Question 1:

1. This code is saved in *q1.py*
2. - This program allows the user to input the final account value, annual interest rate (the unit is in %), and the number of years.
   - This program prompts the user to input each value in either integer or float data type.
   - If the user fails to input the value in integer or float data type, an error message "*The input should be either an integer number or a float number*" will show up and the program will ask the user to reinput the value.
   - User is not suggested to input "*-100*" for the annual interest rate as the division of 0 results an error.
   - This program outputs the initial deposit amount that has to be saved with the following formula:

$$Initial\ Deposit\ Amount = \frac{Final\ Account\ Value}{(1 + Annual\ Interest\ Rate)^{Number\ of\ Years}}$$

3. Execute as following:

```
D:\py\CSC1001\Assignment 1>q1.py
Enter the final account value: 1000
Enter the annual interest rate: 4.25
Enter the number of years: 5
The initial value is  812.1190197993631
```

Question 2:

1. This code is saved in *q2.py*
2. - This program allows the user to input a value.
   - This program prompts the user to input the value as a positive integer.
   - If the user fails to input the value in integer, an error message "*The input should be an integer number*" will show up and the program will ask the user to reinput the value.
   - If the integer value is not a positive number, an error message "*The integer should be positive*" will show up and the program will ask the user to reinput the value.
   - This program ignores leading zeroes (if any) in the number.
   - This program outputs every digit in the number in a separated line.
3. Execute as following:

```
D:\py\CSC1001\Assignment 1>q2.py
Enter a positive integer: 3125
3
1
2
5
```

Question 3:

1. This code is saved in *q3.py*
2. - This program allows the user to input a value m.
   - This program prompts the user to input the value in integer type.
   - If the user fails to input the value in integer, an error message "*The input should be an integer number*" will show up and the program will ask to reinput the value.
   - This program outputs the smallest non-negative number n such that n^2 is larger than m.
3. Execute as following:

```
D:\py\CSC1001\Assignment 1>q3.py
m = 10
n = 4
```

Question 4:

1. This code is saved in *q4.py*
2. - This program allows the user to input a value N.
   - This program prompts the user to input the value as a positive integer.
   - If the user fails to input the value in integer, an error message "*The input should be an integer number*" will show up and the program will ask the user to reinput the value.
   - If the integer value is not a positive number, an error message "*The integer should be positive*" will show up and the program will ask the user to reinput the value.
   - This program outputs N + 1 lines, each with 3 columns. The $(m + 1)^{th}$ line consists of 3 numbers m, m+1, and m^{m+1}.
3. Execute as following:

```
D:\py\CSC1001\Assignment 1>q4.py
N = 5
m       m+1     m**(m+1)
1       2       1
2       3       8
3       4       81
4       5       1024
5       6       15625
```

Question 5:

1. This code is saved in *q5.py*
2. - This program allows the user to input a value N.
   - This program prompts the user to input the value as a positive integer larger than 1.

- If the user fails to input the value in integer, an error message "*The input should be an integer number*" will show up and the program will ask the user to reinput the value.
- If the integer value is not larger than 1, an error message "*The integer should be larger than 1*" will show up and the program will ask the user to reinput the value.
- This program outputs all prime numbers less than N. At most 8 prime numbers will be printed in every single line.

3. Execute as following:

```
N = 10
The prime numbers smaller than 10 include:
2 3 5 7
```

Question 6:

1. This code is saved in *q6.py*
2. - This program allows the user to specify a trigonometric function f, both endpoints of an interval [a, b], and a number of partition n.
   - This program prompts the user to specify the trigonometric function in either sin, cos, or tan.
   - This program prompts the user to input both endpoints in either integer or float type.
   - This program prompts the user to input the partition in integer type.
   - If the user fails to specify the function in either sin, cos, or tan; an error message "*The input should be either sin, cos, or tan*" will show up and the program will ask the user to reinput the value.
   - If the user fails to input both endpoints in either integer or float type, an error message "*The input should be either an integer number or a float number*" will show up and the program will ask the user to reinput the value.
   - If the user fails to input the number of partition in integer type, an error message "*The input should be an integer number*" will show up and the program will ask the user to reinput the value.
   - If the number of partition is not larger than 0, an error message "*The integer should be larger than 0*" will show up and the program will ask the user to reinput the value.
   - This program outputs the integral approximation of function f with lower bound a and upper bound b using midpoint rule with n partitions. The formula used:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} \frac{b-a}{n} f\left(a + \frac{b-a}{n}\left(i - \frac{1}{2}\right)\right)$$

3. Execute as following:

```
D:\py\CSC1001\Assignment 1>q6.py
Specify a trigonometric function f: sin
Input the interval end points a: 0
Input the interval end points b: 1.57
Input the number of sub-intervals n: 10
The numerical integration of sin over [0, 1.57] is 1.0002306353571317
```