CENG204 - Programming Languages Concepts

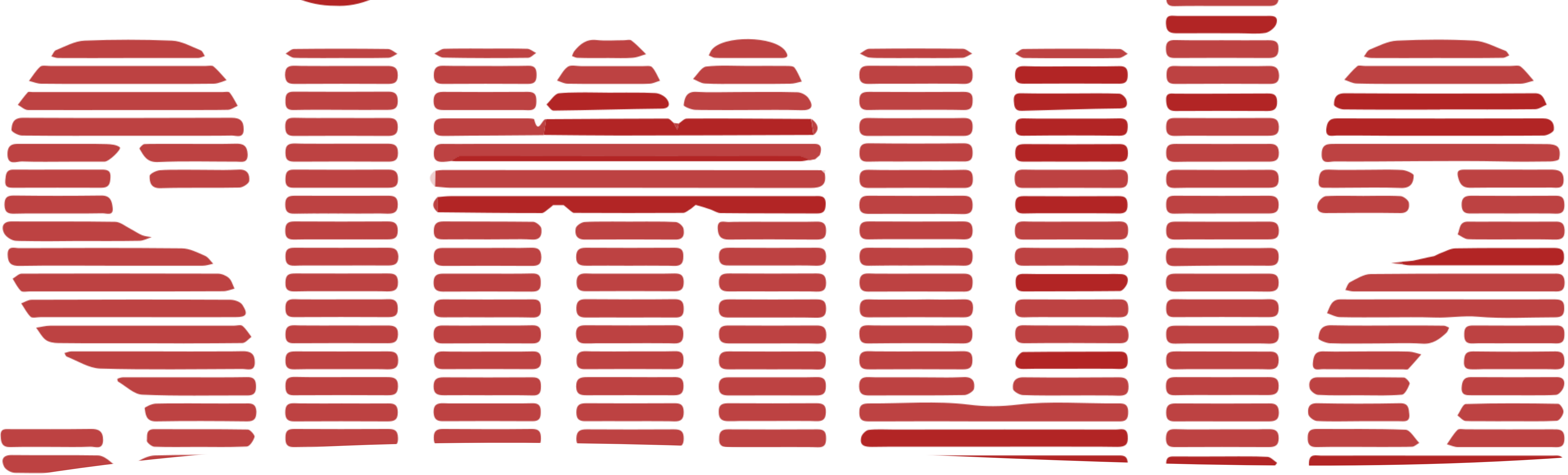
Asst. Prof. Dr. Emre ŞATIR

Lecture 5
Evolution of the Major Programming Languages
(Part 2)

Lecture 5 Topics

- The Beginnings of Data Abstraction: SIMULA 67
- A Discrete Time Simulation Programming Language: GPSS
- Some Early Descendants of the ALGOLs
  - Simplicity by Design: Pascal
  - A Systems Language: C
- Programming Based on Logic: Prolog
- Structured Query Language (SQL)
- The Language of SAP: Abap
- Matlab Environment and Language
- History's Largest Design Effort: Ada
- Object-Oriented Programming: Smalltalk
- Combining Imperative and Object-Oriented Features: C++
  - Objective-C, Eiffel, Delphi.
- An Imperative-Based Object-Oriented Language: Java
- The Flagship .NET Language: C#
- An Open-source Programming Language Designed at Google: Go
- Scripting Languages
- Markup/Programming Hybrid Languages

The Beginnings of Data Abstraction: SIMULA 67



- In the late 1970s, a shift from procedure-oriented to data-oriented program design methodologies began.
- Simply put, data-oriented methods emphasize data design, focusing on the use of abstract data types to solve problems.
- The first language to provide even limited support for data abstraction was SIMULA 67.
- Although SIMULA 67 never achieved widespread use and had little impact on the programmers and computing of its time, some of the constructs it introduced make it historically important.

SIMULA 67

- Two Norwegians, Kristen Nygaard and Ole-Johan Dahl, developed the language SIMULA I between 1962 and 1964 at the Norwegian Computing Center (NCC) in Oslo.
- SIMULA I was designed exclusively for system simulation.
- As soon as the SIMULA I implementation was completed, Nygaard and Dahl began efforts to extend the language by adding new features and modifying some existing constructs in order to make the language useful for general-purpose applications.
- The result of this work was SIMULA 67 (It’s based on ALGOL 60 and SIMULA I).

SIMULA 67

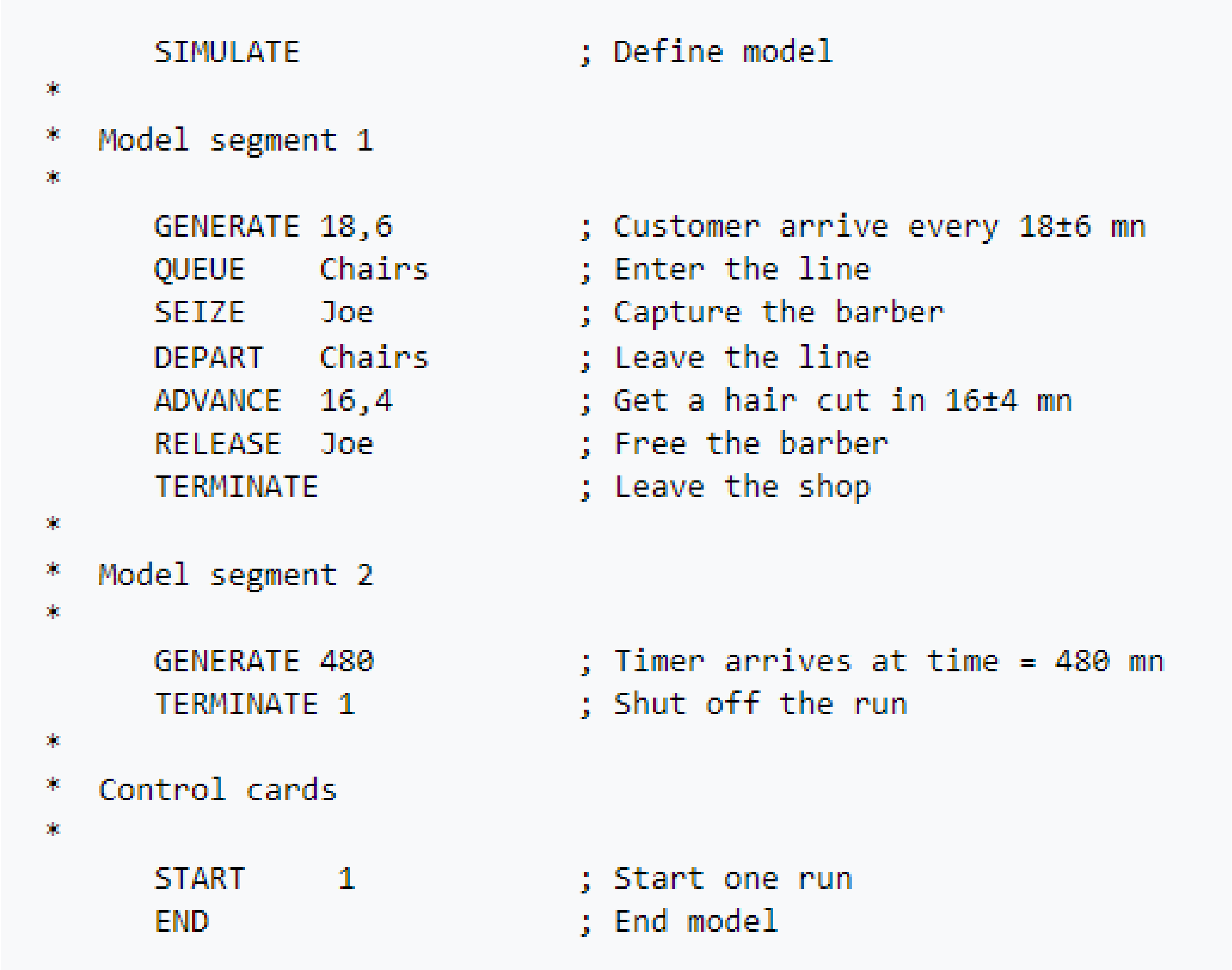
- The primary deficiency of languages at that time for simulation applications was the design of its subprograms.
- Simulation requires subprograms that are allowed to restart at the position where they previously stopped. Subprograms with this kind of control are known as **coroutines** because the caller and called subprograms have a somewhat equal relationship with each other, rather than the rigid master/slave relationship they have in most imperative languages (**subroutines**).
- To provide support for coroutines in SIMULA 67, the "**class**" construct was developed. This was an important development because the concept of "data abstraction" began with it and data abstraction provides "the foundation for object-oriented programming".
- *** It is interesting to note that the important concept of data abstraction was not developed and attributed to the class construct until 1972, when Hoare (1972) recognized the connection.

A Discrete Time Simulation Programming Language: GPSS

GPSS

- **General Purpose Simulation System** (GPSS) is a discrete time simulation programming language.
- **GPSS** was developed by IBM's Geoffrey Gordon at the beginning of the 1960s. He named it **G**ordon's **P**rogrammable **S**imulation **S**ystem. The name was changed when IBM decided to release it as a product.
- The "General Purpose" part of the new name was to create a standard in waiting-line simulations.
- A system is modelled as “transactions” enter the system and are passed from one service (represented by “blocks”) to another.

An Example GPSS Program



```
      SIMULATE               ; Define model
*
*  Model segment 1
*
      GENERATE 18,6          ; Customer arrive every 18±6 mn
      QUEUE    Chairs        ; Enter the line
      SEIZE    Joe           ; Capture the barber
      DEPART   Chairs        ; Leave the line
      ADVANCE  16,4          ; Get a hair cut in 16±4 mn
      RELEASE  Joe           ; Free the barber
      TERMINATE              ; Leave the shop
*
*  Model segment 2
*
      GENERATE 480           ; Timer arrives at time = 480 mn
      TERMINATE 1            ; Shut off the run
*
*  Control cards
*
      START     1            ; Start one run
      END                    ; End model
```

Some Early Descendants of the ALGOLs

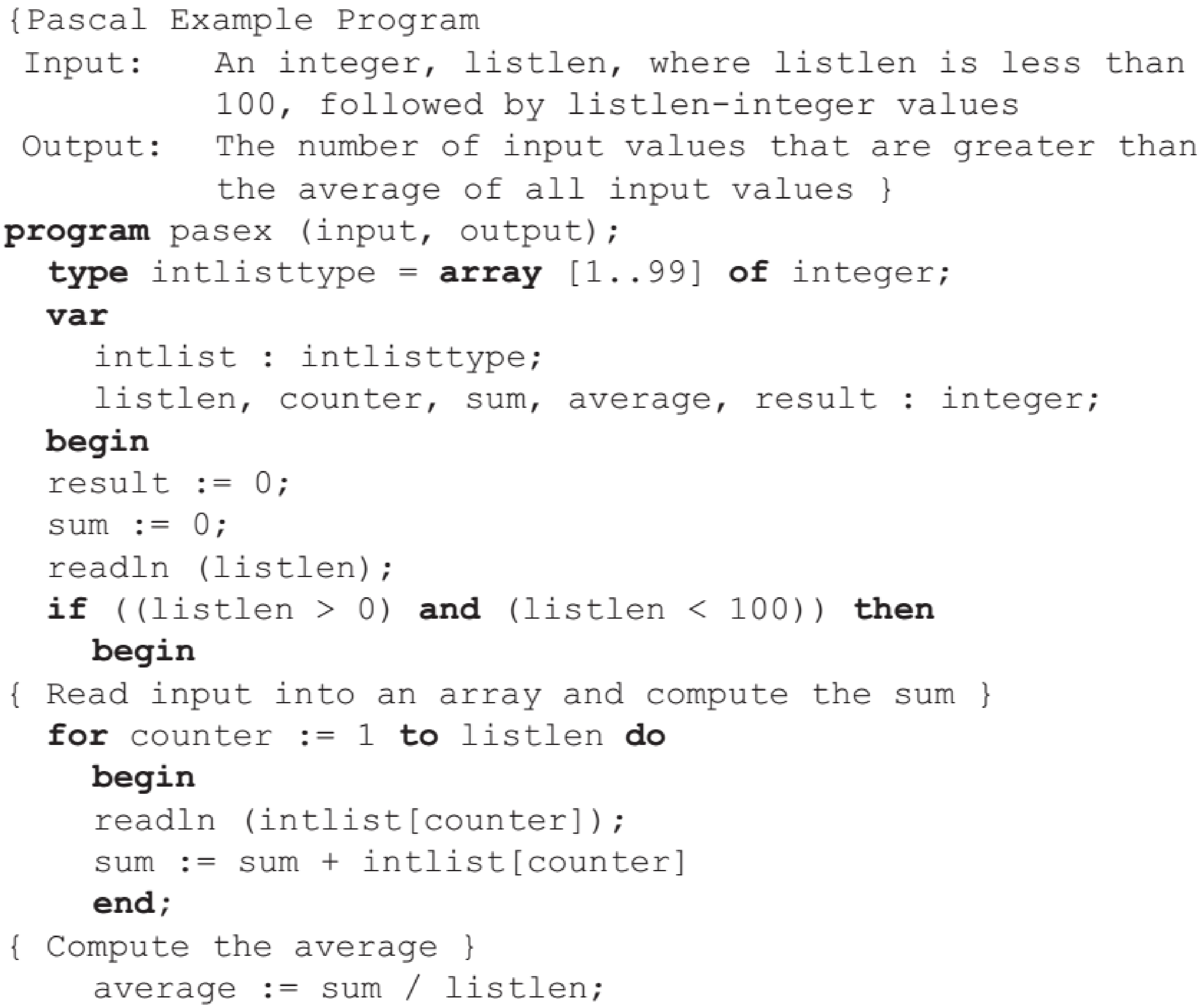
Some Early Descendants of the ALGOLs

- ALGOL 68 was the source of several new ideas in language design, some of which were subsequently adopted by other languages.
- Even though it never achieved widespread use in either Europe or the United States.
- All imperative languages owe some of their design to ALGOL 60 and/or ALGOL 68.
- This section discusses some of the early descendants of these languages.
  - Simplicity by Design: **Pascal**
  - A Systems Language: **C**

