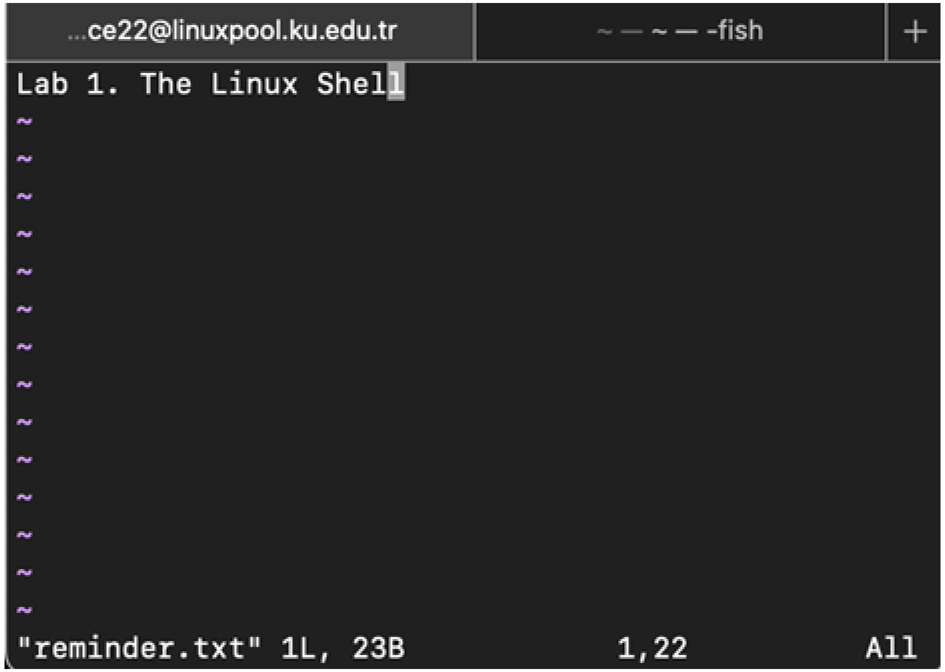
Bits, Ints and Floats, Vim, AI Assistant

COMP201 Lab 2
Spring 2025

Vim

KOÇ
UNIVERSITY

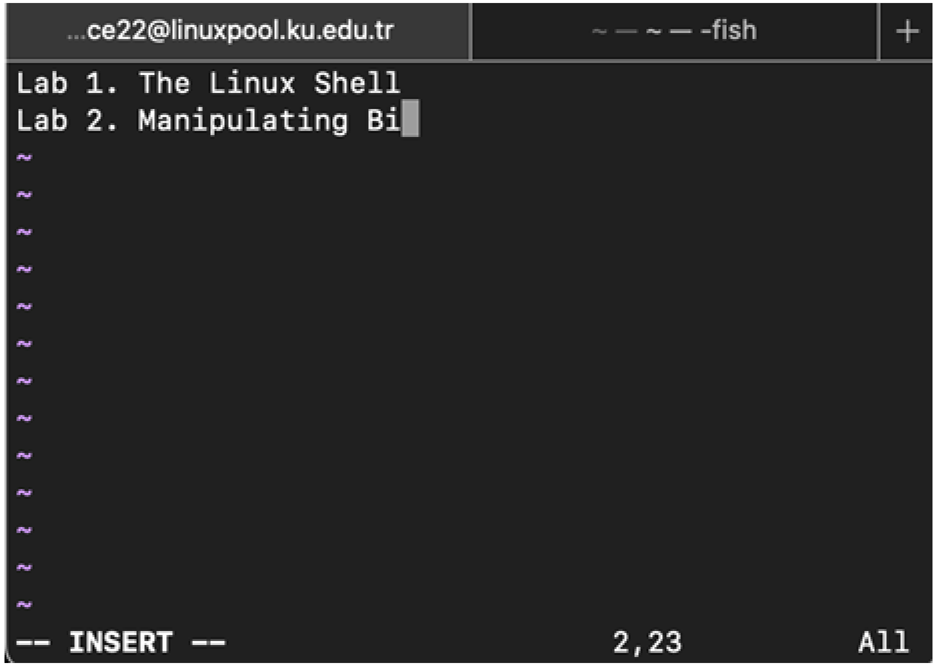
Vi/Vim Reminder



```
Lab 1. The Linux Shell
~
~
~
~
~
~
~
~
~
~
~
~
~
"reminder.txt" 1L, 23B                1,22          All
```

- Normal mode
  - The default mode when launching Vim
  - Mainly allows navigating through text
  - Press **u** or type **:undo** (then Enter) to undo
  - Type **:redo** (then Enter) to redo
  - **Cannot type in this mode!**

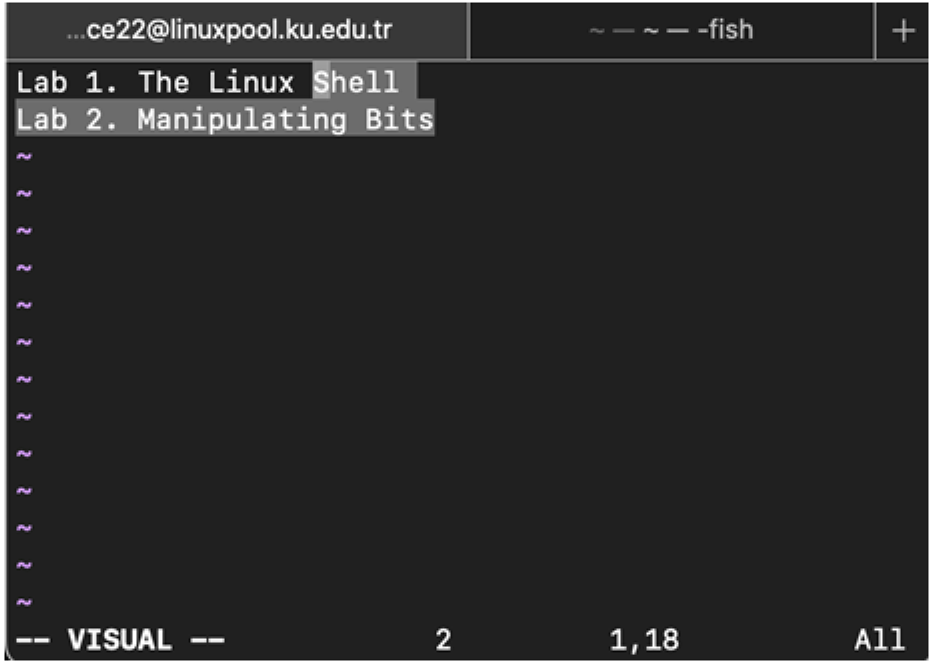
Vi/Vim Reminder



```
Lab 1. The Linux Shell
Lab 2. Manipulating Bi
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                    2,23          All
```

- Insert mode
  - Every character you type is put to the file.
  - Cue the **--INSERT--** on the left bottom
  - To switch from normal mode to insert mode, type **i** in the normal mode.
  - To switch back to normal mode, press **esc**

Vi/Vim Reminder



```
Lab 1. The Linux Shell
Lab 2. Manipulating Bits
~
~
~
~
~
~
~
~
~
~
~
~
~
-- VISUAL --            2          1,18          All
```

- Visual mode
  - Allows selecting a text block with arrow keys.
  - After selecting the block:
    - Type **d** to delete the block
    - Type **x** to cut the block
    - Type **y** to copy the block
    - Type **p** to paste copied (or cut) block
  - To switch from normal mode to visual mode, type **v**.
  - To switch back to normal mode, type **Esc**.

Basic Commands in Vi/Vim (in Normal Mode)

- **Basic navigation:** Arrow keys
- **Navigating across words:** w (next word), b (beginning of word), e (end of word)
- **Jumping in a line:** 0 (beginning of line), $ (end of line)
- **Jumping in a file:** gg (beginning of file), G (end of file), :{num}<Enter> (moving to line number num)
- **Searching for a string:** /{regex}, n (moving forward to find the next match), N (moving backward to find a previous match)
- **Quitting a file without saving:** :q
- **Quitting a file by discarding modification:** :q!
- **Saving a file without quitting the file:** :w
- **Saving a file and quitting it:** :x

Bitwise Operations and Bit Representation of Integers & Floats

KOÇ UNIVERSITY

Bitwise Operations

- Bitwise Operators.
  - & ^ | ~ << >> !
  - Examples of bitwise operations:
    - **Getting least significant 2 bits of 1110:**
      - 1110 & 0011 = 0010

    - **Flipping least significant 2 bits of 1110:**
      - 1110 ^ 0011 = 1101

    - **Arithmetic right shifting 1010 by 2 bits:**
      - 1010 >> 2 = 1110

    - **Getting the most significant 2 bits of 1010:**
      - (1010 >> 2) & 0011 = 1110 & 0011 = 0010

Bitwise Operations at Byte Level

- **Getting the least 4-bits of 0x6e**
  0x6e & 0x0f = 01101110 & 00001111 = 00001110 = 0x0e

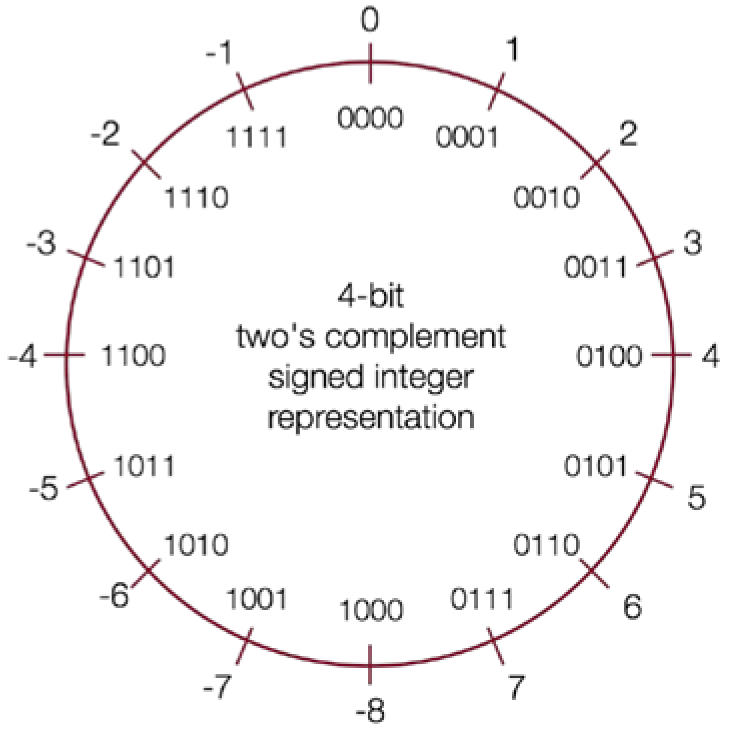
- **Flipping the least significant 4-bits of 0x6e**
  0x6e ^ 0x0f = 01101110 ^ 00001111 = 01100001 = 0x061

- **Arithmetic right shifting 0xee by 4 bits**
  0xee >> 4 = 11101110 >> 4 = 11111110 = 0xfe

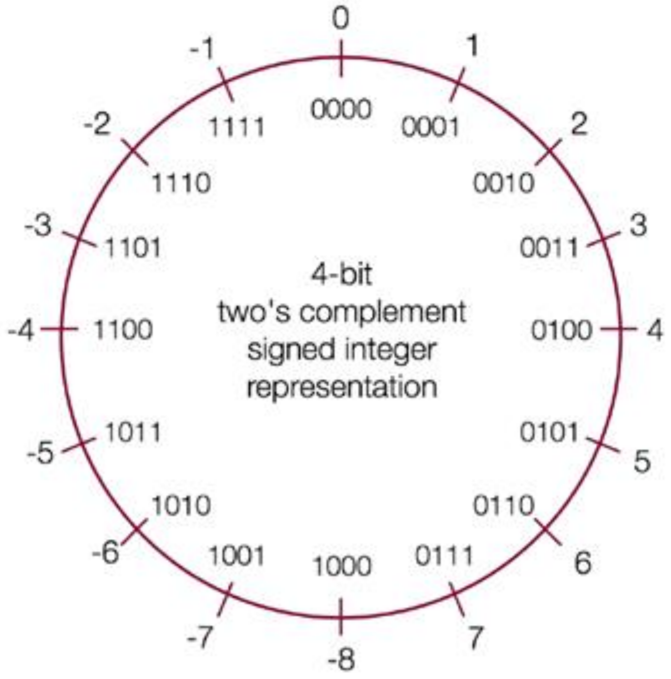
- **Getting the most significant 4 bits of 0xe5**
  (0xe5 >> 4) & 0x0f = (11100101 >> 4) & 00001111 = 11111110 & 00001111 = 00001110 = 0x0e

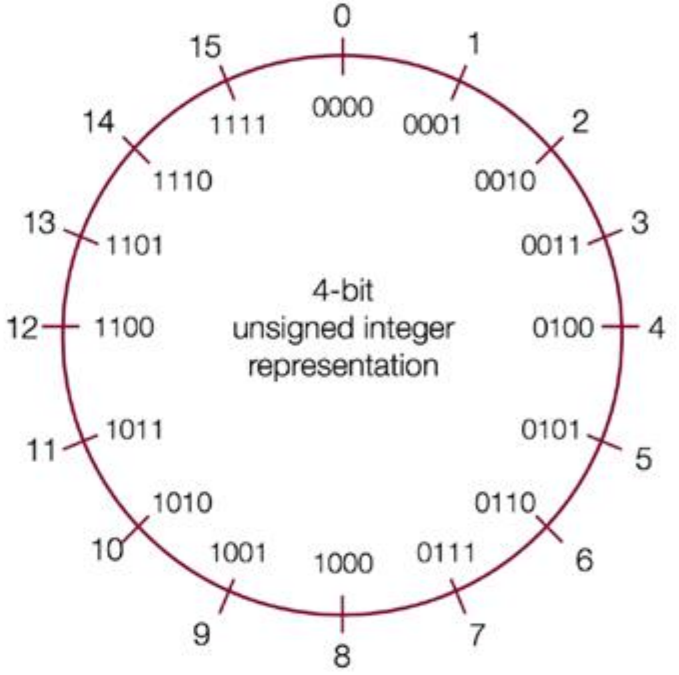
Two’s Complement (Bit Representation of Integers)

- We represent a positive number by itself and a negative number by the two’s complement of the corresponding positive number
- The two’s complement of a number is the binary digits inverted, plus 1.
  - e.g. -0001 (1) = 1111 (-1)
- Standard addition works
  - e.g. 1111 (-1) + 0001 (1) = 0000 (0)
- All bits are used to represent as many numbers as possible (efficient)



Signed vs Unsigned

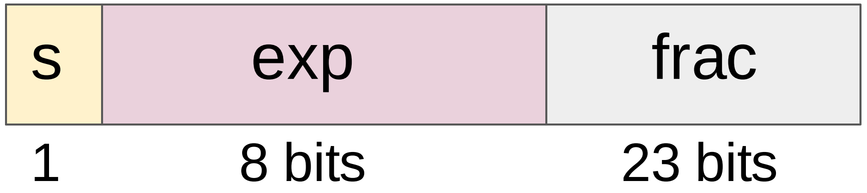




Two’s Complement Exercises

- **minusOne** - return a value of -1
    - Example: minusOne() = -1
    - Legal ops: ! ~ & ^ | + << >>
- **negate** - return -x given x
    - Example: negate(5) = -5, negate(-4) = 4
    - Legal ops: ! ~ & ^ | + << >>
- **fitsShort** - return 1 if x can be represented as a 16-bit, two's complement integer.
    - Examples: fitsShort(33000) = 0, fitsShort(-32768) = 1
    - Legal ops: ! ~ & ^ | + << >>

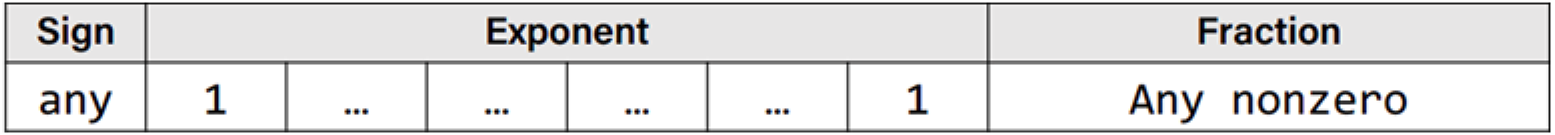
Bit Representation of Floating Point Numbers (32-bits)



- 1 bit is for sign
- 8 bits are for exponent
- 23 bits are for fraction
- Bias = $2^{(8 - 1)} - 1 = 127$
- How to read:
  - If exp > 0 (normalized), floating point number = (s ? -1 : 1) * (1.frac) * $2^{(exp - 127)}$
  - If exp = 0 (denormalized), floating point number = (s ? -1 : 1) * (0.frac) * 2^{-126}

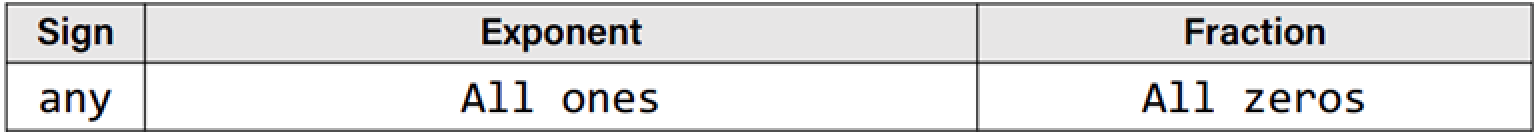
Bit Representation of Floating Point Numbers (32-bits)

- **Not A Number (NaN):**



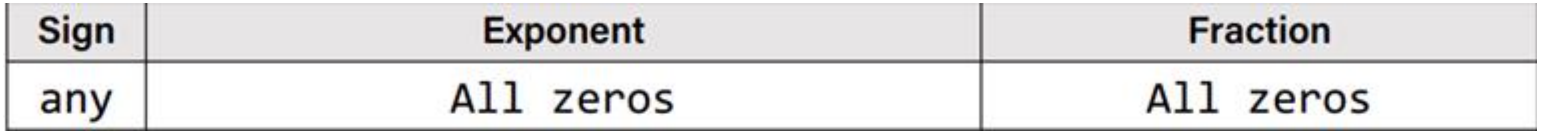
Sign	Exponent						Fraction
any	1	…	…	…	…	1	Any nonzero

- **± Infinity (± ∞):**



Sign	Exponent	Fraction
any	All ones	All zeros

- **Zero (0):**



Sign	Exponent	Fraction
any	All zeros	All zeros

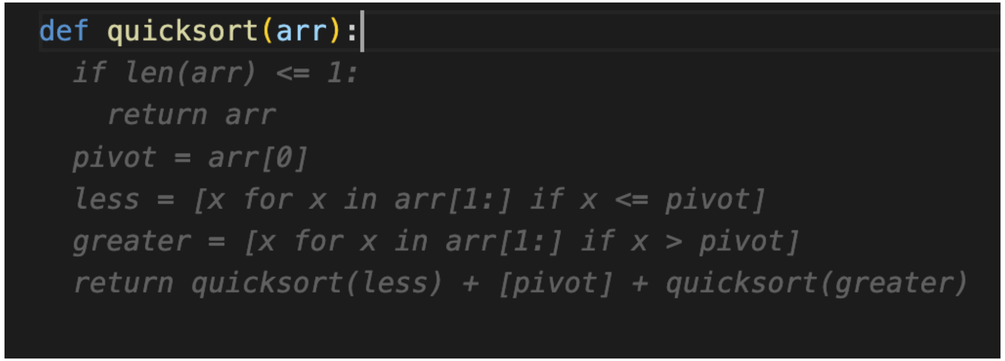
AI Assistants

KOÇ UNIVERSITY

AI Assistans For Coding

There are many LLM models that are specialized or used for general purposes in coding.

- GitHub CoPilot
- CursorAI
- Tabnine
- ChatGPT
- Claude
- DeepSeek



```python
def quicksort(arr):
  if len(arr) <= 1:
    return arr
  pivot = arr[0]
  less = [x for x in arr[1:] if x <= pivot]
  greater = [x for x in arr[1:] if x > pivot]
  return quicksort(less) + [pivot] + quicksort(greater)
```

When to use

It is highly recommended to avoid using AI models while learning to code. While they may enhance your productivity, they can significantly slow down your learning process. Instead, focus on developing a deep understanding of coding concepts through hands-on practice. Once you have built a solid foundation, AI models can be valuable tools in your professional career and additional projects, helping you boost efficiency and innovation.

Links

- https://vim.rtorr.com/
- NeoVim, Nano
- https://www.h-schmidt.net/FloatConverter/IEEE754.html
- https://github.com/features/copilot