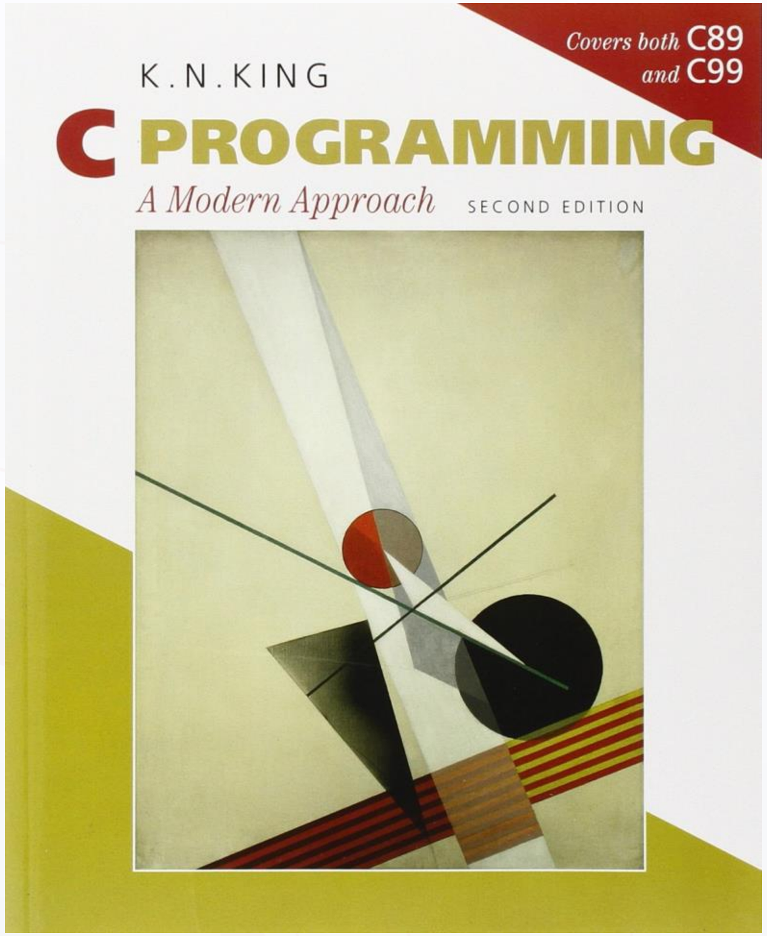
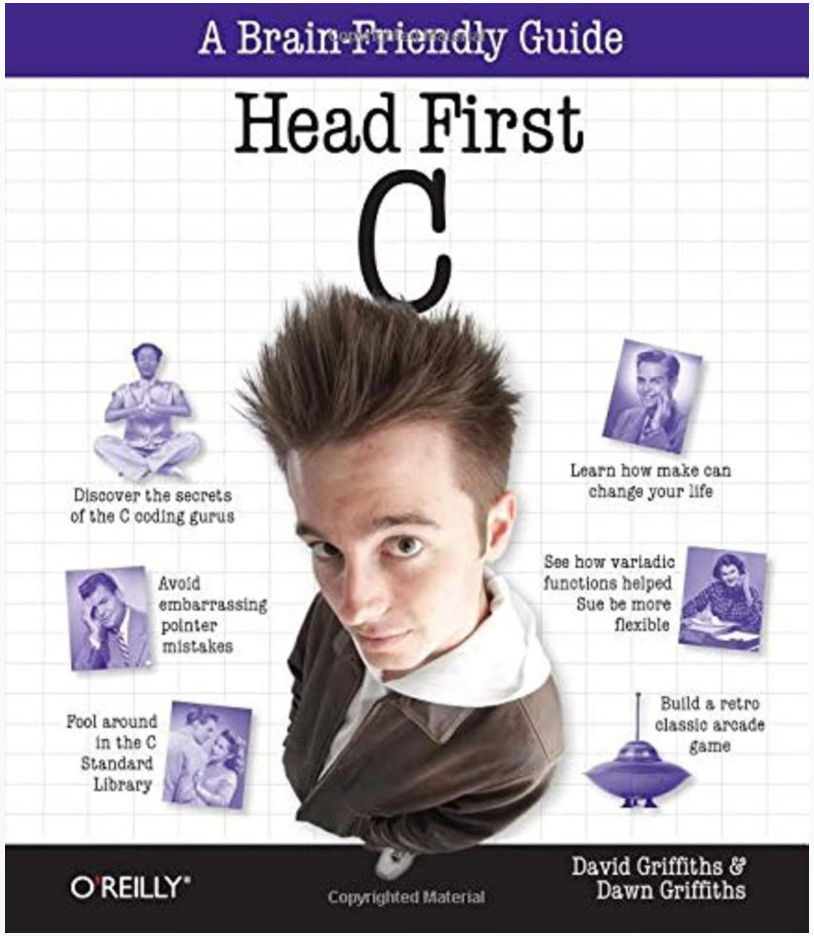
CS 111

Introduction to C Programming

Instructor: Wei Hong **(MAE)**
Lab: Jiawei Huang **(EE)**
TA: TBD

About the course



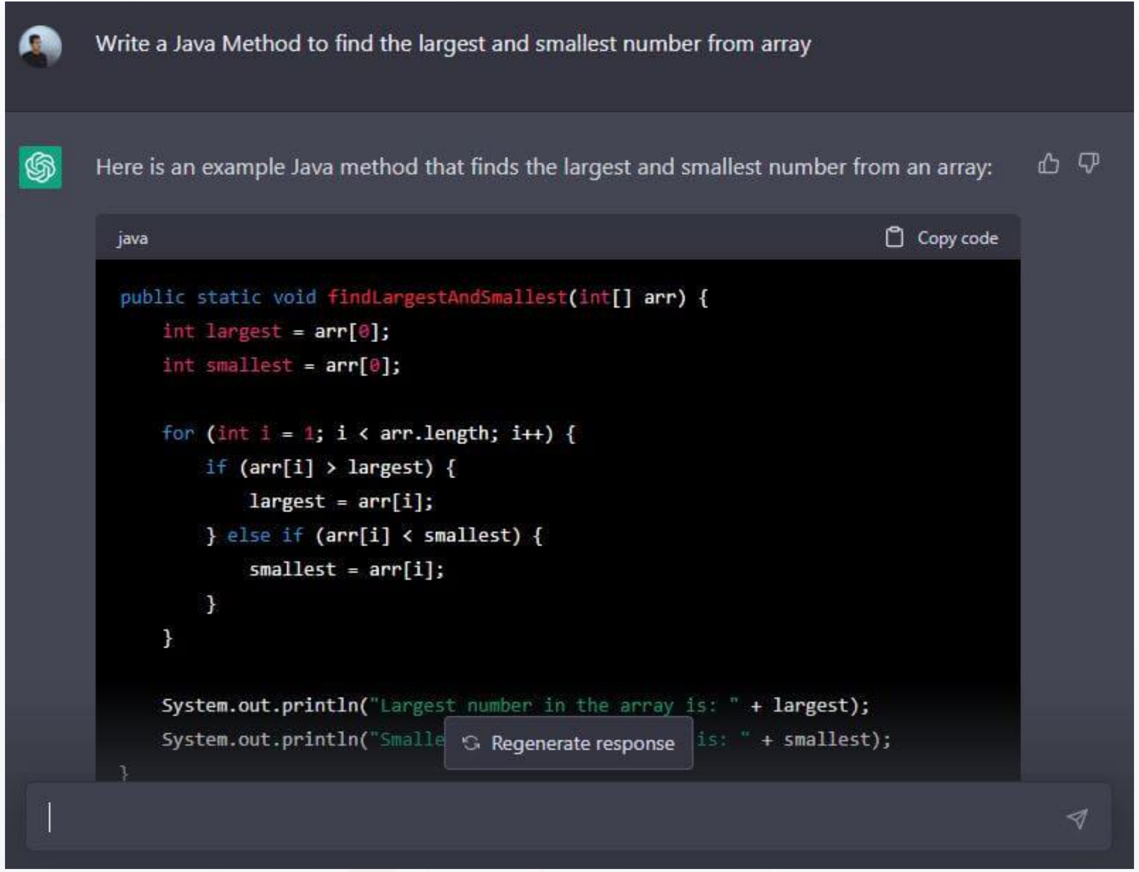

A Brain-Friendly Guide
Head First
C
Discover the secrets of the C coding gurus
Learn how make can change your life
Avoid embarrassing pointer mistakes
See how variadic functions helped Sue be more flexible
Fool around in the C Standard Library
Build a retro classic arcade game
O'REILLY
Copyrighted Material
David Griffiths & Dawn Griffiths

Office Hours:
4:00-5:30 Tue
303 SUSTech Center
(1008 N. Eng. Bldg.)

Credit Distribution:

- In-class performance /quizzes – 10%
- Homework/projects – 50%
- Final Exam – 40%

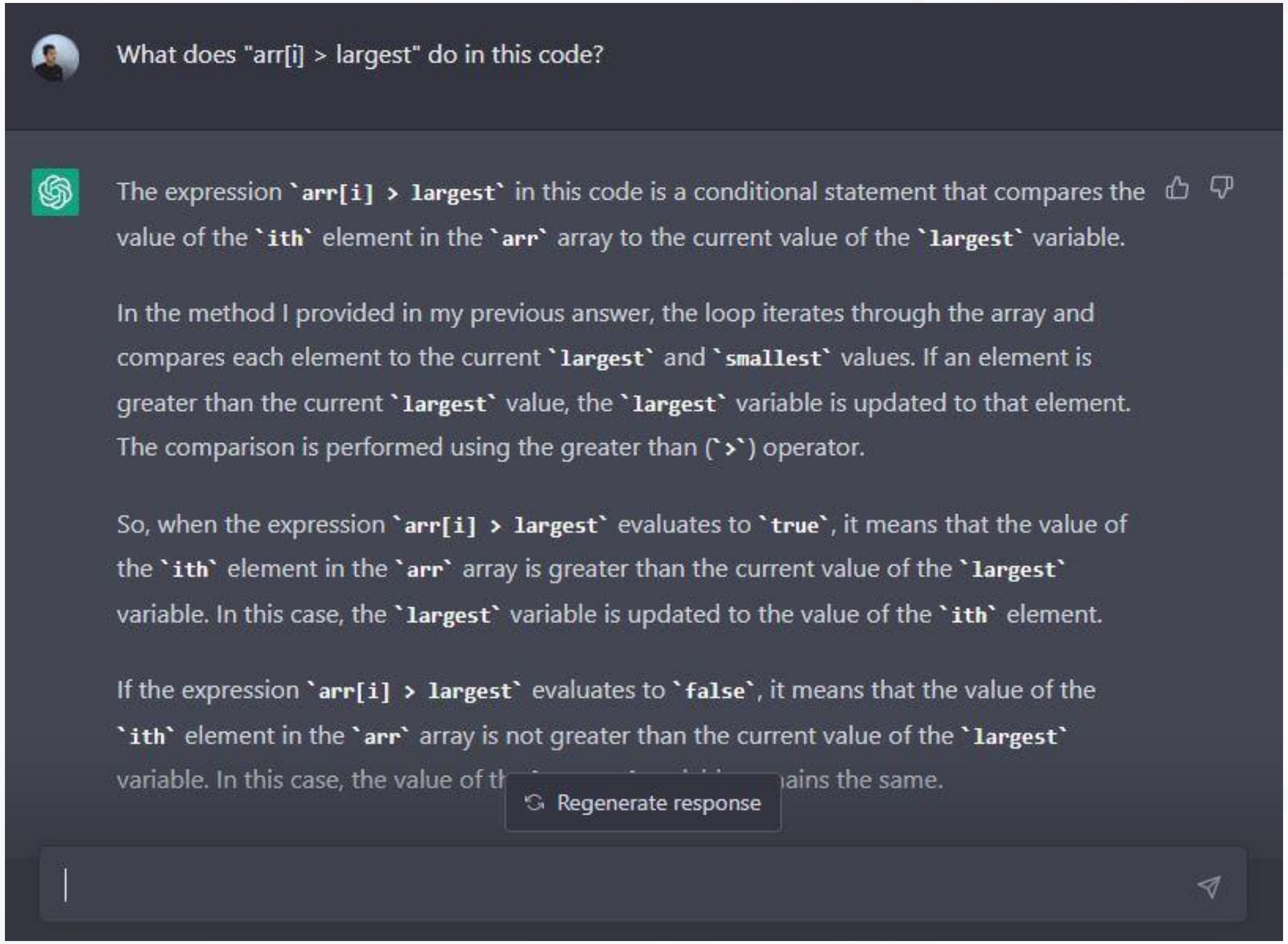
ChatGPT knows coding



Write a Java Method to find the largest and smallest number from array

Here is an example Java method that finds the largest and smallest number from an array:

```java
public static void findLargestAndSmallest(int[] arr) {
    int largest = arr[0];
    int smallest = arr[0];

    for (int i = 1; i < arr.length; i++) {
        if (arr[i] > largest) {
            largest = arr[i];
        } else if (arr[i] < smallest) {
            smallest = arr[i];
        }
    }

    System.out.println("Largest number in the array is: " + largest);
    System.out.println("Smalle                    is: " + smallest);
}
```



What does "arr[i] > largest" do in this code?

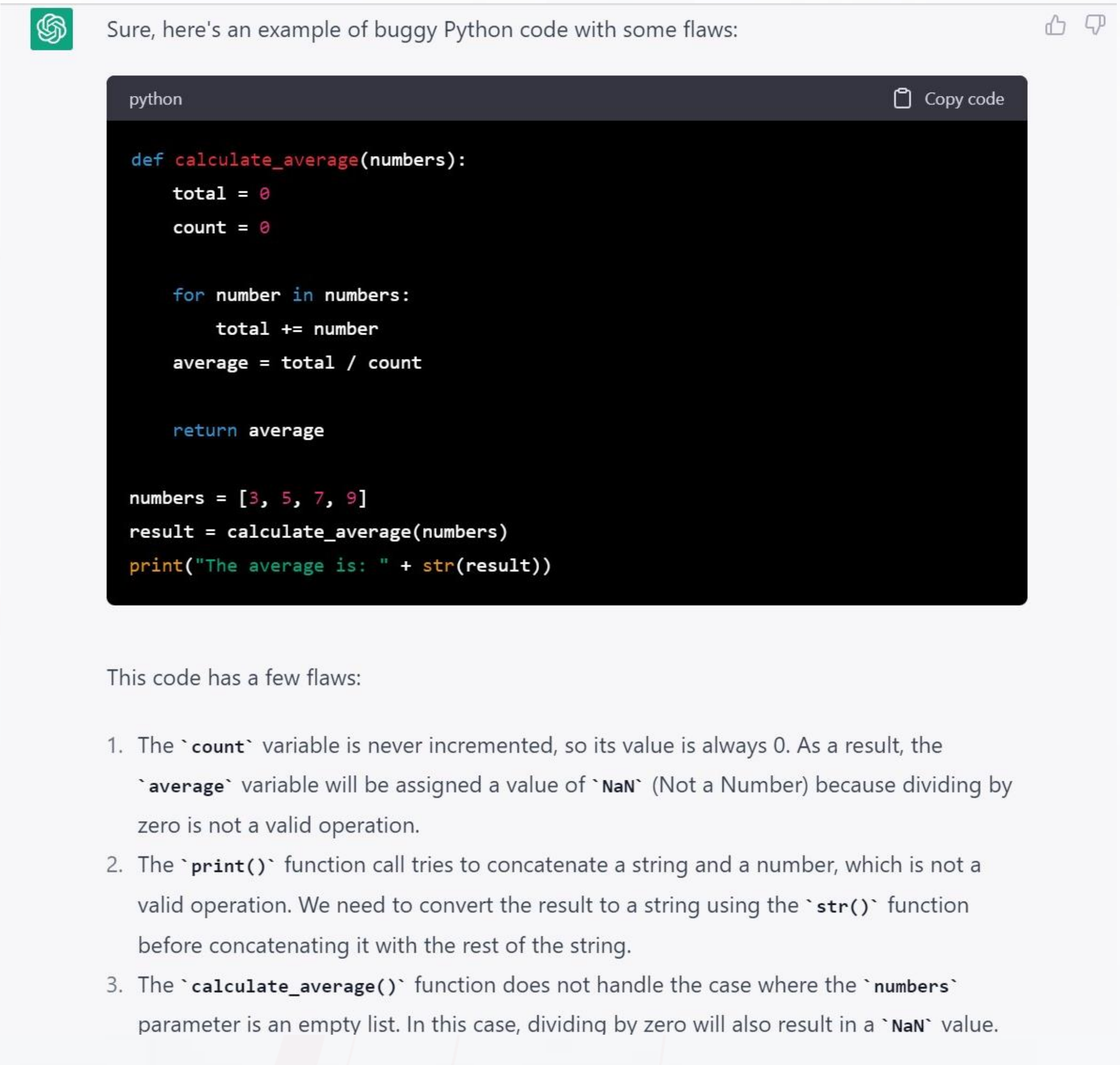
The expression `arr[i] > largest` in this code is a conditional statement that compares the value of the `ith` element in the `arr` array to the current value of the `largest` variable.

In the method I provided in my previous answer, the loop iterates through the array and compares each element to the current `largest` and `smallest` values. If an element is greater than the current `largest` value, the `largest` variable is updated to that element. The comparison is performed using the greater than (`>`) operator.

So, when the expression `arr[i] > largest` evaluates to `true`, it means that the value of the `ith` element in the `arr` array is greater than the current value of the `largest` variable. In this case, the `largest` variable is updated to the value of the `ith` element.

If the expression `arr[i] > largest` evaluates to `false`, it means that the value of the `ith` element in the `arr` array is not greater than the current value of the `largest` variable. In this case, the value of th ains the same.

ChatGPT can debug

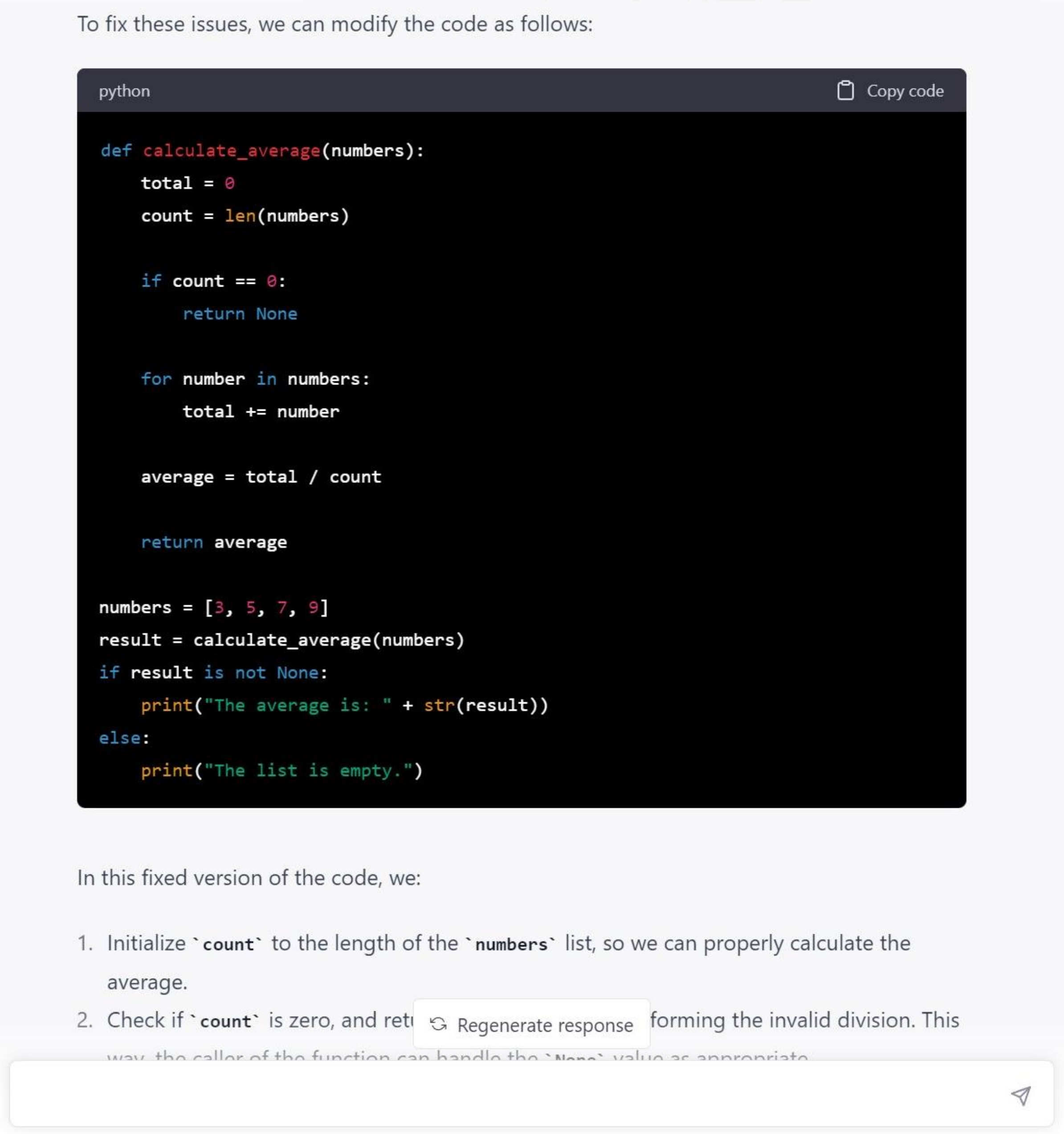


Sure, here's an example of buggy Python code with some flaws:

```python
def calculate_average(numbers):
    total = 0
    count = 0

    for number in numbers:
        total += number
    average = total / count

    return average

numbers = [3, 5, 7, 9]
result = calculate_average(numbers)
print("The average is: " + str(result))
```

This code has a few flaws:

1. The `count` variable is never incremented, so its value is always 0. As a result, the `average` variable will be assigned a value of `NaN` (Not a Number) because dividing by zero is not a valid operation.
2. The `print()` function call tries to concatenate a string and a number, which is not a valid operation. We need to convert the result to a string using the `str()` function before concatenating it with the rest of the string.
3. The `calculate_average()` function does not handle the case where the `numbers` parameter is an empty list. In this case, dividing by zero will also result in a `NaN` value.



To fix these issues, we can modify the code as follows:

```python
def calculate_average(numbers):
    total = 0
    count = len(numbers)

    if count == 0:
        return None

    for number in numbers:
        total += number

    average = total / count

    return average

numbers = [3, 5, 7, 9]
result = calculate_average(numbers)
if result is not None:
    print("The average is: " + str(result))
else:
    print("The list is empty.")
```

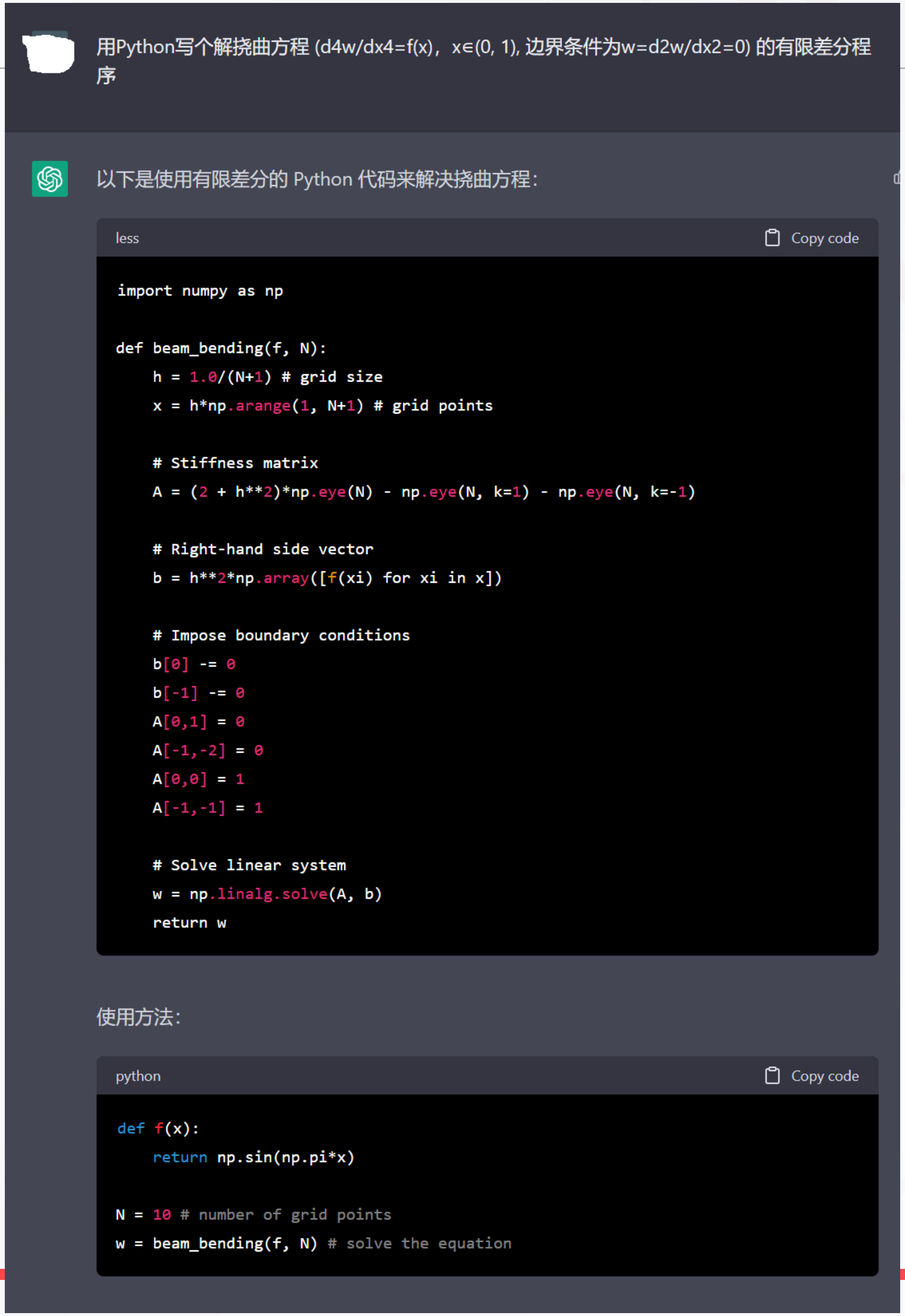
In this fixed version of the code, we:

1. Initialize `count` to the length of the `numbers` list, so we can properly calculate the average.
2. Check if `count` is zero, and retu forming the invalid division. This way, the caller of the function can handle the `None` value as appropriate.

Then why do we have to learn programming?

ChatGPT makes mistakes

which are usually hard to tell



用Python写个解挠曲方程 (d4w/dx4=f(x)，x∈(0, 1), 边界条件为w=d2w/dx2=0) 的有限差分程序

以下是使用有限差分的 Python 代码来解决挠曲方程：

```
import numpy as np

def beam_bending(f, N):
    h = 1.0/(N+1) # grid size
    x = h*np.arange(1, N+1) # grid points

    # Stiffness matrix
    A = (2 + h**2)*np.eye(N) - np.eye(N, k=1) - np.eye(N, k=-1)

    # Right-hand side vector
    b = h**2*np.array([f(xi) for xi in x])

    # Impose boundary conditions
    b[0] -= 0
    b[-1] -= 0
    A[0,1] = 0
    A[-1,-2] = 0
    A[0,0] = 1
    A[-1,-1] = 1

    # Solve linear system
    w = np.linalg.solve(A, b)
    return w
```

使用方法：

```python
def f(x):
    return np.sin(np.pi*x)

N = 10 # number of grid points
w = beam_bending(f, N) # solve the equation
```

How to master programming?

Practice

Programing languages are also languages

Examples of grammatically correct but senseless sentences

Colorless green ideas sleep furiously.

John is a married bachelor.

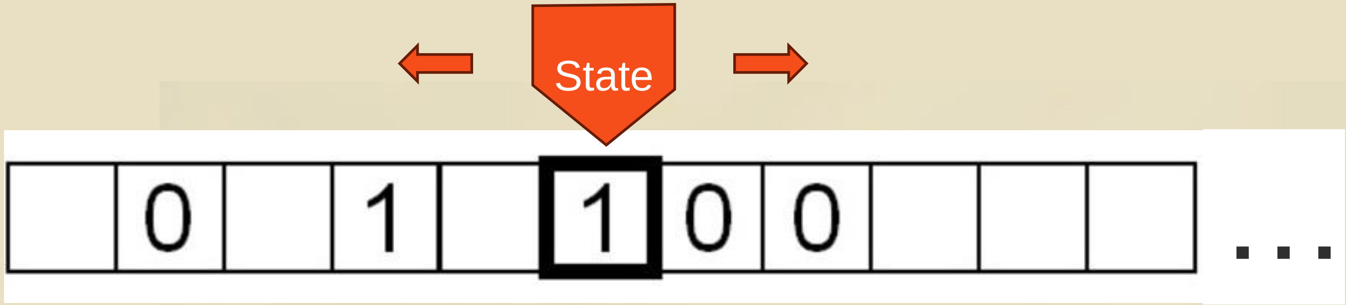
Examples of syntactically correct but semantically undefined codes

```
complex *p = NULL;
complex abs_p = sqrt(*p >> 4 + p->im);
```

Syntax & Semantics

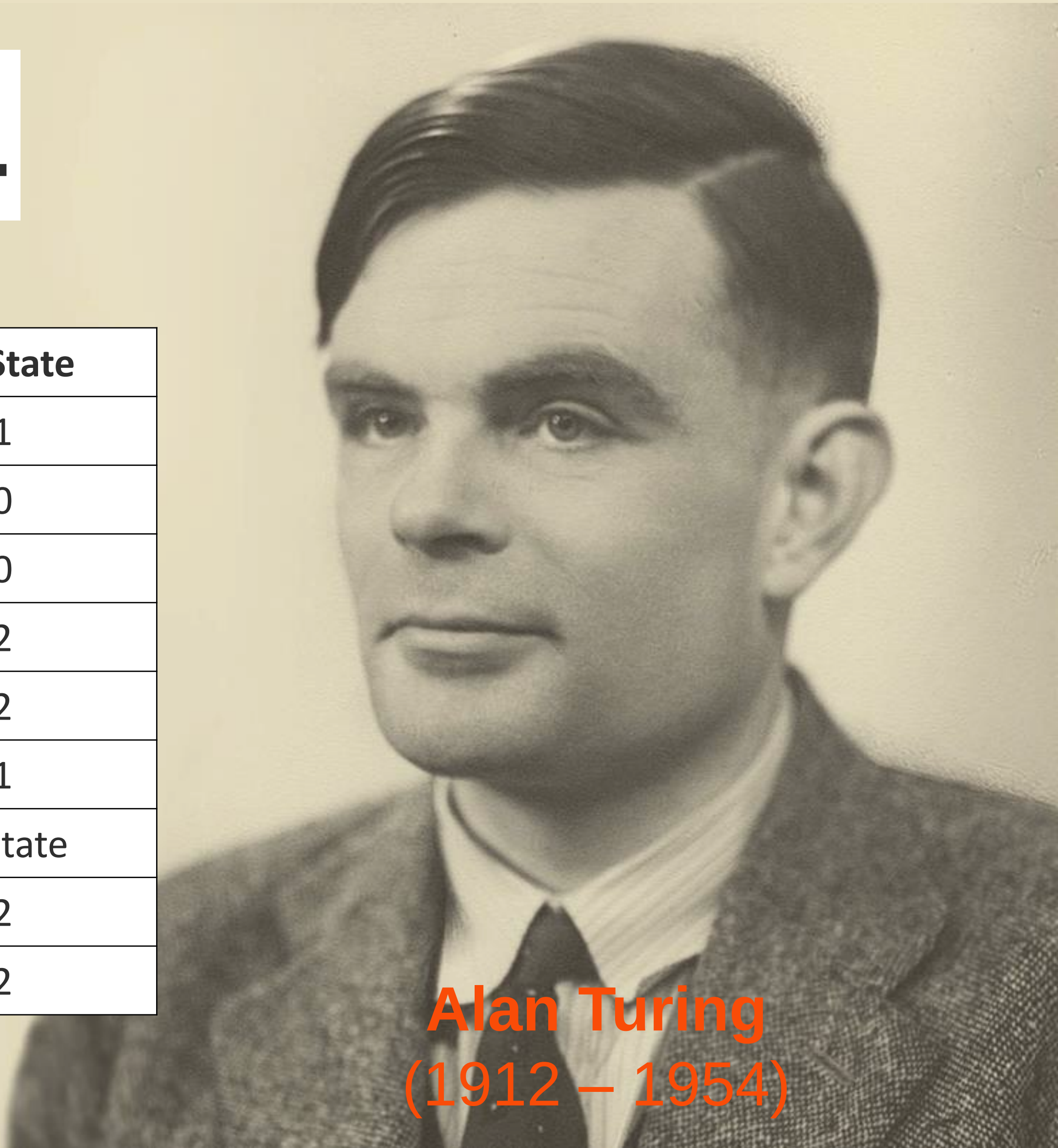
Chomsky, Noam (September 1956). "Three Models for the Description of Language" . *IRE Transactions on Information Theory*. **2** (3): 113–124.

The Turing Machine



Example of a “program” for the Turing Machine

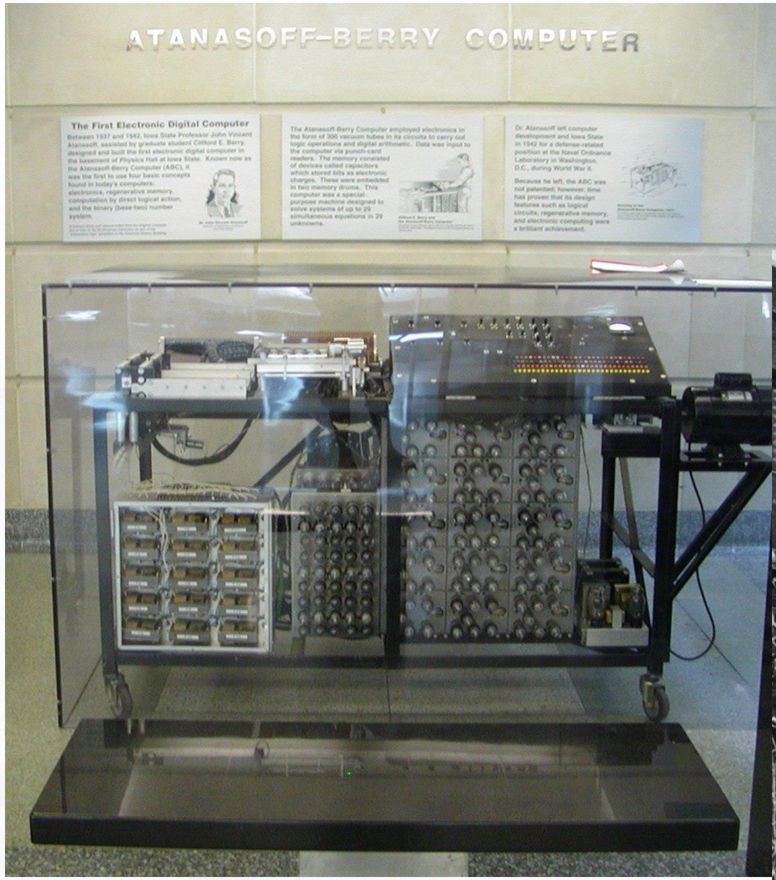
State	Symbol Read	Write Instruction	Move Instruction	Next State
State 0	Blank	Write ‘Blank’	Move tape right	State 1
	0	Write ‘0’	Move tape left	State 0
	1	Write ‘1’	Move tape left	State 0
State 1	Blank	Write ‘1’	Move tape left	State 2
	0	Write ‘1’	Move tape right	State 2
	1	Write ‘0’	Move tape right	State 1
State 2	Blank	Write ‘Blank’	Move tape right	Stop State
	0	Write ‘0’	Move tape left	State 2
	1	Write ‘1’	Move tape left	State 2



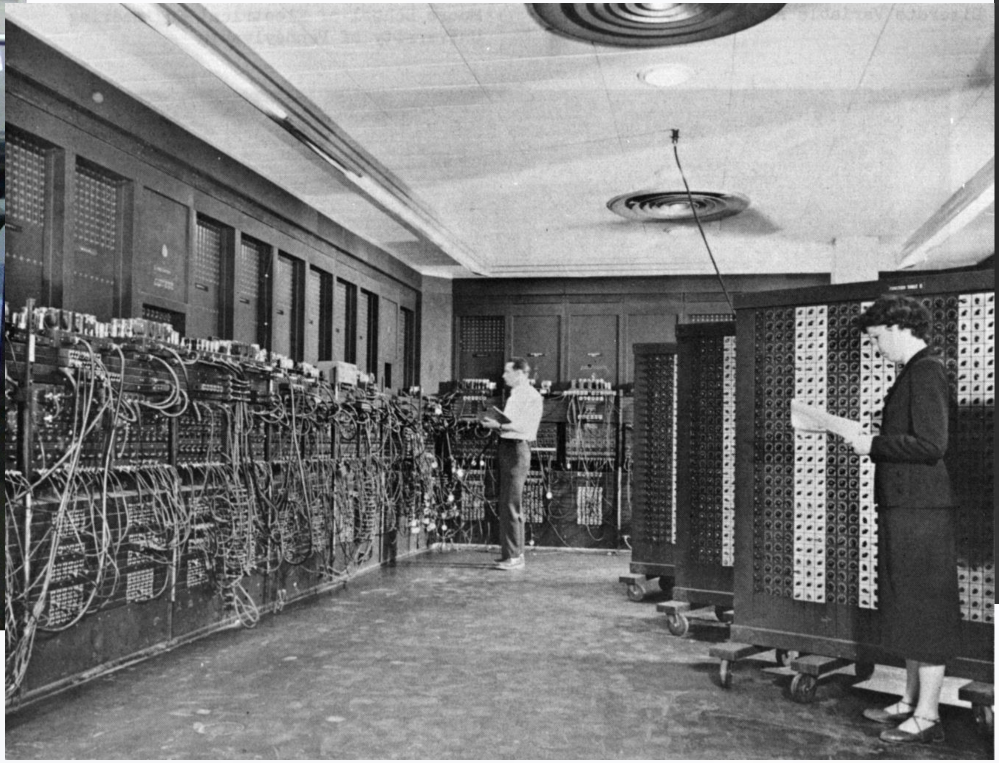
Alan Turing
(1912 – 1954)

Is human brain a Turing Machine?

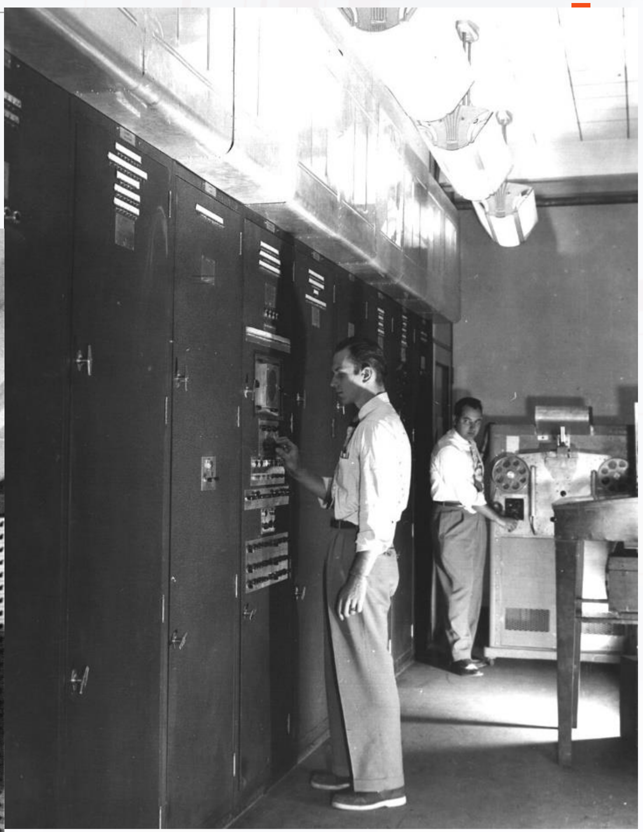
Early Digital Computers



Atanasoff–Berry computer (ABC) the first automatic electronic digital computer, 1942
Non-programmable,
non-Turing-complete

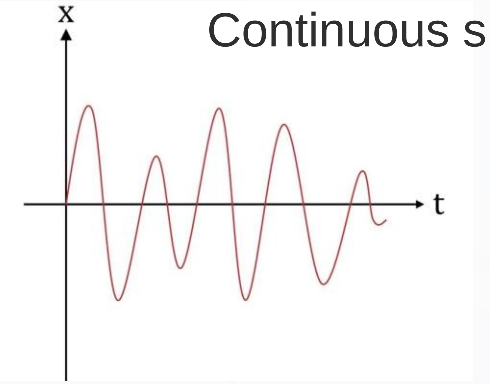


ENIAC (Electronic Numerical Integrator and Computer) the first programmable, electronic, general-purpose digital computer, completed in 1945.
Cost: $487,000 (~$6,600,000 in 2022)

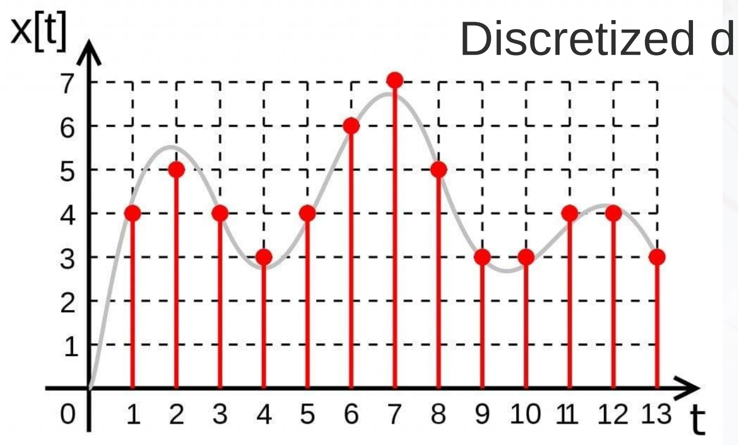


EDVAC (Electronic Discrete Variable Automatic Computer), 1949
Binary, stored-program computer
Cost: $500,000