Simplicity by Design
Pascal

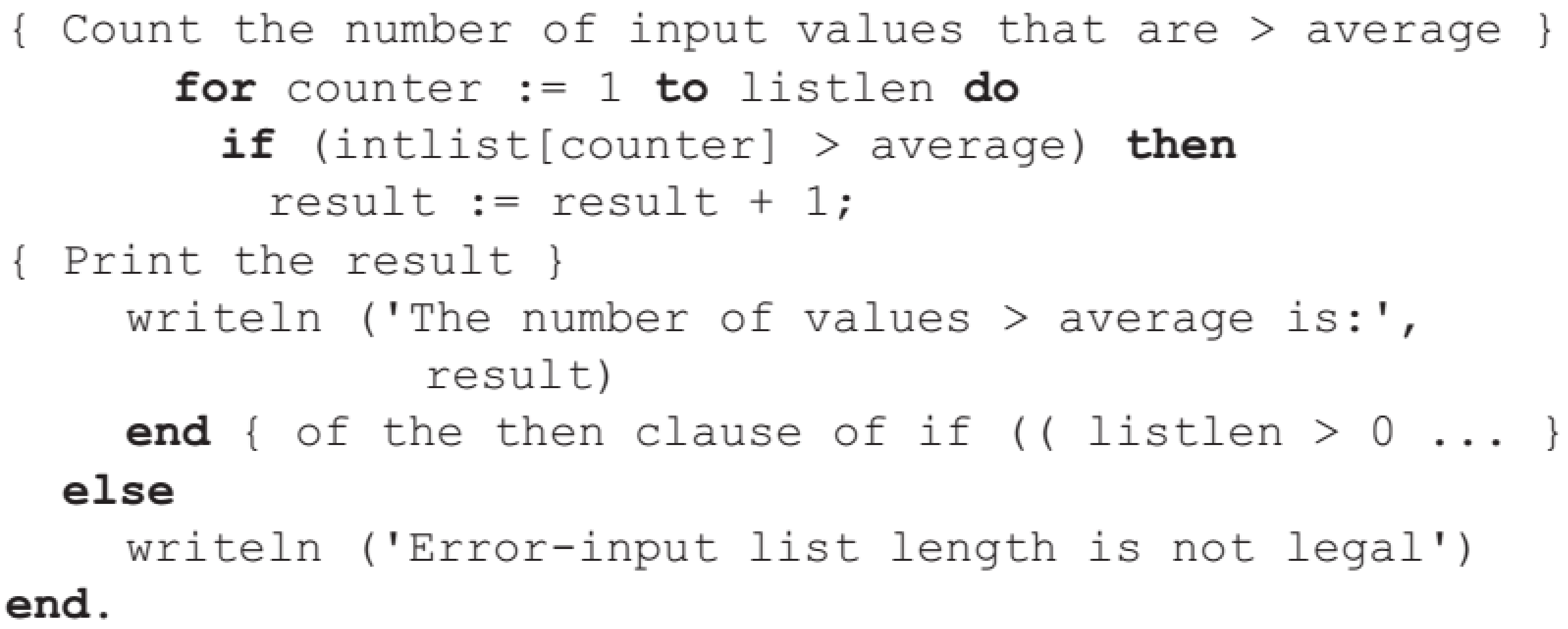
- Developed in 1971 by Niklaus Wirth (a former member of the ALGOL 68 committee).
- Pascal is named after Blaise **Pascal**, a seventeenth-century French philosopher and mathematician who invented the first mechanical adding machine in 1642 (among other things).
- Designed for teaching structured programming.
- Small, simple, nothing really new.
- Largest impact was on teaching programming.
  - From mid-1970s until the late 1990s, it was the most widely used language for teaching programming.

An Example Pascal Program



```
{Pascal Example Program
 Input:   An integer, listlen, where listlen is less than
          100, followed by listlen-integer values
 Output:  The number of input values that are greater than
          the average of all input values }
program pasex (input, output);
  type intlisttype = array [1..99] of integer;
  var
    intlist : intlisttype;
    listlen, counter, sum, average, result : integer;
  begin
  result := 0;
  sum := 0;
  readln (listlen);
  if ((listlen > 0) and (listlen < 100)) then
    begin
{ Read input into an array and compute the sum }
  for counter := 1 to listlen do
    begin
    readln (intlist[counter]);
    sum := sum + intlist[counter]
    end;
{ Compute the average }
    average := sum / listlen;
```

An Example Pascal Program (continued)



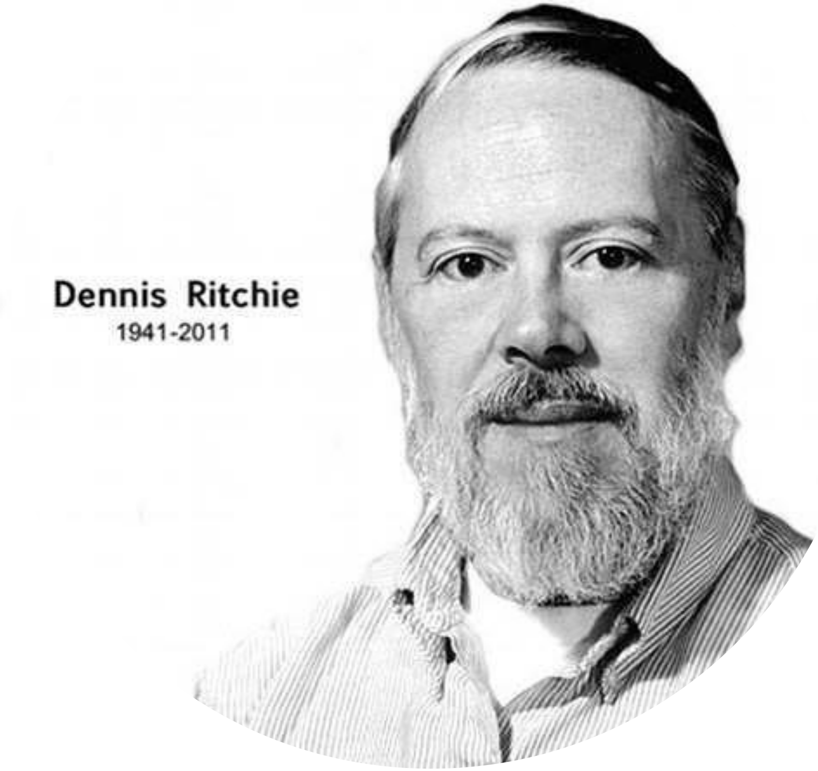
```
{ Count the number of input values that are > average }
      for counter := 1 to listlen do
        if (intlist[counter] > average) then
         result := result + 1;
{ Print the result }
    writeln ('The number of values > average is:',
               result)
    end { of the then clause of if (( listlen > 0 ... }
  else
    writeln ('Error-input list length is not legal')
end.
```

A Systems Language: C

- C’s ancestors include CPL, BCPL, B, and ALGOL 68.
- CPL was developed at Cambridge University in the early 1960s.
- BCPL is a simple systems language, also developed at Cambridge, this time by Martin Richards in 1967.
- The first work on the UNIX operating system was done in the late 1960s by Ken Thompson at Bell Laboratories. The first version was written in assembly language. The first high-level language implemented under UNIX was B, which was based on BCPL.
- B was designed and implemented by Thompson in 1970.

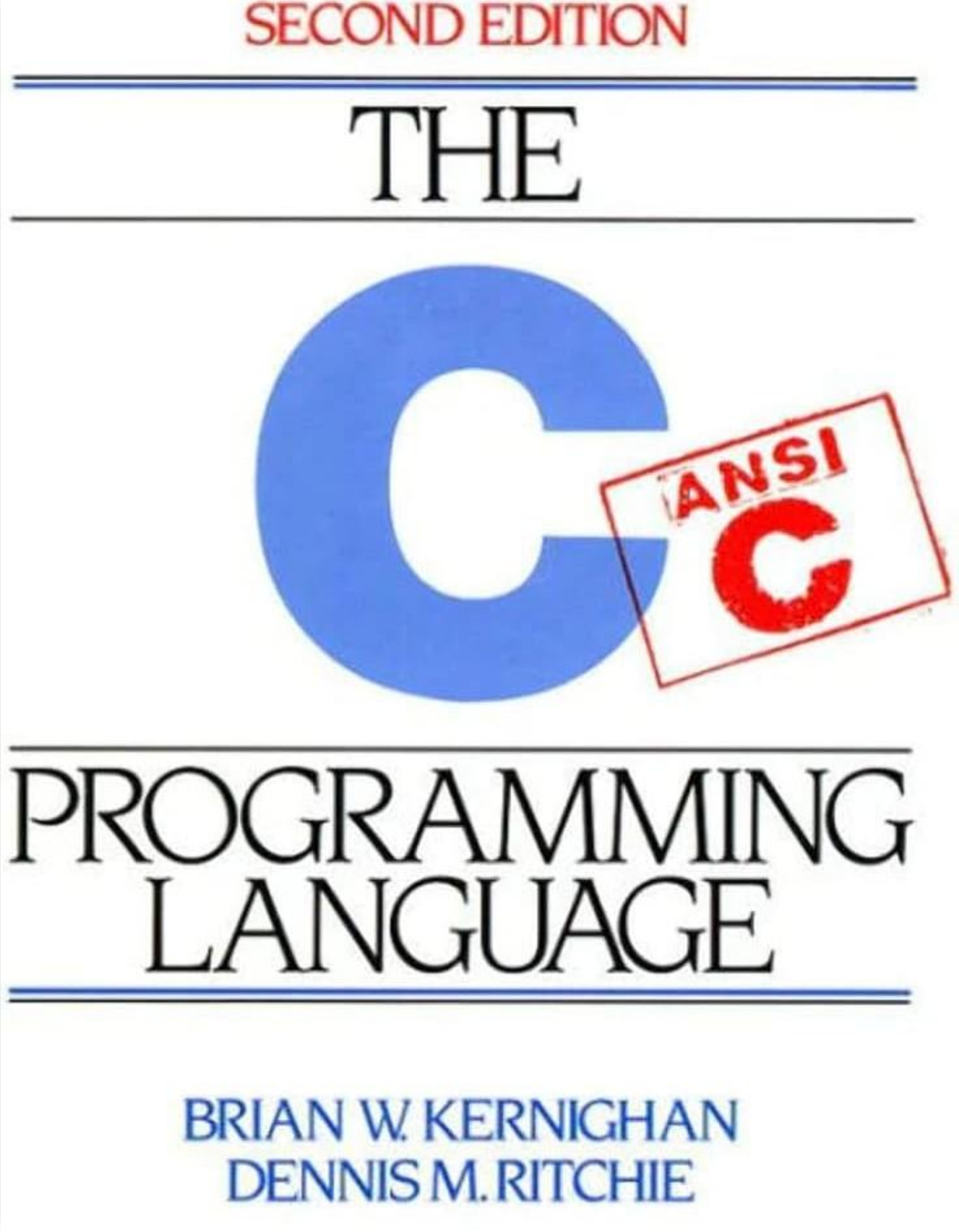
C

- Some problems led to the development of a new typed language based on B (Neither BCPL nor B is a typed language).
- Originally called NB but later named C, it was designed and implemented by Dennis Ritchie at Bell Laboratories in 1972. In some cases, through BCPL, and in other cases directly, C was influenced by ALGOL 68.
- Like Pascal, C contributed little to the previously known collection of language features, but it has been widely used over a long period of time.
- Although originally designed for systems programming, C is well suited for a wide variety of applications.


Dennis Ritchie
1941-2011

C

- C has adequate control statements and data-structuring facilities to allow its use in many application areas.
- It also has a rich set of operators that provide a high degree of expressiveness.
- One of the most important reasons why C is both liked and disliked is its lack of complete type checking (flexible but insecure).



Programming Based on Logic: Prolog

Programming Based on Logic: Prolog

- Simply put, logic programming is the use of a formal logic notation to communicate computational processes to a computer. "**Predicate calculus**" is the notation used in current logic programming languages.
- Programming in logic programming languages is nonprocedural (not imperative). Programs in such languages do not state exactly how a result is to be computed but rather describe the necessary form and/or characteristics of the result.
- What is needed to provide this capability in logic programming languages is a concise means of supplying the computer with both the relevant information and an inferencing process for computing desired results.
- Predicate calculus supplies the basic form of communication to the computer, and the proof method, named **resolution**.

Prolog

- The name Prolog is from **PRO**gramming **LOG**ic.
- Developed, by Comerauer and Roussel (University of Aix-Marseille), with help from Kowalski (University of Edinburgh)
- Programming Domain: AI (like LISP).
- Can be summarized as being an intelligent database system that uses an inferencing process to infer the truth of given queries.
- Few application areas.

Prolog Example

- The database of a Prolog program consists of two kinds of statements: **facts** and **rules**. The following are examples of fact statements:
  - `parent(vern, joanne).`
  - `parent(joanne, jake).`
- These state that vern is the parent of joanne, and joanne is the parent of jake. An example of a rule statement is
  - `grandparent(X, Z) :- parent(X, Y), parent(Y, Z).`
- This states that "it can be deduced that X is the grandparent of Z if it is true that X is the parent of Y and Y is the parent of Z".
- The Prolog database can be interactively queried with goal statements, example of which is
  - `grandparent(vern, jake).` → true

Structured Query Language (SQL)

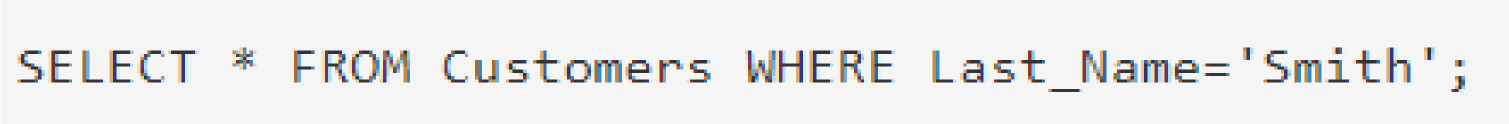
Structured Query Language (SQL)

- Relational database management systems (RDBMSs) store data in the form of tables.
- Queries on such databases are often stated in Structured Query Language (SQL).
- SQL was initially developed at IBM by Donald D. Chamberlin and Raymond F. Boyce after learning about the relational model from Edgar F. Codd in the early 1970s.
- SQL was one of the first commercial languages to use Edgar F. Codd’s relational model. The model was described in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks".

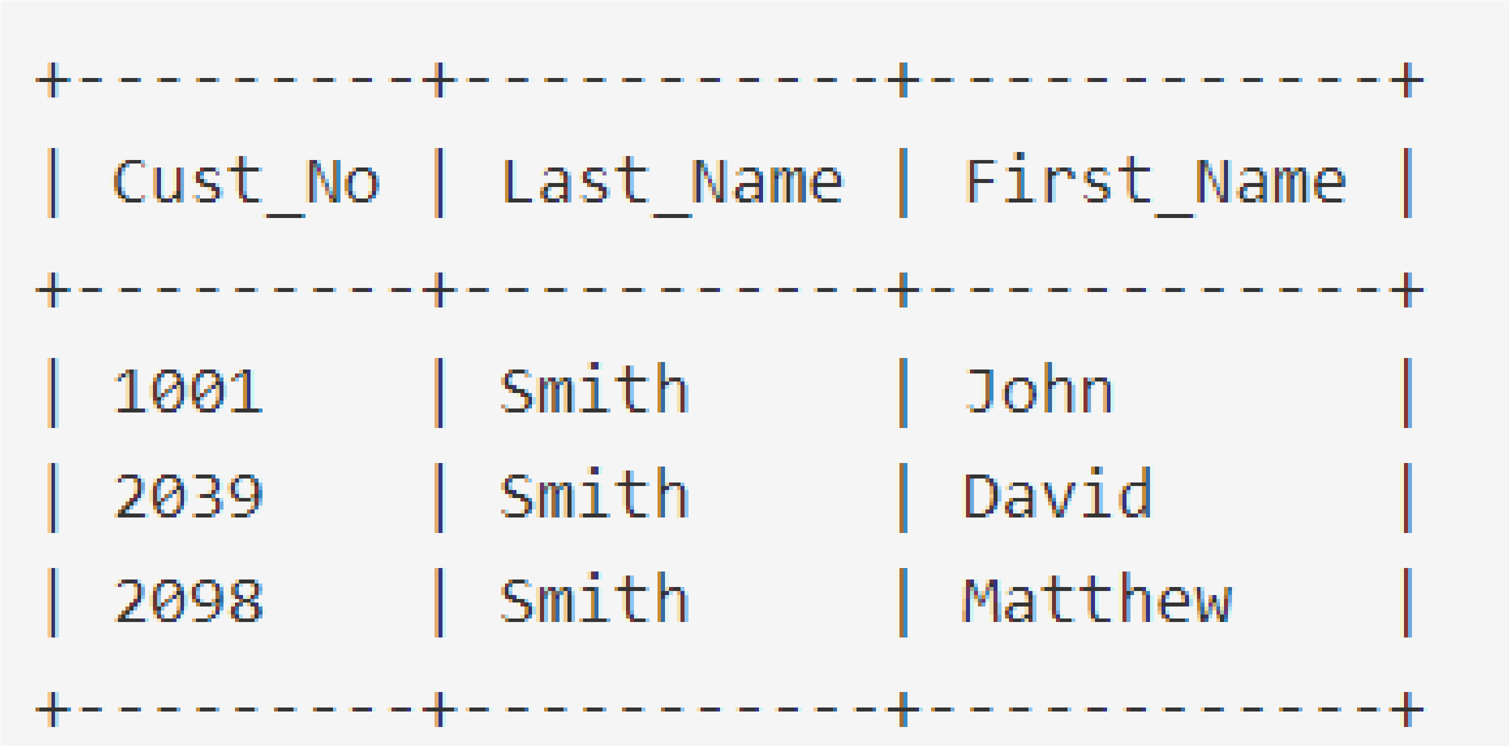
Structured Query Language (SQL)

- Early version, initially called SEQUEL (Structured English Query Language), was designed to manipulate and retrieve data stored in IBM's original database management system, “System R”, which a group at IBM San Jose Research Laboratory had developed during the 1970s.
- SQL is essentially a declarative language (Like GPSS, Prolog, etc.).
- SQL is nonprocedural (not imperative) in the same sense that logic programming is nonprocedural. The user does not describe how to retrieve the answer; rather, he or she describes only the characteristics of the answer.

SQL Example



```
SELECT * FROM Customers WHERE Last_Name='Smith';
```



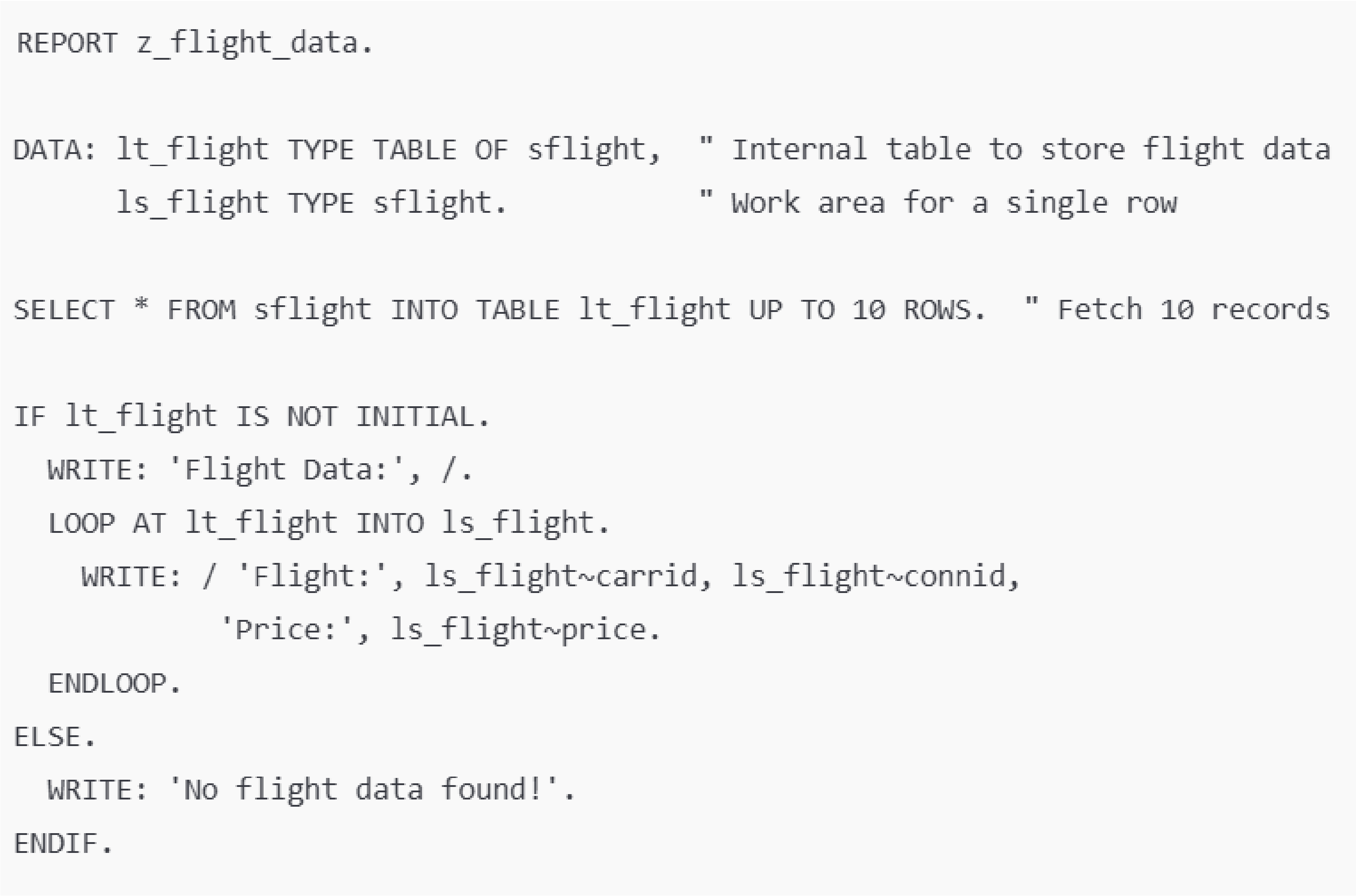
```
+---------+-----------+------------+
| Cust_No | Last_Name | First_Name |
+---------+-----------+------------+
| 1001    | Smith     | John       |
| 2039    | Smith     | David      |
| 2098    | Smith     | Matthew    |
+---------+-----------+------------+
```

The Language of SAP: Abap

The Language of SAP: Abap

- SAP (Systems, Applications, and Products in Data Processing) is a Germany based leading enterprise resource planning (ERP) software that helps businesses manage various processes such as finance, logistics, human resources, and supply chain management.
- ABAP (Advanced Business Application Programming) is a high-level programming language developed by SAP (in 1983) for building applications within the SAP ERP system. It is used for customizing SAP applications, developing reports, and handling data processing.

Abap Example

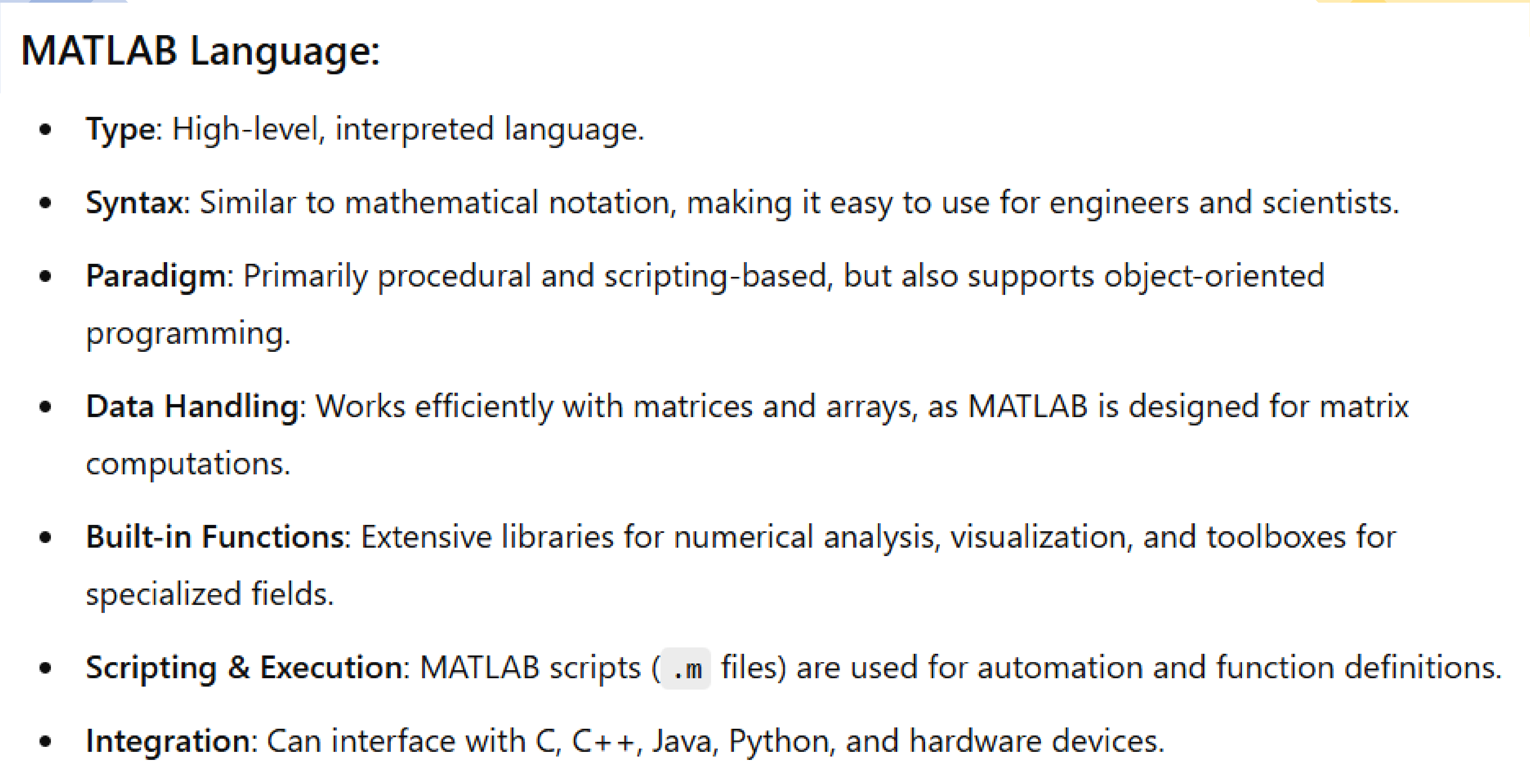


```
REPORT z_flight_data.

DATA: lt_flight TYPE TABLE OF sflight,  " Internal table to store flight data
      ls_flight TYPE sflight.           " Work area for a single row

SELECT * FROM sflight INTO TABLE lt_flight UP TO 10 ROWS.  " Fetch 10 records

IF lt_flight IS NOT INITIAL.
  WRITE: 'Flight Data:', /.
  LOOP AT lt_flight INTO ls_flight.
    WRITE: / 'Flight:', ls_flight~carrid, ls_flight~connid,
            'Price:', ls_flight~price.
  ENDLOOP.
ELSE.
  WRITE: 'No flight data found!'.
ENDIF.
```

Matlab Environment and Language

Matlab Environment and Language

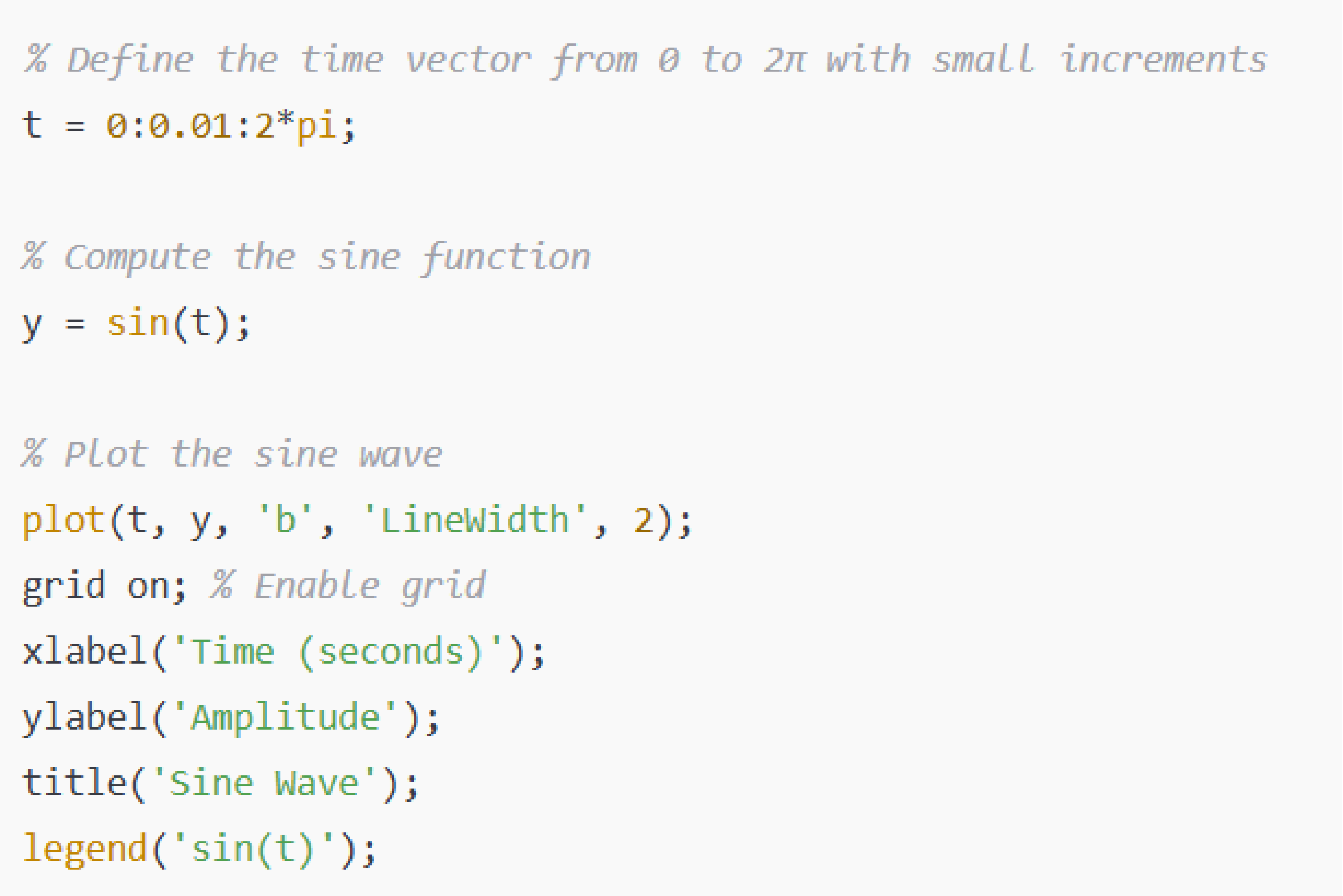
- MATLAB (Matrix Laboratory) is a high-level programming environment and numerical computing platform developed by MathWorks.
- It is widely used for mathematical modeling, simulation, data analysis, signal processing, machine learning, and engineering applications.
- MATLAB was developed in the late 1970s by Cleve Moler, a professor of mathematics, as a simple way for his students to access LINPACK and EISPACK, which are libraries for numerical linear algebra. Initially, MATLAB was a purely academic tool.
- In 1984, Moler, along with Jack Little and Steve Bangert, founded MathWorks to commercialize MATLAB. Over time, it evolved into a powerful computing environment with toolboxes for specialized applications like signal processing, control systems, and machine learning.
- MATLAB has continuously expanded with features such as GUI development, object-oriented programming, parallel computing, and deep learning support, making it one of the most widely used platforms in engineering, science, and industry today.



MATLAB Language:

- **Type**: High-level, interpreted language.
- **Syntax**: Similar to mathematical notation, making it easy to use for engineers and scientists.
- **Paradigm**: Primarily procedural and scripting-based, but also supports object-oriented programming.
- **Data Handling**: Works efficiently with matrices and arrays, as MATLAB is designed for matrix computations.
- **Built-in Functions**: Extensive libraries for numerical analysis, visualization, and toolboxes for specialized fields.
- **Scripting & Execution**: MATLAB scripts (`.m` files) are used for automation and function definitions.
- **Integration**: Can interface with C, C++, Java, Python, and hardware devices.

Matlab Language Example

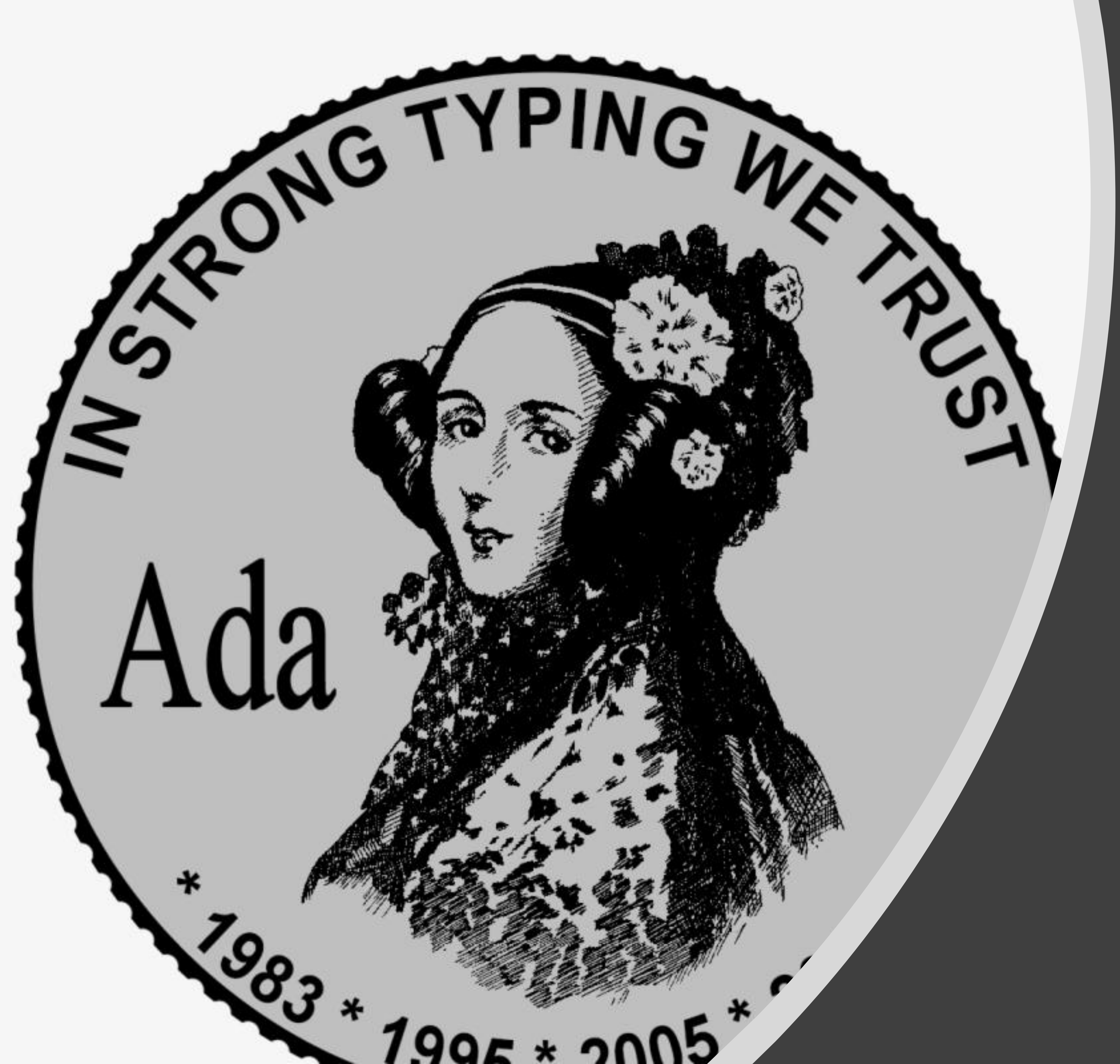


```
% Define the time vector from 0 to 2π with small increments
t = 0:0.01:2*pi;

% Compute the sine function
y = sin(t);

% Plot the sine wave
plot(t, y, 'b', 'LineWidth', 2);
grid on; % Enable grid
xlabel('Time (seconds)');
ylabel('Amplitude');
title('Sine Wave');
legend('sin(t)');
```

History's Largest Design Effort: Ada

Ada

- The Ada language is the result of the most extensive and expensive language design effort ever undertaken.
- Huge design effort, involving hundreds of people, much money, and about eight years.
- The Ada language was developed for the Department of Defense (DoD).
- Named Ada after Ada Lovelace (Byron), the first programmer.



It has built-in language support for design by extremely strong typing.

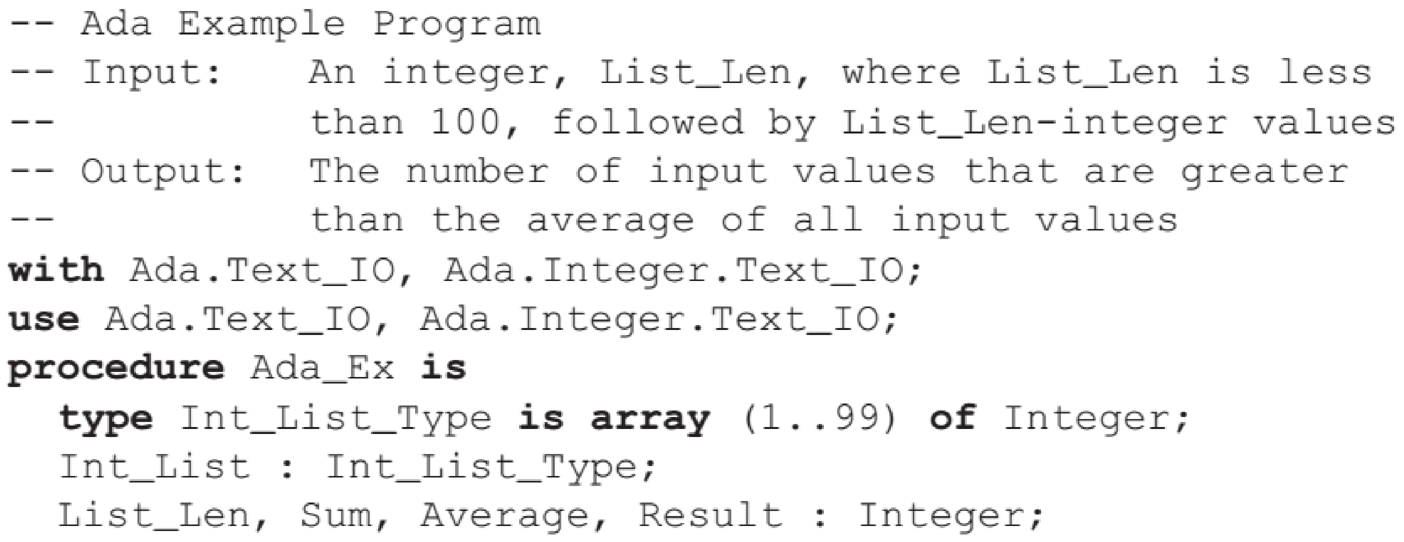
Contributions

- Packages - support for data abstraction.
- Extensive facilities for exception handling.
- Generic program units
- Concurrency - through the tasking model

Ada

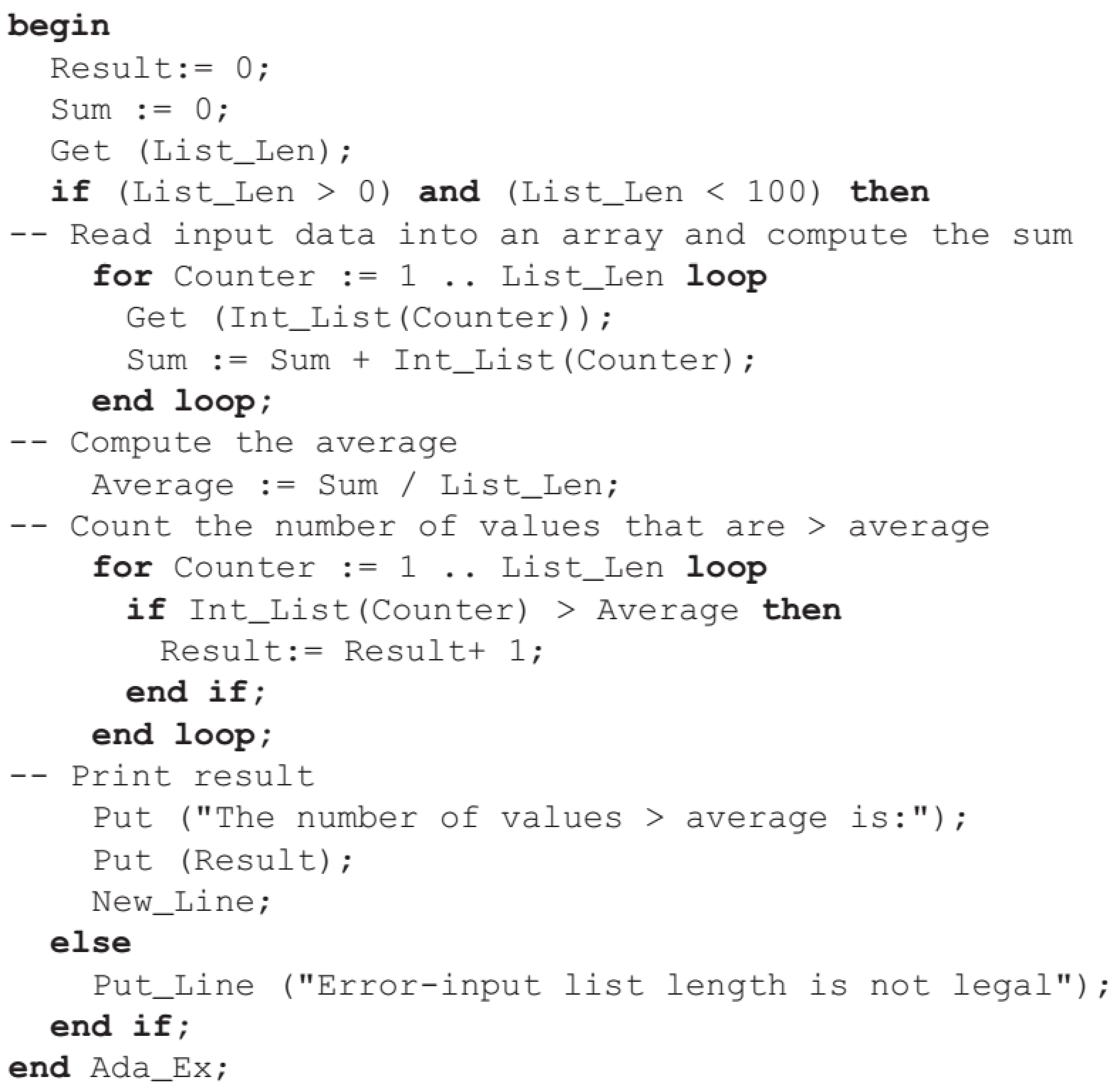
- Ada 95 (began in 1988)
- Ada 2005
- Ada is widely used in both commercial and defense avionics, air traffic control, and rail transportation, as well as in other areas.
- Popularity suffered because the DoD no longer requires its use but also because of popularity of C++.

An Example Ada Program



```
-- Ada Example Program
-- Input:    An integer, List_Len, where List_Len is less
--           than 100, followed by List_Len-integer values
-- Output:   The number of input values that are greater
--           than the average of all input values
with Ada.Text_IO, Ada.Integer.Text_IO;
use Ada.Text_IO, Ada.Integer.Text_IO;
procedure Ada_Ex is
  type Int_List_Type is array (1..99) of Integer;
  Int_List : Int_List_Type;
  List_Len, Sum, Average, Result : Integer;
```

An Example Ada Program (continued)

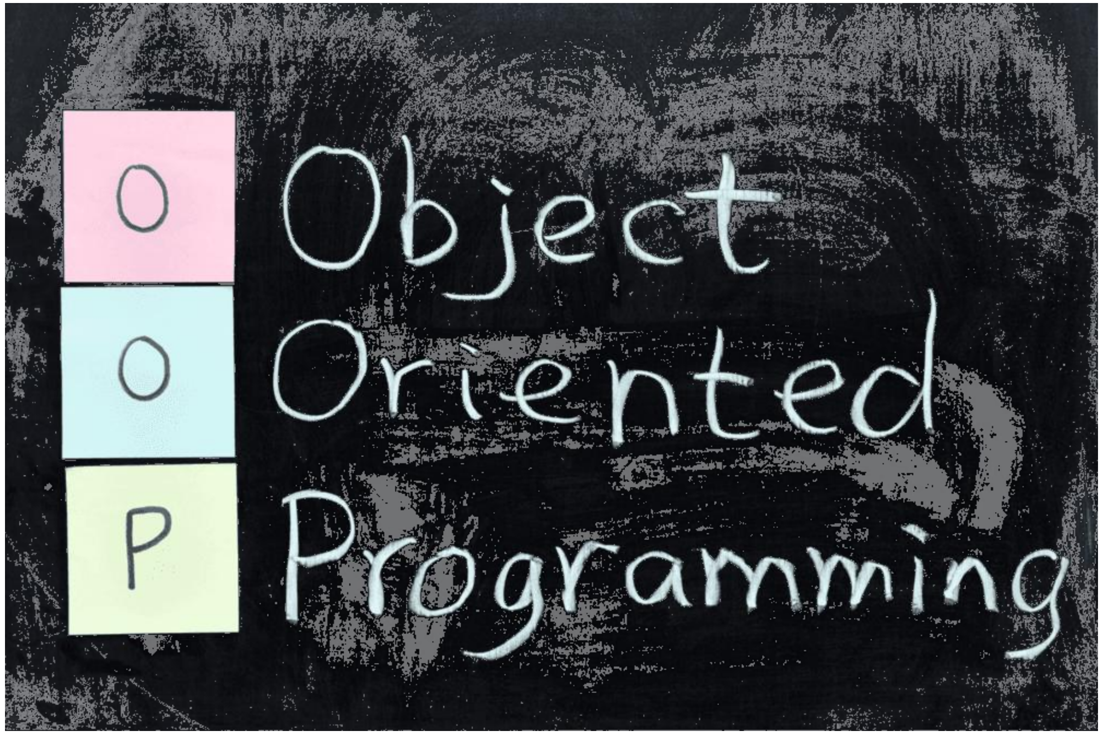


```
begin
  Result:= 0;
  Sum := 0;
  Get (List_Len);
  if (List_Len > 0) and (List_Len < 100) then
-- Read input data into an array and compute the sum
    for Counter := 1 .. List_Len loop
      Get (Int_List(Counter));
      Sum := Sum + Int_List(Counter);
    end loop;
-- Compute the average
    Average := Sum / List_Len;
-- Count the number of values that are > average
    for Counter := 1 .. List_Len loop
      if Int_List(Counter) > Average then
       Result:= Result+ 1;
      end if;
    end loop;
-- Print result
    Put ("The number of values > average is:");
    Put (Result);
    New_Line;
  else
    Put_Line ("Error-input list length is not legal");
  end if;
end Ada_Ex;
```

Object-Oriented Programming: Smalltalk

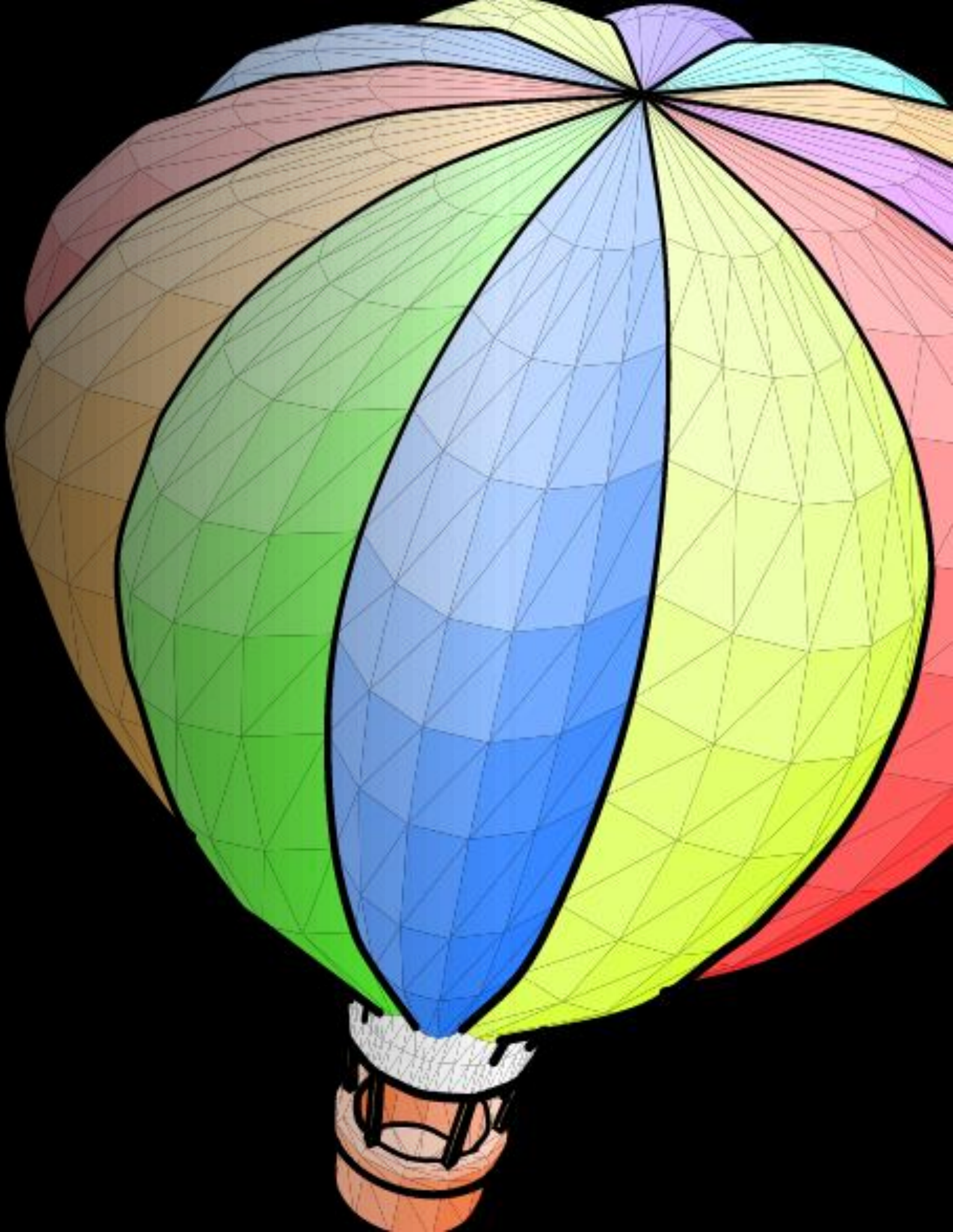
Object-Oriented Programming

- The latest step in the evolution of data-oriented software development, which began in the early 1980s, is object-oriented design.
- Object-oriented methodology begins with data abstraction, which encapsulates processing with data objects and controls access to data and adds inheritance and dynamic method binding.
- Inheritance is a powerful concept that greatly enhances the potential reuse of existing software, thereby providing the possibility of significant increases in software development productivity. This is an important factor in the increase in popularity of object-oriented languages.
- Dynamic (run-time) method binding allows more flexible use of inheritance.
- Smalltalk was the first programming language that fully supported object-oriented programming. It is therefore an important part of any discussion of the evolution of programming languages.

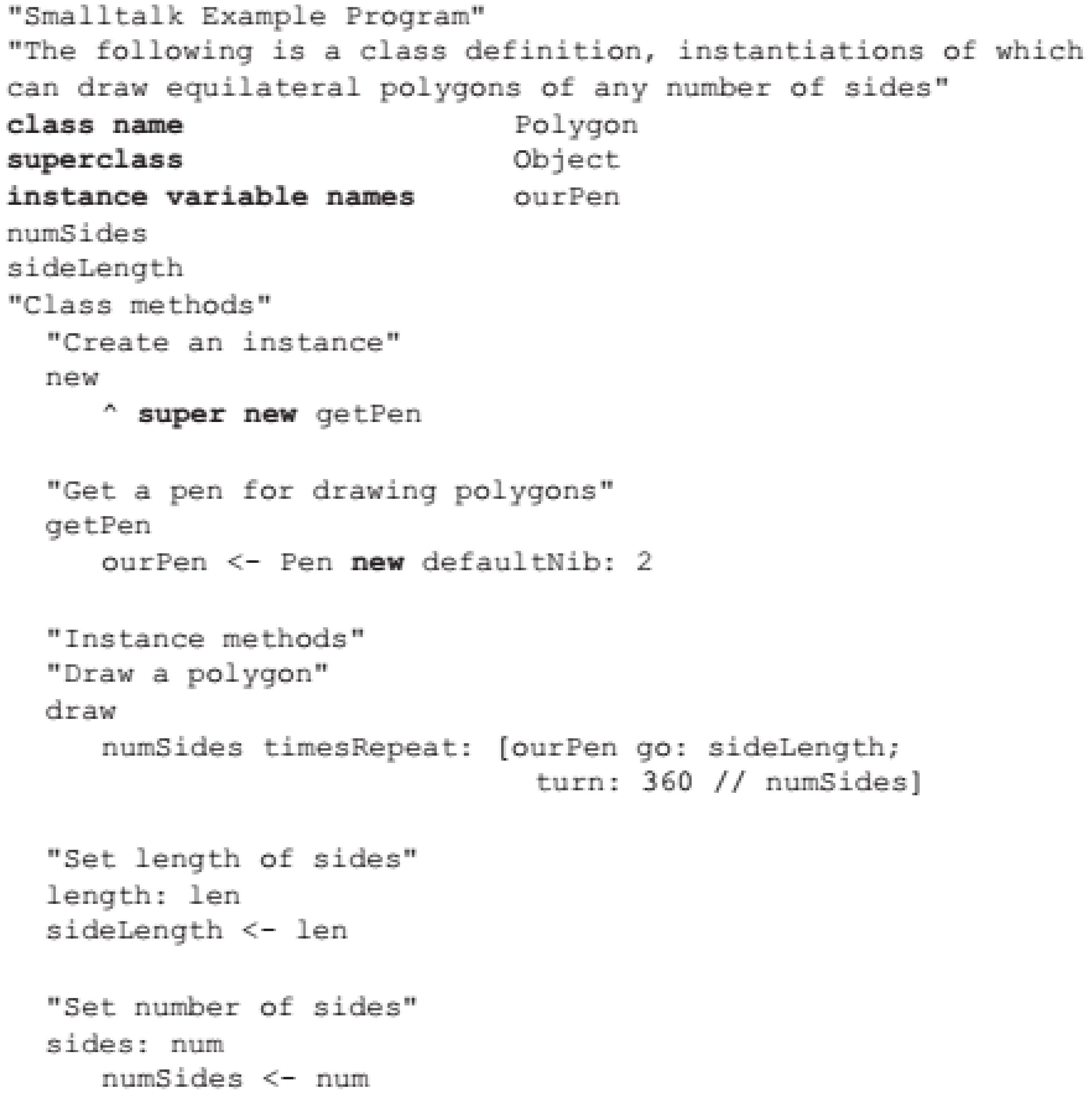


Smalltalk

- Developed at Xerox PARC, initially by Alan Kay, later by Adele Goldberg (1980).
- The pioneer of this language is SIMULA 67.
- First full implementation of an object-oriented language (“data abstraction”, “inheritance”, and “dynamic binding”)



An Example of a Smalltalk Class Definition



```
"Smalltalk Example Program"
"The following is a class definition, instantiations of which
can draw equilateral polygons of any number of sides"
class name                    Polygon
superclass                    Object
instance variable names       ourPen
numSides
sideLength
"Class methods"
  "Create an instance"
  new
     ^ super new getPen

  "Get a pen for drawing polygons"
  getPen
     ourPen <- Pen new defaultNib: 2

  "Instance methods"
  "Draw a polygon"
  draw
     numSides timesRepeat: [ourPen go: sideLength;
                             turn: 360 // numSides]

  "Set length of sides"
  length: len
  sideLength <- len

  "Set number of sides"
  sides: num
     numSides <- num
```

Combining Imperative and Object-Oriented Features: C++

C++

- Developed at Bell Labs by Bjarne Stroustrup in 1980.
- Evolved from C and SIMULA 67.
- A large and complex language, in part because it supports both procedural and OO programming.
- Rapidly grew in popularity, along with OOP.
- Operators in C++ can be overloaded, meaning the user can create operators for existing operators on user-defined types.
- C++ supports multiple inheritance (Java doesn't).
- It is a less secure language than Java (But more flexible).

A Related Language: Objective-C

- Objective-C is another hybrid language with both imperative and object-oriented features (using Smalltalk syntax for that support for OOP).
- Objective-C was designed by Brad Cox and Tom Love in the early 1980s.
- After Steve Jobs left Apple and founded NeXT, he licensed Objective-C and it was used to write the NeXT computer system software. NeXT also released its Objective-C compiler, along with the “NeXTstep development environment” and a library of utilities.
- After the NeXT project failed, Apple bought NeXT and used Objective-C to write MAC OS X.
- Objective-C is the language of all iPhone software, which explains its rapid rise in popularity after the iPhone appeared.

Eiffel

- Eiffel is a hybrid language (imperative and object-oriented) too which designed by Bertrand Meyer (The founder of Eiffel Software).
- Eiffel supports multiple inheritance, genericity, polymorphism, encapsulation and type-safe conversions.
- It is smaller and simpler than C++ but it is a highly expressive and writable language.

Delphi

- In Delphi programming language, imperative and object-oriented programming paradigms have been successfully combined.
- Delphi was originally developed by Borland as a “Rapid Application Development (RAD)” tool for Windows as the successor of Turbo Pascal.
- With the effect of being a Pascal (Object Pascal) derived language, it is a more reliable language than C and C++ in controlling array elements, pointer arithmetic and type issues.

An Imperative-Based Object-Oriented Language: Java

An Imperative-Based Object-Oriented Language: Java

- Java’s designers started with C++, removed some constructs, changed some, and added a few others. The resulting language provides much of the power and flexibility of C++, but in a smaller, simpler, and safer language. Since that initial design, Java has grown considerably.
- In 1990, Sun Microsystems determined there was a need for a programming language for embedded consumer electronic devices, such as toasters, microwave ovens, and interactive TV systems. Reliability was one of the primary goals for such a language.
- After considering C and C++, it was decided that neither would be satisfactory for developing software for consumer electronic devices.

Java

- Although C was relatively small, it did not provide support for object-oriented programming, which they deemed a necessity. C++ supported object-oriented programming, but it was judged to be too large and complex.
- It was also believed that neither C nor C++ provided the necessary level of reliability.
- So, a new language, later named Java, was designed.
- Its design was guided by the fundamental goal of providing greater simplicity and reliability than C++ was believed to provide.

*** The name Java comes from “Java coffee”, a type of coffee from Indonesia.

Java

The Java design team was headed by James Gosling, who had previously designed the UNIX emacs editor and the NeWS windowing system.

Java does not have pointers, but its reference types provide some of the capabilities of pointers. These references are used to point to class instances.

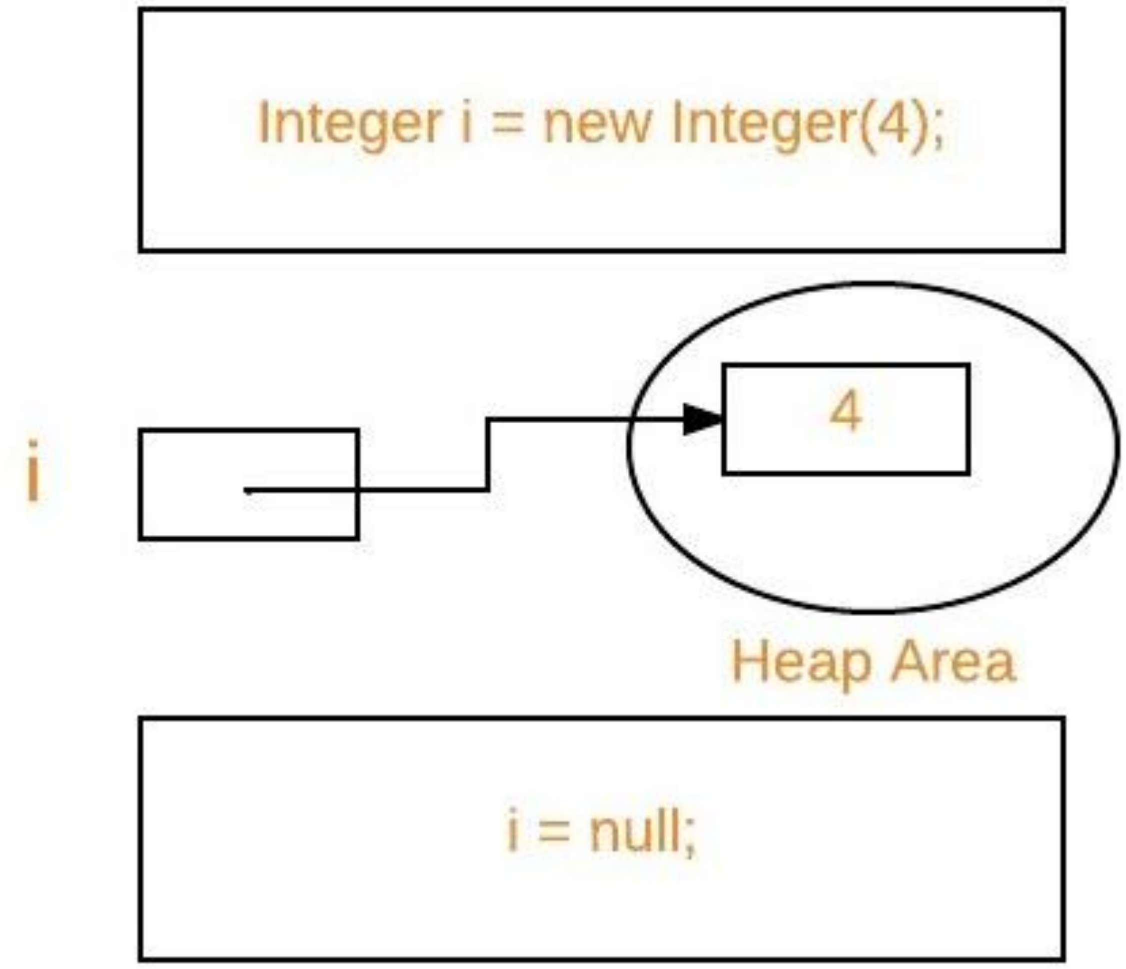
Significantly simplified (does not include struct, union, enum, pointer arithmetic)

Supports only OOP (it is not possible to write stand-alone subprograms in Java. All Java subprograms are methods and are defined in classes. Furthermore, methods can be called through a class or object only)

Portable: Java Virtual Machine concept.

Java

- Java **garbage collection** is the process by which Java programs perform automatic memory management.
- When Java programs run on the JVM, objects are created on the "heap", which is a portion of memory dedicated to the program.
- Eventually, some objects will no longer be needed.
- "The garbage collector" finds these unused objects and deletes them to free up memory.



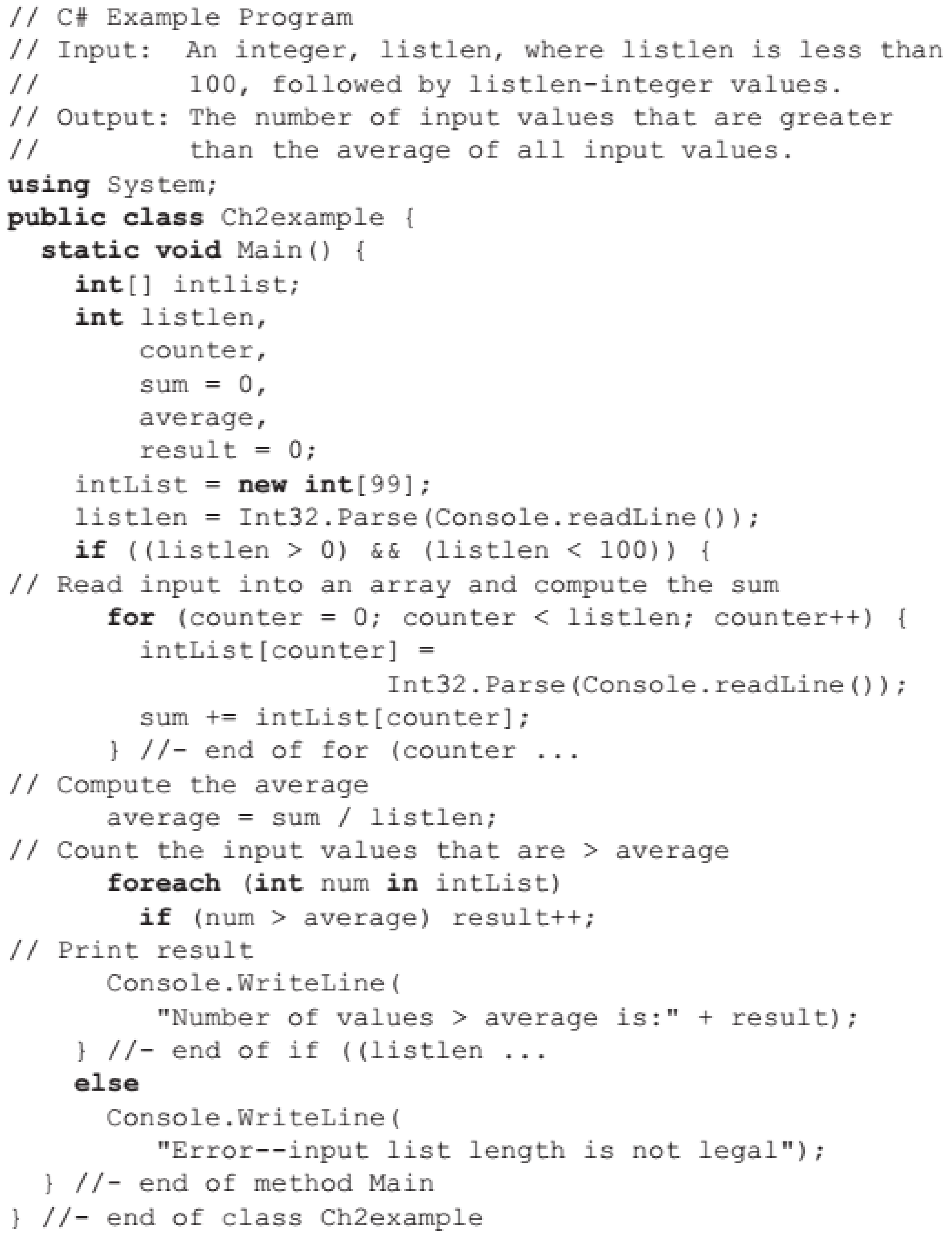
The Flagship .NET Language: C#

- C#, along with the new development platform .NET, was announced by Microsoft in 2000.
- C# is based on C++ and Java but includes some ideas from Delphi and Visual Basic.
- The purpose of C# is to provide a language for “component-based” software development, specifically for such development in the .NET Framework.
- In this environment, components from a variety of languages can be easily combined to form systems.

C#

- All of the .NET languages, which include C#, VB.NET, Managed C++ /C++/CLI, F#, and JScript .NET, use the “common type system” (CTS). The CTS provides a common class library.
- All types in all five .NET languages inherit from a single class root, “System.Object”.
- Compilers that conform to the CTS specification create objects that can be combined into software systems.
- All .NET languages are compiled into the same intermediate form, Intermediate Language (IL).
- Unlike Java, however, the IL is never interpreted. A “Just-in-Time compiler” is used to translate IL into machine code before it is executed.

An Example C# Program

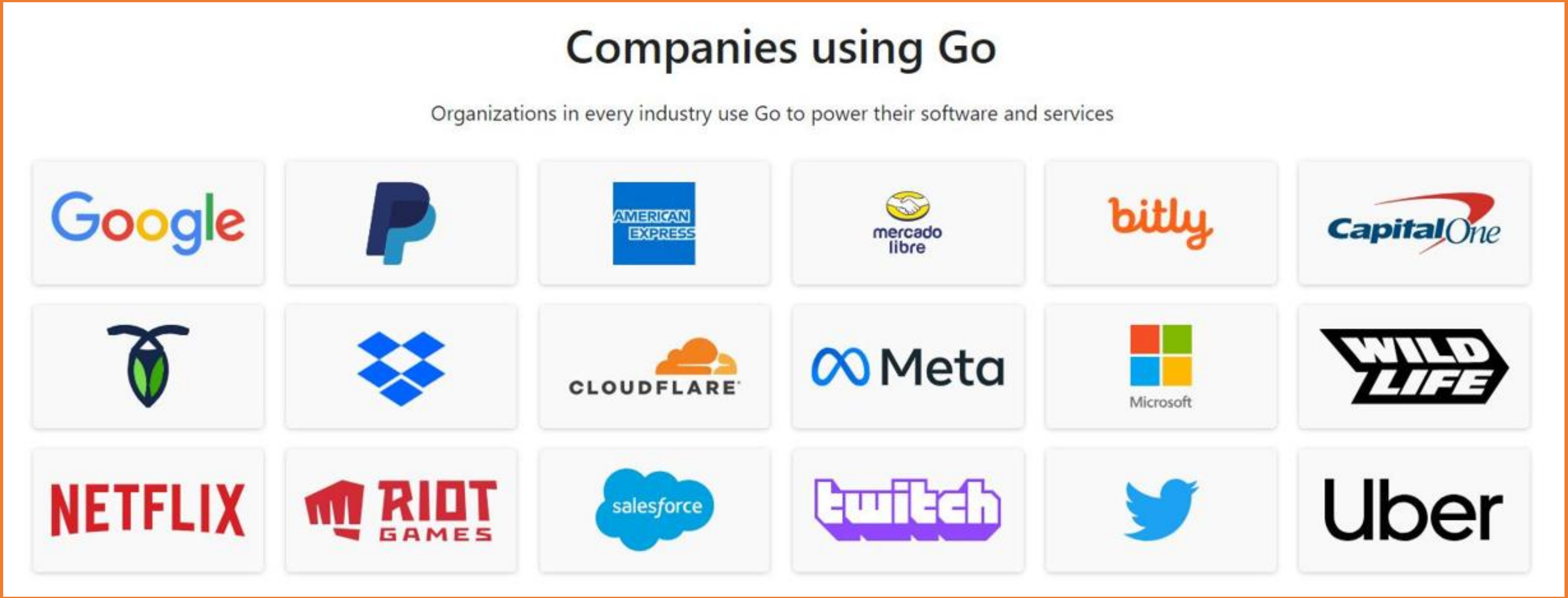


```
// C# Example Program
// Input:  An integer, listlen, where listlen is less than
//         100, followed by listlen-integer values.
// Output: The number of input values that are greater
//         than the average of all input values.
using System;
public class Ch2example {
  static void Main() {
    int[] intlist;
    int listlen,
        counter,
        sum = 0,
        average,
        result = 0;
    intList = new int[99];
    listlen = Int32.Parse(Console.readLine());
    if ((listlen > 0) && (listlen < 100)) {
// Read input into an array and compute the sum
      for (counter = 0; counter < listlen; counter++) {
        intList[counter] =
                       Int32.Parse(Console.readLine());
        sum += intList[counter];
      } //- end of for (counter ...
// Compute the average
      average = sum / listlen;
// Count the input values that are > average
      foreach (int num in intList)
        if (num > average) result++;
// Print result
      Console.WriteLine(
         "Number of values > average is:" + result);
    } //- end of if ((listlen ...
    else
      Console.WriteLine(
         "Error--input list length is not legal");
  } //- end of method Main
} //- end of class Ch2example
```

An Open-source Programming Language Designed at Google: Go

An Open-source Programming Language Designed at Google: Go

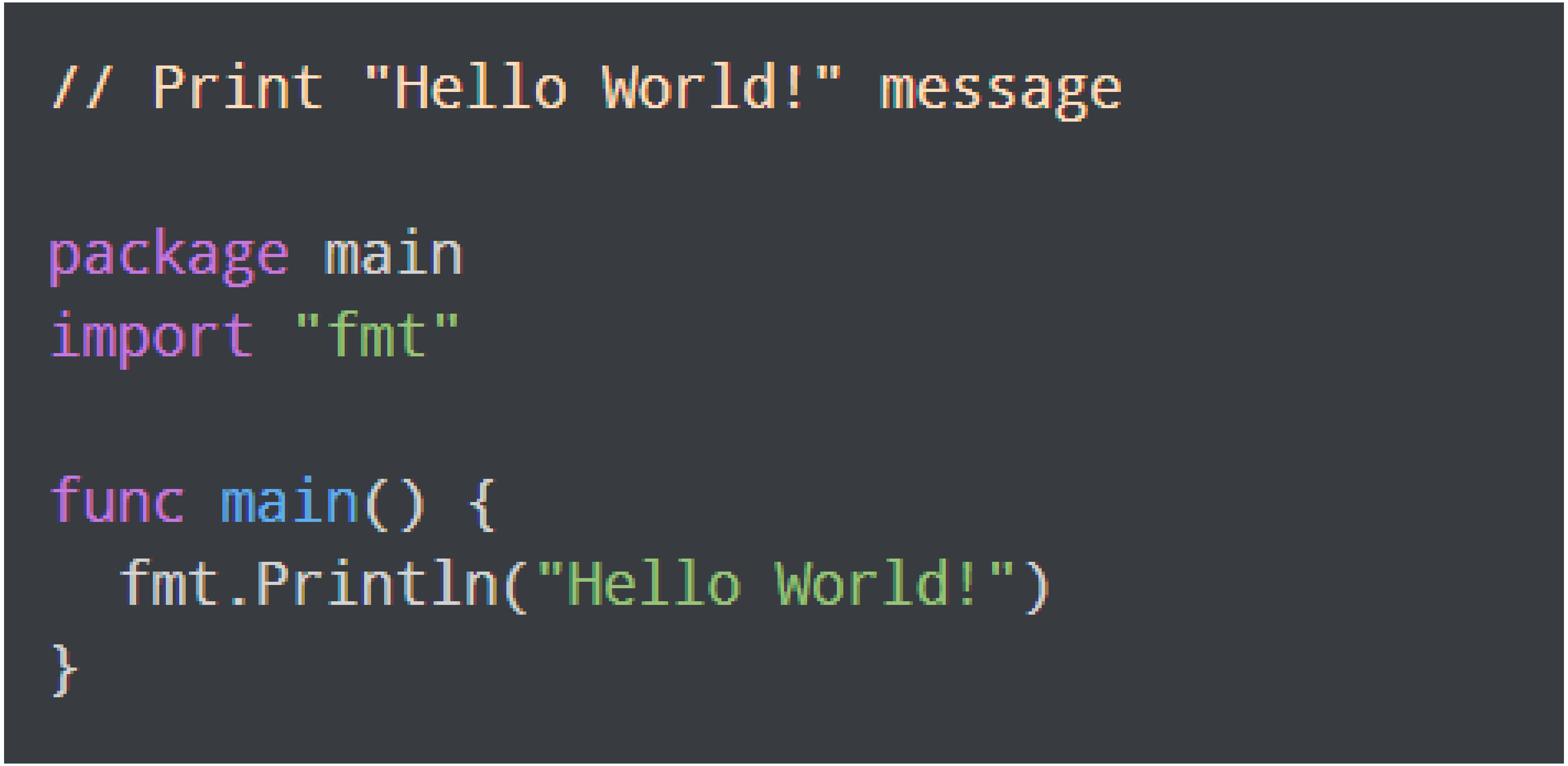
- Go is a statically typed, compiled high-level programming language designed at Google by Robert Griesemer, Rob Pike, and Ken Thompson.
- It is syntactically similar to C, but also has memory safety, garbage collection.
- It is often referred to as “Golang” because of its former domain name, “golang.org” (currently “go.dev”), but its proper name is “Go”.



