AGUIRRE, PAUL VINCENT S.

M001 || CP102 || MW – 4:00 – 6:30

PRELIM OUTPUT

TASK 1 (OOP)

Description:

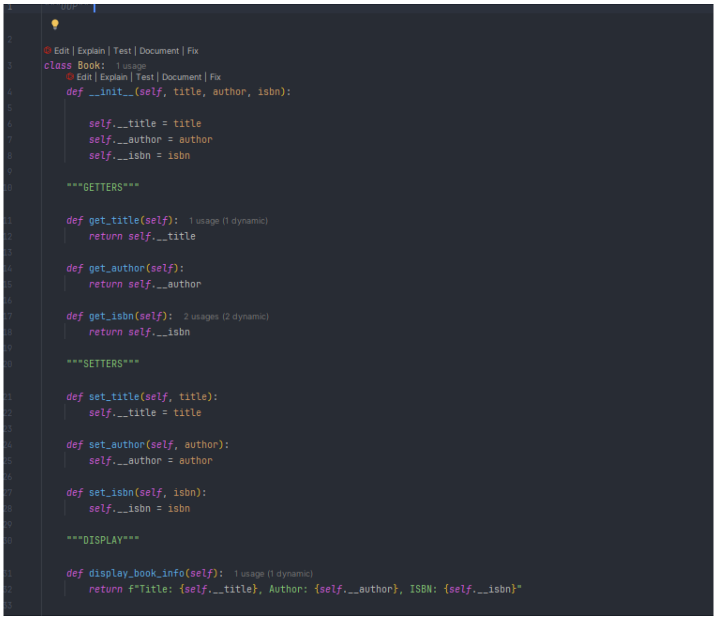
- The Program *Library Management System* is quite simple; it is build using Object-Oriented Program (OOP) principle in Python. It allows user to add, remove, and list books, where each book is represented as an object with attributes such as title, author, and ISBN. The system ensures data encapsulation through private attributes and getters/setters methos.

Objective:

- Implement OOP Concepts like Classes, objects, encapsulation, and methods
- Manage a collection of books using a Library Class
- Provide a menu-driven interface for ease of interaction
- Enable book addition, removal, and listing with proper validation
- Demonstrate the use of private attributes and getter/setter methods

Source Code:

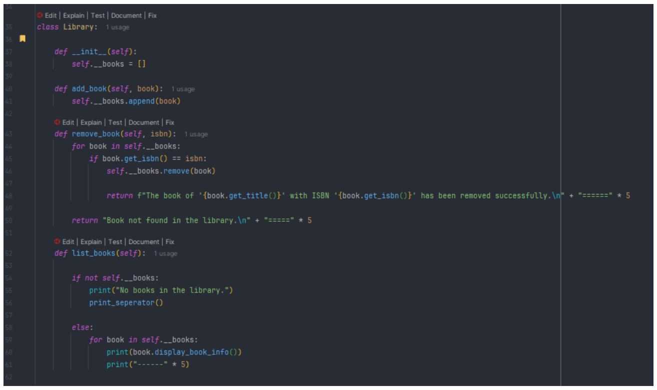
Book Class



Edit | Explain | Test | Document | Fix
1 usage
Edit | Explain | Test | Document | Fix
1 usage (1 dynamic)
2 usages (2 dynamic)
1 usage (1 dynamic)

```python
"""OOP"""

class Book:
    def __init__(self, title, author, isbn):

        self.__title = title
        self.__author = author
        self.__isbn = isbn

    """GETTERS"""

    def get_title(self):
        return self.__title

    def get_author(self):
        return self.__author

    def get_isbn(self):
        return self.__isbn

    """SETTERS"""

    def set_title(self, title):
        self.__title = title

    def set_author(self, author):
        self.__author = author

    def set_isbn(self, isbn):
        self.__isbn = isbn

    """DISPLAY"""

    def display_book_info(self):
        return f"Title: {self.__title}, Author: {self.__author}, ISBN: {self.__isbn}"
```

