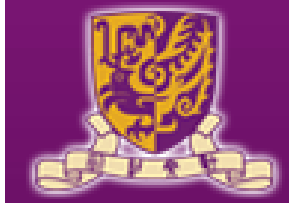


香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

Introduction to Computer Science: Programming Methodology

Lecture 1 Introduction

Tongxin Li

School of Data Science

Who I am (Tongxin Li)

Background

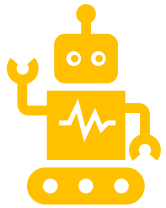
Education: CUHK, Caltech

Working Experience:

- Amazon Web Services (2020, 2021), Applied Scientist Intern
- Assistant Professor (2022-present): SDS, CUHK-SZ

Contact

Email: litongxin@cuhk.edu.cn



Research

General areas: Machine Learning, Control, Sustainability

Topics: control & optimization, online algorithms, reinforcement learning

About this course

- This course is a required course for all SDS students.

- Need to sync between other 5 sessions (*Lecture slides, assignments, exams are the same*)

Learning Objectives

- This course introduces the basics of computer programming using Python

- Students will learn the basic elements of modern computer systems, key programming concepts, problem solving and basic algorithm design

A Message for Freshmen:

- University courses are very different from what you might have been familiar with in your high schools.
  - Languages
  - Assignments
  - Exams

- In your future university life, there are no more 班主任 (head teachers)
  - No one is watching you to finish the assignments
  - It is the right time to start being mature/independent
  - Make best use of tutorials (starting next week)
  - Check your e-mails often (say, once per day)
  - Assignments and important course announcements will be sent out via emails

Key Topics

- Introduction to modern computers
- Preliminary knowledge for computer programming
- Basic introduction to Python language
- Data types and operators in Python language
- Input/output
- Flow control and loop
- Function
- List
- Basic data structure
- Introduction to algorithm design
- Introduction to object oriented programming

Assessment

Assignments × 4	10% × 4
Mid-term quiz	20%
Final exam	40%

Course Materials

- All lecture notes and sample code used in classes will be provided to students via Blackboard (bb.cuhk.edu cn). They can also be found in the course web.

- Recommended readings
  - Online resources: https://www.python.org/doc/
  - Learning Python, 5th Edition, by Mark Lutz, Publisher: O'Reilly media

Course Components

Activity	Hours/week
Lecture	3 * 14
Tutorial	1 * 14

Indicative Teaching Plans

Week	Content/ topic/ activity
1	Introduction to modern computers; Preliminary knowledge for computer programming;
2	Basic introduction to Python language; Data types and operators in Python language; Input/output;
3	Flow control and loop;
4	Function;
5	List;
6	Introduction to object oriented programming, part I
7	Review for mid-term quiz;
8	Introduction to object oriented programming, part II
9	Data Structure, part I;
10	Data Structure, part II;
11	Introduction to algorithm design, part I;
12	Introduction to algorithm design, part II;
13	Introduction to algorithm design, part III;
14	Review for final exam;

Course Web:

www.tongxin.me/CSC1001-2025fall/

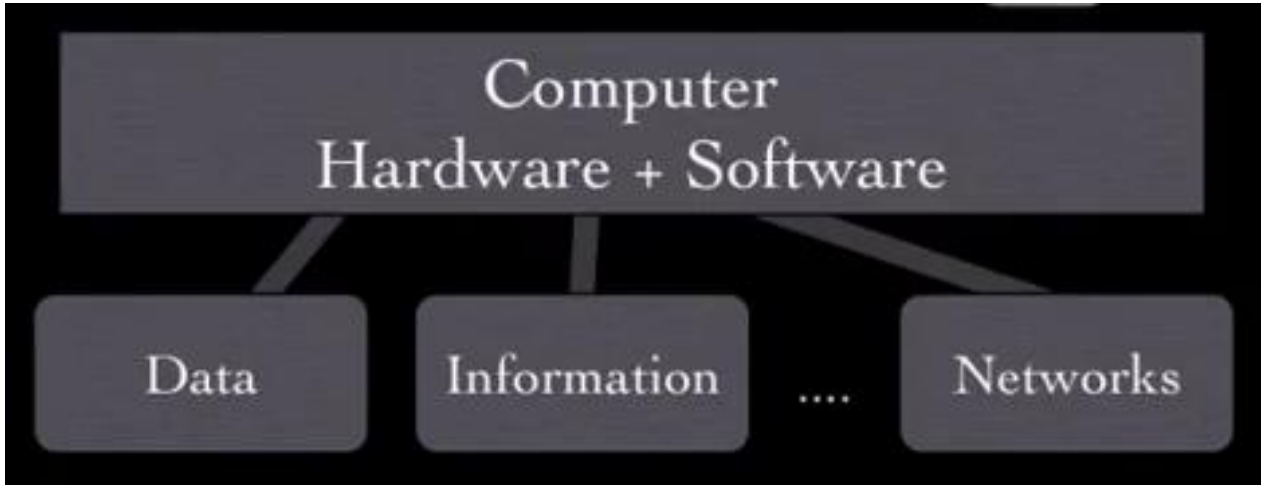
Personal Web: *www.tongxin.me*

Why learn programming?

- Computer is built to help people solve problems
- Computer does not understand what we say
- We need to communicate with computers using their languages (computer programming language)
- Assembly, C, C++, Java and Python

User

Interface



Programmer

- Programmers solve problems like data, information, networks on behalf of users

Programmer

- Professional programmer writes computer programs and develops software
- A junior programmer gets a salary of 10-30k RMB in an INTERNET company like Tencent
- A programmer can earn up to 500k – 1m USD in Google!!
- Software and INTERNET are huge industries.

Programmer

- Professional programmer writes computer programs and develops software

- ~~A junior programmer gets a salary of 10-30k RMB in an INTERNET company like Tencent~~

- ~~A programmer can earn up to 500k – 1m USD in Google!!~~

- Software and INTERNET are still huge industries?

Programmer

- Professional programmer writes computer programs and develops software

- ~~A junior programmer gets a salary of 10-30k RMB in an INTERNET company like Tencent~~

- ~~A programmer can earn up to 500k – 1m USD in Google!!~~

- Software and INTERNET are still huge industries?

Goodbye, $165,000 Tech Jobs. Student Coders Seek Work at Chipotle.

As companies like Amazon and Microsoft lay off workers and embrace A.I. coding tools, computer science graduates say they're struggling to land tech jobs.

The New Trend

BENZINGA

Anthropic CEO Says AI Could Write '90% Of Code' In '3 To 6 Months'—Warns 'Every Industry' Will Be Affected

LaToya Scott
March 27, 2025 • 3 min read

AI could soon write 90% of software code in as few as three to six months, Anthropic CEO **Dario Amodei** said at a Council on Foreign Relations event on March 10. He added that within 12 months, nearly all coding tasks might be handled by AI. Human developers, however, will still be needed to provide design inputs and set operational parameters for these models.

TECH

Satya Nadella says as much as 30% of Microsoft code is written by AI

PUBLISHED TUE, APR 29 2025•9:33 PM EDT | UPDATED TUE, APR 29 2025•9:58 PM EDT

Jordan Novet
@JORDANNOVET

Jonathan Vanian
@IN/JONATHAN-VANIAN-B704432/

SHARE

KEY POINTS

- Microsoft CEO Satya Nadella on Tuesday said that as much as 30% of the company's code is now written by artificial intelligence.

What Should We Do?

The AI Revolution: Our Generation's 'Dot-Com' Moment

Just like the internet boom of the early 2000s, AI is creating a new frontier of opportunity. The landscape is changing, but the core skills for success are more important than ever.

Why Coding is Still Your Key in the Age of AI?

Aim Higher: AI develops code. You develop the AI.

Get Hired: Prove your problem-solving skills in technical interviews.

Work Smarter: Need knowledge to prompt and debug for more effective vibe coding.

What is Code? Software? Program?

- A sequence of instructions
- Computers take the instructions and execute them
- It is a little piece of our intelligence in the computer
- Intelligence which is re-usable

Computers are good at following instructions

- Humans can easily make mistakes when following a set of instructions

- On the contrary, computers (usually) won’t make mistakes, regardless of they are given 10 or 10 billion instructions !!

Computers

Are they computers ?

calculator

router

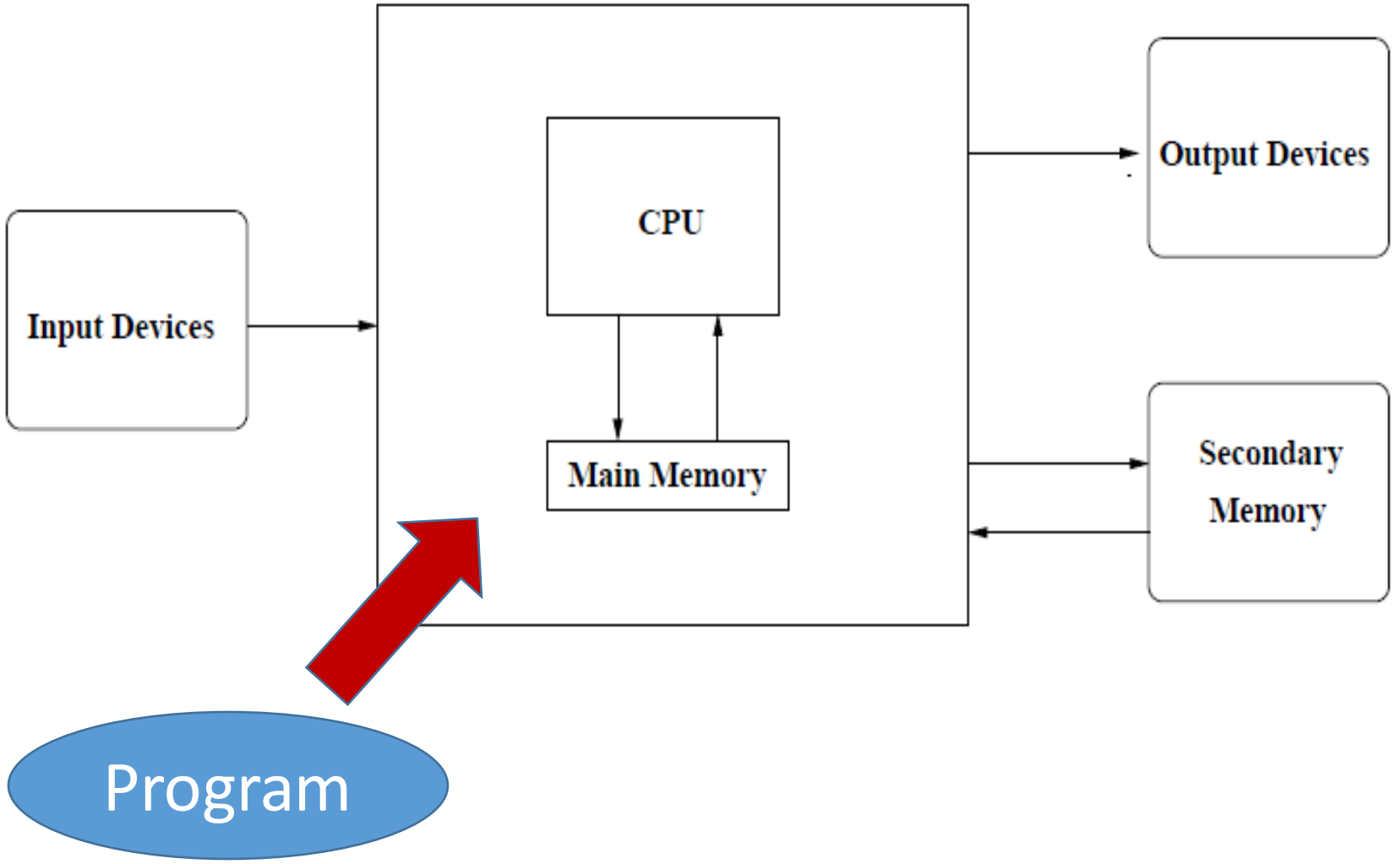
robot

smartwatch

Smart TV

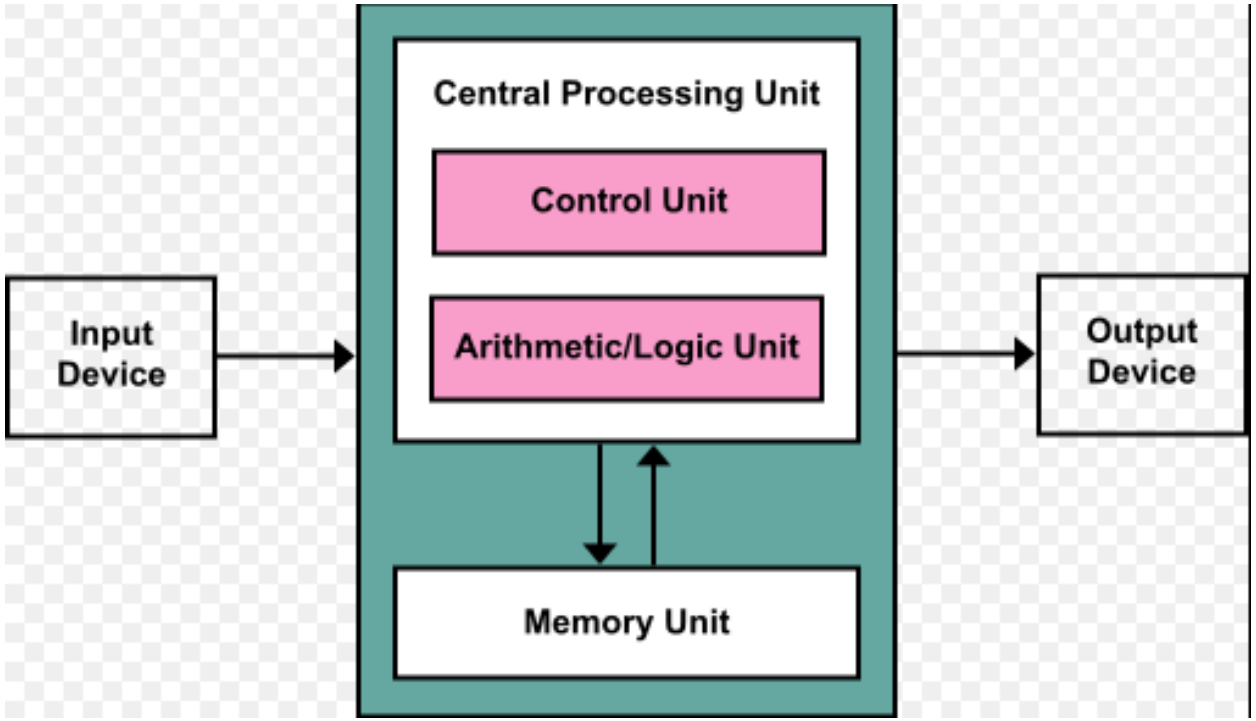
Smart glasses

Computer Hardware

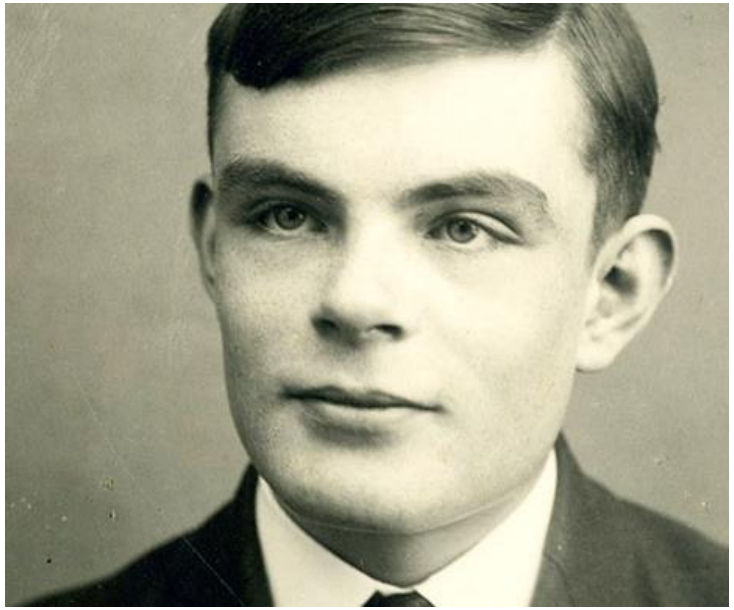


Von Neumann Architecture

- The modern computer architecture is proposed by John Von Neumann



The theoretical foundation of computer science

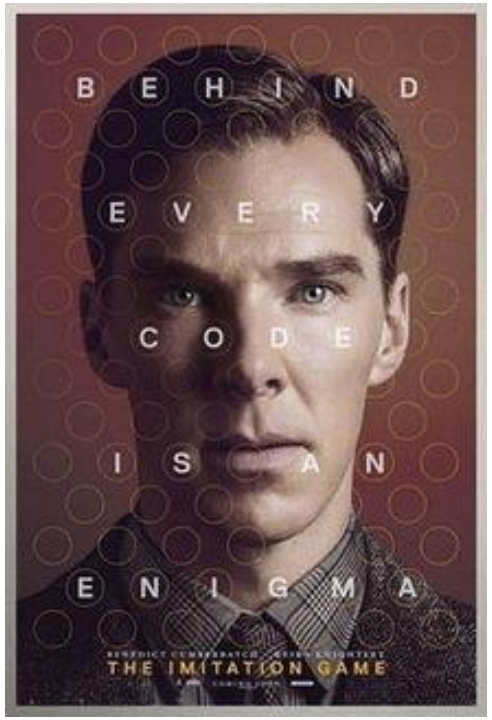


- The theoretical foundation of computer science are built by Alan Turing

- Father of theoretical computer science and artificial intelligence

- Computability theory and Turing test

A movie about Turing



模仿遊戲
The Imitation Game

Also another similar movie about John Nash:　A beautiful mind (美丽心灵)

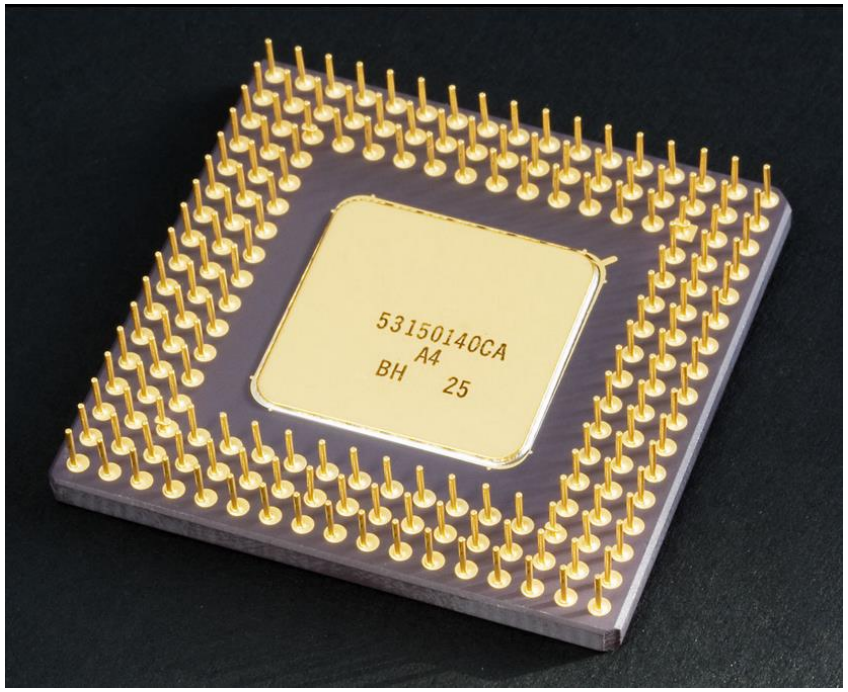
Key components in a computer

- Central processing unit (CPU): execute your program. Similar to human brain, very fast but not that smart
- Input device: take inputs from users or other devices
- Output device: output information to users or other devices
- Main memory: store data, fast and temporary storage
- Secondary memory: slower but large size, permanent storage

Central Processing Unit

- A processor contains two units, a control unit (CU) and an arithmetic/logic unit (ALU)
- CU is used to fetch commands from the memory
- ALU contains the electric circuits which can execute commands

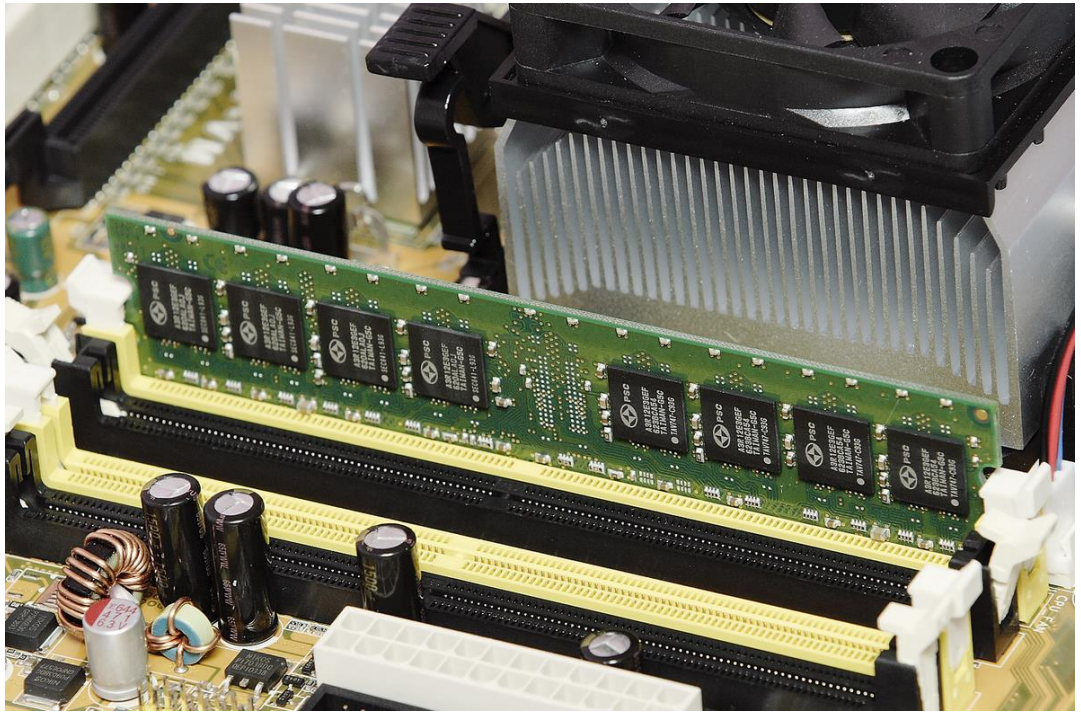
Central Processing Unit



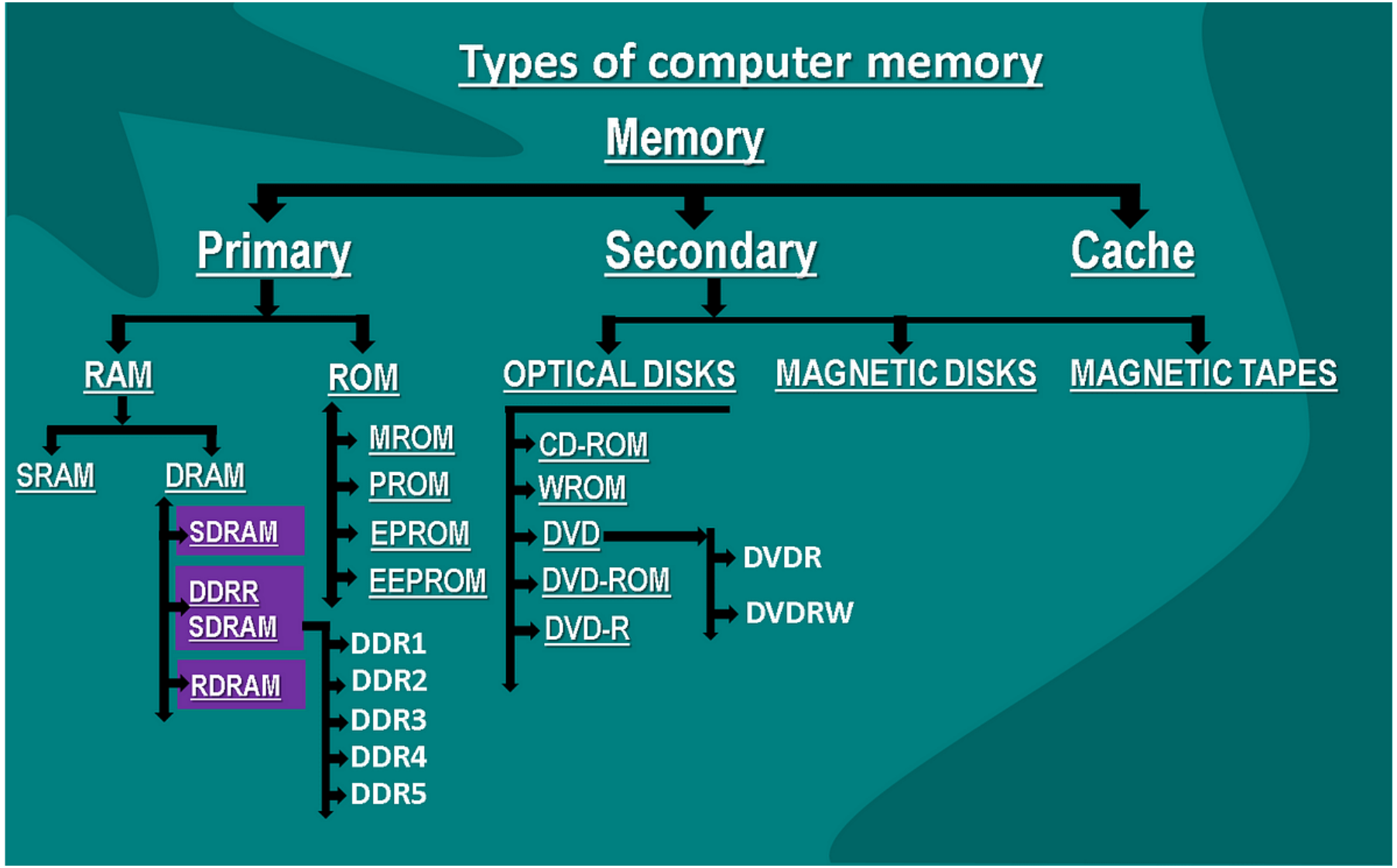
- Processor manufacturer: Intel, AMD, ARM, etc