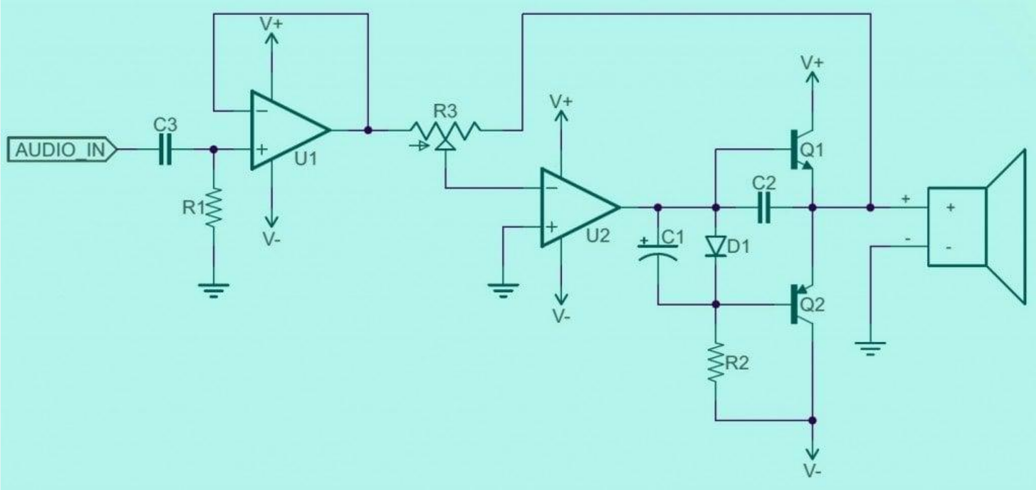
Analog VS. Digital

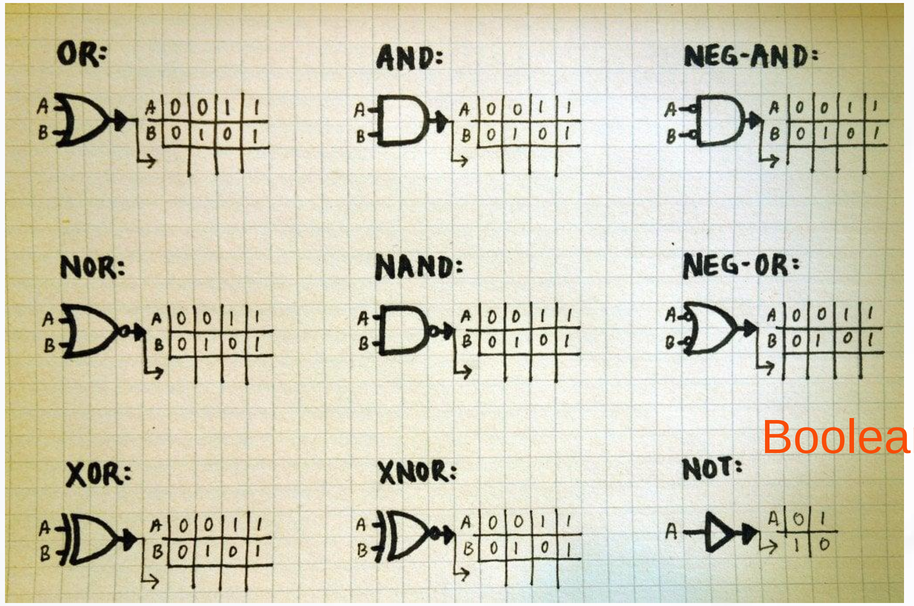


Continuous signal



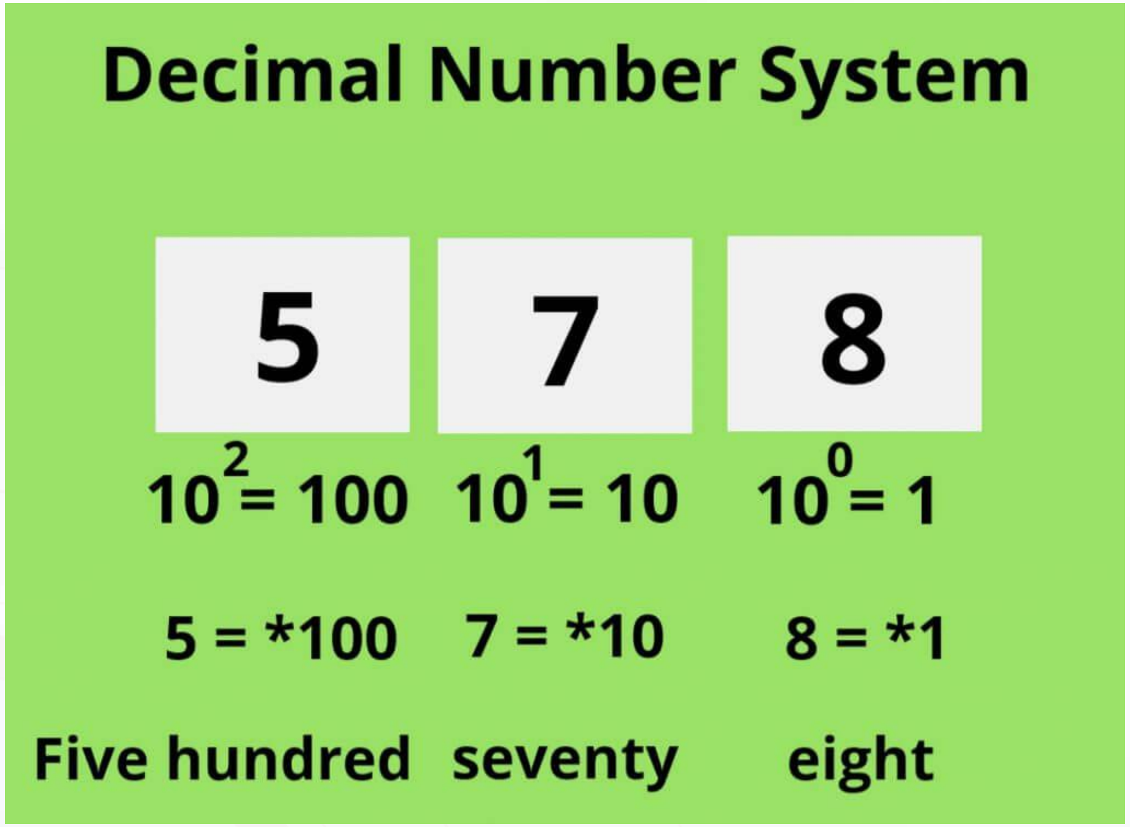
Discretized data





Boolean logic

Digital computers use binary numbers



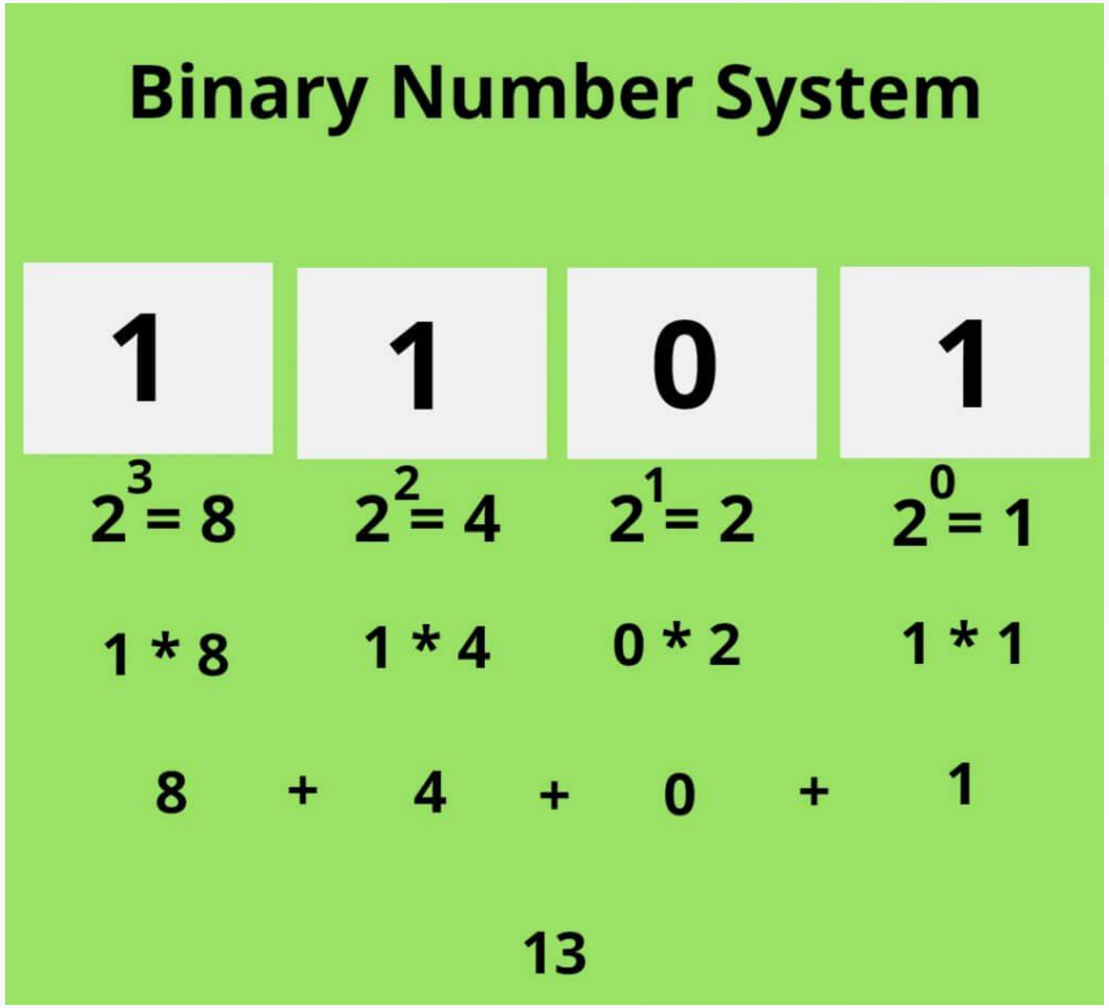
Decimal Number System

5 7 8

$10^2 = 100$ $10^1 = 10$ $10^0 = 1$

5 = *100 7 = *10 8 = *1

Five hundred seventy eight



Binary Number System

1 1 0 1

$2^3 = 8$ $2^2 = 4$ $2^1 = 2$ $2^0 = 1$

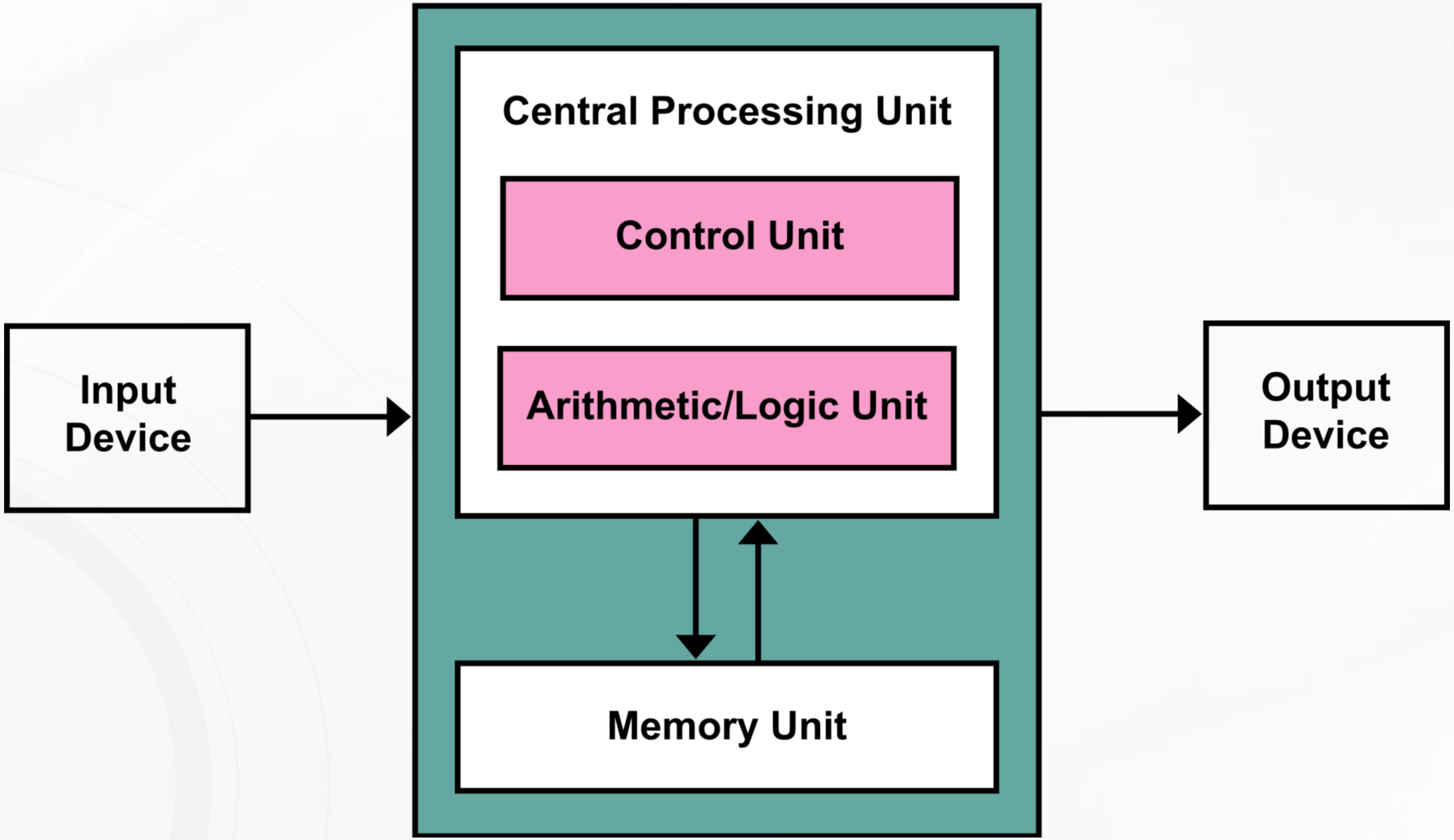
1 * 8 1 * 4 0 * 2 1 * 1

8 + 4 + 0 + 1

13

Hexadecimal: 0x0D

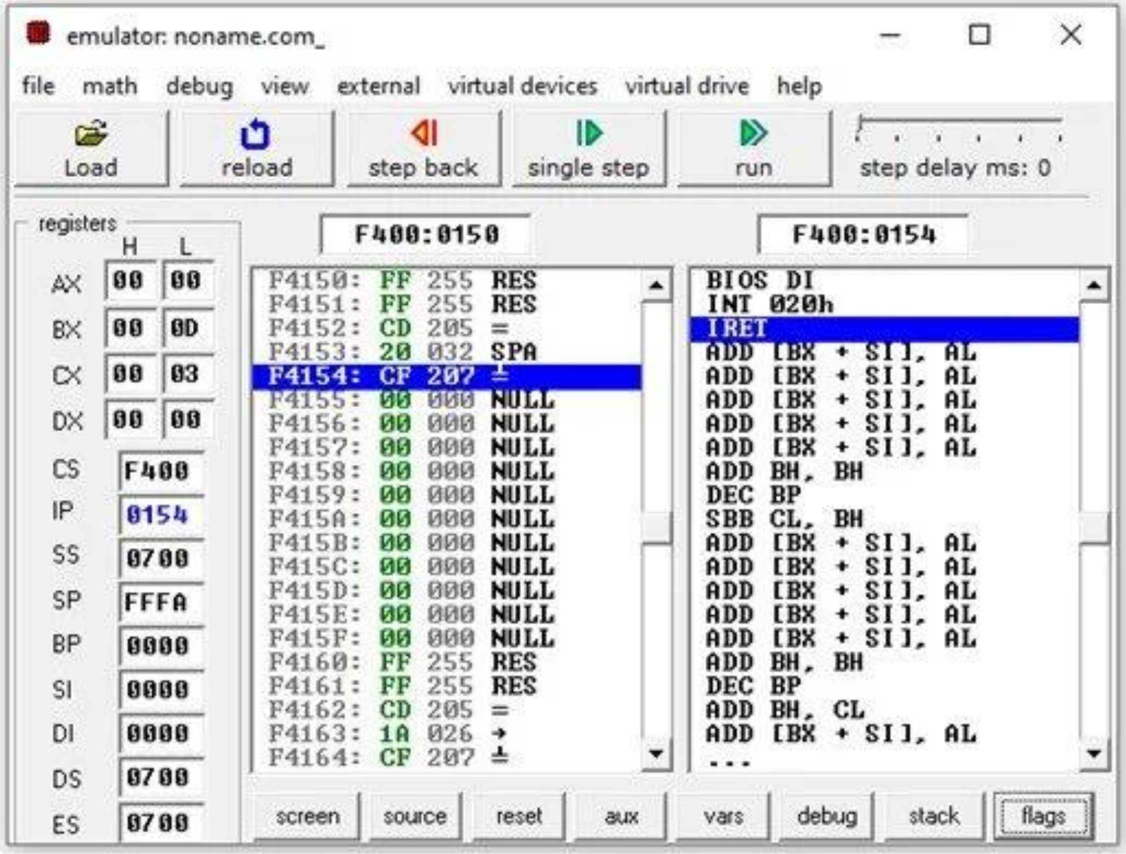
Von Neumann Architecture (Princeton Architecture)



John von Neumann
(1903-1957)

The First Draft of a Report on the EDVAC, John von Neumann (1945)

Machine code & Assembly language