Companies using Go

Organizations in every industry use Go to power their software and services

An Example Go Program



```go
// Print "Hello World!" message

package main
import "fmt"

func main() {
  fmt.Println("Hello World!")
}
```

Scripting Languages

Scripting Languages

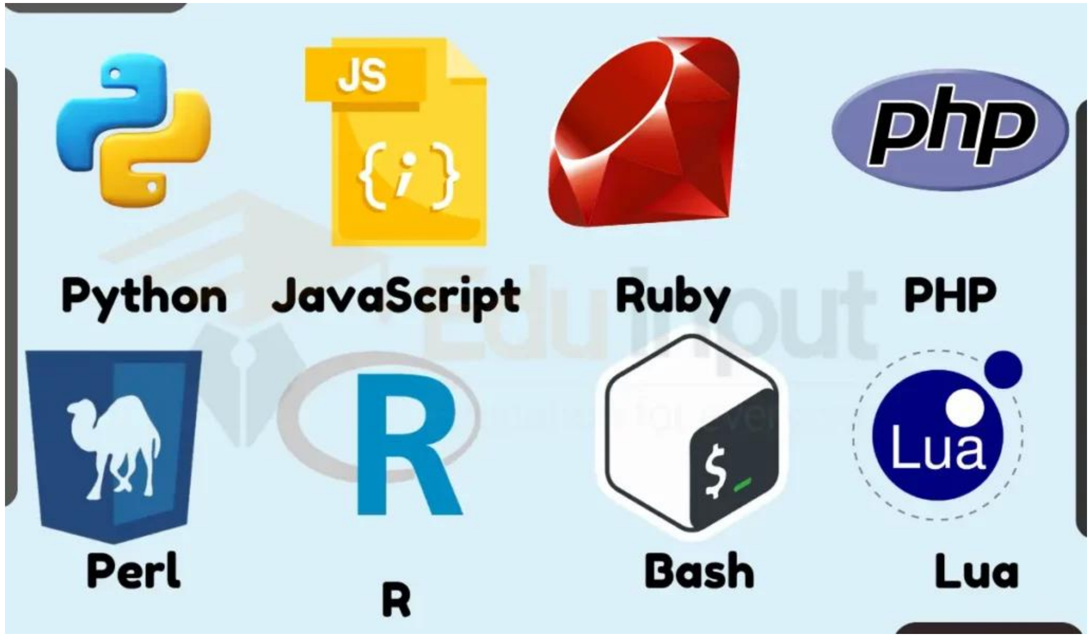
- A **scripting language** is a type of programming language in which the instructions are interpreted individually at runtime.
- With more conventional programming languages, such as C and C++, the code is compiled in advance and in its entirety.
- Scripting languages take a much simpler approach from this perspective.
- However, they're not as efficient as conventional languages.

Scripting Languages

- Early scripting languages were used by putting a list of commands, called a **script**, in a file to be interpreted.
- The first of these languages, named “**sh**” (for shell), began as a small collection of commands that were interpreted as calls to system subprograms that performed utility functions, such as file management and simple file filtering.
- To this were added variables, control flow statements, functions, and various other capabilities, and the result is a complete programming language.
- One of the most powerful and widely known of these is “**ksh**” which was developed by David Korn at Bell Laboratories.
- Another scripting language is “**awk**”, developed by Al Aho, Brian Kernighan, and Peter Weinberger at Bell Laboratories. “awk” began as a report-generation language but later became a more general-purpose language.

Some Important Scripting Languages

- Bash
- Perl
- JavaScript
- PHP
- Phyton
- Ruby
- Lua
- R



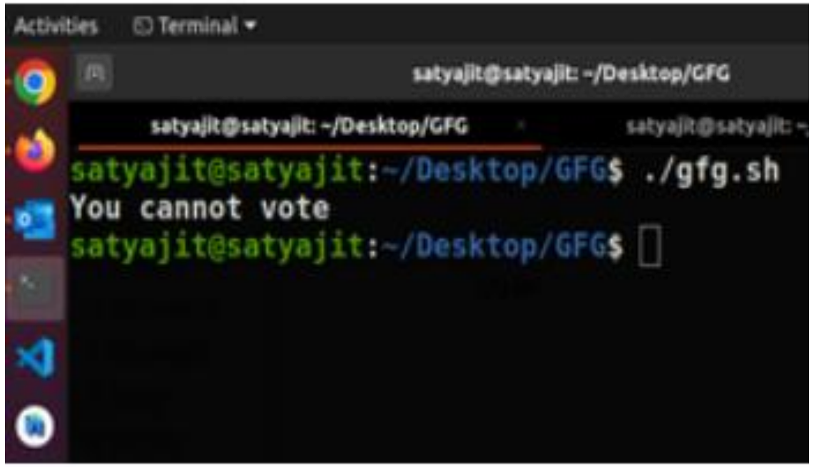
Bash

- **Bash** (**B**ourne **A**gain **Sh**ell) is a command-line interpreter or Unix Shell and it is widely used in GNU/Linux Operating System.
- It is written by Brian Jhan Fox for the GNU Project as a free software replacement for the “Bourne shell”.
- It is used as a default login shell for most Linux distributions.
- Scripting is used to automate the execution of the tasks so that humans do not need to perform them individually.
- Bash scripting is a great way to automate different types of tasks in a system. Developers can avoid doing repetitive tasks using bash scripting.
- Bash scripting supports variables, conditional statements, and loops just like programming languages.

An Example Bash Script

```
#!/bin/bash
Age=17
if [ "$Age" -ge 18 ]; then
    echo "You can vote"
else
    echo "You cannot vote"
fi
```

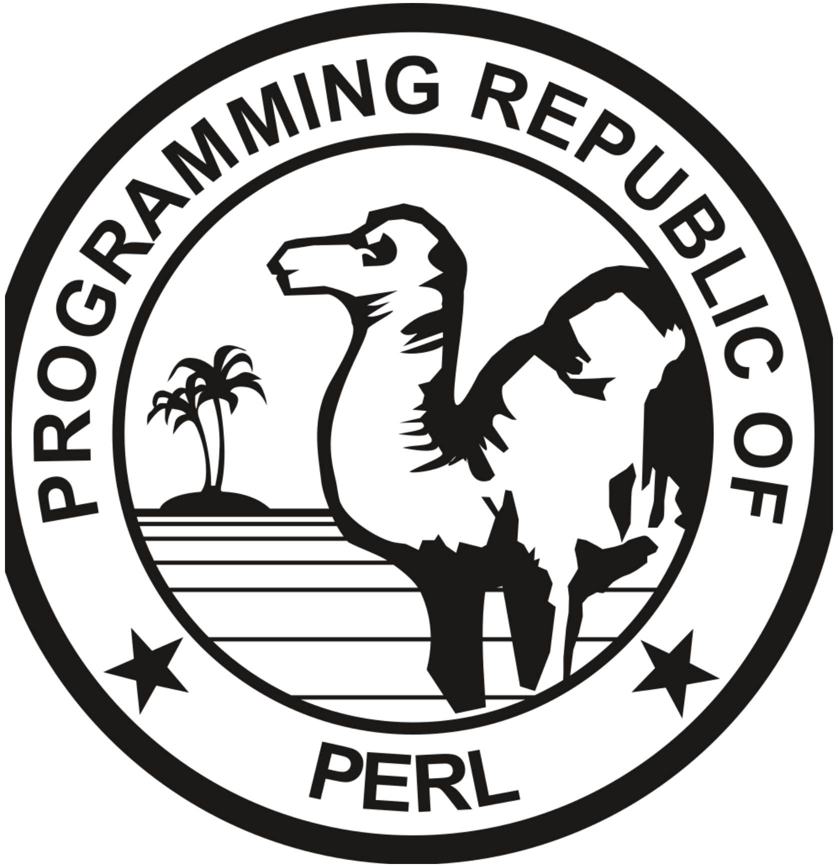
Here is the terminal shell pictorial depiction after executing the above script as follows:


Activities
Terminal
satyajit@satyajit: ~/Desktop/GFG
satyajit@satyajit: ~/Desktop/GFG
satyajit@satyajit: ~

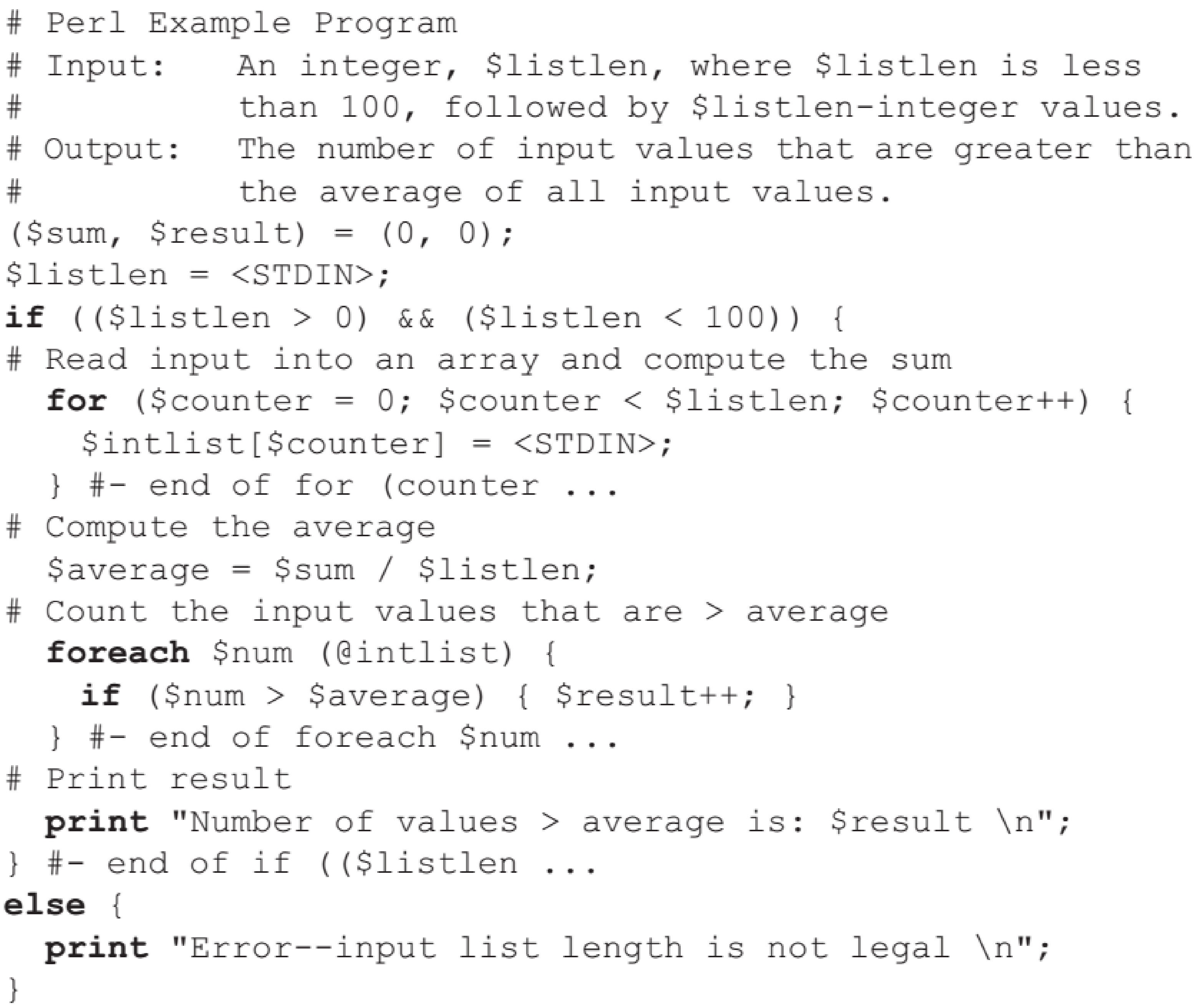
```
satyajit@satyajit:~/Desktop/GFG$ ./gfg.sh
You cannot vote
satyajit@satyajit:~/Desktop/GFG$ 
```

Perl

- The Perl language, developed by Larry Wall (1987), was originally a combination of “sh” and “awk”.
- Perl’s initial use was as a UNIX utility for processing text files. It was and still is widely used as a UNIX system administration tool.
- When the World Wide Web appeared, Perl achieved widespread use as a “common gateway interface (CGI)” language for use with the Web, although it is now rarely used for that purpose.
- Although it is still often called a scripting language, it is actually more similar to a typical imperative language, since it is always compiled, at least into an intermediate language, before it is executed.
- Furthermore, it has all the constructs to make it applicable to a wide variety of areas of computational problems (such as “computational biology” and “artificial intelligence”).



An Example Perl Program



```
# Perl Example Program
# Input:   An integer, $listlen, where $listlen is less
#          than 100, followed by $listlen-integer values.
# Output:  The number of input values that are greater than
#          the average of all input values.
($sum, $result) = (0, 0);
$listlen = <STDIN>;
if (($listlen > 0) && ($listlen < 100)) {
# Read input into an array and compute the sum
  for ($counter = 0; $counter < $listlen; $counter++) {
    $intlist[$counter] = <STDIN>;
  } #- end of for (counter ...
# Compute the average
  $average = $sum / $listlen;
# Count the input values that are > average
  foreach $num (@intlist) {
    if ($num > $average) { $result++; }
  } #- end of foreach $num ...
# Print result
  print "Number of values > average is: $result \n";
} #- end of if (($listlen ...
else {
  print "Error--input list length is not legal \n";
}
```

JavaScript

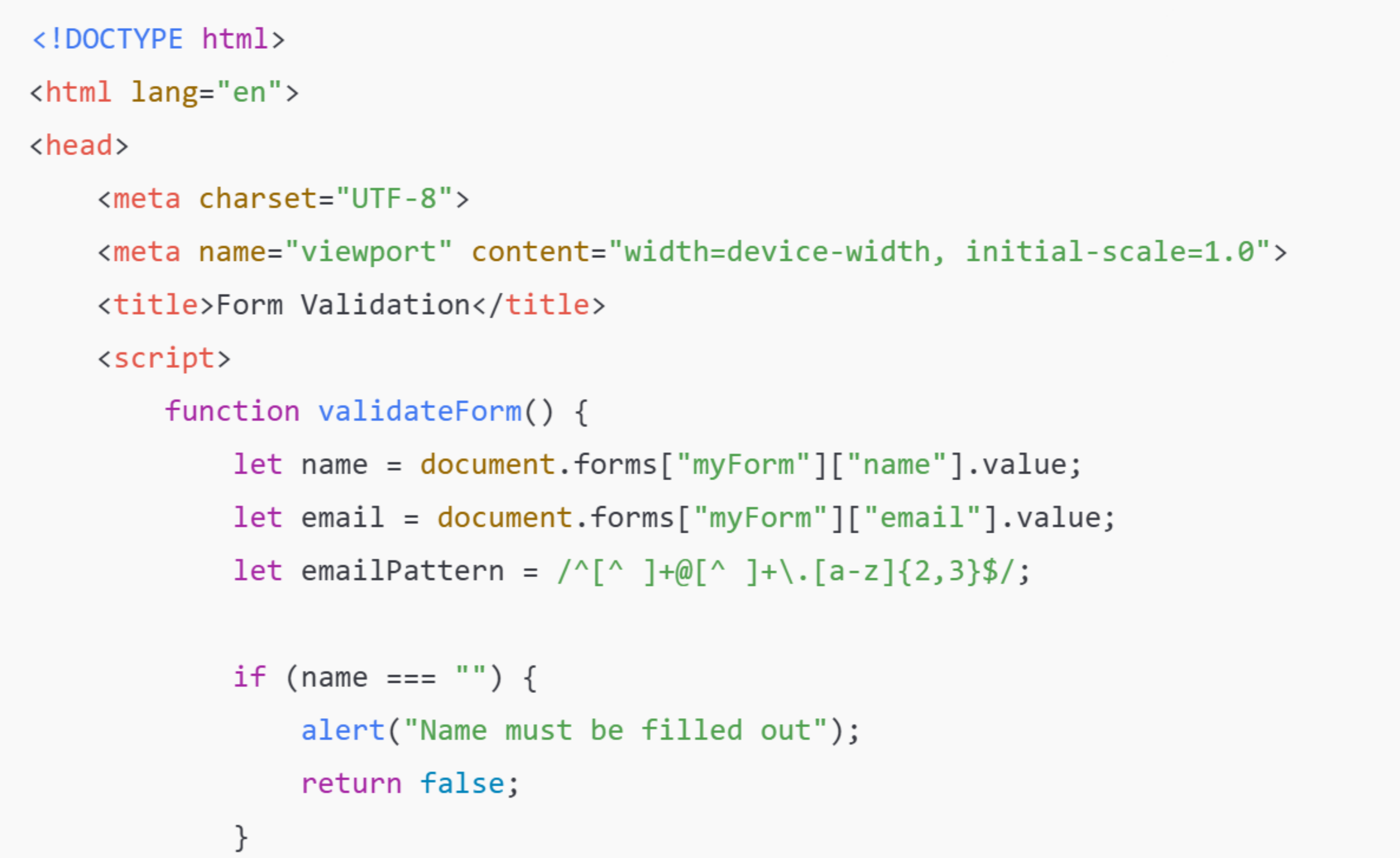
- Use of the Web exploded in the mid-1990s after the first graphical browsers appeared.
- The need for computation associated with HTML documents, which by themselves are completely static, quickly became critical.
- Computation on the server side was made possible with the common gateway interface (CGI), which allowed HTML documents to request the execution of programs on the server, with the results of such computations returned to the browser in the form of HTML documents.
- Computation on the browser end became available with the advent of “Java applets”.

