1.Write a python program with exception handling to input marks for five subjectsphysics,chemistry,biology,mathamatics and computer.calculate the persentage and grae them according to the following

1.Percentage>=90: Grade A
2.Percentage>=80: Grade B
3.Percentage>=70: Grade C
4.Percentage>=60: Grade D
5.Percentage>=50: Grade E
6.Percentage>=40: Grade F

In [1]:
```
import sys
while True:
    try:
        m_physics=int(input('Enter marks of physics subject :'))
        m_chemistry=int(input('Enter marks of chemistry subject :'))
        m_maths=int(input('Enter marks of maths subject :'))
        m_biology=int(input('Enter marks of biology subject :'))
        m_computer=int(input('Enter marks of computer subject :'))
        total_marks=m_physics+m_chemistry+m_maths+m_biology+m_computer
        percentage=total_marks/6
        if percentage>=90:
            print('Grade A')
        elif percentage>=80:
            print('Grade B')
        elif percentage>=70:
            print('Grade C')
        elif percentage>=60:
            print('Grade D')
        elif percentage>=50:
            print('Grade E')
        elif percentage<40:
            print('Grade F')
    except ValueError:
        print('Please enter valid marks')
        print(sys.exc_info()[1])
    finally:
        print()
    break
```

```
Enter marks of physics subject :85
Enter marks of chemistry subject :72
Enter marks of maths subject :55
Enter marks of biology subject :92
Enter marks of computer subject :ss
Please enter valid marks
invalid literal for int() with base 10: 'ss'
```

2.Write a python program for exception handling to input electricity unit charges and calculate total electricity bill accordingly to the given condition

1.for first 50 units RS.0.50/unit
2.for the next time 100 units RS.0.75/unit
3.for the next time 100 units RS.1.20/unit
4.for units above 250 units RS.1.50/unit
5.An additinal subcharge of 20% is added to the bill

In [23]:
```
import sys

try:
    units=int(input('Enter the number of units of current : '))

    if units>=50:
        subcharge=0.50
    elif units>50 and units<150:
        subcharge=0.75
    elif units>150 and units<250:
        subcharge=1.20
    elif units>=250:
        subcharge=1.50
    charge=units*subcharge
    additional_charge=charge*0.20
    final_bill=charge+additional_charge
    print('Charge with out including the additional charge is ',charge)
    print('Additional_charge is ',additional_charge)
    print('The Electricity bill is ', final_bill)
except ValueError as e:
        print('Please enter the valid units')
        print(sys.exc_info()[2])
        print(e)
```

```
Enter the number of units of current : 100
Charge with out including the additional charge is  50.0
Additional_charge is  10.0
The Electricity bill is  60.0
```

3. Write a python program to input the week number and print the week day

In [31]:
```
import sys
try:
    week_number=int(input('Enter the week number : '))
    if week_number>7 or week_number<=0:
        raise ValueError('Week number should be between 1 and 7 only')
    if week_number==1:
        print('Today is sunday ')
    elif week_number==2:
        print('Today is monday ')
    elif week_number==3:
        print('Today is tuesday ')
    elif week_number==4:
        print('Today is wednesday ')
    elif week_number==5:
        print('Today is thursday ')
    elif week_number==6:
        print('Today is friday ')
    elif week_number==7:
        print('Today is saturday ')
except ValueError as ve:
    print(f"Error: {ve}")
    print(sys.exc_info()[2])
```

```
Enter the week number : 8
Error: Week number should be between 1 and 7 only
<traceback object at 0x000001DFF0B5EA80>
```

5. Write a Python program for finding the most frequent words in a text read from a file.

1. Initially open the text file in read mode.
2. Make all the letters in the document into lower letters and split t he words in each line.
3. Get the words in an order.
4. Sort the words for finding the most frequent words in a file.
5. Enter the most frequent words in a file

In [38]:
```
with open('input.txt', 'r') as file:
    content = file.read().lower()
words = content.split()
word_count = {}
for word in words:
    if word in word_count:
        word_count[word] += 1
    else:
        word_count[word] = 1
sorted_words = sorted(word_count.items(), key=lambda x: x[1], reverse=True)
print("The most frequent words in the file are:")
for word, frequency in sorted_words[:10]:
    print(f"{word}: {frequency} times")
```

```
The most frequent words in the file are:
is: 2 times
allah: 1 times
omnipotent,: 1 times
and: 1 times
he: 1 times
above: 1 times
anything: 1 times
else!: 1 times
```

4.

In [43]:

In [ ]:

6

In [35]:
```
import sys

# Function to process the input file and write to the output file
def process_files(input_file_path, output_file_path):
    try:
        # Open and process the input file
        with open(input_file_path, 'r') as input_file:
            data = input_file.read()

        # Process the data here (you can modify this part as needed)
        processed_data = data.upper() # Example: Convert text to uppercase

        # Write the processed data to the output file
        with open(output_file_path, 'w') as output_file:
            output_file.write(processed_data)

    except FileNotFoundError:
        print("Error: Input file not found.")
        sys.exit(1)

    except PermissionError:
        print("Error: Permission issue when writing to the output file.")
        sys.exit(1)

    except Exception as e:
        print(f"An error occurred: {str(e)}")
        sys.exit(1)

if __name__ == "__main__":
    # Check if both input and output file paths are provided as command-line ar
    if len(sys.argv) != 3:
        print("Usage: python program.py <input_file_path> <output_file_path>")
        sys.exit(1)

    input_file_path = sys.argv[1]
    output_file_path = sys.argv[2]

    # Call the function to process the files
    process_files(input_file_path, output_file_path)



########################################
import unittest
import subprocess

class TestFileProcessing(unittest.TestCase):
    def test_valid_paths(self):
        # Test with valid input and output file paths
        result = subprocess.run(["python", "your_program.py", "input.txt", "out
        self.assertEqual(result.returncode, 0) # The program should exit with c

    def test_missing_input_file(self):
        # Test with a missing input file
        result = subprocess.run(["python", "your_program.py", "nonexistent_inpu
        self.assertNotEqual(result.returncode, 0) # The program should exit wit
```

```
    def test_permission_issue(self):
        # Test with a permission issue when writing to the output file
        result = subprocess.run(["python", "your_program.py", "input.txt", "/ro
        self.assertNotEqual(result.returncode, 0) # The program should exit wit

if __name__ == '__main__':
    unittest.main()
```

```
    21 except PermissionError:

SystemExit: 1

During handling of the above exception, another exception occurred:

AttributeError                            Traceback (most recent call last)
    [... skipping hidden 1 frame]

File ~\anaconda3\Lib\site-packages\IPython\core\interactiveshell.py:2092, i
n InteractiveShell.showtraceback(self, exc_tuple, filename, tb_offset, exce
ption_only, running_compiled_code)
   2089 if exception_only:
   2090     stb = ['An exception has occurred, use %tb to see '
   2091            'the full traceback.\n']
-> 2092     stb.extend(self.InteractiveTB.get_exception_only(etype,
   2093                                                      value))
   2094 else:
   2095     try:
   2096         # Exception classes can customise their traceback - we
```

In [ ]:

In [ ]: