Download the ipynb at Github:
https://github.com/wingyeung0317/EEE4463/blob/master/BD_Labs_V3/Lab5/Yeung_Wing_lab5A

Download the ipynb at Github:
https://github.com/wingyeung0317/EEE4463/blob/master/BD_Labs_V3/Lab5/Yeung_Wing_lab5A

Download the ipynb at Github:
https://github.com/wingyeung0317/EEE4463/blob/master/BD_Labs_V3/Lab5/Yeung_Wing_lab5A

Program 1 (P1) – Printing stars

1. Two questions are asked: • How many stars are needed? • 1-sided or 2-sided?
2. If the answers are 5 [stars] and 2 [-sided], the output is:

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

3. If the answers are 8 [stars] and 1 [-sided], the output is:

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

4. You're encouraged to handle exceptions like non-numeric input.

In [ ]:

```python
def print_stars(stars, sided):
    try:
        stars = int(stars)
        sided = int(sided)
        for i in range(1, stars + 1):
            print('*' * i)
        if sided == 2:
            for i in range(stars - 1, 0, -1):
                print('*' * i)
    except ValueError:
        print("Value Error: Please input a number.")

stars = input("Number of stars:")
sided = input("Side(1 or 2):")

print_stars(stars, sided)
```

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

Program 2 (P2) – Solving quadratic equations

1. The coefficients of several quadratic equations (〖ax〗 ^2+bx+c=0) are given in the CSV file ‘abc_lab5_p2_input.csv’.

2. The content of the file is:

a	b	c
1	5	3
2	-4	2
3	2	-1
3	2	1

3. The program reads in the CSV file and outputs the following content on the screen:

   Basic requirement:

a	b	c	D	p	q	`REMARK`
1	5	3	13	-4.303	-0.697	`(Real roots: -4.303 and -0.697)`
2	-4	2	0	1	1	`(Repeated roots: 1)`
3	2	-1	16	-1	0.333	`(Real roots: -1 and 0.333)`
3	2	1	-8	NaN	NaN	`(Complex roots are not calculated)`

   The alignment should be more nice-looking.

   The precision or the number of decimal places shown is for reference only. We are not taking a Mathematics course.

   Advanced requirement:

a	b	c	D	p	q	`REMARK`
1	5	3	13	-4.303	-0.697	`(Real roots: -4.303 and -0.697)`
2	-4	2	0	1	1	`(Repeated roots: 1)`
3	2	-1	16	-1	0.333	`(Real roots: -1 and 0.333)`
3	2	1	-8	-0.333	0.471	`(Complex roots: -0.333 + 0.471j and -0.333 – 0.471j)`

4. The program also outputs the following content to the CSV file 'abc_lab5_p2_output.csv':

   Basic requirement:

a	b	c	D	p	q	REMARK
1	5	3	13	-4.303	-0.697	(Real roots: -4.303 and -0.697)
2	-4	2	0	1	1	(Repeated roots: 1)
3	2	-1	16	-1	0.333	(Real roots: -1 and 0.333)
3	2	1	-8			(Complex roots are not saved)

   Advanced requirement:

a	b	c	D	p	q	REMARK
1	5	3	13	-4.303	-0.697	(Real roots: -4.303 and -0.697)
2	-4	2	0	1	1	(Repeated roots: 1)
3	2	-1	16	-1	0.333	(Real roots: -1 and 0.333)
3	2	1	-8	-0.333	0.471	(Complex roots: -0.333 + 0.471j and -0.333 – 0.471j)

5. You're encouraged to handle exceptions like linear equation, non-numeric input.

```
In [ ]: import pandas as pd
        import numpy as np
        from IPython.display import display

        df = pd.read_csv('ab_lab5_p2_input.csv')

        df['D'] = np.power(df['b'],2) - 4*df['a']*df['c']
```

```
In [ ]: df['sqrt_D'] = np.where(df['D'] >= 0, np.sqrt(df['D']), np.sqrt(-df['D'])*1j)
        df['p'] = (-df['b'] - df['sqrt_D']) / (2*df['a'])
        df['q'] = (-df['b'] + df['sqrt_D']) / (2*df['a'])

        df['p'] = df['p'].apply(lambda x: f"{x.real:.3f}").astype(float)
        df['q'] = df['q'].apply(lambda x: f"{x.real:.3f}" if np.isreal(x) else f"{x.imag:.3
```

```
In [ ]: output = df[['a', 'b', 'c', 'D', 'p', 'q']]

        display(output)
        output.to_csv('ab_lab5_p2_output.csv', index=False, columns=['a', 'b', 'c', 'D', 'p
```

	a	b	c	D	p	q
0	1	5	3	13	-4.303	-0.697
1	2	-4	2	0	1.000	1.000
2	3	2	-1	16	-1.000	0.333
3	3	2	1	-8	-0.333	0.471