Exercise 1

In [5]:
```
def read_file(file_name):
    with open(file_name, 'r') as file:
        content = file.read()
        print(content)
```

In [9]:
```
read_file('Ass7.txt')
```

```
Python is a high-level, versatile programming language celebrated for its simplic
ity and readability. Designed to be intuitive, it allows developers to express co
ncepts in fewer lines of code compared to other languages. This makes it an excel
lent choice for beginners and experienced programmers alike. Python's extensive l
ibraries and frameworks enable efficient development across various applications,
including data science, web development, automation, and artificial intelligence.

In addition to its user-friendly syntax, Python boasts a vibrant community that a
ctively contributes to its growth. With a wealth of resources, tutorials, and thi
rd-party modules available, programmers can easily find support and tools to enha
nce their projects. Its adaptability ensures that Python remains a popular choice
among developers in diverse fields.
```

Exercise 2

In [13]:
```
def copy_file(source_file, target_file):
    with open(source_file, 'r') as src:
        content = src.read()
    with open(target_file, 'w') as tgt:
        tgt.write(content)
```

In [15]:
```
copy_file('Ass7.txt', 'Ass7_copy.txt')
```

In [17]:
```
read_file('Ass7_copy.txt')
```

```
Python is a high-level, versatile programming language celebrated for its simplic
ity and readability. Designed to be intuitive, it allows developers to express co
ncepts in fewer lines of code compared to other languages. This makes it an excel
lent choice for beginners and experienced programmers alike. Python's extensive l
ibraries and frameworks enable efficient development across various applications,
including data science, web development, automation, and artificial intelligence.

In addition to its user-friendly syntax, Python boasts a vibrant community that a
ctively contributes to its growth. With a wealth of resources, tutorials, and thi
rd-party modules available, programmers can easily find support and tools to enha
nce their projects. Its adaptability ensures that Python remains a popular choice
among developers in diverse fields.
```

Exercise 3

In [21]:
```
def count_words(file_name):
    with open(file_name, 'r') as file:
        content = file.read()
        words = content.split()
        print(f"Total number of words: {len(words)}")
```

In [23]:
```
count_words('Ass7.txt')
```

```
Total number of words: 116
```

Exercise 4

In [27]:
```
def count_word_occurrences(file_name, word):
    with open(file_name, 'r') as file:
        content = file.read().lower()
        word_count = content.count(word.lower())
        print(f"The word '{word}' occurred {word_count} times.")
```

In [29]:
```
count_word_occurrences('Ass7.txt', 'Python')
```

```
The word 'Python' occurred 4 times.
```

In [31]:
```
count_word_occurrences('Ass7.txt', 'in')
```

```
The word 'in' occurred 12 times.
```

Exercise 5

In [35]:
```
def convert_to_integer():
    try:
        user_input = input("Enter a string to convert to an integer: ")
        number = int(user_input)
        print(f"The integer is: {number}")
    except ValueError:
        print("The input is not a valid integer.")
```

In [37]:
```
convert_to_integer()
```

```
The integer is: 57
```

In [39]:
```
convert_to_integer()
```

```
The input is not a valid integer.
```

Exercise 6

In [43]:
```
def check_negative_integers():
    try:
        numbers = list(map(int, input("Enter a list of integers separated by spa
        for number in numbers:
            if number < 0:
                raise ValueError("Negative integers are not allowed!")
        print("All integers are non-negative.")
    except ValueError as e:
        print(e)
```

In [45]:
```
check_negative_integers()
```

```
Negative integers are not allowed!
```

In [47]:
```python
check_negative_integers()
```

```
All integers are non-negative.
```

Exercise 7

In [51]:
```python
def compute_average():
    try:
        numbers = list(map(int, input("Enter a list of integers separated by spa
        if not numbers:
            raise ValueError("List cannot be empty!")
        average = sum(numbers) / len(numbers)
        print(f"The average is: {average}")
    except ValueError as e:
        print(e)
    finally:
        print("Program has finished running.")
```

In [53]:
```python
compute_average()
```

```
The average is: 17.0
Program has finished running.
```

In [55]:
```python
compute_average()
```

```
List cannot be empty!
Program has finished running.
```

Exercise 8

In [59]:
```python
def write_to_file():
    try:
        file_name = input("Enter the filename: ")
        content = input("Enter the string to write to the file: ")
        with open(file_name, 'w') as file:
            file.write(content)
        print("File written successfully.")
    except Exception as e:
        print(f"An error occurred: {e}")
    else:
        print("Welcome! No exceptions occurred.")
```

In [61]:
```python
write_to_file()
```

```
File written successfully.
Welcome! No exceptions occurred.
```

In [63]:
```python
read_file('Ass7_copy.txt')
```

```
I'm learing Python.
```

Exercise 9

In [81]:
```
class Course:
    def __init__(self, course_code, course_name, credit_hours):
        self.course_code = course_code
        self.course_name = course_name
        self.credit_hours = credit_hours

    def __str__(self):
        return f"{self.course_code}: {self.course_name} ({self.credit_hours} cre

class CoreCourse(Course):
    def __init__(self, course_code, course_name, credit_hours, required_for_majo
        super().__init__(course_code, course_name, credit_hours)
        self.required_for_major = required_for_major

    def __str__(self):
        return f"{super().__str__()} - Required for Major: {self.required_for_ma

class ElectiveCourse(Course):
    def __init__(self, course_code, course_name, credit_hours, elective_type):
        super().__init__(course_code, course_name, credit_hours)
        self.elective_type = elective_type

    def __str__(self):
        return f"{super().__str__()} - Elective Type: {self.elective_type}"
```

In [83]:
```
core = CoreCourse("CS101", "Introduction to Computer Science", 3, True)
elective = ElectiveCourse("ART100", "Basic Art", 2, "liberal arts")
```

In [85]:
```
print(core)
print(elective)
```

```
CS101: Introduction to Computer Science (3 credits) - Required for Major: True
ART100: Basic Art (2 credits) - Elective Type: liberal arts
```

In [ ]: