COL100
Introduction to Computer Science

2023-24 Semester II, Groups 21-30
Instructor: LH121: Subodh Kumar (Computer Sc & Engg., Bharti 429)
LH114: Prem Kalra (Computer Sc & Engg., Bharti 401/431)
Lecture: Mo 330-5, Th 330-5
Labs: LH504 (Group-wise: M-F, 9-11, 11-1)

Contact

- moodle moodlenew.iitd.ac.in
  - Main and persistent source of information: slides, schedule, references, etc.
  - Programming exercises and tests
  - Discussion forum: ask (and answer) course-related questions
- ✉ col100admin@iitd.ac.in
  - Send email for help, will reach instructors and TAs
- Instructor open hours: available after class, in lab
- Mentor TA (Will be announced on Moodle), reach out for individual help

Tentative Evaluation Plan

ID required

- Retests will be 1.25 times harder
- Only one retest will be allowed (none for labs)
- Only if 75% attendance at the end

Pre Mid-Term	In class	5	~20 minutes 12 Feb
Mid-Term	As scheduled	20	
Pre Final	In class	6	~22 minutes 16 Apr
Final	As scheduled	30	
Programming Test	In Lab, On Moodle	3×8	~45 minutes 3/2, 2/3, 6/4, 20/4
Lab Participation	Weekly In Lab, On Moodle	10×1.5	10 out of 13/14

Other Policies

Strict policies

- D on 30, E on 20
  - Up to 2 extra credit mark for those between 18 and 28
  - Additional lab/tutorial sessions
- Attend lectures (and labs)

Retest only if 75% attendance at the end

  - Self study more efficient?

Varies from topic to topic. Slides not sufficient.

  - Fallen behind, cannot understand?

Absence will not help you catch up

  - Not important enough?

Think again

- Unfair practice has very high risk

F if caught

  - Discussion is allowed for in-lab exercises only (first ~1.5 hour only)

General Rules and Etiquette

- No side-discussions in lecture
  - Discussion allowed for the first 1.5 hours of the lab
- No arriving late or leaving early
  - Entry to the lab will be closed 15 minutes past the start

Allowed only in your assigned slot

- No mobile use in class (lectures and lab)
  - No ringing in lecture, mobile to be left outside in lab
- When sending email to course admin, use COL100 in the subject
- Upload medical certificate; link will be provided on Moodle (only if you have missed a test for which make up is possible)

Introduction to CS

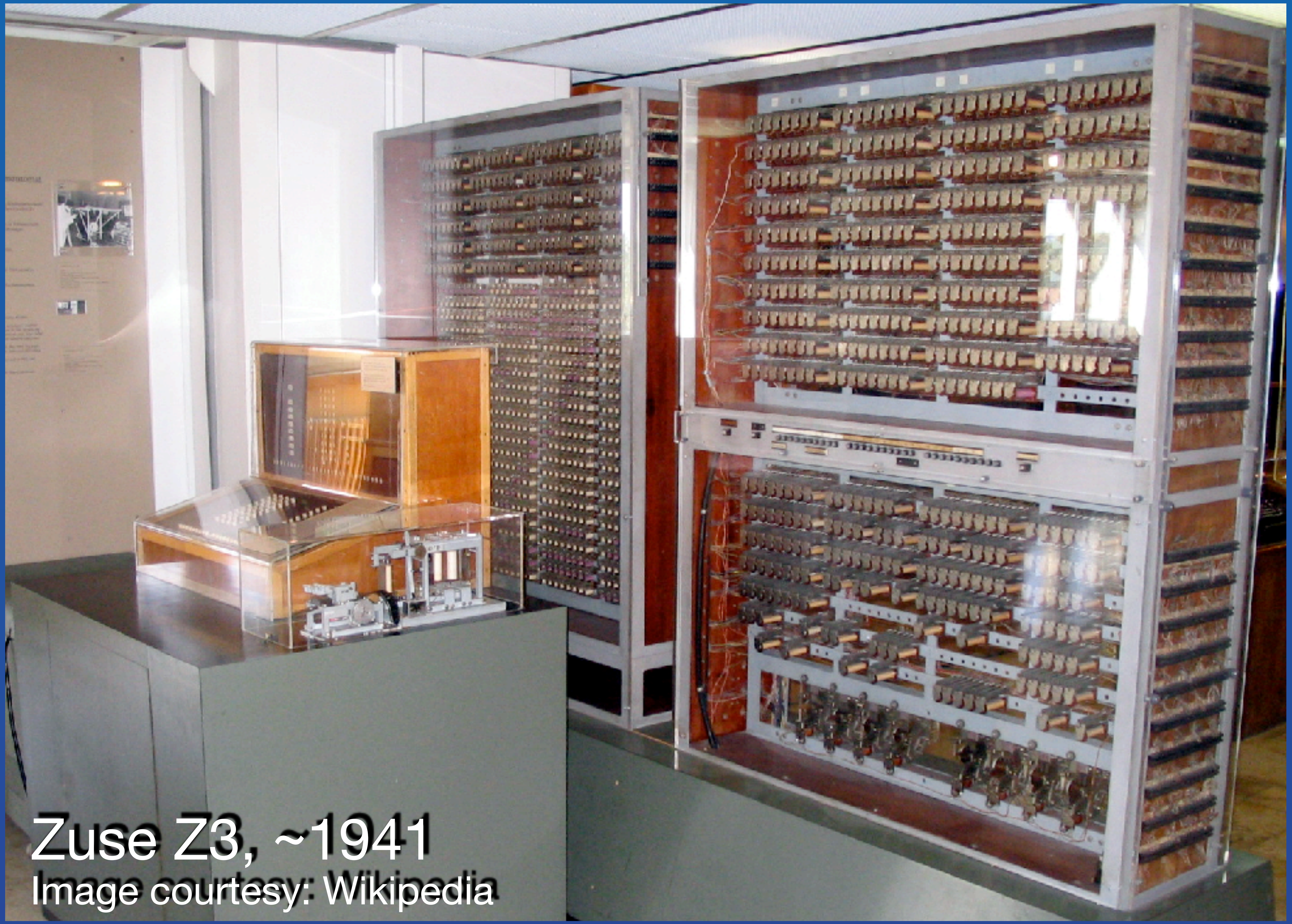
- How to solve Computational problems

  Take values and produce values

  - What is computable?
  - Methods to compute what is computable
    - Best way to compute what is computable
      - Define metric, Measurement procedure
  - Computers (Hardware, Systems, Theoretical models)
- Examples
  - Computing abstraction | Algorithms & Data | Program expression & verification | Data analysis & prediction | Computer architecture | Operating systems | Security | Networks | System modeling | Numerical computation | Interfaces

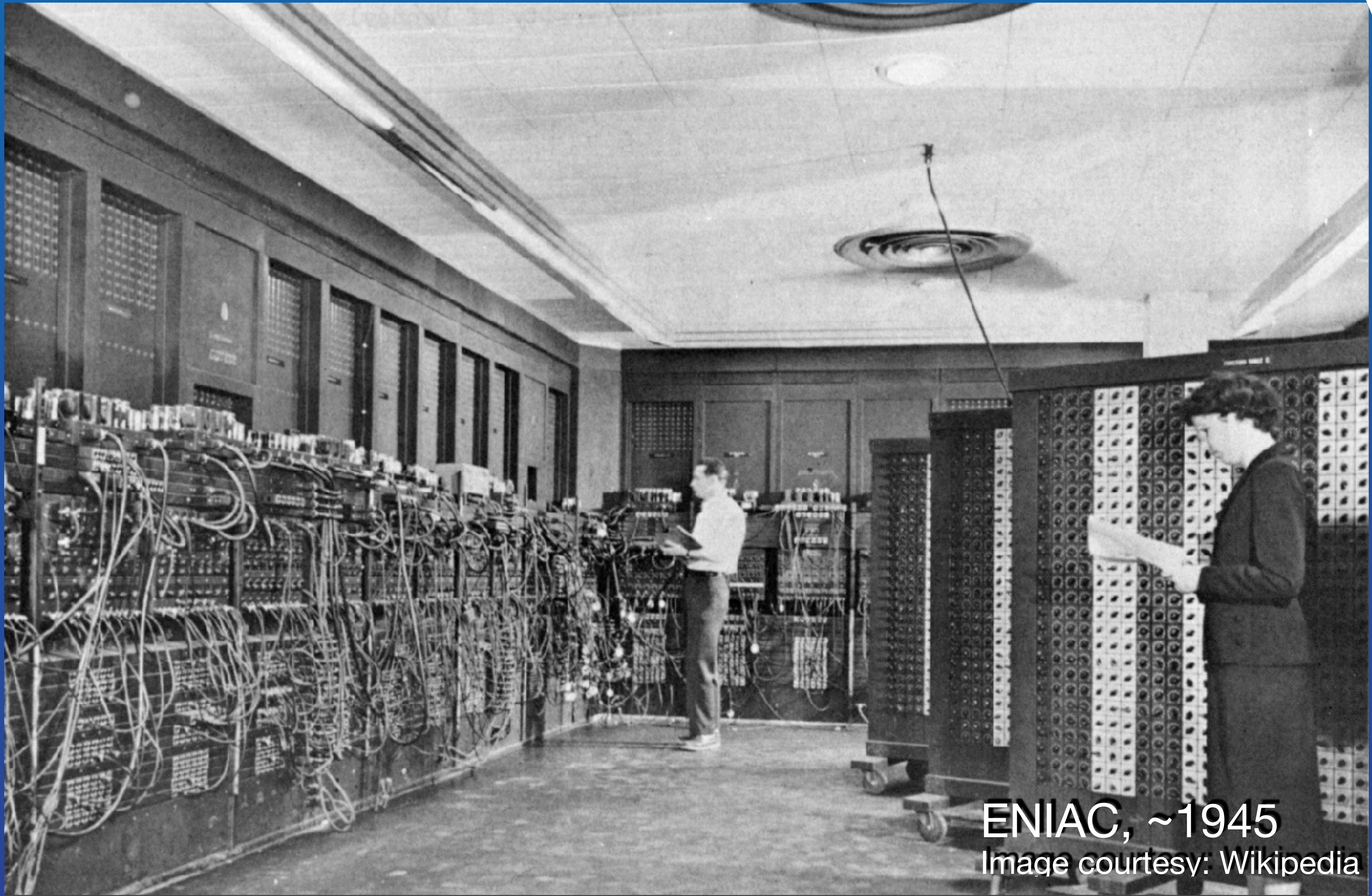
Computer?

BLETCHLY PARK BOMBE, ~1940
Image courtesy: Wikipedia



Zuse Z3, ~1941
Image courtesy: Wikipedia

Computer?



ENIAC, ~1945
Image courtesy: Wikipedia

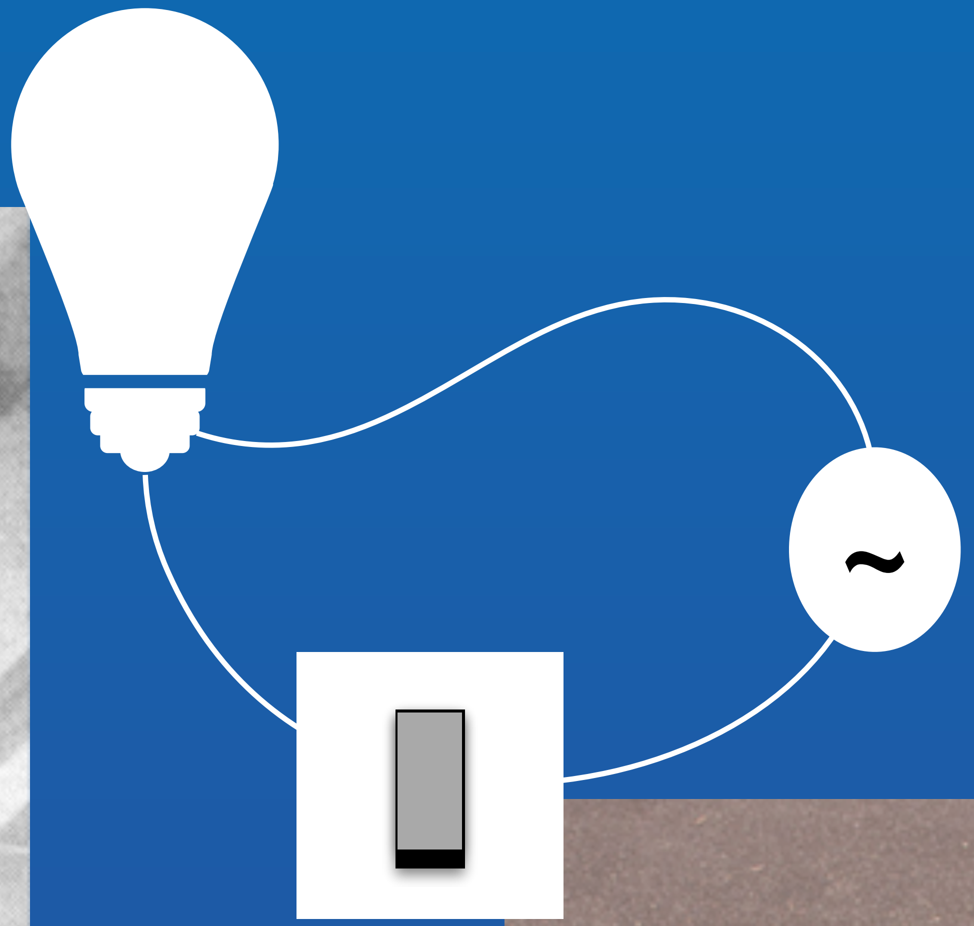


Image:Jeepika

Image: JulianVilla26

Image: Wikipedia

Computer

- Early Usage:
  - (Richard Brathwait, 1613) “I [have] read the truest computer of Times, and the best Arithmetician that [ever] breathed, and he reduceth thy [days] into a short number.”
  - A person who carried out calculations or computations
  - By 1943, most computers were women (Erika Smith, 2013)
- For us:
  - A tool with a defined set of operations
    - Arithmetic and Logical operations
    - Storage, Interfaces (Keyboard, Screen, Mouse, Network, Camera, Mic, Speaker, ..)

COL100

- Learn how to precisely specify computational problem
- Learn simple solution (algorithms) and their design
  - Understand the requirement (+ assumptions), and available operations
  - Continue to completion (⇒ must complete, produce output)
  - Argue that the results are always correct, and produced “quickly”

What steps on what data:
Input ➧ Operations ➧ Output

- Translate algorithms into programs
  - Start with high-level expression
  - Know typical constructs for common steps (and sequences of steps)
  - Verify correctness, speed

Should be easy and secure to use. And to modify.

Learning Goals

Will be tested

- Be able to formulate problems in a precise and concise manner
- Be able to design correct and efficient algorithms solving the problem
  - Mostly using standard algorithmic components
- Be able to categorize cases about and actions on abstract items
- Be able to recognize program and data constructs suitable for algorithm steps
- Be able to write the required program in Python (at least)
- Be able to test, debug, and modify programs
- Be able to evaluate problem formulation, algorithm, and program

References

See Moodle

- Think Python by A. Downey
  - https://greenteapress.com/wp/think-python-2e
- How to solve by computers by RG Dromey
  - https://www.edutechlearners.com/download/books/How%20To%20Solve%20It%20By%20Computer.pdf
- MITx: Introduction to Computer Science and Programming Using Python
  - https://www.edx.org/learn/computer-science/massachusetts-institute-of-technology-introduction-to-computer-science-and-programming-using-python
- SAK's notes: https://www.cse.iitd.ac.in/~sak/courses/ics/ics-2013.pdf

Keys to Success

- Ask questions — repeatedly
    - In class, In lab, online, outside class
    - Seek help (Instructors and TAs)
- Practice programming in the lab and outside
    - Learn to recognize common errors (and parse error messages)
        - Try again and again .. (sometimes a short break helps)
- Attend, Take notes, Review slides and in-class programs
    - Be regular — Catching up later is harder
- Do not miss emails; follow instructions carefully **Keep your institute login ID working**