Memory/Storage

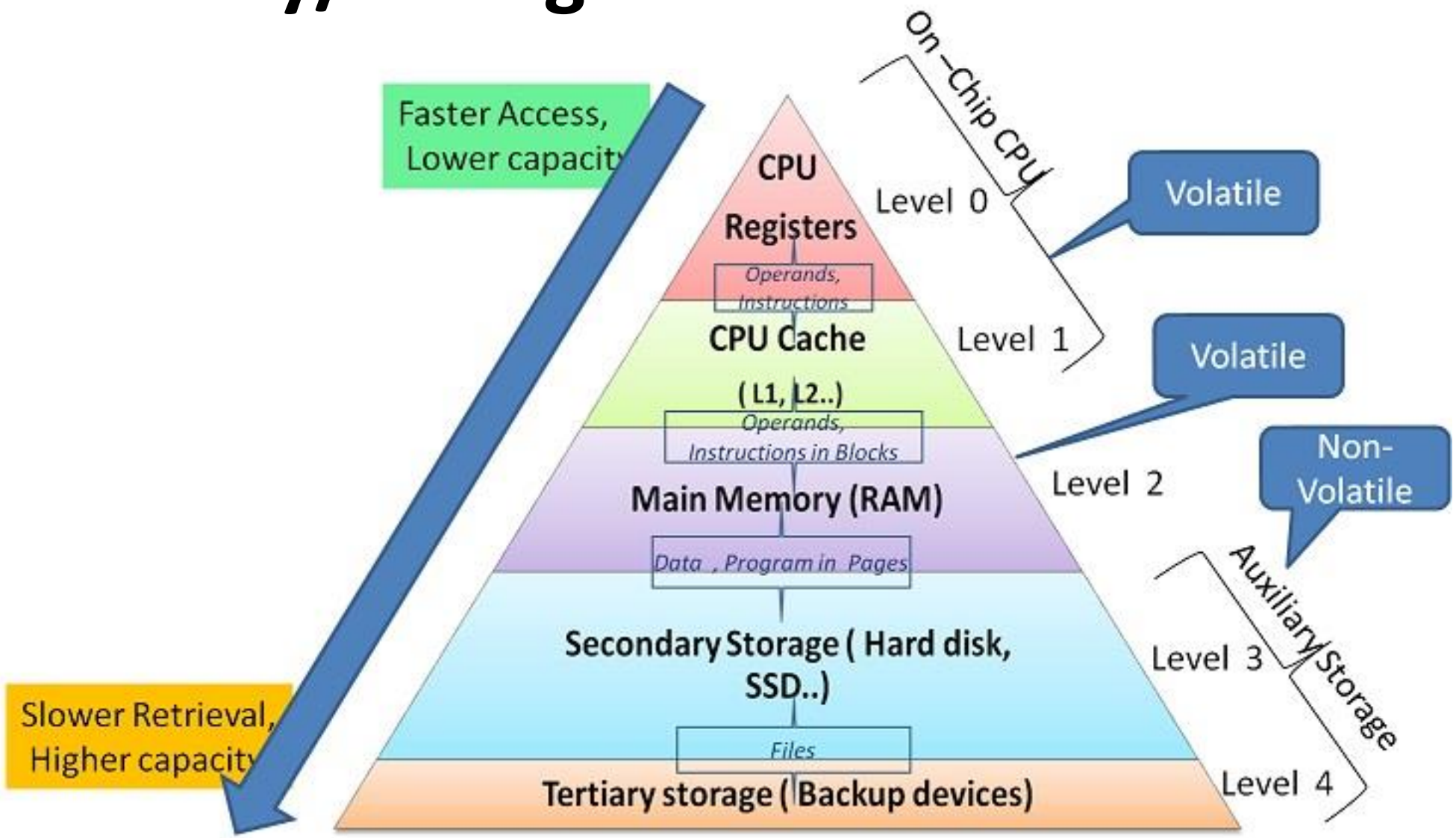
- High speed cache
- ROM
- RAM
- Flash
- Hard disk



Memory/Storage



Memory/Storage



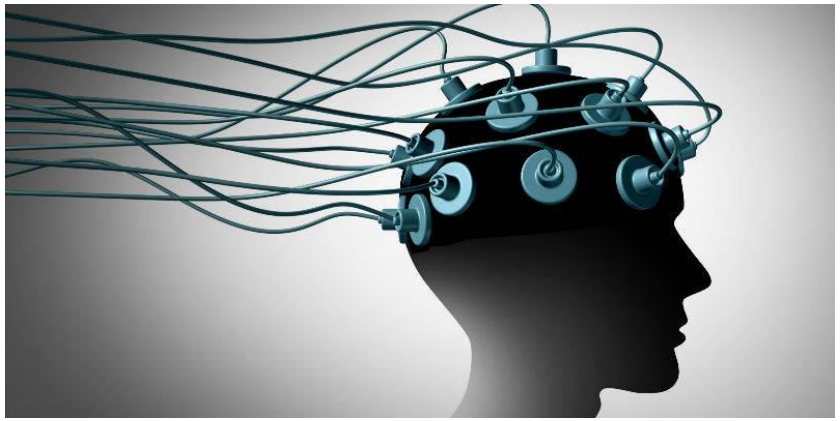
Input/output devices

- Input devices: mouse, keyboard, panel, touch screen, audio input, mind reading, etc

- Output devices: screen, audio output, etc

Any other input devices?

Any other input devices?



Any other input devices?

Any other output devices?

VR

Holographic projection

How the hard disk works

http://v.youku.com/v_show/id_XNjA4NzMxNDk2.html?from=s1.8-1-1.2

What can a computer actually understand?

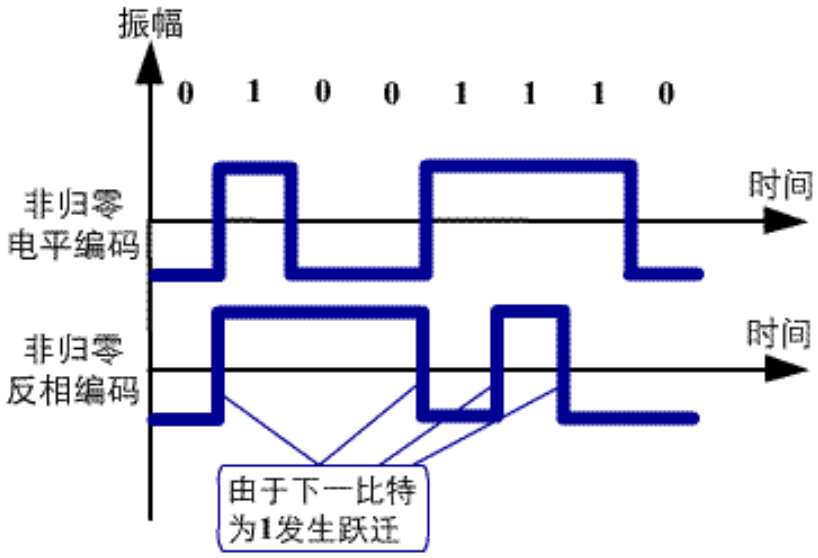
- The computers used nowadays can understand only binary number (i.e. 0 and 1)
- Computers use voltage levels to represent 0 and 1
- NRZL and NRZI coding
- The instructions expressed in binary code is called machine language