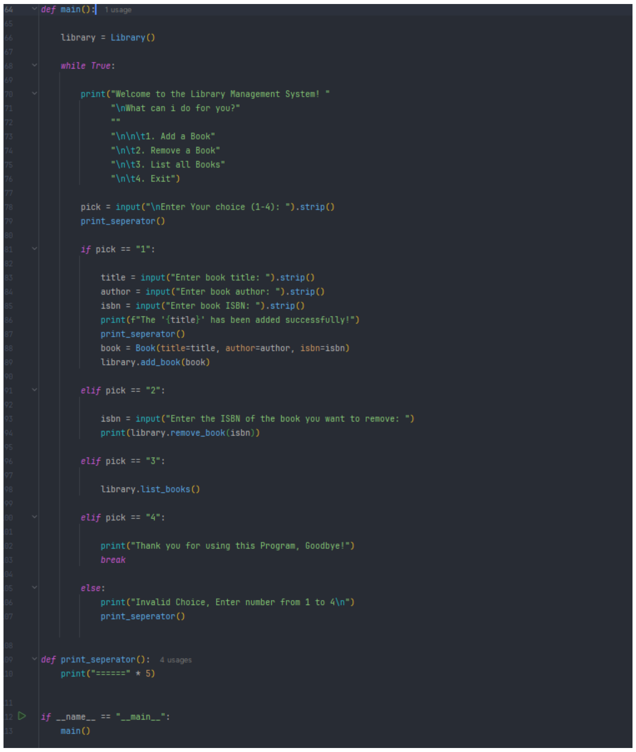
Library Class



Edit | Explain | Test | Document | Fix
1 usage
1 usage
Edit | Explain | Test | Document | Fix
1 usage
Edit | Explain | Test | Document | Fix
1 usage

```python
class Library:

    def __init__(self):
        self.__books = []

    def add_book(self, book):
        self.__books.append(book)

    def remove_book(self, isbn):
        for book in self.__books:
            if book.get_isbn() == isbn:
                self.__books.remove(book)

                return f"The book of '{book.get_title()}' with ISBN '{book.get_isbn()}' has been removed successfully.\n" + "======" * 5

        return "Book not found in the library.\n" + "=====" * 5

    def list_books(self):

        if not self.__books:
            print("No books in the library.")
            print_seperator()

        else:
            for book in self.__books:
                print(book.display_book_info())
                print("------" * 5)
```

Main

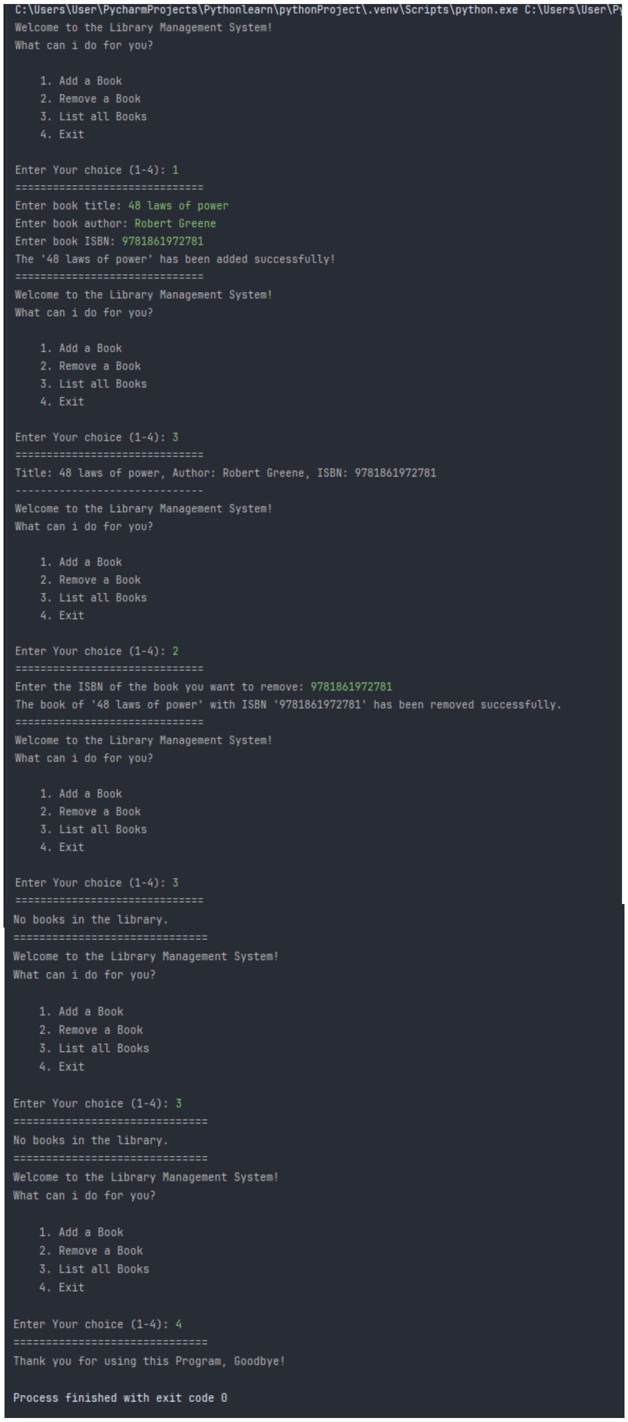


```python
def main():

    library = Library()

    while True:

        print("Welcome to the Library Management System! "
              "\nWhat can i do for you?"
              ""
              "\n\n\t1. Add a Book"
              "\n\t2. Remove a Book"
              "\n\t3. List all Books"
              "\n\t4. Exit")

        pick = input("\nEnter Your choice (1-4): ").strip()
        print_seperator()

        if pick == "1":

            title = input("Enter book title: ").strip()
            author = input("Enter book author: ").strip()
            isbn = input("Enter book ISBN: ").strip()
            print(f"The '{title}' has been added successfully!")
            print_seperator()
            book = Book(title=title, author=author, isbn=isbn)
            library.add_book(book)

        elif pick == "2":

            isbn = input("Enter the ISBN of the book you want to remove: ")
            print(library.remove_book(isbn))

        elif pick == "3":

            library.list_books()

        elif pick == "4":

            print("Thank you for using this Program, Goodbye!")
            break

        else:
            print("Invalid Choice, Enter number from 1 to 4\n")
            print_seperator()


def print_seperator():
    print("======" * 5)


if __name__ == "__main__":
    main()
```

1. Class Structure (Composition)
   - Library class contains a list of Books Object
2. OOP features
   - Attributes & Constructor
     - Books has title, author, isbn
     - Library has a private list __books

   - Encapsulation
     - Getters and setters
3. Main Program
   - The main function creates a library object
   - And it allows users to:
     - Add books
     - Remove Books
     - List Books
     - Exit Program

Sample Output



```
C:\Users\User\PycharmProjects\Pythonlearn\pythonProject\.venv\Scripts\python.exe C:\Users\User\Py
Welcome to the Library Management System!
What can i do for you?

    1. Add a Book
    2. Remove a Book
    3. List all Books
    4. Exit

Enter Your choice (1-4): 1
==============================
Enter book title: 48 laws of power
Enter book author: Robert Greene
Enter book ISBN: 9781861972781
The '48 laws of power' has been added successfully!
==============================
Welcome to the Library Management System!
What can i do for you?

    1. Add a Book
    2. Remove a Book
    3. List all Books
    4. Exit

Enter Your choice (1-4): 3
==============================
Title: 48 laws of power, Author: Robert Greene, ISBN: 9781861972781
------------------------------
Welcome to the Library Management System!
What can i do for you?

    1. Add a Book
    2. Remove a Book
    3. List all Books
    4. Exit

Enter Your choice (1-4): 2
==============================
Enter the ISBN of the book you want to remove: 9781861972781
The book of '48 laws of power' with ISBN '9781861972781' has been removed successfully.
==============================
Welcome to the Library Management System!
What can i do for you?

    1. Add a Book
    2. Remove a Book
    3. List all Books
    4. Exit

Enter Your choice (1-4): 3
==============================
No books in the library.
==============================
Welcome to the Library Management System!
What can i do for you?

    1. Add a Book
    2. Remove a Book
    3. List all Books
    4. Exit

Enter Your choice (1-4): 3
==============================
No books in the library.
==============================
Welcome to the Library Management System!
What can i do for you?

    1. Add a Book
    2. Remove a Book
    3. List all Books
    4. Exit

Enter Your choice (1-4): 4
==============================
Thank you for using this Program, Goodbye!

Process finished with exit code 0
```

TASK 2 (REG EX)

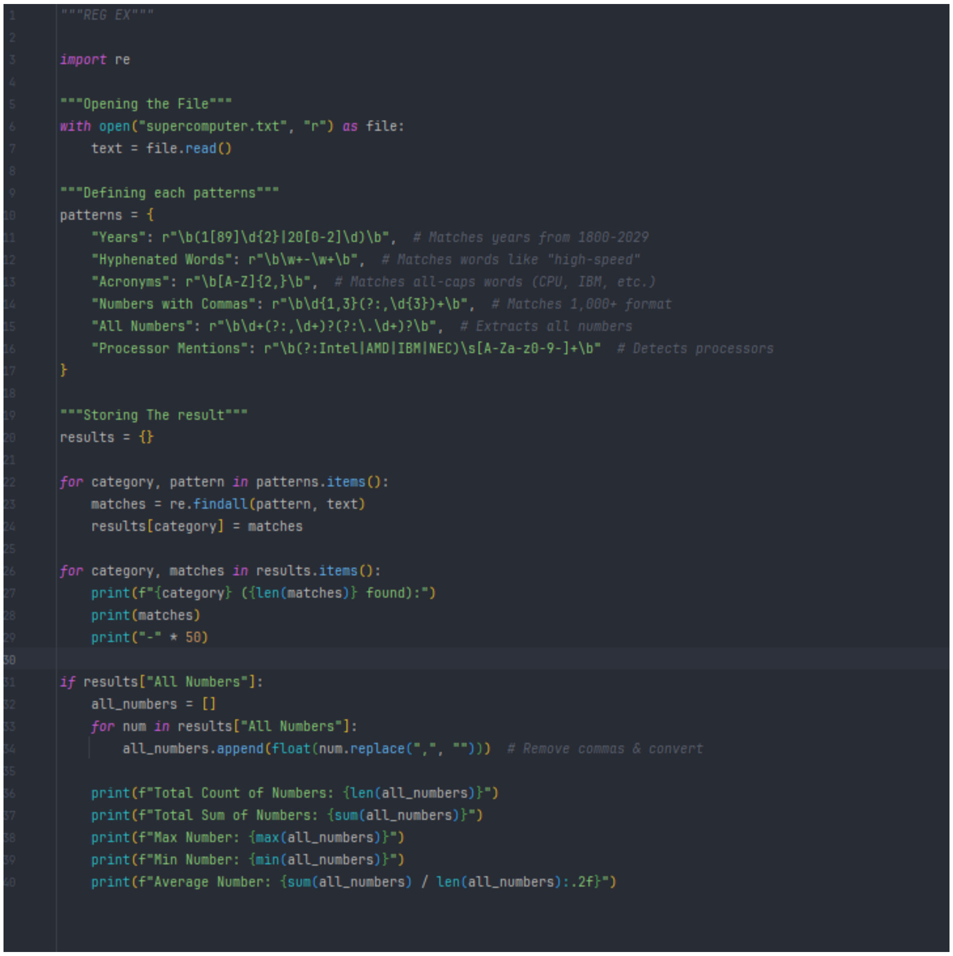
Description:

- The program analyzes and processes a text file called, supercomputer.txt and by extracting a specific pattern using regular expression (regex) it identifies and analyzes elements such as years, hyphenated words, acronyms, number with commas, and processor mention. Additionally, it performs data processing task like counting occurrences, finding minimum and maximum values, computer averages, and summing numbers.

