

#List of gems available in the store
gems_list=["Emerald","Ivory","Jasper","Ruby","Garnet"]

#Price of gems available in the store. gems_list and price_list have one-to-one correspondence
price_list=[1760,2119,1599,3920,3999]

#List of gems required by the customer
reqd_gems=["Ivory","Emerald","Garnet"]

#Quantity of gems required by the customer. reqd_gems and reqd_quantity have one-to-one correspondence
reqd_quantity=[3,10,12]

bill_amount=calculate_bill_amount(gems_list, price_list, reqd_gems, reqd_quantity)
print(bill_amount)
```

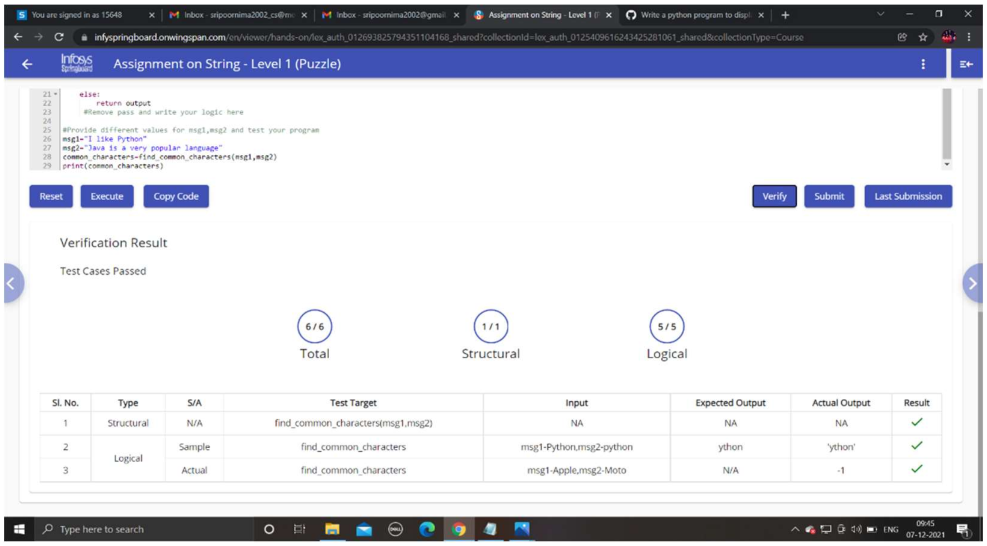
## 2). Assignment on String Level-1:



```
#lex_auth_012693825794351104168

def find_common_characters(msg1,msg2):
    list=[]
    for x in msg1:
        if x==" ":
            continue
        else:
            for y in msg2:
                if x == " ":
                    continue
                elif x == y:
                    if x in list:
                        break
                    else:
                        list.append(x)
                        break
    output="".join(list)
    if len(output)==0:
        return -1
    else:
        return output
     #Remove pass and write your logic here

#Provide different values for msg1,msg2 and test your program
msg1="I like Python"
msg2="Java is a very popular language"
common_characters=find_common_characters(msg1,msg2)
print(common_characters)
```

## Code:

```
#lex_auth_012693825794351104168
def find_common_characters(msg1,msg2):
    list=[]
    for x in msg1:
        if x==" ":
            continue
        else:
            for y in msg2:
                if x == " ":
                    continue
        elif x == y:
          if x in list:
            break
          else:
            list.append(x)
            break
  output="".join(list)
  if len(output)==0:
    return -1
  else:
    return output
   #Remove pass and write your logic here
#Provide different values for msg1,msg2 and test your program
msg1="I like Python"
msg2="Java is a very popular language"
common_characters=find_common_characters(msg1,msg2)
print(common_characters)
```



Assignment on String - Level 1 (Puzzle)

```
    else:
        return output
     #Remove pass and write your logic here

#Provide different values for msg1,msg2 and test your program
msg1="I like Python"
msg2="Java is a very popular language"
common_characters=find_common_characters(msg1,msg2)
print(common_characters)
```

Verification Result

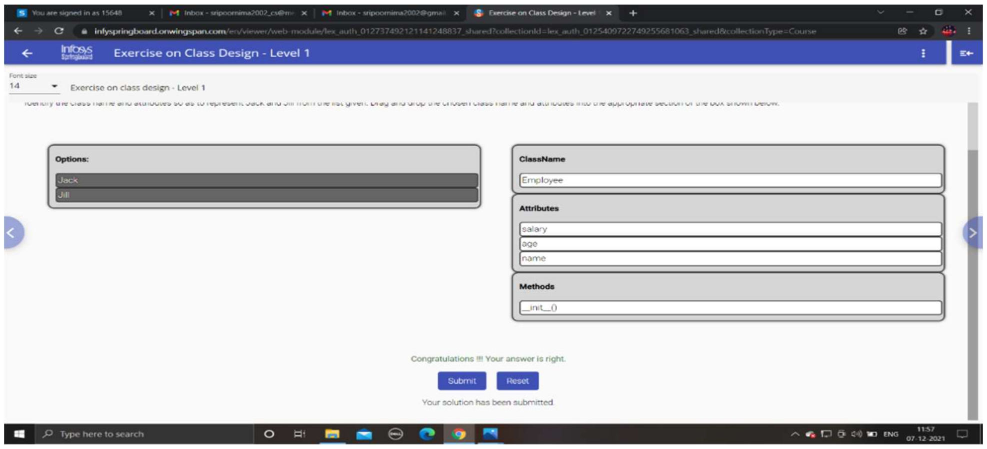
Test Cases Passed

6 / 6 Total 1 / 1 Structural 5 / 5 Logical

| Sl. No. | Type | S/A | Test Target | Input | Expected Output | Actual Output | Result |
|---|---|---|---|---|---|---|---|
| 1 | Structural | N/A | find_common_characters(msg1,msg2) | NA | NA | NA | ✓ |
| 2 | Logical | Sample | find_common_characters | msg1-Python,msg2-python | ython | 'ython' | ✓ |
| 3 | | Actual | find_common_characters | msg1-Apple,msg2-Moto | N/A | -1 | ✓ |

## ASSIGNMENT SET-4:

## 1). Exercise on class design-Level 1:



Exercise on Class Design - Level 1

Exercise on class design - Level 1

Identify the class name and attributes so as to represent Jack and Jill from the list given. Drag and drop the chosen class name and attributes into the appropriate section of the box shown below.

Options:
Jack
Jill

ClassName
Employee

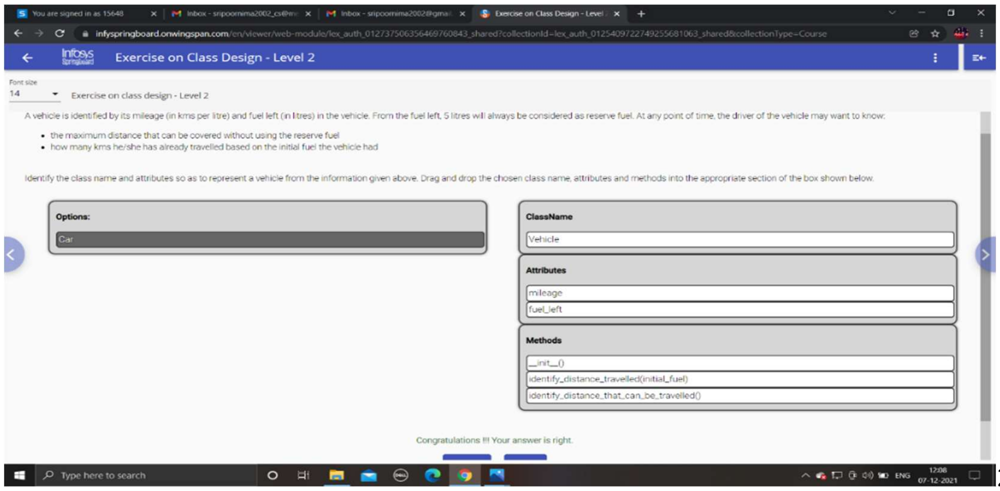
Attributes
salary
age
name

Methods
__init__()

Congratulations !!! Your answer is right.

Your solution has been submitted.

2). Exercise on class design-Level2:



Exercise on Class Design - Level 2

Exercise on class design - Level 2

A vehicle is identified by its mileage (in kms per litre) and fuel left (in litres) in the vehicle. From the fuel left, 5 litres will always be considered as reserve fuel. At any point of time, the driver of the vehicle may want to know:

- the maximum distance that can be covered without using the reserve fuel
- how many kms he/she has already travelled based on the initial fuel the vehicle had

Identify the class name and attributes so as to represent a vehicle from the information given above. Drag and drop the chosen class name, attributes and methods into the appropriate section of the box shown below.

Options: Car

ClassName: Vehicle

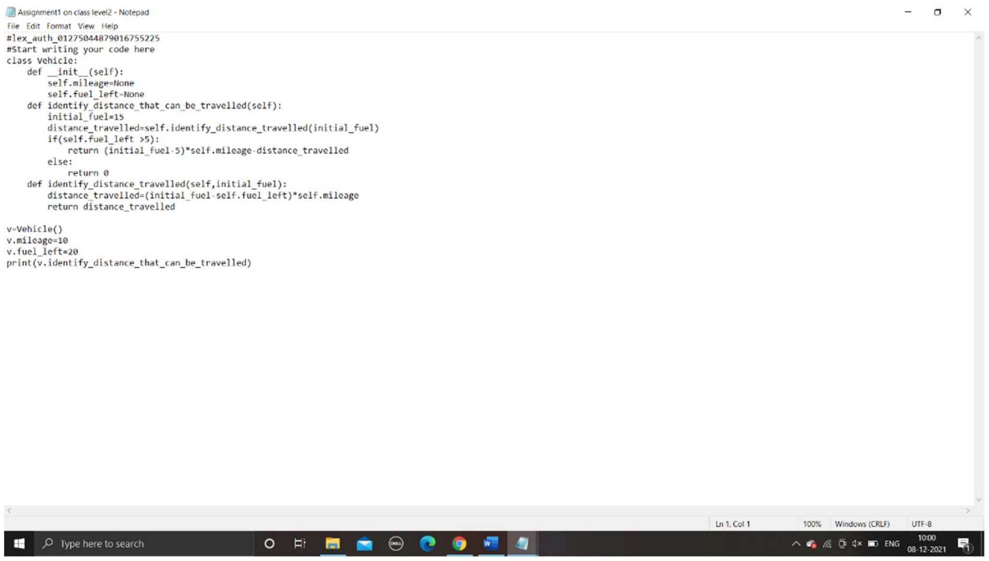
Attributes: mileage, fuel_left

Methods: __init__(), identify_distance_travelled(initial_fuel), identify_distance_that_can_be_travelled()

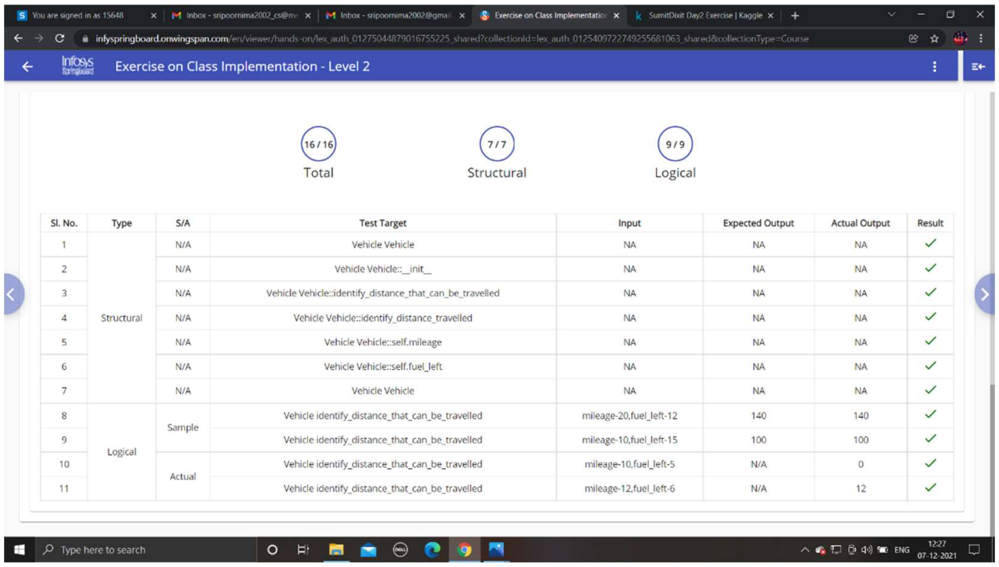
Congratulations !!! Your answer is right.

2).

3). Exercise on class Implementation-Level2:



```
#lex_auth_012750448790167555225
#Start writing your code here
class Vehicle:
    def __init__(self):
        self.mileage=None
        self.fuel_left=None
    def identify_distance_that_can_be_travelled(self):
        initial_fuel=15
        distance_travelled=self.identify_distance_travelled(initial_fuel)
        if(self.fuel_left >5):
            return (initial_fuel-5)*self.mileage-distance_travelled
        else:
            return 0
    def identify_distance_travelled(self,initial_fuel):
        distance_travelled=(initial_fuel-self.fuel_left)*self.mileage
        return distance_travelled

v=Vehicle()
v.mileage=10
v.fuel_left=20
print(v.identify_distance_that_can_be_travelled)
```



Exercise on Class Implementation - Level 2

16 / 16 Total | 7 / 7 Structural | 9 / 9 Logical

| Sl. No. | Type | S/A | Test Target | Input | Expected Output | Actual Output | Result |
|---|---|---|---|---|---|---|---|
| 1 | Structural | N/A | Vehicle Vehicle | NA | NA | NA | ✓ |
| 2 | | N/A | Vehicle Vehicle::__init__ | NA | NA | NA | ✓ |
| 3 | | N/A | Vehicle Vehicle::identify_distance_that_can_be_travelled | NA | NA | NA | ✓ |
| 4 | | N/A | Vehicle Vehicle::identify_distance_travelled | NA | NA | NA | ✓ |
| 5 | | N/A | Vehicle Vehicle::self.mileage | NA | NA | NA | ✓ |
| 6 | | N/A | Vehicle Vehicle::self.fuel_left | NA | NA | NA | ✓ |
| 7 | | N/A | Vehicle Vehicle | NA | NA | NA | ✓ |
| 8 | Logical | Sample | Vehicle identify_distance_that_can_be_travelled | mileage-20,fuel_left-12 | 140 | 140 | ✓ |
| 9 | | | Vehicle identify_distance_that_can_be_travelled | mileage-10,fuel_left-15 | 100 | 100 | ✓ |
| 10 | | Actual | Vehicle identify_distance_that_can_be_travelled | mileage-10,fuel_left-5 | N/A | 0 | ✓ |
| 11 | | | Vehicle identify_distance_that_can_be_travelled | mileage-12,fuel_left-6 | N/A | 12 | ✓ |

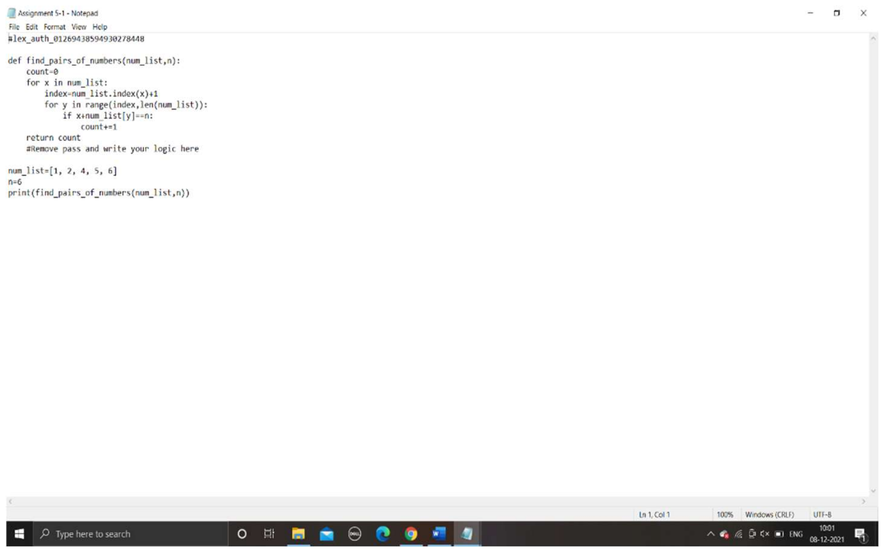
Code:

```
#lex_auth_01275044879016755225

#Start writing your code here

class Vehicle:

    def __init__(self):

        self.mileage=None

        self.fuel_left=None

    def identify_distance_that_can_be_travelled(self):

        initial_fuel=15

        distance_travelled=self.identify_distance_travelled(initial_fuel)

        if(self.fuel_left >5):

            return (initial_fuel-5)*self.mileage-distance_travelled

        else:

            return 0

    def identify_distance_travelled(self,initial_fuel):

        distance_travelled=(initial_fuel-self.fuel_left)*self.mileage

        return distance_travelled


v=Vehicle()

v.mileage=10

v.fuel_left=20

print(v.identify_distance_that_can_be_travelled)
```

## **PYTHON FUNDAMENTALS – PART 2:**

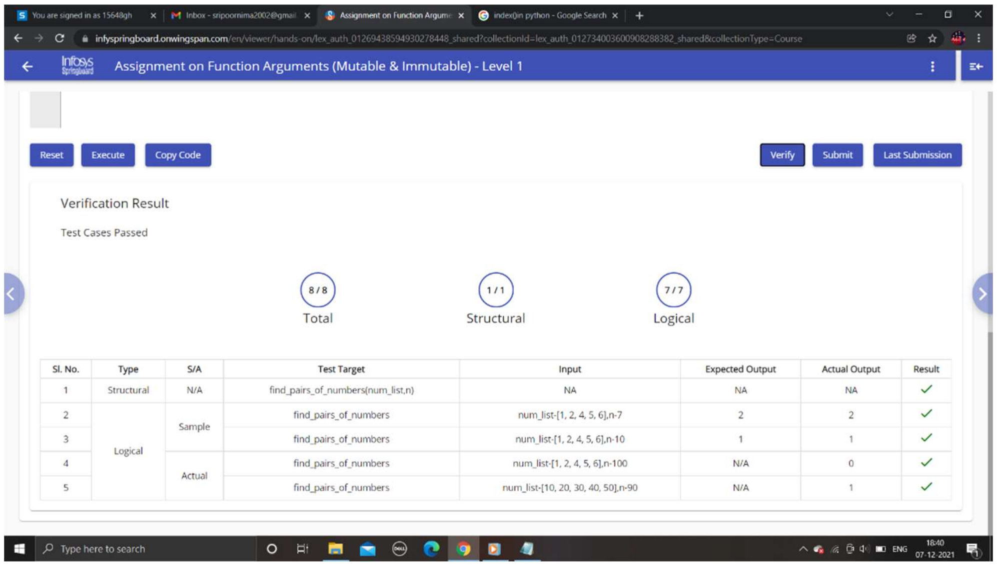
## ASSIGNMENT SET-1:

### 1). Assignment on Function Arguments:



Assignment 5-1 - Notepad

```
#lex_auth_01269438594930278448

def find_pairs_of_numbers(num_list,n):
    count=0
    for x in num_list:
        index=num_list.index(x)+1
        for y in range(index,len(num_list)):
            if x+num_list[y]==n:
                count+=1
    return count
    #Remove pass and write your logic here

num_list=[1, 2, 4, 5, 6]
n=6
print(find_pairs_of_numbers(num_list,n))
```


Assignment on Function Arguments (Mutable & Immutable) - Level 1
Reset
Execute
Copy Code
Verify
Submit
Last Submission

Verification Result

Test Cases Passed

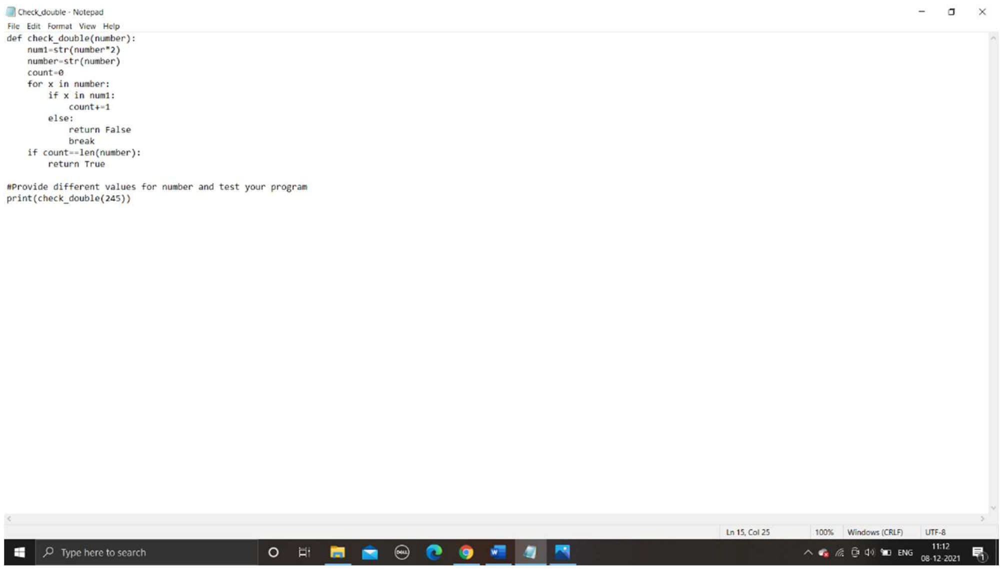
8 / 8 Total
1 / 1 Structural
7 / 7 Logical

| Sl. No. | Type | S/A | Test Target | Input | Expected Output | Actual Output | Result |
|---|---|---|---|---|---|---|---|
| 1 | Structural | N/A | find_pairs_of_numbers(num_list,n) | NA | NA | NA | ✓ |
| 2 | Logical | Sample | find_pairs_of_numbers | num_list-[1, 2, 4, 5, 6],n-7 | 2 | 2 | ✓ |
| 3 | | | find_pairs_of_numbers | num_list-[1, 2, 4, 5, 6],n-10 | 1 | 1 | ✓ |
| 4 | | Actual | find_pairs_of_numbers | num_list-[1, 2, 4, 5, 6],n-100 | N/A | 0 | ✓ |
| 5 | | | find_pairs_of_numbers | num_list-[10, 20, 30, 40, 50],n-90 | N/A | 1 | ✓ |

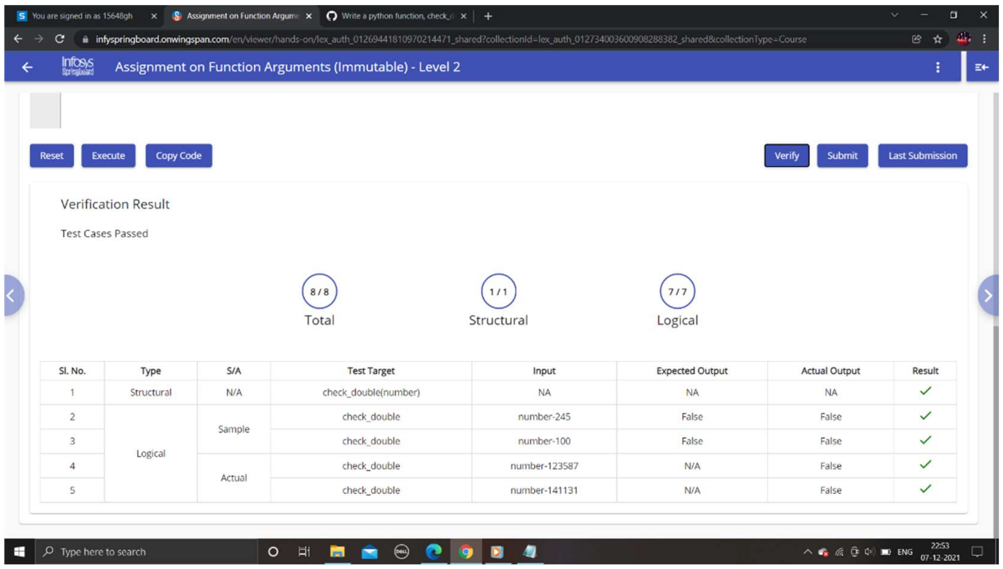
## Code:

```
#lex_auth_012694385949302784448

def find_pairs_of_numbers(num_list,n):

   count=0

   for x in num_list:

       index=num_list.index(x)+1

       for y in range(index,len(num_list)):

           if x+num_list[y]==n:

               count+=1

   return count

   #Remove pass and write your logic here


num_list=[1, 2, 4, 5, 6]

n=6

print(find_pairs_of_numbers(num_list,n))
```

## 2). Assignment on Function Arguments(Immutable):



```
def check_double(number):
    num1=str(number*2)
    number=str(number)
    count=0
    for x in number:
        if x in num1:
            count+=1
        else:
            return False
            break
    if count==len(number):
        return True

#Provide different values for number and test your program
print(check_double(245))
```



Verification Result

Test Cases Passed

8 / 8 Total    1 / 1 Structural    7 / 7 Logical

| Sl. No. | Type | S/A | Test Target | Input | Expected Output | Actual Output | Result |
|---|---|---|---|---|---|---|---|
| 1 | Structural | N/A | check_double(number) | NA | NA | NA | ✓ |
| 2 | Logical | Sample | check_double | number-245 | False | False | ✓ |
| 3 | | | check_double | number-100 | False | False | ✓ |
| 4 | | Actual | check_double | number-123587 | N/A | False | ✓ |
| 5 | | | check_double | number-141131 | N/A | False | ✓ |

## Code:

```
#lex_auth_01269441810970214471

def check_double(number):

   num1=str(number*2)

   number=str(number)

   count=0

   for x in number:

      if x in num1:

         count+=1

      else:

         return False

         break

   if count==len(number):

      return True

   #Remove pass and write your logic here


#Provide different values for number and test your program

print(check_double(125874))
```

## ASSIGNMENT SET-2:

## 1). Assignment on Local Scope-Level1:

## Code:

```
#lex_auth_012693816779112448160

def calculate(distance,no_of_passengers):

   cost=rest=0

   cost=70*distance/10

   rest = no_of_passengers * 80

   if cost <= rest :

      return abs(cost - rest)

   else :

      return -1
```

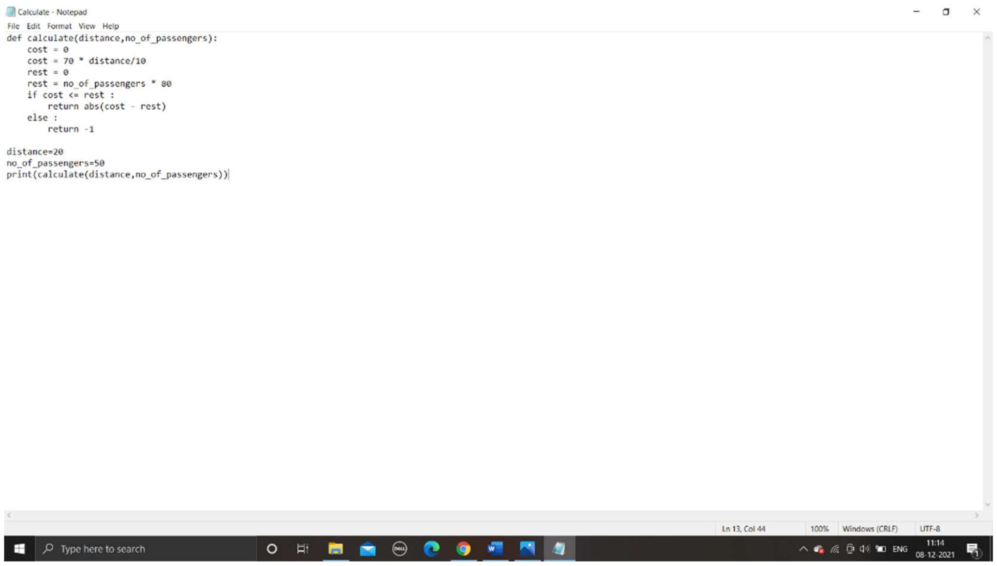
    #Remove pass and write your logic here

 #Provide different values for distance, no_of_passenger and test your program

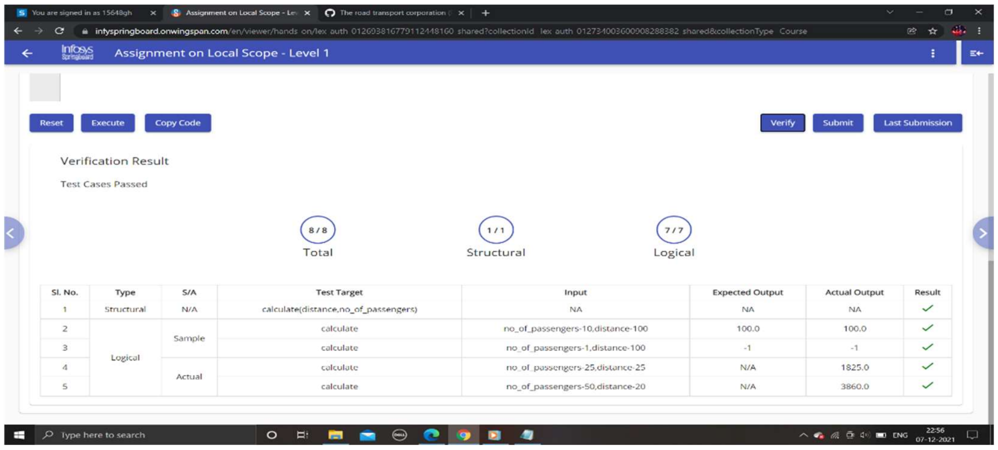
distance=20

no_of_passengers=50

print(calculate(distance,no_of_passengers))



```
def calculate(distance,no_of_passengers):
    cost = 0
    cost = 70 * distance/10
    rest = 0
    rest = no_of_passengers * 80
    if cost <= rest :
        return abs(cost - rest)
    else :
        return -1

distance=20
no_of_passengers=50
print(calculate(distance,no_of_passengers))
```



Assignment on Local Scope - Level 1

Verification Result

Test Cases Passed

8 / 8 Total    1 / 1 Structural    7 / 7 Logical

| Sl. No. | Type | S/A | Test Target | Input | Expected Output | Actual Output | Result |
|---|---|---|---|---|---|---|---|
| 1 | Structural | N/A | calculate(distance,no_of_passengers) | NA | NA | NA | ✓ |
| 2 | Logical | Sample | calculate | no_of_passengers-10,distance-100 | 100.0 | 100.0 | ✓ |
| 3 |  |  | calculate | no_of_passengers-1,distance-100 | -1 | -1 | ✓ |
| 4 |  | Actual | calculate | no_of_passengers-25,distance-25 | N/A | 1825.0 | ✓ |
| 5 |  |  | calculate | no_of_passengers-50,distance-20 | N/A | 3860.0 | ✓ |

## <u>2). Assignment on Recursion-Level1</u>:

## <u>Code:</u>

#lex_auth_01269442114344550475

def is_palindrome(word):

    if (len(word) < 1):

        return True

    else:

        if word[0] == word[-1]:

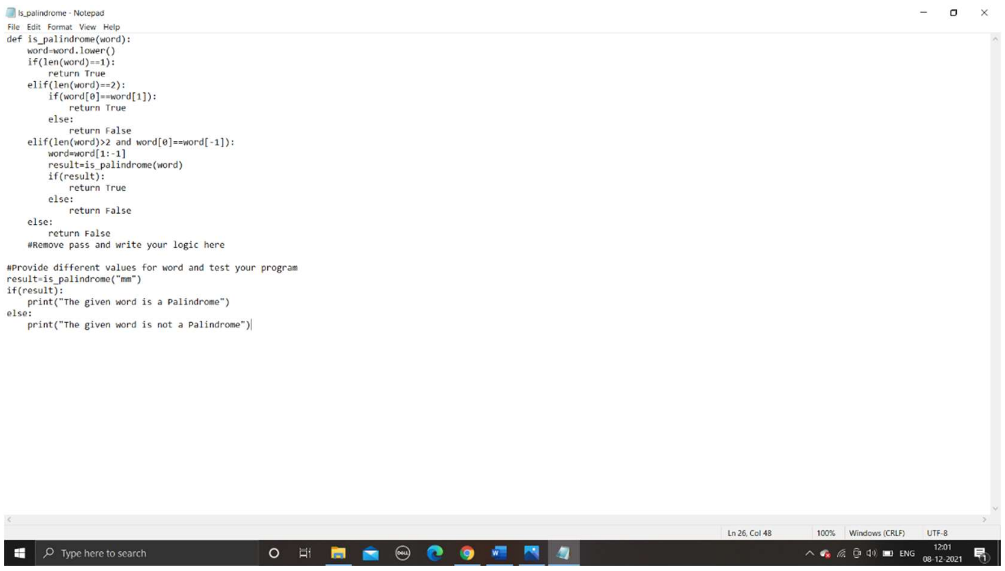
            return is_palindrome(word[1:-1])

        else:

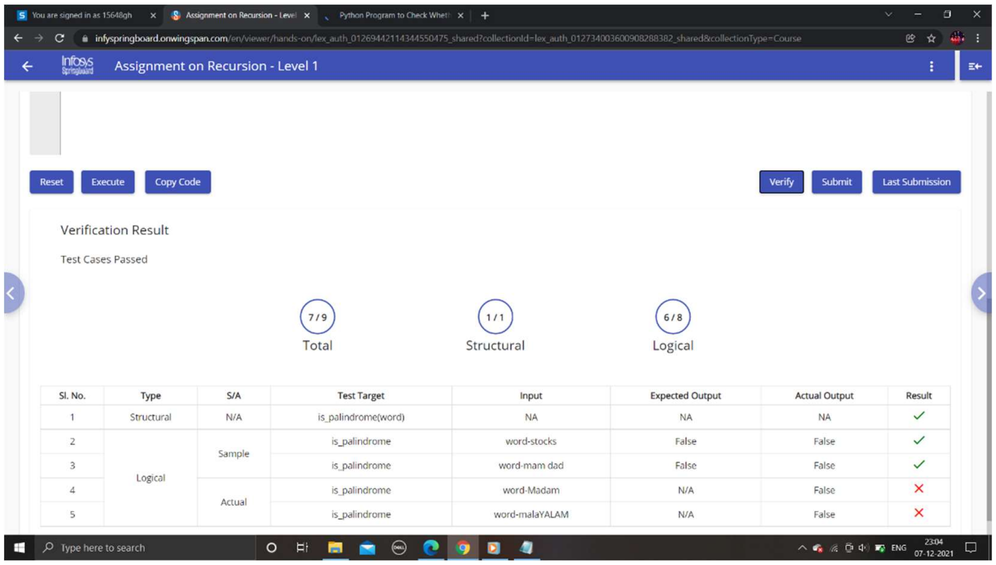
            return False

    #Remove pass and write your logic here

```
#Provide different values for word and test your program

result=is_palindrome("MadAMa")

if(result):

    print("The given word is a Palindrome")

else:

    print("The given word is not a Palindrome")
```



```
def is_palindrome(word):
    word=word.lower()
    if(len(word)==1):
        return True
    elif(len(word)==2):
        if(word[0]==word[1]):
            return True
        else:
            return False
    elif(len(word)>2 and word[0]==word[-1]):
        word=word[1:-1]
        result=is_palindrome(word)
        if(result):
            return True
        else:
            return False
    else:
        return False
    #Remove pass and write your logic here