```
     H   L
AX  00  00
BX  00  0D
CX  00  03
DX  00  00
CS  F400
IP  0154
SS  0700
SP  FFFA
BP  0000
SI  0000
DI  0000
DS  0700
ES  0700
```

```
F400:0150
F4150: FF 255 RES
F4151: FF 255 RES
F4152: CD 205 =
F4153: 20 032 SPA
F4154: CF 207 ±
F4155: 00 000 NULL
F4156: 00 000 NULL
F4157: 00 000 NULL
F4158: 00 000 NULL
F4159: 00 000 NULL
F415A: 00 000 NULL
F415B: 00 000 NULL
F415C: 00 000 NULL
F415D: 00 000 NULL
F415E: 00 000 NULL
F415F: 00 000 NULL
F4160: FF 255 RES
F4161: FF 255 RES
F4162: CD 205 =
F4163: 1A 026 →
F4164: CF 207 ±
```

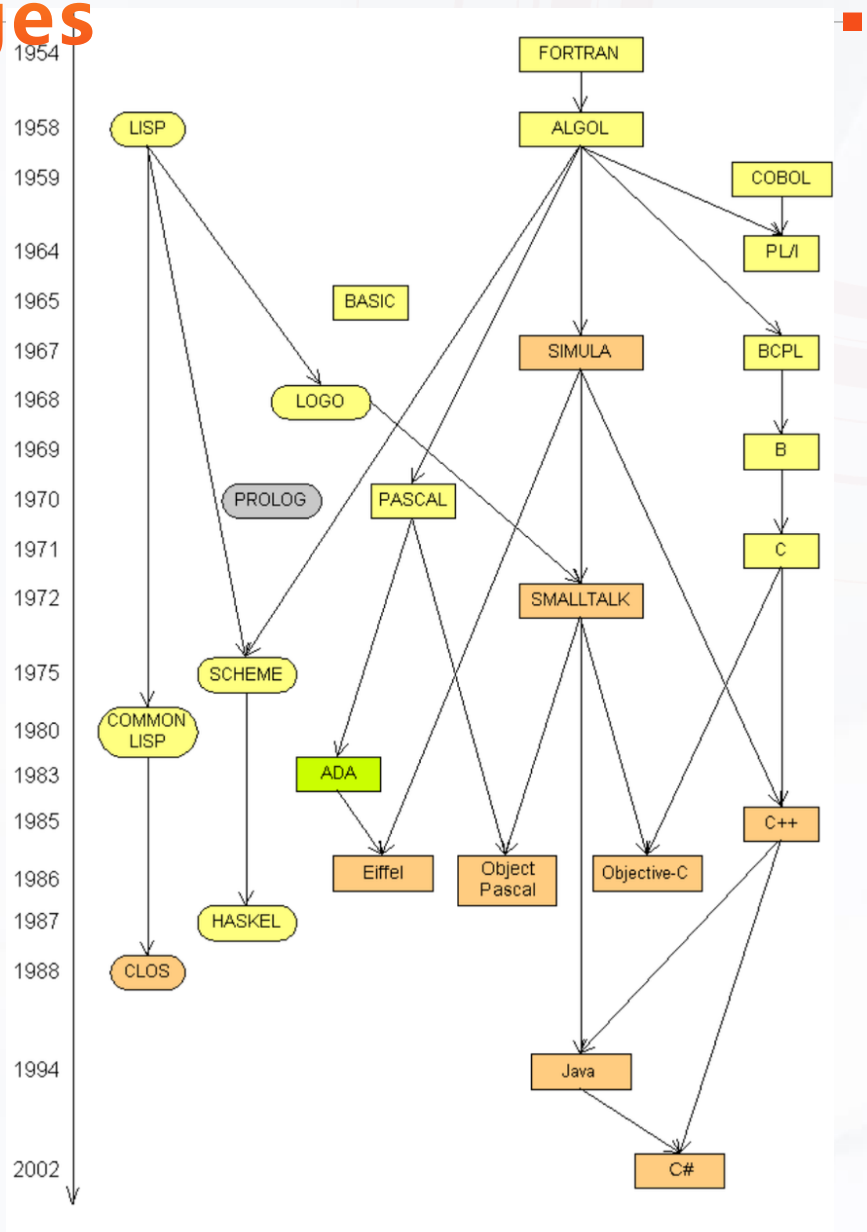
```
F400:0154
BIOS DI
INT 020h
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD BH, BH
DEC BP
SBB CL, BH
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD BH, BH
DEC BP
ADD BH, CL
ADD [BX + SI], AL
...
```

8086/8088 instructions

Instruction	Meaning	Opcode
`ADD`	Add	0x00...0x05, 0x80/0...0x81/0, 0x82/0...0x83/0 (since 80186)
`ADC`	Add with carry	0x10...0x15, 0x80...0x81/2, 0x82...0x83/2 (since 80186)
`AND`	Logical and	0x20...0x25, 0x80...0x81/4, 0x82...0x83/4 (since 80186)
`JMP`	Jump	0xE9...0xEB, 0xFF/4, 0xFF/5
`MOV`	Move	0xA0...0xA3
`MUL`	Unsigned multiply	0xF7/4, 0xF6/4
`SUB`	Subtraction	0x28...0x2D, 0x80...0x81/5, 0x82...0x83/5 (since 80186)

High level programming languages

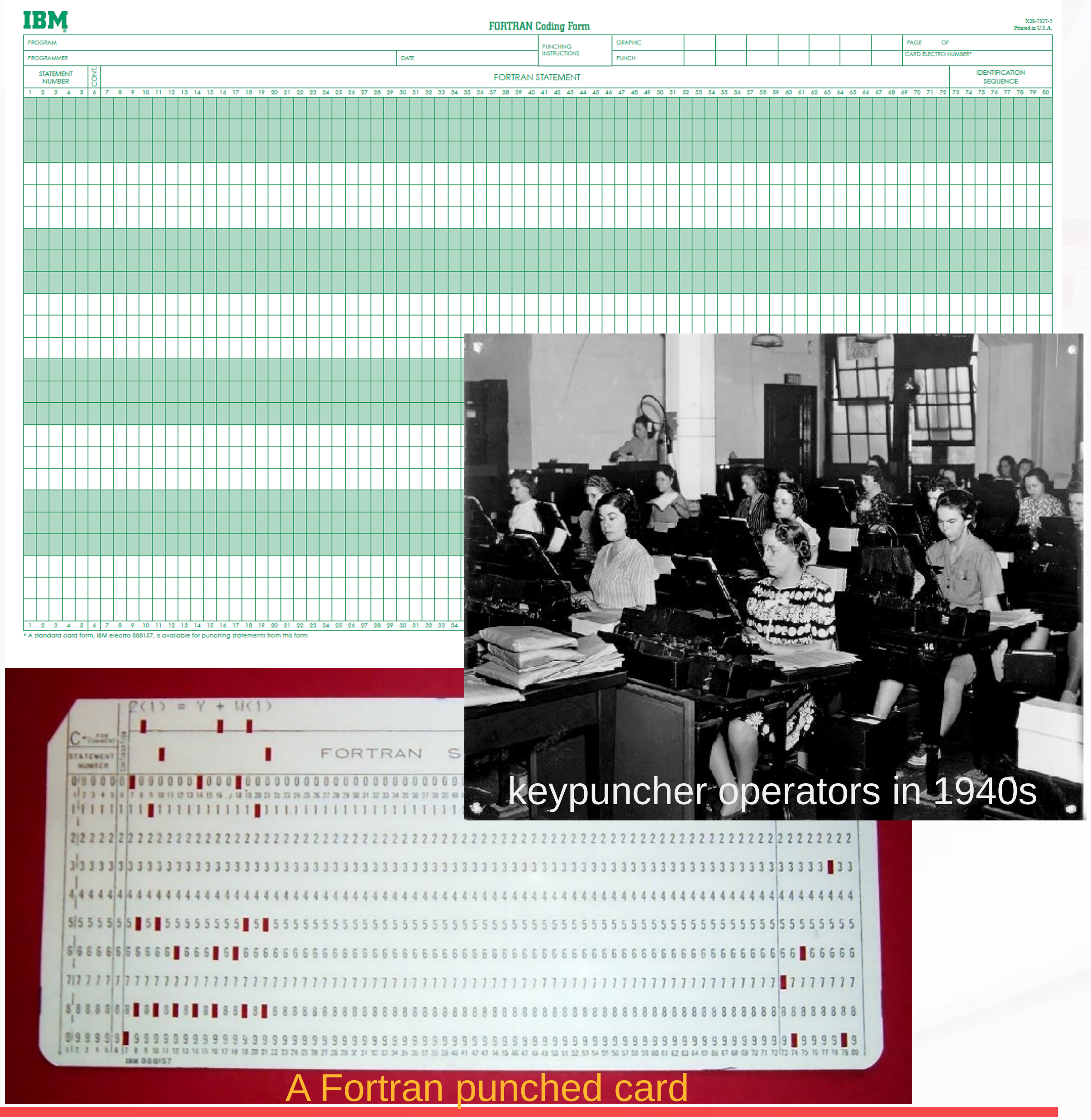
IBM 704 – first general purpose computer to support floating-point arithmetic



FORTRAN

FORmula TRANslating System

```
C AREA OF A TRIANGLE WITH A STANDARD SQUARE ROOT FUNCTION
C INPUT - TAPE READER UNIT 5, INTEGER INPUT
C OUTPUT - LINE PRINTER UNIT 6, REAL OUTPUT
C INPUT ERROR DISPLAY ERROR OUTPUT CODE 1 IN JOB CONTROL LISTING
      READ INPUT TAPE 5, 501, IA, IB, IC
  501 FORMAT (3I5)
C IA, IB, AND IC MAY NOT BE NEGATIVE OR ZERO
C FURTHERMORE, THE SUM OF TWO SIDES OF A TRIANGLE
C MUST BE GREATER THAN THE THIRD SIDE, SO WE CHECK FOR THAT, TOO
      IF (IA) 777, 777, 701
  701 IF (IB) 777, 777, 702
  702 IF (IC) 777, 777, 703
  703 IF (IA+IB-IC) 777, 777, 704
  704 IF (IA+IC-IB) 777, 777, 705
  705 IF (IB+IC-IA) 777, 777, 799
  777 STOP 1
C USING HERON'S FORMULA WE CALCULATE THE
C AREA OF THE TRIANGLE
  799 S = FLOATF (IA + IB + IC) / 2.0
      AREA = SQRTF( S * (S - FLOATF(IA)) * (S - FLOATF(IB)) *
     +   (S - FLOATF(IC)))
      WRITE OUTPUT TAPE 6, 601, IA, IB, IC, AREA
  601 FORMAT (4H A= ,I5,5H  B= ,I5,5H  C= ,I5,8H  AREA= ,F10.2,
     +    13H SQUARE UNITS)
      STOP
      END
```

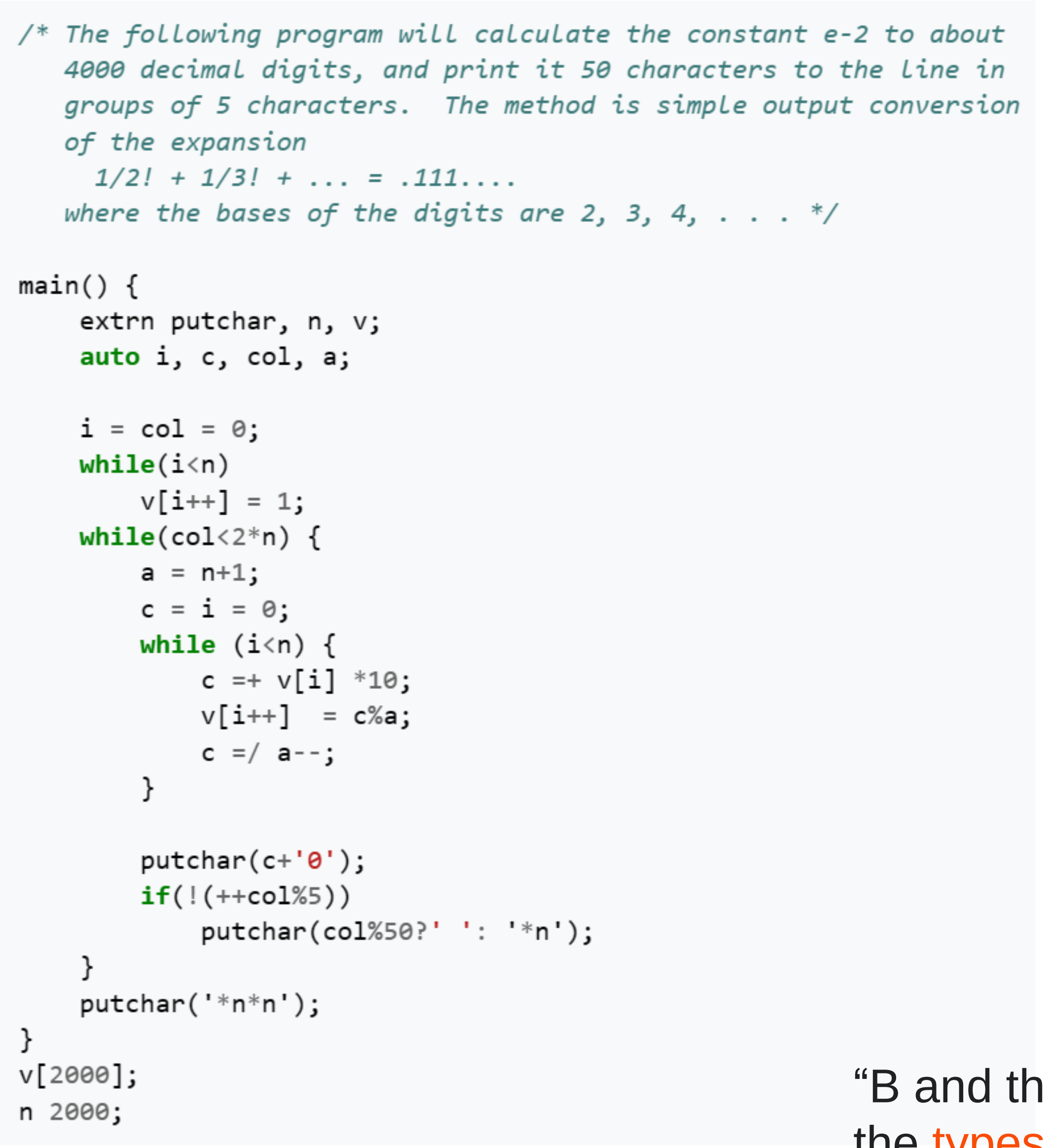


keypuncher operators in 1940s

A Fortran punched card

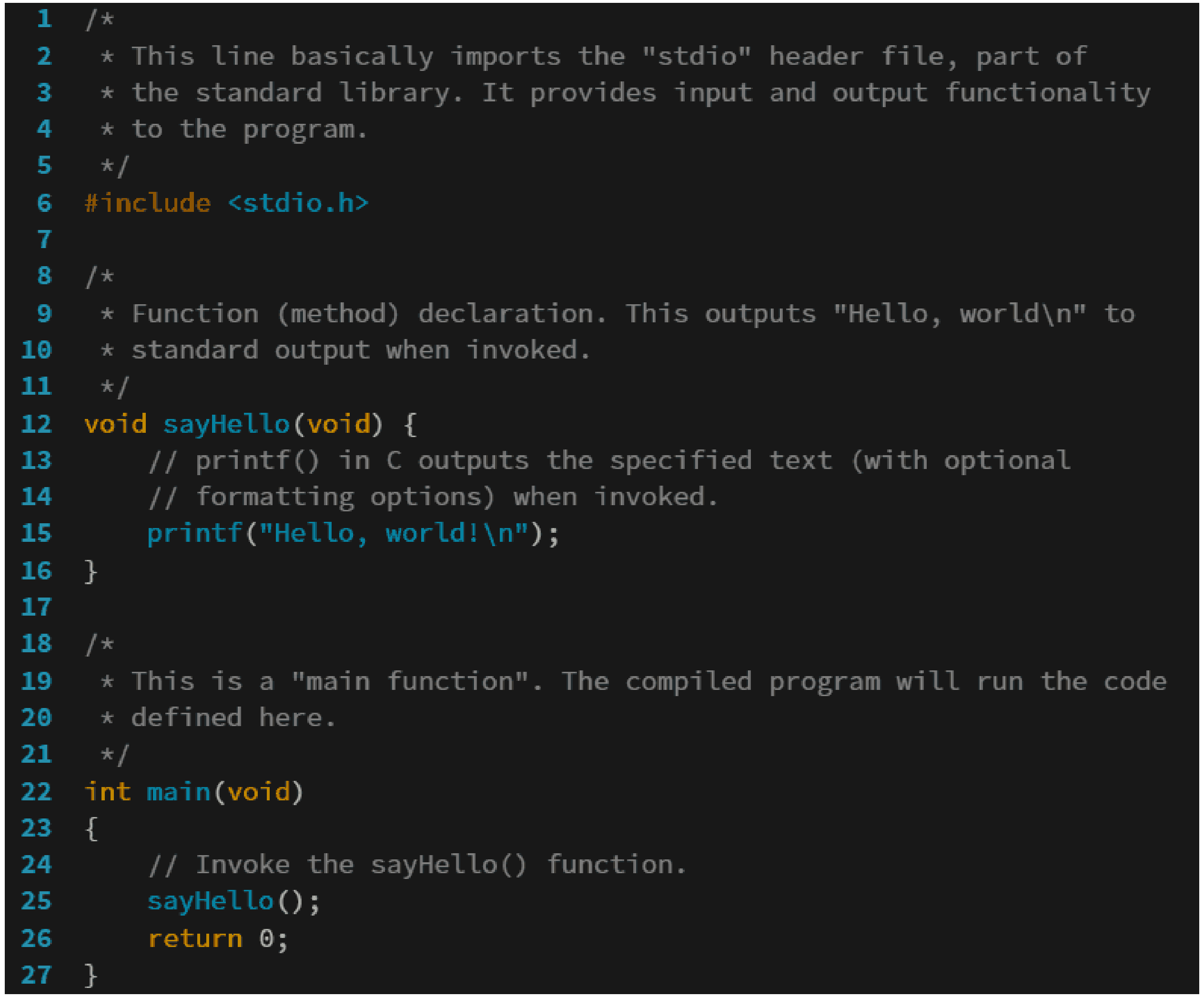
“A byproduct of UNIX”

The B (Bell Lab) language



```
/* The following program will calculate the constant e-2 to about
   4000 decimal digits, and print it 50 characters to the line in
   groups of 5 characters.  The method is simple output conversion
   of the expansion
     1/2! + 1/3! + ... = .111....
   where the bases of the digits are 2, 3, 4, . . . */

main() {
    extrn putchar, n, v;
    auto i, c, col, a;

    i = col = 0;
    while(i<n)
        v[i++] = 1;
    while(col<2*n) {
        a = n+1;
        c = i = 0;
        while (i<n) {
            c =+ v[i] *10;
            v[i++]  = c%a;
            c =/ a--;
        }

        putchar(c+'0');
        if(!(++col%5))
            putchar(col%50?' ': '*n');
    }
    putchar('*n*n');
}
v[2000];
n 2000;
```

The C language



```
/*
 * This line basically imports the "stdio" header file, part of
 * the standard library. It provides input and output functionality
 * to the program.
 */
#include <stdio.h>

/*
 * Function (method) declaration. This outputs "Hello, world\n" to
 * standard output when invoked.
 */
void sayHello(void) {
    // printf() in C outputs the specified text (with optional
    // formatting options) when invoked.
    printf("Hello, world!\n");
}

/*
 * This is a "main function". The compiled program will run the code
 * defined here.
 */
int main(void)
{
    // Invoke the sayHello() function.
    sayHello();
    return 0;
}
```

“B and the old old C were very very similar languages except for all the types” - *Ken Thompson* in *Users’ reference to B*

Standardization/development of C

- *K&R C*
  - Described in Kernighan and Ritchie, *The C Programming Language* (1978)
  - De facto standard
- *C89/C90*
  - ANSI standard X3.159-1989 (completed in 1988; formally approved in December 1989)
  - International standard ISO/IEC 9899:1990
- *C99*
  - International standard ISO/IEC 9899:1999
  - Incorporates changes from Amendment 1 (1995)
- *C11*
  - International standard ISO/IEC 9899:2011
  - Incorporates features already supported by common contemporary compilers, and includes a detailed memory model to better support multi-thread execution
- *C17*
  - International standard ISO/IEC 9899:2018
  - Fixes numerous minor defects without introducing new features
- *To be superseded by C23 (ISO/IEC 9899:2023) in 2024*

The differences between the different versions aren't huge, and the compilers are usually backward compatible.

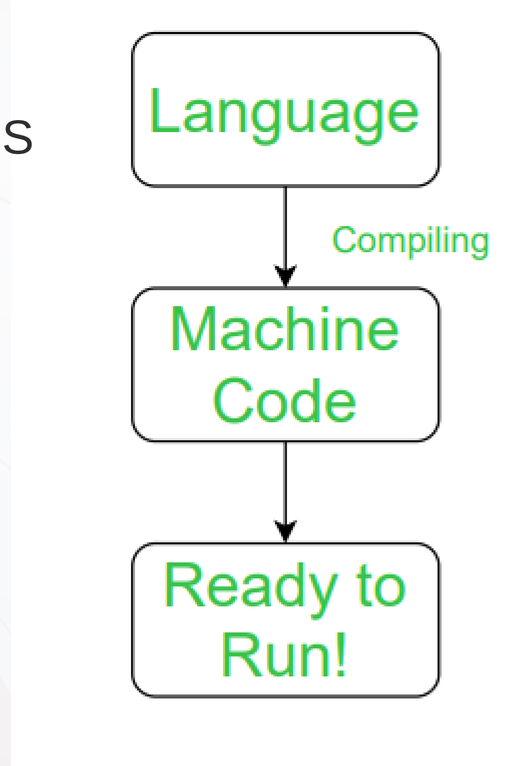
Decedents of C

- ***C++*** includes all the features of C, but adds classes and other features to support object-oriented programming.
- ***Java*** is based on C++ and therefore inherits many C features.
- ***C#*** is a more recent language derived from C++ and Java.
- ***Perl*** has adopted many of the features of C.
- …

Compiled Language

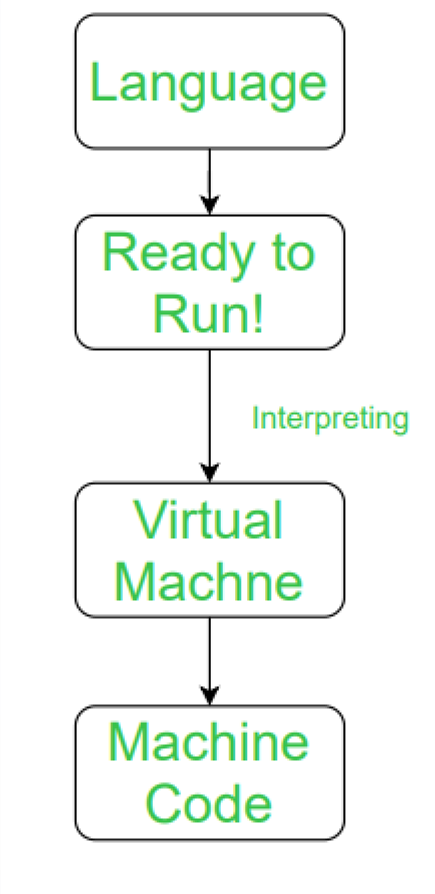
- Faster
- Closer to OS

- Platform dependent
- Steeper learning curves
- Difficult to debug / modify


Language
Compiling
Machine Code
Ready to Run!

VS.

Interpreted Language


Language
Ready to Run!
Interpreting
Virtual Machne
Machine Code

- Cross-platform
- Run directly
- Usually easier
- Run-time debugging

- Slower
- Source code visible to user

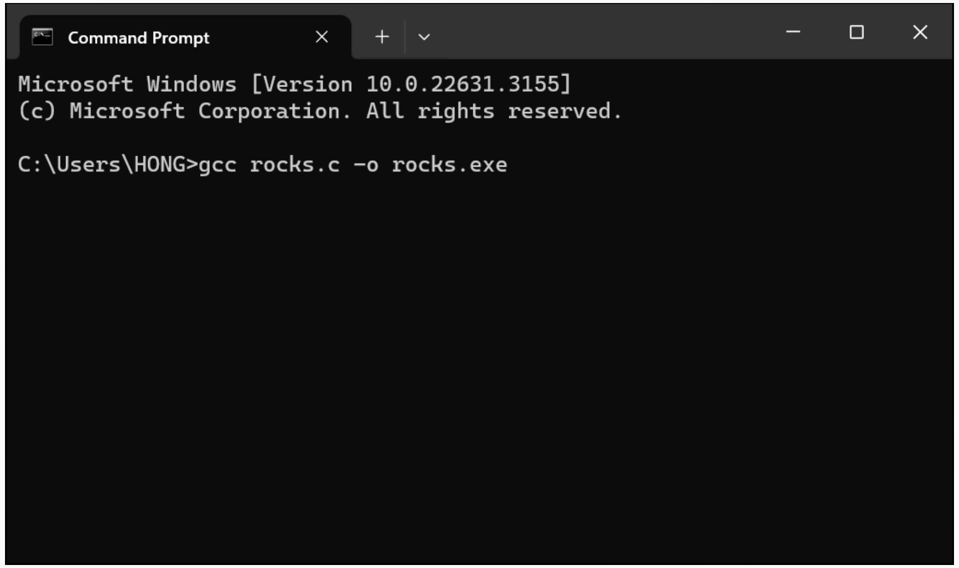
e.g. Fortran, C, C++, C#, COBOL, Rust…

e.g. Python, matlab, Java script, PHP, BASIC…

The way C works

rocks.c

```c
#include <stdio.h>
int main()
{
      puts("C Rocks!");
      return 0;
}
```



```
Microsoft Windows [Version 10.0.22631.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HONG>gcc rocks.c -o rocks.exe
```

rocks.exe

Source
Edit a source file (with a file name ending by .c)

Compile (& Link)

Executable
A file containing machine codes, ending by .exe in Windows.