JavaScript

- JavaScript was originally developed by Brendan Eich at Netscape but later became a joint venture of Netscape and Sun Microsystems.
- Its original name was “Mocha”. It was later renamed “LiveScript” and finally JavaScript.
- JavaScript code is embedded in HTML documents and interpreted by the browser when the documents are displayed.
- The primary uses of JavaScript in Web programming are to validate form input data and create dynamic HTML documents.
- So, JavaScript is “a client-side HTML-embedded scripting language”, often used to create dynamic HTML documents.
- Purely interpreted.

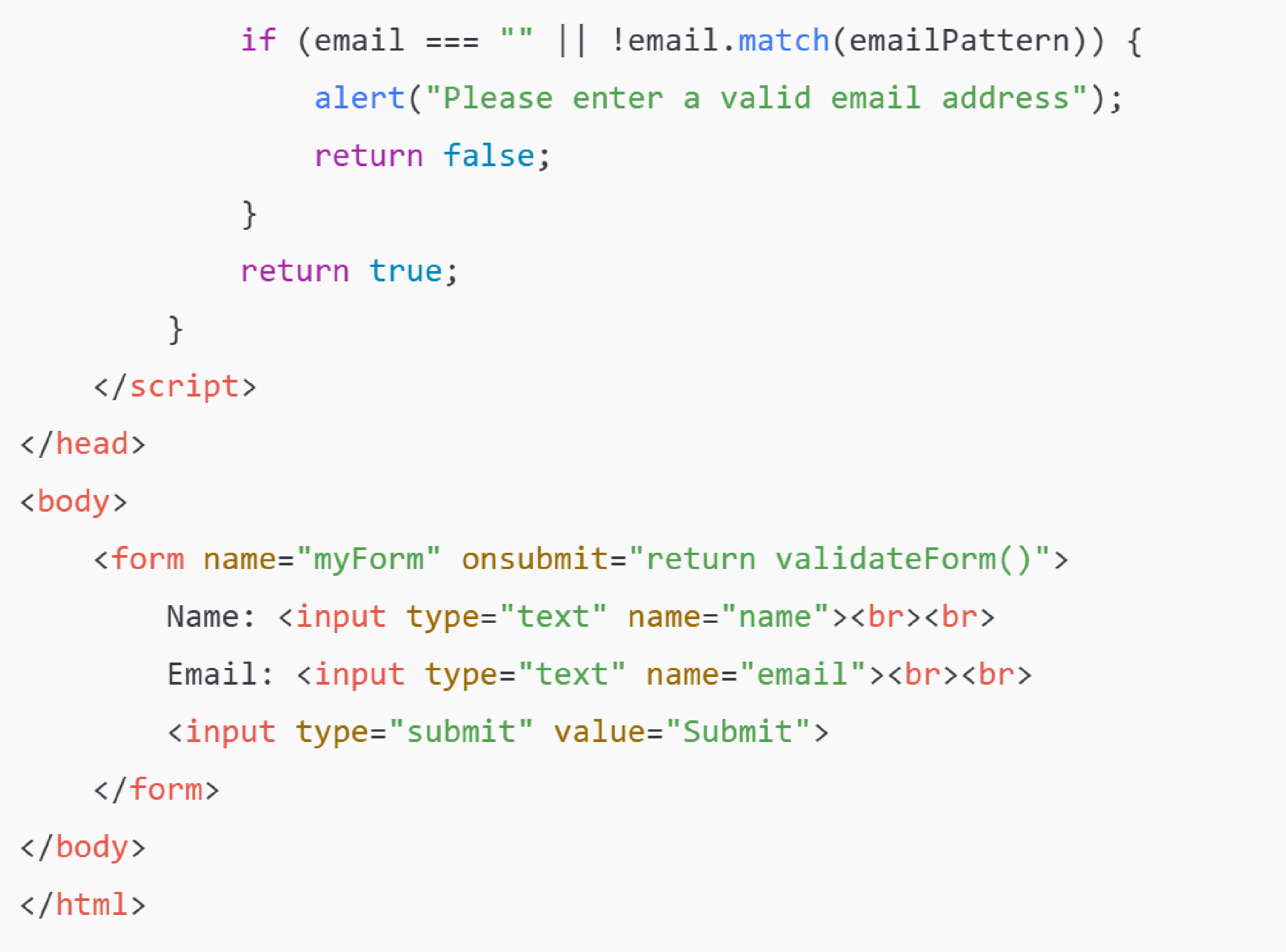
***Related to Java only through similar syntax (not structurally).

An Example JavaScript Program



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Form Validation</title>
    <script>
        function validateForm() {
            let name = document.forms["myForm"]["name"].value;
            let email = document.forms["myForm"]["email"].value;
            let emailPattern = /^[^ ]+@[^ ]+\.[a-z]{2,3}$/;

            if (name === "") {
                alert("Name must be filled out");
                return false;
            }
```

An Example JavaScript Program (continued)



```
            if (email === "" || !email.match(emailPattern)) {
                alert("Please enter a valid email address");
                return false;
            }
            return true;
        }
    </script>
</head>
<body>
    <form name="myForm" onsubmit="return validateForm()">
        Name: <input type="text" name="name"><br><br>
        Email: <input type="text" name="email"><br><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Php

Designed by Rasmus Lerdorf, a member of the Apache Group (1994).

PHP was an abbreviation for “Personal Home Page”. Later, its user community began using the recursive name PHP: Hypertext Preprocessor.

PHP is now developed, distributed, and supported as an open-source product.

It’s “a server-side HTML-embedded scripting language”, often used for form processing and database access through the Web.

It’s purely interpreted.

PHP allows simple access to HTML form data, so form processing is easy with PHP. PHP provides support for many different database management systems. This makes it a useful language for building programs that need Web access to databases.

Python is a relatively recent object-oriented interpreted scripting language.

Its initial design was by Guido van Rossum at Stichting Mathematisch Centrum in the Netherlands in the early 1990s. Its development is now being done by the “Python Software Foundation”.

Python is being used for the same kinds of applications as Perl: system administration, CGI programming, and other relatively small computing tasks.

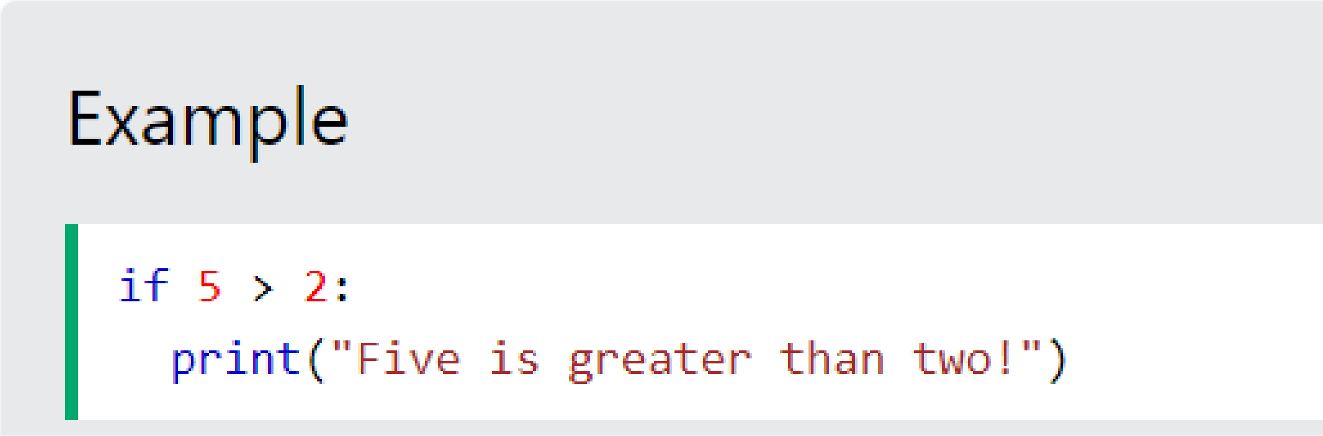
Python is an open-source system and is available for most common computing platforms.

Python

- Python’s syntax is not based directly on any commonly used language.
- Instead of arrays, Python includes “lists”. There is a collection of list methods, such as append, insert, remove, and sort.
- Especially the fact that many libraries used in the field of artificial intelligence are written in Python language makes the language important in this field.
- One of the most interesting features of Python is that it can be easily extended by any user. The modules that support the extensions can be written in any compiled language. Extensions can add functions, variables, and object types. These extensions are implemented as “additions to the Python interpreter”.

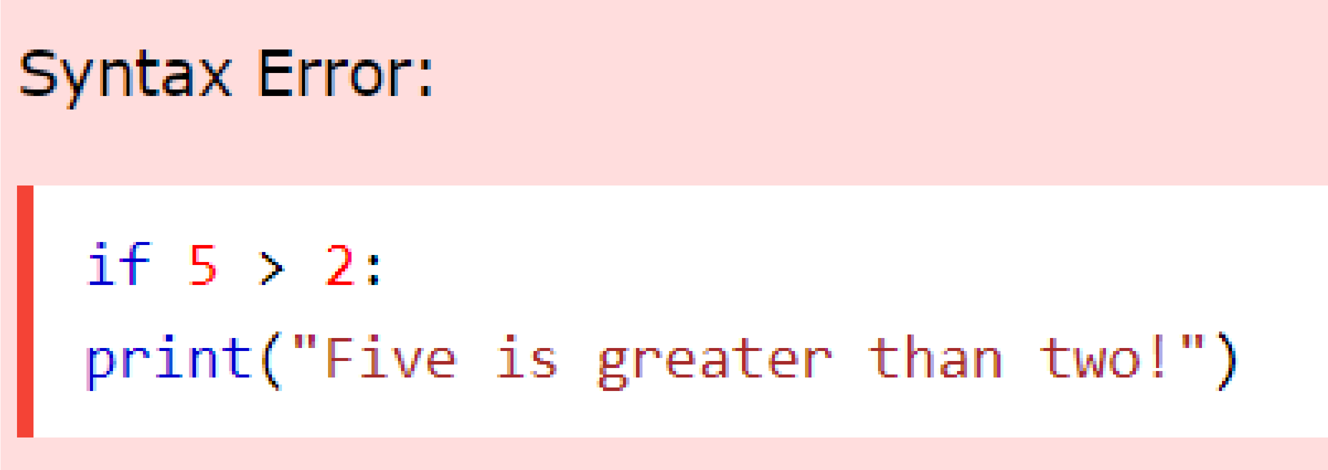
Python Syntax

- “Indentation” refers to the spaces at the beginning of a code line. Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important. Python uses indentation to “indicate a block of code”.
- Python will give you an error if you skip the indentation:



Example

```
if 5 > 2:
  print("Five is greater than two!")
```



Syntax Error:

```
if 5 > 2:
print("Five is greater than two!")
```

Ruby

- Ruby was designed by Yukihiro Matsumoto in the early 1990s and released in 1996.
- Ruby is culturally interesting because it is the first programming language designed in Japan that has achieved relatively widespread use in the United States.
- The primary characterizing feature of Ruby is that it is a pure object-oriented language (unlike Perl and Python). All data are objects.
- Purely interpreted.

Lua

- Lua was designed in the early 1990s by Roberto Lerusalimschy, Waldemar Celes, and Luis Henrique de Figueiredo at the Pontifical University of Rio de Janeiro in Brazil.
- The name Lua is derived from the Portuguese word for “moon”.
- Easily extendable.
- Lua is a relatively small and simple language, having only 21 reserved words. The design philosophy of the language was to provide the bare essentials and relatively simple ways to extend the language to allow it to fit a variety of application areas.

R

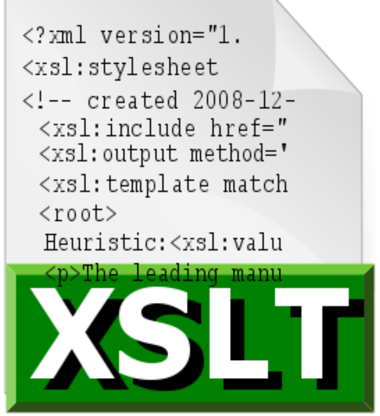
- R is a programming language for statistical computing.
- It has been adopted in the fields of data mining, bioinformatics, and data analysis.
- R software is open-source and free. It is licensed by the GNU Project and available under the GNU General Public License.
- As an interpreted language, R has a native command line interface. Moreover, multiple third-party graphical user interfaces are available, such as “Rstudio”—an integrated development environment.

Markup-Programming Hybrid Languages

Markup-Programming Hybrid Languages

- We've mentioned earlier that a markup language (HTML, XML, etc.) is not considered a programming language.
- However, in recent years, a new category of languages has emerged, the **markup-programming hybrid languages**.
- A markup-programming hybrid language is a markup language in which some of the elements can specify programming actions, such as control flow and computation.
- The following subsections introduce two such hybrid languages, **XSLT** and **JSTL**.

XSLT (e**X**tensible **S**tylesheet **L**anguage Transformations)



- e**X**tensible **M**arkup **L**anguage (XML) is a **metamarkup language**. Such a language is used to define markup languages.
- "XML-derived markup languages" are used to define XML data documents. Although XML documents are human readable, they are processed by computers. This processing sometimes consists only of transformations to alternative forms that can be effectively displayed or printed. In many cases, such transformations are to HTML, which can be displayed by a Web browser.
- The transformation of XML documents to HTML documents is specified in another markup language, **eXtensible stylesheet language transformations (XSLT)**.
- XSLT can specify programming-like operations. Therefore, XSLT is a "markup-programming hybrid language".
- XSLT has "programming constructs". For example, a looping construct is included, which allows repeated parts of the XML document to be selected. There is also a sort process.
- These constructs are specified with XSLT tags, such as `<for-each>`.

JSTL (Java Server Pages Standard Tag Library

- The “core” part of the **JSTL (Java Server Pages Standard Tag Library)** is another markup-programming hybrid language, although its form and purpose are different from those of XSLT.
- *JSTL is a* ***JSP*** **(Java Server Pages)** *library*, includes programming constructs in the form of HTML elements (`<if>`, `<forEach>`, etc.).
- JSP is a collection of technologies designed to support dynamic Web documents and provide other processing needs of Web documents. (A JSP document is often a mixture of “HTML” and “Java”).
- The JSTL defines a collection of XML action elements that control the processing of the JSP document on the Web server.