#Provide different values for word and test your program
result=is_palindrome("mm")
if(result):
    print("The given word is a Palindrome")
else:
    print("The given word is not a Palindrome")
```



Assignment on Recursion - Level 1

Verification Result

Test Cases Passed

7 / 9 Total

1 / 1 Structural

6 / 8 Logical

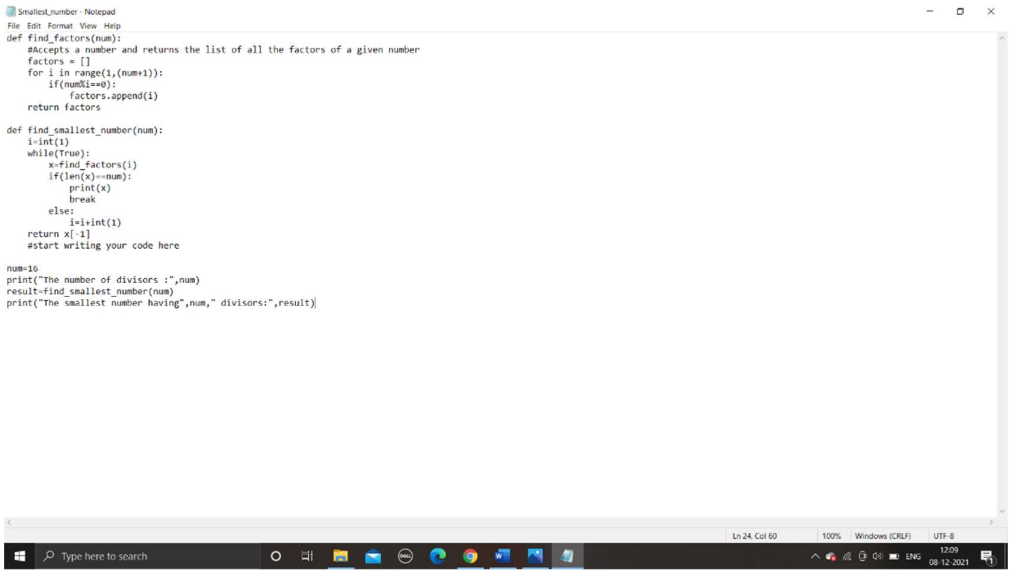
| Sl. No. | Type | S/A | Test Target | Input | Expected Output | Actual Output | Result |
|---|---|---|---|---|---|---|---|
| 1 | Structural | N/A | is_palindrome(word) | NA | NA | NA | ✓ |
| 2 | Logical | Sample | is_palindrome | word-stocks | False | False | ✓ |
| 3 | | | is_palindrome | word-mam dad | False | False | ✓ |
| 4 | | Actual | is_palindrome | word-Madam | N/A | False | ✗ |
| 5 | | | is_palindrome | word-malaYALAM | N/A | False | ✗ |

## ASSIGNMENT SET-3:

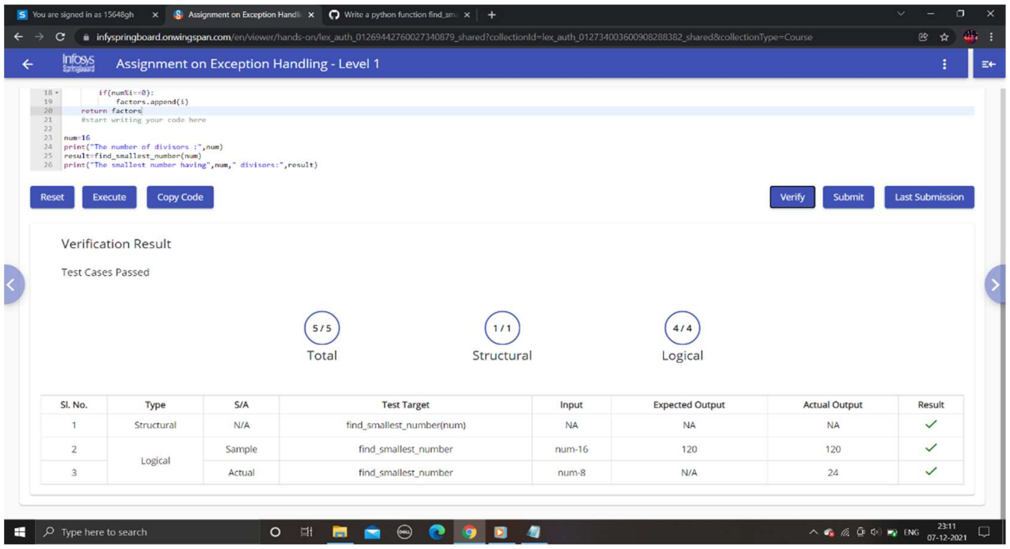
## 1). Assignment on Exception Handling:

## Code:

```
#lex_auth_01269442760027340879

def find_smallest_number(num):

    i=int(1)

    while(True):

        x=find_factors(i)

        if(len(x)==num):

            print(x)

            break
        else:

            i=i+int(1)

    return x[-1]


def find_factors(num):

    #Accepts a number and returns the list of all the factors of a given number

    factors = []

    for i in range(1,(num+1)):

        if(num%i==0):

            factors.append(i)

    return factors

    #start writing your code here


num=16

print("The number of divisors :",num)

result=find_smallest_number(num)

print("The smallest number having",num," divisors:",result)
```



```
def find_factors(num):
    #Accepts a number and returns the list of all the factors of a given number
    factors = []
    for i in range(1,(num+1)):
        if(num%i==0):
            factors.append(i)
    return factors

def find_smallest_number(num):
    i=int(1)
    while(True):
        x=find_factors(i)
        if(len(x)==num):
            print(x)
            break
        else:
            i=i+int(1)
    return x[-1]
    #start writing your code here

num=16
print("The number of divisors :",num)
result=find_smallest_number(num)
print("The smallest number having",num," divisors:",result)
```


You are signed in as 15648gh
Assignment on Exception Handli...
Write a python function find_sm...
infyspringboard.onwingspan.com/en/viewer/hands-on/lex_auth_01269442760027340879_shared?collectionId=lex_auth_01273400360090828382_shared&collectionType=Course
Infosys Springboard
Assignment on Exception Handling - Level 1
Reset
Execute
Copy Code
Verify
Submit
Last Submission
Type here to search
ENG
23:11
07-12-2021

```
        if(num%i==0):
            factors.append(i)
    return factors
    #start writing your code here

num=16
print("The number of divisors :",num)
result=find_smallest_number(num)
print("The smallest number having",num," divisors:",result)
```

Verification Result

Test Cases Passed

5 / 5 Total

1 / 1 Structural

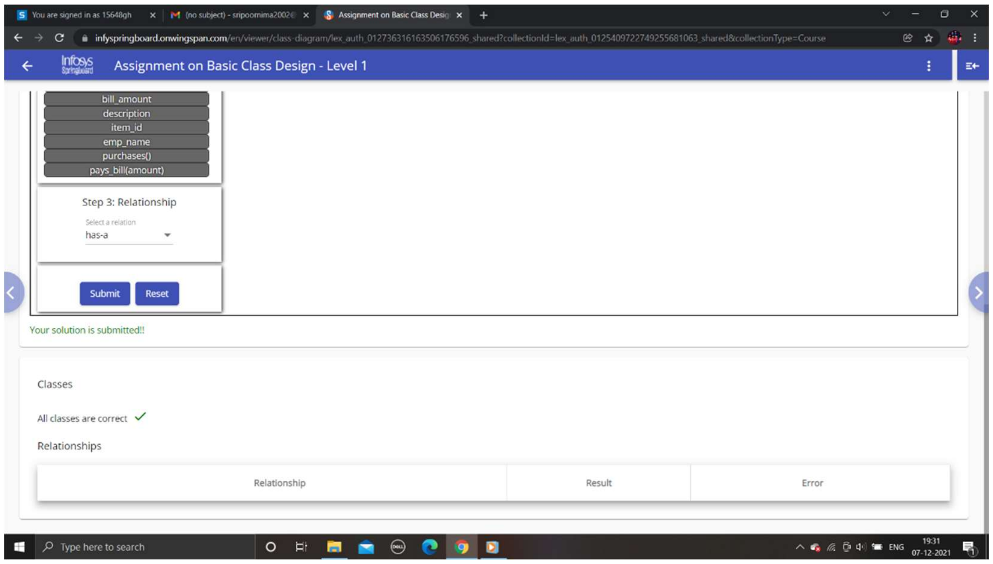
4 / 4 Logical

| Sl. No. | Type | S/A | Test Target | Input | Expected Output | Actual Output | Result |
|---|---|---|---|---|---|---|---|
| 1 | Structural | N/A | find_smallest_number(num) | NA | NA | NA | ✓ |
| 2 | Logical | Sample | find_smallest_number | num-16 | 120 | 120 | ✓ |
| 3 |  | Actual | find_smallest_number | num-8 | N/A | 24 | ✓ |

## **<u>OOPS PROGRAMMING IN PYTHON:</u>**

ASSIGNMENT SET-1:

1). Assignment on Basic Class Design-Level1:


Assignment on Basic Class Design - Level 1

bill_amount
description
item_id
emp_name
purchases()
pays_bill(amount)

Step 3: Relationship

Select a relation
has-a

Submit Reset

Your solution is submitted!!

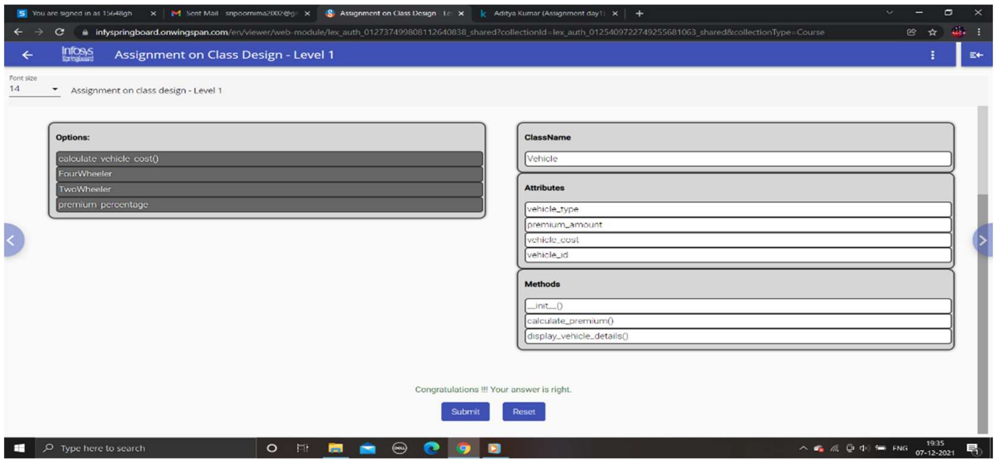
Classes

All classes are correct ✓

Relationships

| Relationship | Result | Error |
|---|---|---|

2). Assignment on Class Design-Level1:


Assignment on Class Design - Level 1

Font size 14

Assignment on class design - Level 1

Options:
calculate_vehicle_cost()
FourWheeler
TwoWheeler
premium_percentage

ClassName
Vehicle

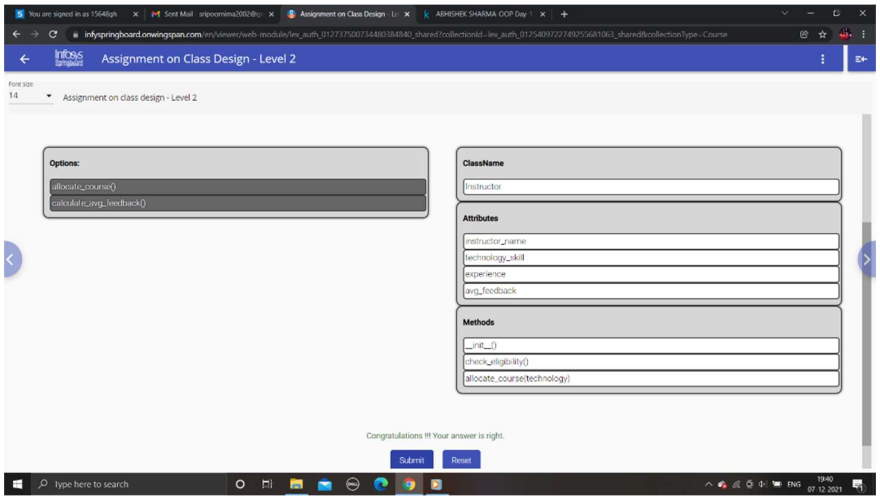
Attributes
vehicle_type
premium_amount
vehicle_cost
vehicle_id

Methods
__init__()
calculate_premium()
display_vehicle_details()

Congratulations !!! Your answer is right.

Submit Reset

3). Assignment on Class Design-Level2:


Assignment on Class Design - Level 2

Font size 14

Assignment on class design - Level 2

Options:
allocate_course()
calculate_avg_feedback()

ClassName
Instructor

Attributes
instructor_name
technology_skill
experience
avg_feedback

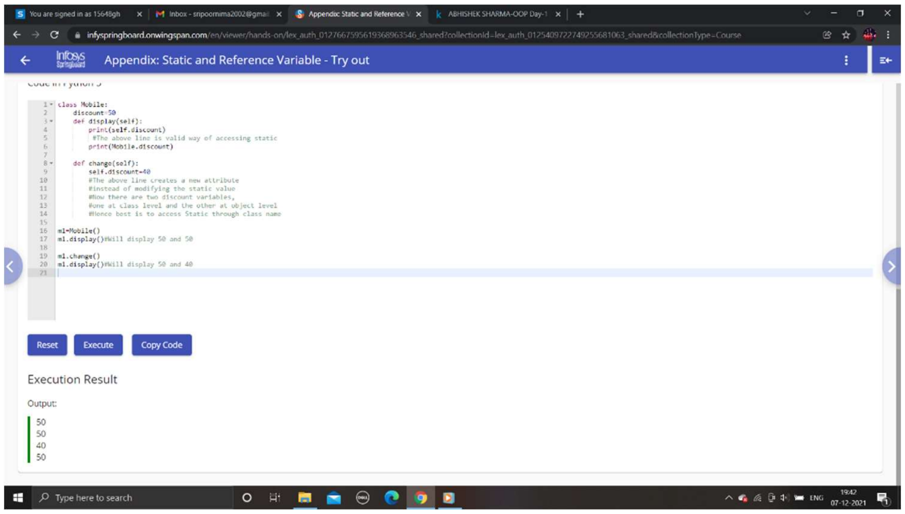
Methods
__init__()
check_eligibility()
allocate_course(technology)

Congratulations !!! Your answer is right.

Submit Reset

## ASSIGNMENT SET-2:

### 1). Static and Reference Variable:



Appendix: Static and Reference Variable - Try out

```python
class Mobile:
    discount=50
    def display(self):
        print(self.discount)
         #The above line is valid way of accessing static
        print(Mobile.discount)

    def change(self):
        self.discount=40
        #The above line creates a new attribute
        #instead of modifying the static value
        #Now there are two discount variables,
        #one at class level and the other at object level
        #Hence best is to access Static through class name

m1=Mobile()
m1.display()#Will display 50 and 50

m1.change()
m1.display()#Will display 50 and 40
```

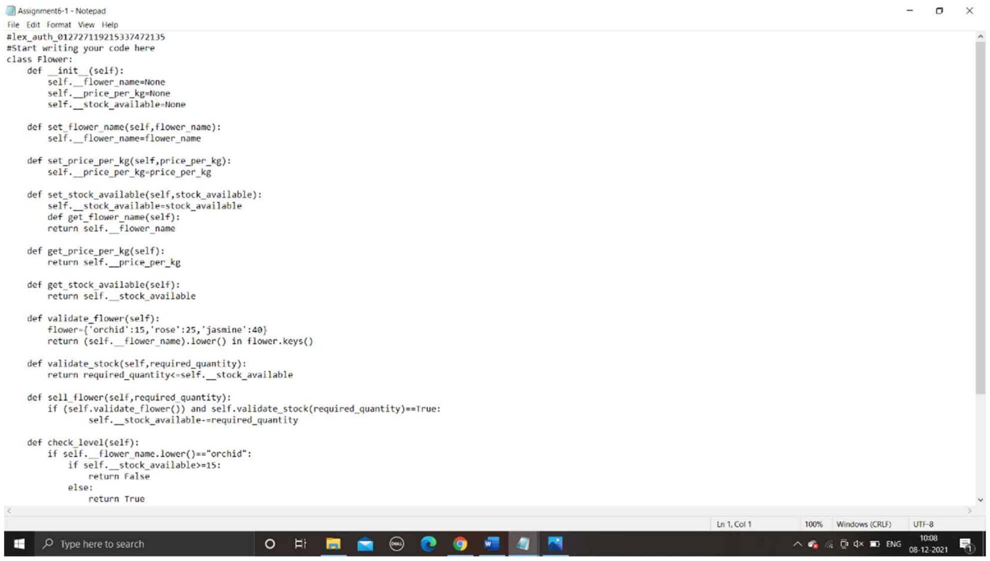
Execution Result

Output:

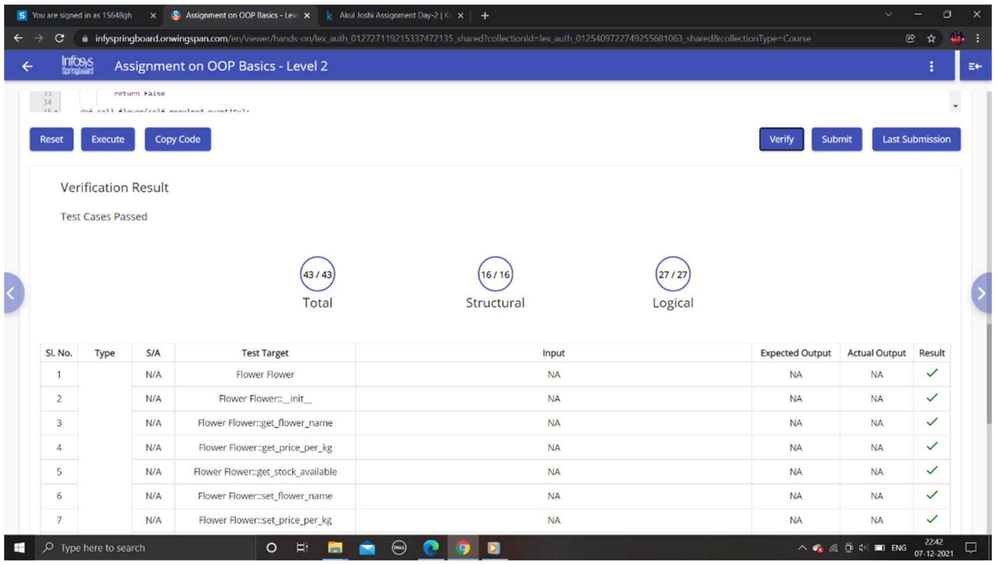
```
50
50
40
50
```

## ASSIGNMENT SET-3:

### 1). Assignment on OOP Basics-Level2:



```python
#lex_auth_0127271192153337472135
#Start writing your code here
class Flower:
    def __init__(self):
        self.__flower_name=None
        self.__price_per_kg=None
        self.__stock_available=None

    def set_flower_name(self,flower_name):
        self.__flower_name=flower_name

    def set_price_per_kg(self,price_per_kg):
        self.__price_per_kg=price_per_kg

    def set_stock_available(self,stock_available):
        self.__stock_available=stock_available
        def get_flower_name(self):
        return self.__flower_name

    def get_price_per_kg(self):
        return self.__price_per_kg

    def get_stock_available(self):
        return self.__stock_available

    def validate_flower(self):
        flower={'orchid':15,'rose':25,'jasmine':40}
        return (self.__flower_name).lower() in flower.keys()

    def validate_stock(self,required_quantity):
        return required_quantity<=self.__stock_available

    def sell_flower(self,required_quantity):
        if (self.validate_flower()) and self.validate_stock(required_quantity)==True:
                self.__stock_available-=required_quantity

    def check_level(self):
        if self.__flower_name.lower()=="orchid":
            if self.__stock_available>=15:
                return False
            else:
                return True
```


You are signed in as 15648gh
Assignment on OOP Basics - Lev
Akul Joshi Assignment Day-2 | K
infyspringboard.onwingspan.com/en/viewer/hands-on/lex_auth_012727119215337472135_shared?collectionId=lex_auth_0125409722749255681063_shared&collectionType=Course
Infosys Springboard
Assignment on OOP Basics - Level 2
Reset
Execute
Copy Code
Verify
Submit
Last Submission
Type here to search
ENG
22:42
07-12-2021

```
return False
```

Verification Result

Test Cases Passed

43 / 43 Total

16 / 16 Structural

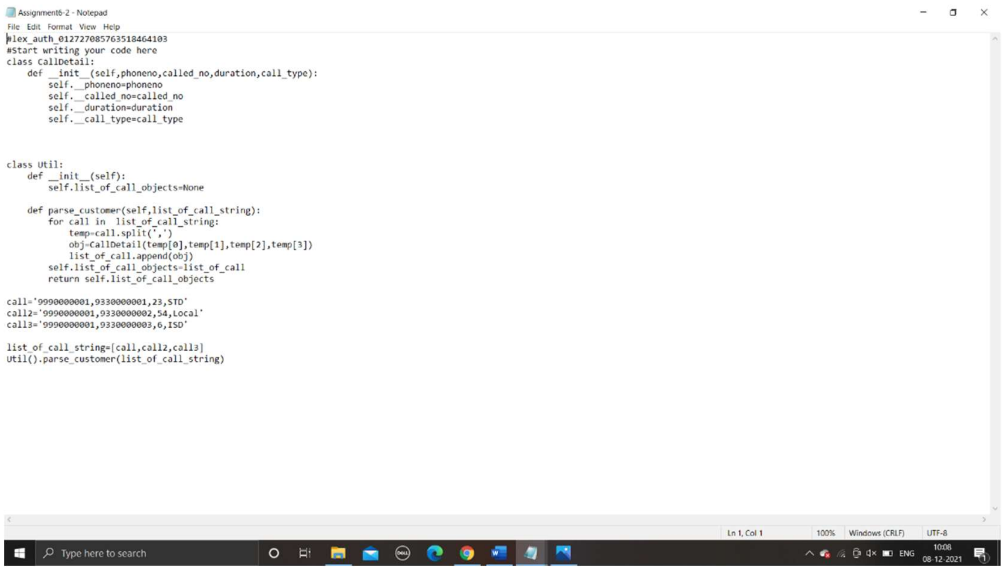
27 / 27 Logical

| Sl. No. | Type | S/A | Test Target | Input | Expected Output | Actual Output | Result |
|---|---|---|---|---|---|---|---|
| 1 | | N/A | Flower Flower | NA | NA | NA | ✓ |
| 2 | | N/A | Flower Flower::__init__ | NA | NA | NA | ✓ |
| 3 | | N/A | Flower Flower::get_flower_name | NA | NA | NA | ✓ |
| 4 | | N/A | Flower Flower::get_price_per_kg | NA | NA | NA | ✓ |
| 5 | | N/A | Flower Flower::get_stock_available | NA | NA | NA | ✓ |
| 6 | | N/A | Flower Flower::set_flower_name | NA | NA | NA | ✓ |
| 7 | | N/A | Flower Flower::set_price_per_kg | NA | NA | NA | ✓ |

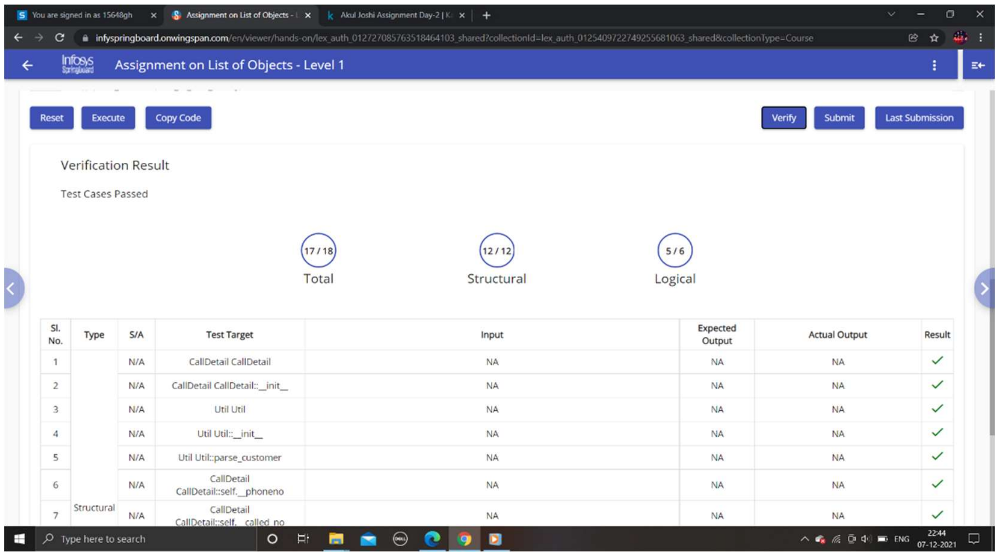
## Code:

```
#lex_auth_012727119215337472135

#Start writing your code here

class Flower:

    def __init__(self):

        self.__flower_name=None

        self.__price_per_kg=None

        self.__stock_available=None


    def set_flower_name(self,flower_name):

        self.__flower_name=flower_name


    def set_price_per_kg(self,price_per_kg):

        self.__price_per_kg=price_per_kg


    def set_stock_available(self,stock_available):

        self.__stock_available=stock_available

        def get_flower_name(self):

        return self.__flower_name


    def get_price_per_kg(self):

        return self.__price_per_kg


    def get_stock_available(self):

        return self.__stock_available


    def validate_flower(self):

        flower={'orchid':15,'rose':25,'jasmine':40}

        return (self.__flower_name).lower() in flower.keys()


    def validate_stock(self,required_quantity):
        return required_quantity<=self.__stock_available


    def sell_flower(self,required_quantity):

        if (self.validate_flower()) and self.validate_stock(required_quantity)==True:

            self.__stock_available-=required_quantity


    def check_level(self):

        if self.__flower_name.lower()=="orchid":

            if self.__stock_available>=15:

                return False

            else:

                return True

        elif self.__flower_name.lower()=="rose":

            if self.__stock_available>=25:

                return False

            else:

                return True

        elif self.__flower_name.lower()=="jasmine":

            if self.__stock_available>=40:

                return False

            else:

                return True

        else:

            return False
```

## 2). Assignment on list of Objects-Level1:



```
#lex_auth_01272708576351846410З
#Start writing your code here
class CallDetail:
    def __init__(self,phoneno,called_no,duration,call_type):
        self.__phoneno=phoneno
        self.__called_no=called_no
        self.__duration=duration
        self.__call_type=call_type



class Util:
    def __init__(self):
        self.list_of_call_objects=None

    def parse_customer(self,list_of_call_string):
        for call in  list_of_call_string:
            temp=call.split(',')
            obj=CallDetail(temp[0],temp[1],temp[2],temp[3])
            list_of_call.append(obj)
        self.list_of_call_objects=list_of_call
        return self.list_of_call_objects

call='9990000001,9330000001,23,STD'
call2='9990000001,9330000002,54,Local'
call3='9990000001,9330000003,6,ISD'

list_of_call_string=[call,call2,call3]
Util().parse_customer(list_of_call_string)
```



Verification Result

Test Cases Passed

17 / 18 Total

12 / 12 Structural

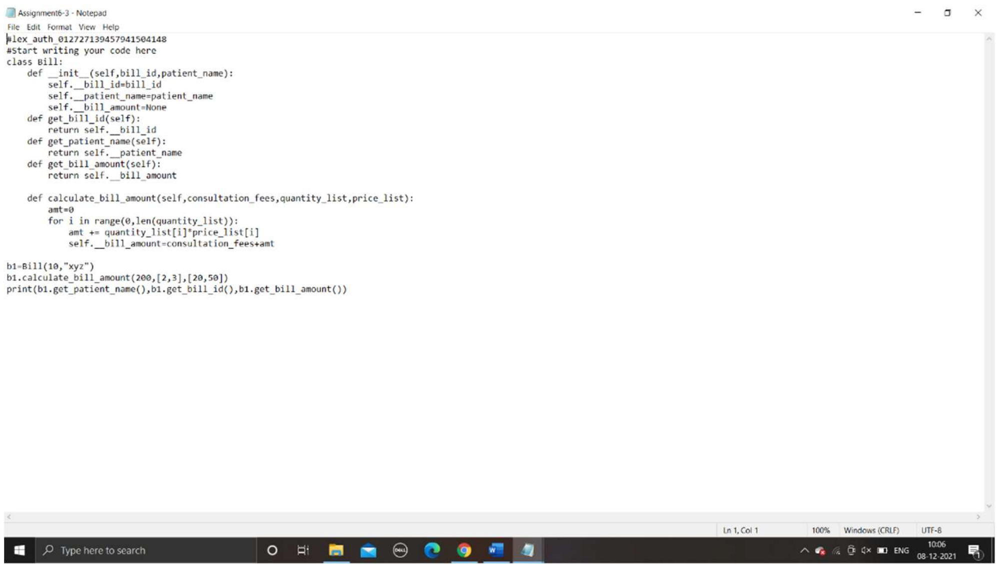
5 / 6 Logical

| Sl. No. | Type | S/A | Test Target | Input | Expected Output | Actual Output | Result |
|---|---|---|---|---|---|---|---|
| 1 | | N/A | CallDetail CallDetail | NA | NA | NA | ✓ |
| 2 | | N/A | CallDetail CallDetail::__init__ | NA | NA | NA | ✓ |
| 3 | | N/A | Util Util | NA | NA | NA | ✓ |
| 4 | | N/A | Util Util::__init__ | NA | NA | NA | ✓ |
| 5 | | N/A | Util Util::parse_customer | NA | NA | NA | ✓ |
| 6 | | N/A | CallDetail CallDetail::self.__phoneno | NA | NA | NA | ✓ |
| 7 | Structural | N/A | CallDetail CallDetail::self.__called_no | NA | NA | NA | ✓ |

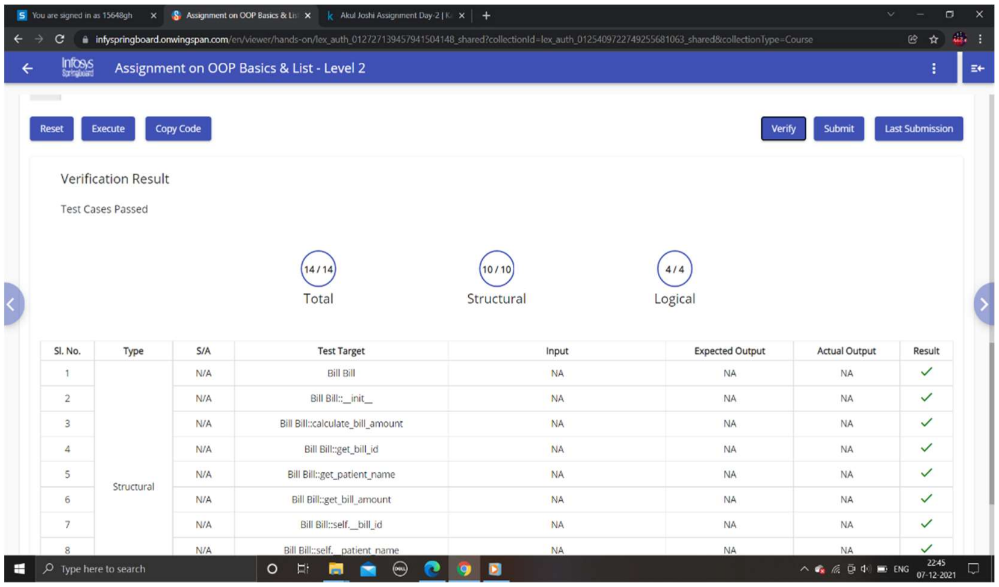
## <u>Code:</u>

```
#lex_auth_012727085763518464103

#Start writing your code here

class CallDetail:

    def __init__(self,phoneno,called_no,duration,call_type):

        self.__phoneno=phoneno

        self.__called_no=called_no

        self.__duration=duration

        self.__call_type=call_type





class Util:

    def __init__(self):

        self.list_of_call_objects=None


    def parse_customer(self,list_of_call_string):

        for call in  list_of_call_string:

            temp=call.split(',')

            obj=CallDetail(temp[0],temp[1],temp[2],temp[3])

            list_of_call.append(obj)

        self.list_of_call_objects=list_of_call

        return self.list_of_call_objects


call='9990000001,9330000001,23,STD'

call2='9990000001,9330000002,54,Local'

call3='9990000001,9330000003,6,ISD'


list_of_call_string=[call,call2,call3]

Util().parse_customer(list_of_call_string)
```

## 3). Assignment on OOP basics and List-Level2:



```
#lex_auth_012727139457941504148
#Start writing your code here
class Bill:
    def __init__(self,bill_id,patient_name):
        self.__bill_id=bill_id
        self.__patient_name=patient_name
        self.__bill_amount=None
    def get_bill_id(self):
        return self.__bill_id
    def get_patient_name(self):
        return self.__patient_name
    def get_bill_amount(self):
        return self.__bill_amount

    def calculate_bill_amount(self,consultation_fees,quantity_list,price_list):
        amt=0
        for i in range(0,len(quantity_list)):
            amt += quantity_list[i]*price_list[i]
            self.__bill_amount=consultation_fees+amt

b1=Bill(10,"xyz")
b1.calculate_bill_amount(200,[2,3],[20,50])
print(b1.get_patient_name(),b1.get_bill_id(),b1.get_bill_amount())
```


You are signed in as 15648gh
Assignment on OOP Basics & Lis
Akul Joshi Assignment Day-2 | K
infyspringboard.onwingspan.com/en/viewer/hands-on/lex_auth_012727139457941504148_shared?collectionId=lex_auth_0125409722749255681063_shared&collectionType=Course
Assignment on OOP Basics & List - Level 2
Reset
Execute
Copy Code
Verify
Submit
Last Submission
Type here to search
22:45
07-12-2021

Verification Result

Test Cases Passed

14 / 14 Total

10 / 10 Structural

4 / 4 Logical

| Sl. No. | Type | S/A | Test Target | Input | Expected Output | Actual Output | Result |
|---|---|---|---|---|---|---|---|
| 1 | Structural | N/A | Bill Bill | NA | NA | NA | ✓ |
| 2 | | N/A | Bill Bill::__init__ | NA | NA | NA | ✓ |
| 3 | | N/A | Bill Bill::calculate_bill_amount | NA | NA | NA | ✓ |
| 4 | | N/A | Bill Bill::get_bill_id | NA | NA | NA | ✓ |
| 5 | | N/A | Bill Bill::get_patient_name | NA | NA | NA | ✓ |
| 6 | | N/A | Bill Bill::get_bill_amount | NA | NA | NA | ✓ |
| 7 | | N/A | Bill Bill::self.__bill_id | NA | NA | NA | ✓ |
| 8 | | N/A | Bill Bill::self.__patient_name | NA | NA | NA | ✓ |

## Code:

```
#lex_auth_012727139457941504148

#Start writing your code here

class Bill:

    def __init__(self,bill_id,patient_name):

        self.__bill_id=bill_id

        self.__patient_name=patient_name

        self.__bill_amount=None

    def get_bill_id(self):

        return self.__bill_id

    def get_patient_name(self):

        return self.__patient_name

    def get_bill_amount(self):

        return self.__bill_amount


    def calculate_bill_amount(self,consultation_fees,quantity_list,price_list):
        amt=0
        for i in range(0,len(quantity_list)):
            amt += quantity_list[i]*price_list[i]
            self.__bill_amount=consultation_fees+amt

b1=Bill(10,"xyz")
b1.calculate_bill_amount(200,[2,3],[20,50])
print(b1.get_patient_name(),b1.get_bill_id(),b1.get_bill_amount())
```