Objectives:

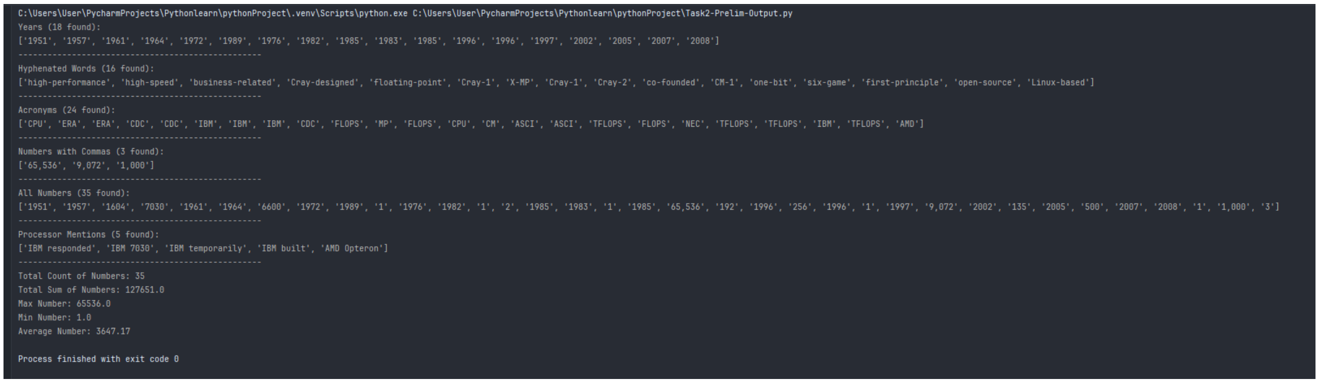
- Utilizing file handling to read and process text files
- Apply regular expressions (regex) to extract relevant data
- Using functions such as search () and findall () pattern matching
- Perform data analysis on extracted numbers (sum, average, min/max values)

Sources Code:



