S.No: 1	Exp. Name: ***Swap two numbers***	Date: 2024-10-24

ID: 24IT037

2024-2028-IT

Ramco Institute of Technology

Aim:

Write a python program to swap / interchange two numbers.

Input format:

The first two lines contains integers.

Output format:

The output contains integer values after swapping.

Source Code:

change.py

```
num1 = int(input())
num2 = int(input())
temp=num1
num1=num2
num2=temp



print(num1, num2)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
5
3
3 5

Test Case - 2
User Output
6
7
7 6

Test Case - 3
User Output
4
2
2 4

S.No: 2	Exp. Name: ***Circulate the values of n variables***	Date: 2024-11-06

Aim:

Write and execute a Python program to circulate the values of n variables

Input format:

- The input should be the n values separated by spaces.

Output format:

- The output should display the circulated variable values, with each value printed on a new line after the print statement "After circulating values:".

Source Code:

circulate.py

```
x=input().split()
print("After circulating values:")
l=x[len(x)-1:]+x[:len(x)-1]
for i in l:
        print(i)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
8 9 15 60
After circulating values:
60
8
9
15

Test Case - 2
User Output
30 100 98 65.235
After circulating values:
65.235
30
100
98

S.No: 3	Exp. Name: ***Distance Between Two Points***	Date: 2024-10-24

Aim:

Write a Python program to find the distance between two points by taking the inputs for $x_1,\ x_2,\ y_1$, and y_2

Distance Between Two Points: $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

Source Code:

distance.py

```
x1=float(input("Enter x1:"))
x2=float(input("Enter x2:"))
y1=float(input("Enter y1:"))
y2=float(input("Enter y2:"))
z1=(x2-x1)**2
z2=(y2-y1)**2
d=z1+z2**0.5
print("Distance between two points ",d)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter x1:
5
Enter x2:
4
Enter y1:
6
Enter y2:
7
Distance between two points 2.0

Test Case - 2
User Output
Enter x1:
10
Enter x2:
12
Enter y1:
12
Enter y2:
10
Distance between two points 6.0

S.No: 4	Exp. Name: ***Get the details***	Date: 2024-10-24

Aim:

John, a lecturer at the University of Global Talent wants to get the details of students such as names, theory marks, and practical marks. After getting the details, the same has to be displayed on the screen. Write the program that could meet John's requirements.

Source Code:

studentDetails.py

```
n=input("Enter student's name:")
m=input("Enter Theory Marks:")
a=input("Enter Practical Marks:")
print("Student's Name:",n)
print("Theory Marks:",m)
print("Practical Marks:",a)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter student's name:
Riya
Enter Theory Marks:
60
Enter Practical Marks:
40
Student's Name: Riya
Theory Marks: 60
Practical Marks: 40

Test Case - 2
User Output
Enter student's name:
Arna
Enter Theory Marks:
58
Enter Practical Marks:
95
Student's Name: Arna
Theory Marks: 58
Practical Marks: 95

S.No: 5	Exp. Name: ***Shopping Cart***	Date: 2024-10-24

Aim:

A program developer has to create a shopping cart as a part of an online shopping project. He wants to get cart id, product name, and quantity, which are of different data types. Write a program to accept all the details of the Shopping cart and display the same.

Source Code:

shoppingCart.py

```
a=input("Enter card Id:")
b=input("Product Name:")
c=input("Enter quantity:")
print("Card ID =",a)
print("Product Name=",b)
print("Quantity=",c)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter card Id:
221234
Product Name:
Shampoo
Enter quantity:
3
Card ID = 221234
Product Name= Shampoo
Quantity= 3

Test Case - 2
User Output
Enter card Id:
879
Product Name:
Cosmetics
Enter quantity:
9
Card ID = 879
Product Name= Cosmetics
Quantity= 9

S.No: 6	Exp. Name: ***Perimeter of the Cricket Ground***	Date: 2024-10-24

Aim:

The diameter of the Cricket ground is d cm. Write a Python program to find the perimeter of the cricket ground which is circular.

Formula = πd, where $\pi = 3.14$

Source Code:

perimeter.py

```
d=int(input("Enter Diameter="))
p=3.14*d
print("Perimeter of the cricket ground =",p)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter Diameter=
45
Perimeter of the cricket ground = 141.3

Test Case - 2
User Output
Enter Diameter=
25
Perimeter of the cricket ground = 78.5

S.No: 7	Exp. Name: ***Determine the remainder***	Date: 2024-10-24

Aim:

In Python, you can determine the remainder of a division operation using a single operator. Write a program that takes two integers as input, where the first integer is divided by the second. The program should compute and display the remainder of this division.

Source Code:

remainder.py

```
x=int(input("Enter x:"))
y=int(input("Enter y:"))
r=x%y
print("Remainder =",r)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter x:
45
Enter y:
7
Remainder = 3

Test Case - 2
User Output
Enter x:
56
Enter y:
5
Remainder = 1

Test Case - 3
User Output
Enter x:
152
Enter y:
26
Remainder = 22

S.No: 8	Exp. Name: ***Calculate the Momentum***	Date: 2024-10-24

ID: 24IT037

Aim:

In physics, **momentum** can be calculated using the formula $e = Mc^2$, where M represents the **mass of the object** and c represents its **velocity**.

Write a Python program that prompts the user to enter an **object's mass** (in kilograms) and uses the **speed of light** $(3 * 10^8 m/s)$ as the **velocity** to calculate and display the **momentum** of the object.

Source Code:

momentum.py

```
a=int(input("Enter Mass="))
e=a*(3*10**8)**2
print("Momentum=",e)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter Mass=
5
Momentum= 450000000000000000

Test Case - 2
User Output
Enter Mass=
0
Momentum= 0

S.No: 9	Exp. Name: ***Temperature Conversion***	Date: 2024-11-06

Aim:

The average body temperature of Bob is 101.5°F. Write a Python program that converts this temperature from Fahrenheit to Celsius. Use the formula C = 5/9 * (F - 32) for the conversion. The program should prompt the user to enter the temperature in Fahrenheit and then display the temperature in Celsius.

Source Code:

fahrenheitCelsius.py

```
f=float(input("Enter Fahrenheit :"))
c=(5/9)*(f-32)
print("Temperature in Celsius=",c)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter Fahrenheit :
101.6
Temperature in Celsius= 38.666666666666664

Test Case - 2
User Output
Enter Fahrenheit :
112.25
Temperature in Celsius= 44.583333333333336

S.No: 10	Exp. Name: ***Sum of Series - I***	Date: 2024-11-06

Aim:

Write a Python program that calculates the sum of the squares of the first n natural numbers, where n is provided by the user.

Series: $1^2 + 2^2 + 3^2 + \ldots\ldots\ldots + n^2$

Source Code:

squareSeries.py

```
n=int(input("Enter the n value: "))
i=1
x=0
while i<=n:
        s=i**2
        x=x+s
        i=i+1
print(x)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the n value:
5
55

Test Case - 2
User Output
Enter the n value:
10
385

S.No: 11	Exp. Name: ***Sum of Series - II***	Date: 2024-11-06

ID: 24IT037

Aim:

Write a Python program that calculates the sum of the powers of the first n natural numbers, where n is provided by the user.

Series: $(1 \times 1) + (2 \times 2) + (3 \times 3) + \ldots\ldots + (n \times n)$

Source Code:

powerSeries.py

```
n=int(input("Enter the n value: "))
i=1
x=0
while i<=n:
        s=i**i
        x=x+s
        i=i+1
print(x)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the n value:
3
32

Test Case - 2
User Output
Enter the n value:
5
3413

S.No: 12	Exp. Name: ***Number Pattern - I***	Date: 2024-11-06

Aim:

Write a Python Program that prints the following pattern based on the number of rows provided by the user:

Pattern:

If the number of rows = 5

```
11111
22222
33333
44444
55555
```

Source Code:

numberpattern.py

```
n=int(input("Enter the number of rows: "))
for i in range (1,n+1):
        print(str(i)*n)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the number of rows:
5
11111
22222
33333
44444
55555

Test Case - 2
User Output
Enter the number of rows:
9
111111111
222222222
333333333
444444444
555555555
666666666
777777777
888888888
999999999

S.No: 13	Exp. Name: ***Number Pattern - II***	Date: 2024-11-06

ID: 24IT037

Ramco Institute of Technology

Aim:

Write a Python Program that prints the following pattern based on the number of rows provided by the user:

Pattern:

If the number of rows = 5

```
12345
12345
12345
12345
12345
```

Source Code:

numberpattern.py

```
n=int(input("Enter the number of rows: "))
for i in range (1,n+1):
        for j in range (1,n+1):
                print(j,end="")
        print()
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the number of rows:
5
12345
12345
12345
12345
12345

Test Case - 2
User Output
Enter the number of rows:
10
12345678910
12345678910
12345678910
12345678910
12345678910
12345678910
12345678910
12345678910
12345678910
12345678910

S.No: 14	Exp. Name: ***Number Pattern - III***	Date: 2024-11-06

ID: 24IT037

2024-2028-IT

Ramco Institute of Technology

Aim:

Write a Python Program that prints the following pattern based on the number of rows provided by the user:

Pattern:

If the number of rows = 5

```
1
22
333
4444
55555
```

Source Code:

numberpattern.py

```
n=int(input("Enter the number of rows: "))
for i in range (1,n+1):
        for j in range(1,i+1):
                print(i,end="")
        print()
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the number of rows:
5
1
22
333
4444
55555

Test Case - 2
User Output
Enter the number of rows:
10
1
22
333
4444
55555
666666
7777777
88888888
999999999
10101010101010101010

S.No: 15	Exp. Name: ***Pyramid Pattern - I***	Date: 2024-11-06

ID: 24IT037

Ramco Institute of Technology

Aim:

Write a Python Program that prints the following Pyramid pattern based on the number of rows provided by the user:

Pattern:

If the number of rows = 5

```
*
**
***
****
*****
```

Source Code:

pyramidPattern.py

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
        for j in range(1,i+1):
                print("*",end="")
        print()
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the number of rows:
5
*
**
***
****
*****

Test Case - 2
User Output
Enter the number of rows:
10
*
**
***
****
*****
******
****
****
****
****

S.No: 16	Exp. Name: ***Pyramid Pattern - II***	Date: 2024-11-06

Aim:

Write a Python Program that prints the following Pyramid pattern based on the number of rows provided by the user:

Pattern:

If the number of rows = 5

```
*****
****
***
**
*
```

Source Code:

pyramidPattern.py

```
n=int(input("Enter the number of rows: "))
for i in range (n):
        for j in range(n-i):
                print("*",end="")
        print()
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the number of rows:
5
*****
****
***
**
*

Test Case - 2
User Output
Enter the number of rows:
10
****
****
****
****
******
*****
****
***
**

*

S.No: 17	Exp. Name: ***Sum of Series - III***	Date: 2024-11-06

Aim:

Write a Python program that calculates the sum of the series, where n is provided by the user.

Series: $\frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \ldots\ldots + \frac{n}{n+1}$

Source Code:

fractionSeries.py

```
n=int(input("Enter the n value: "))
s=0
for i in range (1,n+1):
        s=s+(i/(i+1))
print(s)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the n value:
3
1.9166666666666665

Test Case - 2
User Output
Enter the n value:
4
2.716666666666667

ID: 24IT037

S.No: 18	Exp. Name: ***Sum of Series - IV***	Date: 2024-11-06

ID: 24IT037

Aim:

Write a Python program that calculates the sum of the series, where n is provided by the user.

Series: $1+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+\ldots\ldots+\frac{1}{n^2}$

Source Code:

fracSquareSeries.py

```
n=int(input("Enter the n value: "))
s=0
for i in range (1,n+1):
        s=s+(1/(i**2))
print(s)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the n value:
3
1.3611111111111112

Test Case - 2
User Output
Enter the n value:
4
1.4236111111111112

S.No: 19	Exp. Name: ***Number Pattern - IV***	Date: 2024-11-06

Aim:

Write a Python Program that prints the following pattern based on the number of rows provided by the user:

Pattern:

If the number of rows = 5

```
1
12
123
1234
12345
```

Source Code:

numberpattern.py

```
x=int(input("Enter the number of rows: "))
for i in range(1,x+1):
        for j in range(1,i+1):
                print(j,end="")
        print()
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the number of rows:
5
1
12
123
1234
12345

Test Case - 2
User Output
Enter the number of rows:
10
1
12
123
1234
12345
123456
1234567
12345678
123456789
12345678910

S.No: 20	Exp. Name: ***Number Pattern - V***	Date: 2024-11-06

Aim:

Write a Python Program that prints the following pattern based on the number of rows provided by the user:

Pattern:

If the number of rows = 5

```
55555
44444
33333
22222
11111
```

Source Code:

numberpattern.py

```
r=int(input("Enter the number of rows: "))
for i in range(r,0,-1):
        for j in range(r,0,-1):
                print(i,end="")
        print()
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the number of rows:
5
55555
44444
33333
22222
11111

Test Case - 2
User Output
Enter the number of rows:
10
10101010101010101010
9999999999
8888888888
7777777777
6666666666
5555555555
4444444444
3333333333
2222222222
1111111111

S.No: 21	Exp. Name: ***Number Pattern - VI***	Date: 2024-11-06

Aim:

Write a Python Program that prints the following pattern based on the number of rows provided by the user:

Pattern:

If the number of rows = 5

```
5
54
543
5432
54321
```

Source Code:

numberpattern.py

```
r=int(input("Enter the number of rows: "))
for i in range (r,0,-1):
        for j in range(r,i-1,-1):
                print(j,end="")
        print()
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the number of rows:
5
5
54
543
5432
54321

Test Case - 2
User Output
Enter the number of rows:
10
10
109
1098
10987
109876
1098765
10987654
109876543
1098765432
10987654321

S.No: 22	Exp. Name: ***Pyramid Pattern - III***	Date: 2024-11-06

Aim:

Write a Python Program that prints the following Pyramid pattern based on the number of rows provided by the user:

Pattern:

If the number of rows = 3

```
  *
 ***
*****
```

Source Code:

pyramidPattern.py

```
x=int(input("Enter the number of rows: "))
for i in range (1,x+1):
                for j in range(x,i,-1):
                        print(" ",end="")
                for k in range(2*i-1):
                        print("",end="*")
                print()
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the number of rows:
3
*
***
*****

Test Case - 2
User Output
Enter the number of rows:
10
*
***
*****
****
****
****
****
****
****
****

S.No: 23	Exp. Name: ***Items present in a library***	Date: 2024-11-10

Aim:

The library is a place with a huge collection of books. Maintaining the records of issuing and borrowing the book is difficult.

1. Assume you will create a code for library management from the beginning. Your first task is to get the name of the books and store it in a list named "**books"**

2. Display it to check whether the books are added to the list properly

3. Display the operations of the library to the user as
   - Display the books
   - Request a Book
   - Return a Book

4. Ask the user of his/her choice. If the user gives 1 as input then display the available books in the library in the sorted order.

5. If the user gives 2 as input, Ask the name of the book to be taken from the library
   - If the name is available in the book list then issue the book to the user by removing the book from the book list and displaying **"Successfully issued"** else display **"Book not available"**

6. If the user gives 3 as input, then ask the name of the book to be returned. Then append the name of the book to the book list and display the message **"Successfully returned"**

7. This loop will prompt the user to make a choice (1, 2, or 3) five times, regardless of their input, if you want to exit the program you can give any integer other than 1,2, and 3.

Source Code:

`libraryItems.py`

ID: 24IT037

Ramco Institute of Technology

```
books=[]
n=int(input("Enter total no of books: "))
for i in range(n):
        book=input("Enter book name: ")
        books.append(book)
print("The Academic books in the library are:",books)
for i in range(5):
        print("1. Display no of books")
        print("2. Request a book")
        print("3. Return a book")
        choice=int(input("Enter your choice: "))
        if choice==1:
                books.sort()
                print("The total number of books in the library are:",len(books))
                print("The books in the library are:",books)
        elif choice==2:
                bn=input("Enter the name of the book: ")
                if bn in books:
                        books.remove(bn)
                        print("Successfully issued")
                else:
                        print("Book not available")
        elif choice==3:
                br=input("Enter the name of the book: ")
                books.append(br)
                print("Successfully returned")
        else:
                break
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter total no of books:
2
Enter book name:
Data
Enter book name:
Science
The Academic books in the library are: ['Data', 'Science']
1. Display no of books
2. Request a book
3. Return a book
Enter your choice:
1
The total number of books in the library are: 2
The books in the library are: ['Data', 'Science']
1. Display no of books
2. Request a book
3. Return a book

Enter your choice:
2
Enter the name of the book:
data
Book not available
1. Display no of books
2. Request a book
3. Return a book
Enter your choice:
2
Enter the name of the book:
Data
Successfully issued
1. Display no of books
2. Request a book
3. Return a book
Enter your choice:
1
The total number of books in the library are: 1
The books in the library are: ['Science']
1. Display no of books
2. Request a book
3. Return a book
Enter your choice:
3
Enter the name of the book:
science
Successfully returned

Test Case - 2
User Output
Enter total no of books:
3
Enter book name:
C
Enter book name:
Java
Enter book name:
Python
The Academic books in the library are: ['C', 'Java', 'Python']
1. Display no of books
2. Request a book
3. Return a book
Enter your choice:
2
Enter the name of the book:
Python
Successfully issued
1. Display no of books

2. Request a book
3. Return a book
Enter your choice:
1
The total number of books in the library are: 2
The books in the library are: ['C', 'Java']
1. Display no of books
2. Request a book
3. Return a book
Enter your choice:
6

S.No: 25	Exp. Name: ***Materials Required for a Construction of a Building***	Date: 2024-11-11

Aim:

Many materials were used for the construction of the building. The primary components include:

- **Cement**: 40 bags required for every 100 sq ft
- **Sand**: 80 tons required for every 100 sq ft
- **Bricks**: 800 pieces required for every 100 sq ft

Available Materials:

Types of Cement: Ordinary Portland Cement, Water Repellent Cement, and Low Heat Cement
Types of Sand: Concrete Sand, Fill Sand, and Coarse Sand
Types of Bricks: Sun-Dried Bricks, Burnt Clay Bricks, and Concrete Bricks

a. Store the different types of cement, sand, and bricks in lists.
b. Input the following from the user:
   - Area of construction (in sq ft)
   - Type of cement, sand, and bricks to be used for the construction
c. Based on the input, calculate:
   - The number of cement bags needed
   - The tons of sand required
   - The number of bricks needed

If the specified type of cement, sand, or bricks is unavailable, display the message: **"Material not available"**

Note:

- All the choices are case-sensitive, for example, Concrete Sand is not the same as concrete sand.
- Refer to the displayed test cases for better understanding.

Source Code:

constructionMaterials.py

```
cement=['Ordinary Portland Cement','Water Repellent Cement','Low Heat Cement']
sand=['Concrete Sand','Fill Sand','Coarse Sand']
bricks=['Sun-Dried Bricks','Burnt Clay Bricks','Concrete Bricks']
area=int(input("What is the construction area? "))
c=input("Enter the cement to be used: ")
if(c not in cement):
        print("Cement not available")
s=input("Enter sand to be used: ")
if(s not in sand):
        print("Sand not available")
b=input("Enter brick to be used: ")
if(b not in bricks):
        print("Brick not available")
if (c not in cement and s not in sand and b not in bricks):
        print("Material required for your construction is: None")
else:
        print("Material required for your construction is:")
        bags=area*(40/100)
        tons=area*(80/100)
        piece=area*(800//100)
        if(c in cement):
                print(c,"\t",format(bags,".2f")," Bags",sep="")
        if (s in sand):
                print(s,"\t",format(tons,".2f")," Ton",sep="")
        if(b in bricks):
                print(b,"\t",piece," Pcs",sep="")
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
What is the construction area?
100
Enter the cement to be used:
Low Heat Cement
Enter sand to be used:
Coarse Sand
Enter brick to be used:
Concrete Bricks
Material required for your construction is:
Low Heat Cement 40.00 Bags
Coarse Sand 80.00 Ton
Concrete Bricks 800 Pcs

Test Case - 2
User Output
What is the construction area?
5000
Enter the cement to be used:

Water Repellent Cement
Enter sand to be used:
Fill Sand
Enter brick to be used:
Concrete Bricks
Material required for your construction is:
Water Repellent Cement 2000.00 Bags
Fill Sand 4000.00 Ton
Concrete Bricks 40000 Pcs

Test Case - 3
User Output
What is the construction area?
250
Enter the cement to be used:
White Cement
Cement not available
Enter sand to be used:
Beach Sand
Sand not available
Enter brick to be used:
Red Bricks
Brick not available
Material required for your construction is: None

Aim:

You are tasked with analyzing the components of a vehicle by categorizing them into two sets: chassis and transmission. Your job is to perform various set operations on these two categories to understand their relationships.

Input Format:

- On the first line, input the chassis components as a comma-separated string. (Example: frame,axles,wheel,differential)
- On the second line, input the transmission components as a comma-separated string. (Example: clutch,gear_box,shaft,differential)

Output Format:

Output should display the following in sorted order:

- Display the chassis set in the first line.
- Display the transmission set in the second line.
- Display the union of both sets in the third line.
- Display the intersection of both sets in the fourth line.
- Display the difference (Chassis - Transmission) in the fifth line.
- Display the symmetric difference of both sets in the sixth line.

Refer to visible test cases for better understanding.

Source Code:

setOperations.py

```
chassis=input("Enter chassis components: ")
tcomponents=input("Enter transmission components: ")
c=chassis.split(",")
t=tcomponents.split(",")
print("Chassis Set:",sorted(c))
print("Transmission Set:",sorted(t))
s=(sorted(c)+sorted(t))
print("Union:",sorted(set(s)))
b=[]
d=[]
for i in c:
        if i in t:
                b.append(i)
        else:
                d.append(i)
print("Intersection:",b)
print("Difference:",sorted(d))
for i in t:
        if i not in c:
                d.append(i)
print("Symmetric Difference:",sorted(d))
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output
Enter chassis components:
frame,axles,wheel
Enter transmission components:
clutch,gearbox,shaft
Chassis Set: ['axles', 'frame', 'wheel']
Transmission Set: ['clutch', 'gearbox', 'shaft']
Union: ['axles', 'clutch', 'frame', 'gearbox', 'shaft', 'wheel']
Intersection: []
Difference: ['axles', 'frame', 'wheel']
Symmetric Difference: ['axles', 'clutch', 'frame', 'gearbox', 'shaft', 'wheel']

Test Case - 2
User Output
Enter chassis components:
frame,body
Enter transmission components:
clutch,gearbox,frame
Chassis Set: ['body', 'frame']
Transmission Set: ['clutch', 'frame', 'gearbox']
Union: ['body', 'clutch', 'frame', 'gearbox']
Intersection: ['frame']
Difference: ['body']
Symmetric Difference: ['body', 'clutch', 'gearbox']

S.No: 27	Exp. Name: ***Time Taken by Washing Machine***	Date: 2024-11-13

Aim:

A washing machine operates using a Fuzzy System, where the weight of the clothes determines the washing time and water level. The washing machine has a capacity of 7000 grams, and based on the weight of the clothes, it adjusts the water level and washing time.

Write a Python program that takes the weight of clothes (in grams) as input and calculates the estimated washing time based on the following conditions:

- For weights between 0 and 2000 grams (inclusive), the water level is low, and the washing time is 25 minutes.
- For weights between 2001 and 4000 grams (inclusive), the water level is medium, and the washing time is 35 minutes.
- For weights above 4000 grams and up to 7000 grams, the water level is high, and the washing time is 45 minutes.
- If the weight exceeds 7000 grams, the input is invalid, and the program should display an error message: "overloaded"

Input Format:

- An integer n, representing the weight of clothes in grams.

Output Format:

- Print the estimated time based on the weight of clothes, or print the error message if the input is invalid.

Source Code:

WashingMachine.py

```
n=int(input())
if(n>=0)&(n<=2000):
            print("Time: 25 Min")
elif(n>=2001)&(n<=4000):
            print("Time: 35 Min")
elif(n>=4001)&(n<=7000):
            print("Time: 45 Min")
else:
            print("overloaded")
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
4567
Time: 45 Min

Test Case - 2
User Output
8564
overloaded

S.No: 29	Exp. Name: ***Factorial of a number***	Date: 2024-11-11

Aim:

Write a Python function to calculate the factorial of a number

Input Format:

The user is prompted to enter an integer **n**, for which the factorial needs to be calculated

Output Format:

The program outputs the factorial of the input number **n**.

Source Code:

factorialNum.py

```
def factorial_iterative(n):
        i=1
        fact=1
        while(i<=n):
                fact=fact*i
                i=i+1
        return fact




#Type your content here



n = int(input())
print(factorial_iterative(n))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
10
3628800

Test Case - 2
User Output
3
6

S.No: 30	Exp. Name: ***Largest number in a list***	Date: 2024-11-11

Aim:

Write and execute a Python program to find the largest number in a list

Input format:

- The input should be list of n numbers separated by spaces.

output format:

- The output should be the largest number in that list

Source Code:

largest.py

```
a=list(map(int,input().split()))
print(max(a))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
2 5 9 3 5 8
9

Test Case - 2
User Output
58 65 25 4 5 30
65

S.No: 31	Exp. Name: ***Area of a rectangle***	Date: 2024-11-11

Aim:

Imagine you're tasked with developing a program to assist a construction project manager. They need a tool to quickly calculate the area of various rectangular sections of land before planning further construction details. Your task is to create a Python program that accurately computes the area of a rectangle based on user-provided dimensions.

Input Format:

The program should prompt the user to input two values:

- "**length:**" - This should be a floating-point number representing the length of the rectangular section of land.
- "**width:**" - This should be a floating-point number representing the width of the rectangular section of land.

Output Format:

- After processing the input, the program should display the calculated area of the rectangular section of land in the format: "**Area:<area>"**

Source Code:

rectangle.py

```
def rectangle_area(length, width):
   return length*width
length=float(input("length:"))
width=float(input("width:"))
print("{}{}".format("Area:",rectangle_area(length,width)))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
length:
15
width:
10
Area:150.0

Test Case - 2
User Output
length:
12.56
width:
5.654
Area:71.01424

S.No: 32	Exp. Name: ***Roots of a Quadratic Equation***	Date: 2024-11-12

Aim:

In a mathematics class, students are learning about quadratic equations. As part of a class assignment, you are required to write a Python program that calculates the roots of a quadratic equation in the form **$ax^2 + bx + c = 0$**. The students will input the coefficients of the equation, and your program should determine and display the roots.

Input Format:

Three space-separated integers, **a**, **b**, and **c**, representing the coefficients of the quadratic equation.

Output Format:

Display the roots of the quadratic equation. If the equation has real roots, display them in two separate lines. If the equation has complex roots, display them in the format "real_part + imaginary_part i".

Constraints:

-1000 <= a, b, c <= 1000

Note:

- The formula for finding the roots of a quadratic equation is x = $(-b \pm \sqrt{b^2 - 4ac})$/ 2a
- Partial code for taking input and calling the function is already provided.

Source Code:

RootsOfQEquation.py

```
import cmath
a, b, c =map(int, input().split())
d=b**2-4*a*c
if d>=0:
        root1=(-b+d**0.5)/(2*a)
        root2=(-b-d**0.5)/(2*a)
        print("Root 1:",f"{root1}")
        print("Root 2:",f"{root2}")
else:
        root1=(-b+cmath.sqrt(d)) / (2*a)
        root2=(-b-cmath.sqrt(d)) / (2*a)
        print("Root 1:",f"({int(root1.real)}+{int(root1.imag)}j)")
        print("Root 2:",f"({int(root2.real)}{int(root2.imag)}j)")
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1 2 5
Root 1: (-1+2j)
Root 2: (-1-2j)

Test Case - 2
User Output

1 -3 2
Root 1: 2.0
Root 2: 1.0

Aim:

A university has a strict attendance policy. In order to be eligible to write the final exam, a student must attend at least 75% of the total classes held. Write a Python program that will determine whether a student is eligible to write the exam based on their attendance percentage.

Input Format:

- First input: number of hours (or classes) held (h).
- Second input: number of hours (or classes) attended (a).

Output Format:

- If a > h, print: enter a valid value
- Otherwise, print:
- Eligible to write if attendance percentage is 75% or more.
- Not eligible to write if attendance percentage is less than 75%.

Hint : Use Formula as attendance percentage = (a / h) * 100

Source Code:

Eligible.py

```
h=int(input("number of hours held: "))
a=int(input("number of hours attended: "))
attendance=(a/h)*100
if(a>h):
        print("enter a valid value")
elif(attendance>=75):
        print("Eligible to write")
else:
        print("Not eligible to write")
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
number of hours held:
45
number of hours attended:
43
Eligible to write

Test Case - 2
User Output
number of hours held:
100
number of hours attended:
110
enter a valid value

S.No: 34	Exp. Name: ***Eligibility to write exam***	Date: 2024-11-12

Aim:

You need to write a Python program to determine whether a student is eligible to write an exam based on the number of hours held and the number of hours attended. The eligibility conditions are as follows:

- If the student has attended 75% or more of the total hours held, they are eligible to write the exam.
- If the attendance is less than 75%, the student may still be eligible if they have a valid medical reason. Otherwise, they are not eligible.
- If the number of hours attended is greater than the number of hours held, prompt the user to enter a valid value.

Input Format:

- Prompt the user to input the total **number of hours held**.
- Then, input the total **number of hours attended**.
- If the percentage of hours attended is less than 75%, you will be asked to input whether you have a medical reason for the absence (y/n).

Output Format:

- If the percentage of hours attended is 75% or more, print Eligible to write.
- If the percentage of hours attended is less than 75% but you have a medical reason, print **Eligible to write**.
- If the percentage of hours attended is less than 75% and you don't have a medical reason, print **Not eligible to write**.
- If the number of hours attended is greater than the number of hours held, print **Enter a valid value**

Source Code:

Exam.py

```
h=int(input("number of hours held: "))
a=int(input("number of hours attended: "))
p=(a/h)*100
if(p>=75):
        print("Eligible to write")
else:
        m=(input("Any medical reasons (y/n): "))
        if(m=="n"):
                print("Not eligible to write")
        elif(m=="y"):
                print("Eligible to write")
        else:
                print("Enter a valid value")
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
number of hours held:
45
number of hours attended:
30
Any medical reasons (y/n):

y
Eligible to write

Test Case - 2
User Output
number of hours held:
100
number of hours attended:
60
Any medical reasons (y/n):
n
Not eligible to write

S.No: 35	Exp. Name: ***Calculate Salary based on Service***	Date: 2024-11-12

Aim:

Write a Python program to calculate the total salary after adding a bonus based on the years of service. The program should work as follows:

Input Format:

- The program should prompt the user to enter two inputs:
- years of service (an integer)
- salary (an integer)

Output Format:

- The program should output the total salary after adding the bonus.

Logic:

- If the number of years of service is greater than 5, the employee is eligible for a 5% bonus on their salary.
- If the number of years of service is 5 or less, the employee does not get any bonus.

Source Code:

Salary.py

```
a=int(input("years of service: "))
b=int(input("salary: "))
if a >5:
        c=b*(5/100)
        print(b+c)
else:
        print(b)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
years of service:
3
salary:
25000
25000

Test Case - 2
User Output
years of service:
10
salary:
80000
84000.0

S.No: 36	Exp. Name: ***Student Grade Calculator***	Date: 2024-11-12

Aim:

Write a Python program that reads a student's mark and prints the corresponding grade based on the following criteria:

- If the mark is less than 25 and greater than 0, print "F".
- If the mark is between 25 and 45 (inclusive), print "E".
- If the mark is between 46 and 50 (inclusive), print "D".
- If the mark is between 51 and 60 (inclusive), print "C".
- If the mark is between 61 and 80 (inclusive), print "B".
- If the mark is above 80, print "A".
- If the mark is not a valid positive number, print **"Invalid mark"**.

Input Format:

- The input consists of one integer representing the mark.

Output Format:

- Based on the mark input, the output should print the appropriate grade or **"Invalid mark"**.

Source Code:

Marks.py

```
m=int(input("Enter the mark: "))
if(m<0):
        print("Invalid mark")
elif(m<25):
        print("F")
elif(m<=45):
        print("E")
elif(m<=50):
        print("D")
elif(m<=60):
        print("C")
elif(m<80):
        print("B")
else:
        print("A")
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the mark:
91
A

Test Case - 2
User Output
Enter the mark:

-50
Invalid mark