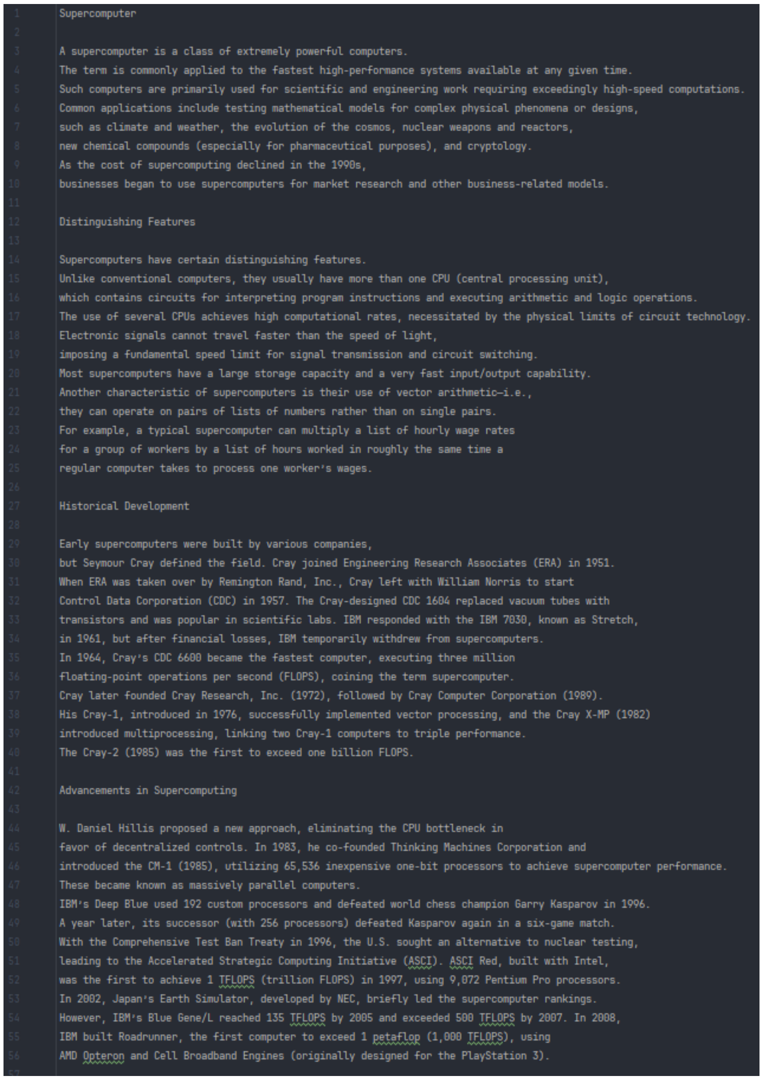
```python
"""REG EX"""

import re

"""Opening the File"""
with open("supercomputer.txt", "r") as file:
    text = file.read()

"""Defining each patterns"""
patterns = {
    "Years": r"\b(1[89]\d{2}|20[0-2]\d)\b",  # Matches years from 1800-2029
    "Hyphenated Words": r"\b\w+-\w+\b",  # Matches words like "high-speed"
    "Acronyms": r"\b[A-Z]{2,}\b",  # Matches all-caps words (CPU, IBM, etc.)
    "Numbers with Commas": r"\b\d{1,3}(?:,\d{3})+\b",  # Matches 1,000+ format
    "All Numbers": r"\b\d+(?:,\d+)?(?:\.\d+)?\b",  # Extracts all numbers
    "Processor Mentions": r"\b(?:Intel|AMD|IBM|NEC)\s[A-Za-z0-9-]+\b"  # Detects processors
}

"""Storing The result"""
results = {}

for category, pattern in patterns.items():
    matches = re.findall(pattern, text)
    results[category] = matches

for category, matches in results.items():
    print(f"{category} ({len(matches)} found):")
    print(matches)
    print("-" * 50)

if results["All Numbers"]:
    all_numbers = []
    for num in results["All Numbers"]:
        all_numbers.append(float(num.replace(",", "")))  # Remove commas & convert

    print(f"Total Count of Numbers: {len(all_numbers)}")
    print(f"Total Sum of Numbers: {sum(all_numbers)}")
    print(f"Max Number: {max(all_numbers)}")
    print(f"Min Number: {min(all_numbers)}")
    print(f"Average Number: {sum(all_numbers) / len(all_numbers):.2f}")
```

Sample Output:



```
C:\Users\User\PycharmProjects\Pythonlearn\pythonProject\.venv\Scripts\python.exe C:\Users\User\PycharmProjects\Pythonlearn\pythonProject\Task2-Prelim-Output.py
Years (18 found):
['1951', '1957', '1961', '1964', '1972', '1989', '1976', '1982', '1985', '1983', '1985', '1996', '1996', '1997', '2002', '2005', '2007', '2008']
--------------------------------------------------
Hyphenated Words (16 found):
['high-performance', 'high-speed', 'business-related', 'Cray-designed', 'floating-point', 'Cray-1', 'X-MP', 'Cray-1', 'Cray-2', 'co-founded', 'CM-1', 'one-bit', 'six-game', 'first-principle', 'open-source', 'Linux-based']
--------------------------------------------------
Acronyms (24 found):
['CPU', 'ERA', 'ERA', 'CDC', 'CDC', 'IBM', 'IBM', 'IBM', 'CDC', 'FLOPS', 'MP', 'FLOPS', 'CPU', 'CM', 'ASCI', 'ASCI', 'TFLOPS', 'FLOPS', 'NEC', 'TFLOPS', 'TFLOPS', 'IBM', 'TFLOPS', 'AMD']
--------------------------------------------------
Numbers with Commas (3 found):
['65,536', '9,072', '1,000']
--------------------------------------------------
All Numbers (35 found):
['1951', '1957', '1604', '7030', '1961', '1964', '6600', '1972', '1989', '1', '1976', '1982', '1', '2', '1985', '1983', '1', '1985', '65,536', '192', '1996', '256', '1996', '1', '1997', '9,072', '2002', '135', '2005', '500', '2007', '2008', '1', '1,000', '3']
--------------------------------------------------
Processor Mentions (5 found):
['IBM responded', 'IBM 7030', 'IBM temporarily', 'IBM built', 'AMD Opteron']
--------------------------------------------------
Total Count of Numbers: 35
Total Sum of Numbers: 127651.0
Max Number: 65536.0
Min Number: 1.0
Average Number: 3647.17

Process finished with exit code 0
```

Source Text:



```
Supercomputer

A supercomputer is a class of extremely powerful computers.
The term is commonly applied to the fastest high-performance systems available at any given time.
Such computers are primarily used for scientific and engineering work requiring exceedingly high-speed computations.
Common applications include testing mathematical models for complex physical phenomena or designs,
such as climate and weather, the evolution of the cosmos, nuclear weapons and reactors,
new chemical compounds (especially for pharmaceutical purposes), and cryptology.
As the cost of supercomputing declined in the 1990s,
businesses began to use supercomputers for market research and other business-related models.

Distinguishing Features

Supercomputers have certain distinguishing features.
Unlike conventional computers, they usually have more than one CPU (central processing unit),
which contains circuits for interpreting program instructions and executing arithmetic and logic operations.
The use of several CPUs achieves high computational rates, necessitated by the physical limits of circuit technology.
Electronic signals cannot travel faster than the speed of light,
imposing a fundamental speed limit for signal transmission and circuit switching.
Most supercomputers have a large storage capacity and a very fast input/output capability.
Another characteristic of supercomputers is their use of vector arithmetic—i.e.,
they can operate on pairs of lists of numbers rather than on single pairs.
For example, a typical supercomputer can multiply a list of hourly wage rates
for a group of workers by a list of hours worked in roughly the same time a
regular computer takes to process one worker’s wages.

Historical Development

Early supercomputers were built by various companies,
but Seymour Cray defined the field. Cray joined Engineering Research Associates (ERA) in 1951.
When ERA was taken over by Remington Rand, Inc., Cray left with William Norris to start
Control Data Corporation (CDC) in 1957. The Cray-designed CDC 1604 replaced vacuum tubes with
transistors and was popular in scientific labs. IBM responded with the IBM 7030, known as Stretch,
in 1961, but after financial losses, IBM temporarily withdrew from supercomputers.
In 1964, Cray’s CDC 6600 became the fastest computer, executing three million
floating-point operations per second (FLOPS), coining the term supercomputer.
Cray later founded Cray Research, Inc. (1972), followed by Cray Computer Corporation (1989).
His Cray-1, introduced in 1976, successfully implemented vector processing, and the Cray X-MP (1982)
introduced multiprocessing, linking two Cray-1 computers to triple performance.
The Cray-2 (1985) was the first to exceed one billion FLOPS.

Advancements in Supercomputing

W. Daniel Hillis proposed a new approach, eliminating the CPU bottleneck in
favor of decentralized controls. In 1983, he co-founded Thinking Machines Corporation and
introduced the CM-1 (1985), utilizing 65,536 inexpensive one-bit processors to achieve supercomputer performance.
These became known as massively parallel computers.
IBM’s Deep Blue used 192 custom processors and defeated world chess champion Garry Kasparov in 1996.
A year later, its successor (with 256 processors) defeated Kasparov again in a six-game match.
With the Comprehensive Test Ban Treaty in 1996, the U.S. sought an alternative to nuclear testing,
leading to the Accelerated Strategic Computing Initiative (ASCI). ASCI Red, built with Intel,
was the first to achieve 1 TFLOPS (trillion FLOPS) in 1997, using 9,072 Pentium Pro processors.
In 2002, Japan’s Earth Simulator, developed by NEC, briefly led the supercomputer rankings.
However, IBM’s Blue Gene/L reached 135 TFLOPS by 2005 and exceeded 500 TFLOPS by 2007. In 2008,
IBM built Roadrunner, the first computer to exceed 1 petaflop (1,000 TFLOPS), using
AMD Opteron and Cell Broadband Engines (originally designed for the PlayStation 3).
```

Link: [https://www.britannica.com/technology/supercomputer]