0 0 0 1 numerical value 2^0

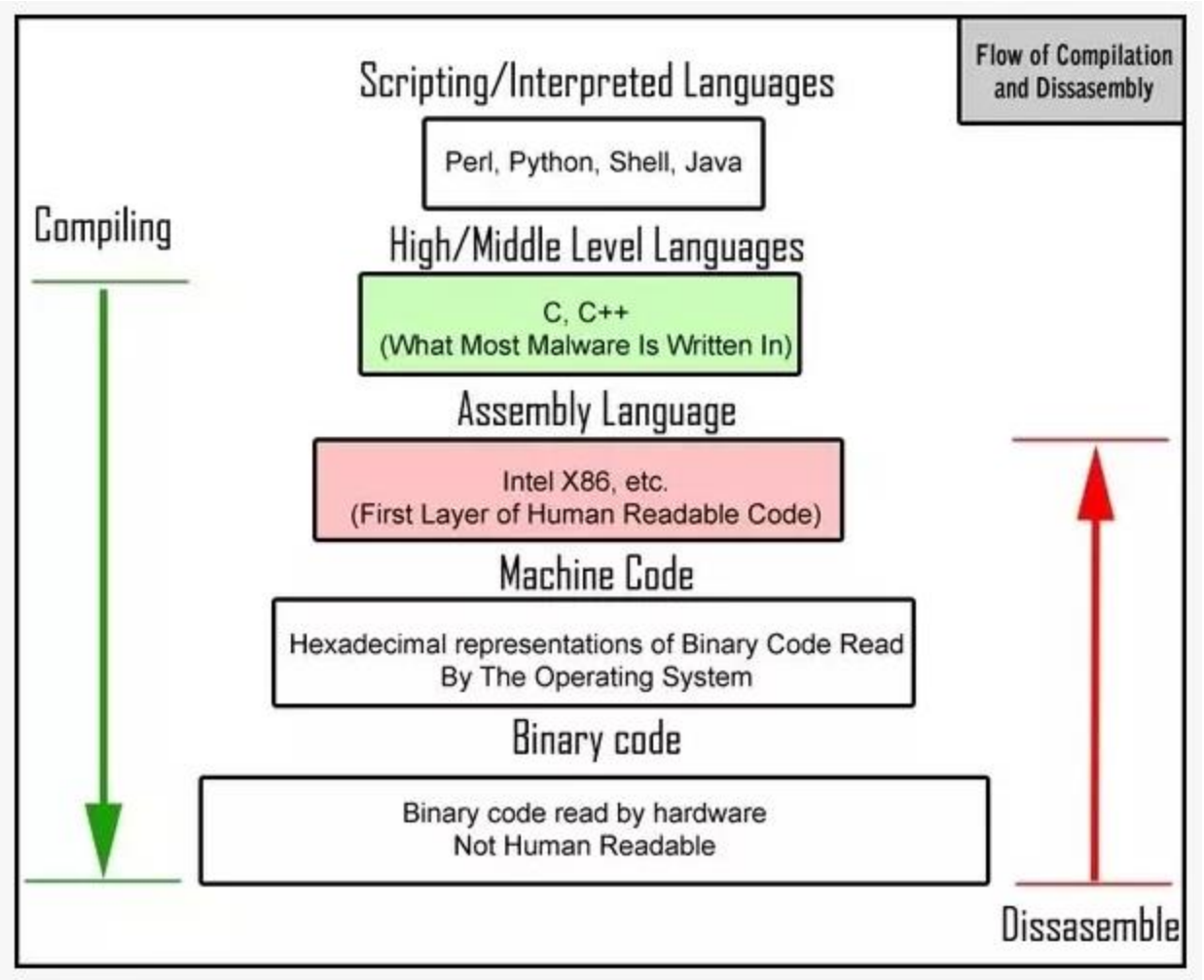
0 0 1 0 numerical value 2^1

0 1 0 0 numerical value 2^2

1 0 0 0 numerical value 2^3



Programing Language



https://www.quora.com/I-am-an-11th-grader-I-find-it-quite-difficult-to-write-C++-code-especially-when-the-only-way-to-practice-is-to-solve-maths-problems-Should-I-keep-learning-C++-or-drop-it-for-C

Review of last lecture

- **Von Neumann Architecture**
- **CPU and memory**
- **Input devices and output devices**

Low level language – Assembly Language

- An assembly language is a low-level programming language, in which there is a very strong (generally one-to-one) correspondence between the language and machine code instructions.

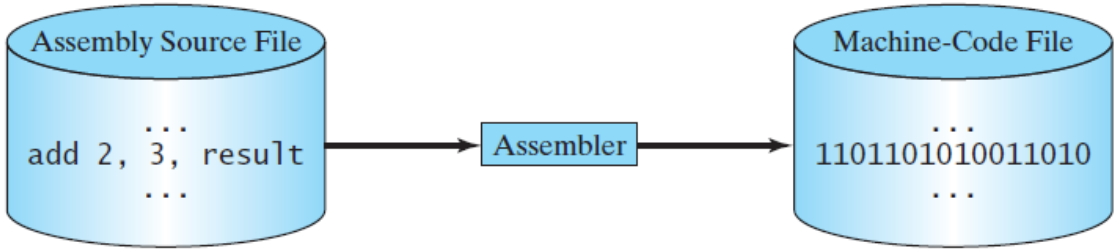
- Each assembly language is specific to a particular computer architecture

- Assembly language is converted into executable machine code by a utility program referred to as an assembler

```
                        ****************************************
                        * FUNCTION: INHEX - INPUT HEX DIGIT
                        * INPUT: none
                        * OUTPUT: Digit in acc A
                        * CALLS: INCH
                        * DESTROYS: acc A
                        * Returns to monitor if not HEX input

C01E 8D F0      INHEX   BSR     INCH        GET A CHAR
C020 81 30              CMP A   #'0         ZERO
C022 2B 11              BMI     HEXERR      NOT HEX
C024 81 39              CMP A   #'9         NINE
C026 2F 0A              BLE     HEXRTS      GOOD HEX
C028 81 41              CMP A   #'A
C02A 2B 09              BMI     HEXERR      NOT HEX
C02C 81 46              CMP A   #'F
C02E 2E 05              BGT     HEXERR
C030 80 07              SUB A   #7          FIX A-F
C032 84 0F      HEXRTS  AND A   #$0F        CONVERT ASCII TO DIGIT
C034 39                 RTS

C035 7E C0 AF   HEXERR  JMP     CTRL        RETURN TO CONTROL LOOP
```



C Language (1969 - 1973)

- C was developed by Dennis Ritchie between 1969 and 1973 at AT&T Bell Labs
- One of the early high-level programming language
- Somewhere between assembly and other high level languages
- Provide powerful functionalities for low level memory manipulations
- Have the highest efficiency within high level languages
- Very widely used in low level applications, such as operating systems, embedded programming, super computers, etc

C++ Language (1979)

- C++ was developed by Bjarne Stroustrup at Bell Labs since 1979
- Inherent major features of C
- An object oriented programming language, supporting code reuse
- High efficiency and powerful in low level memory manipulation
- Still could be platform dependent

Java Language (1995)

- Java was developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995
- A new generation of general-purpose object oriented programming language
- Platform independent, “write once, run anywhere” (WORA)
- Java is one of the most popular programming languages currently in use

Popular Java Software?

Popular Java Software?

Most games use C++

Python (1991)

- Developed by Guido van Rossum in 1989, and formally released in 1991
- An open source, object oriented programming language
- Powerful libraries
- Powerful interfaces to integrate other programming languages (C/C++, Java, and many other languages)
- In AI research, people mainly use Python.

Popular Python Software?

Popular Python Software?

Do they use Python 100%?

Popular Python Software?

Python (1991)

- **Python is evolving …**

- The best way to keep track of the updates is to learn by really using Python (not by taking lectures)

From https://www.python.org/doc/versions/

Python Documentation by Version

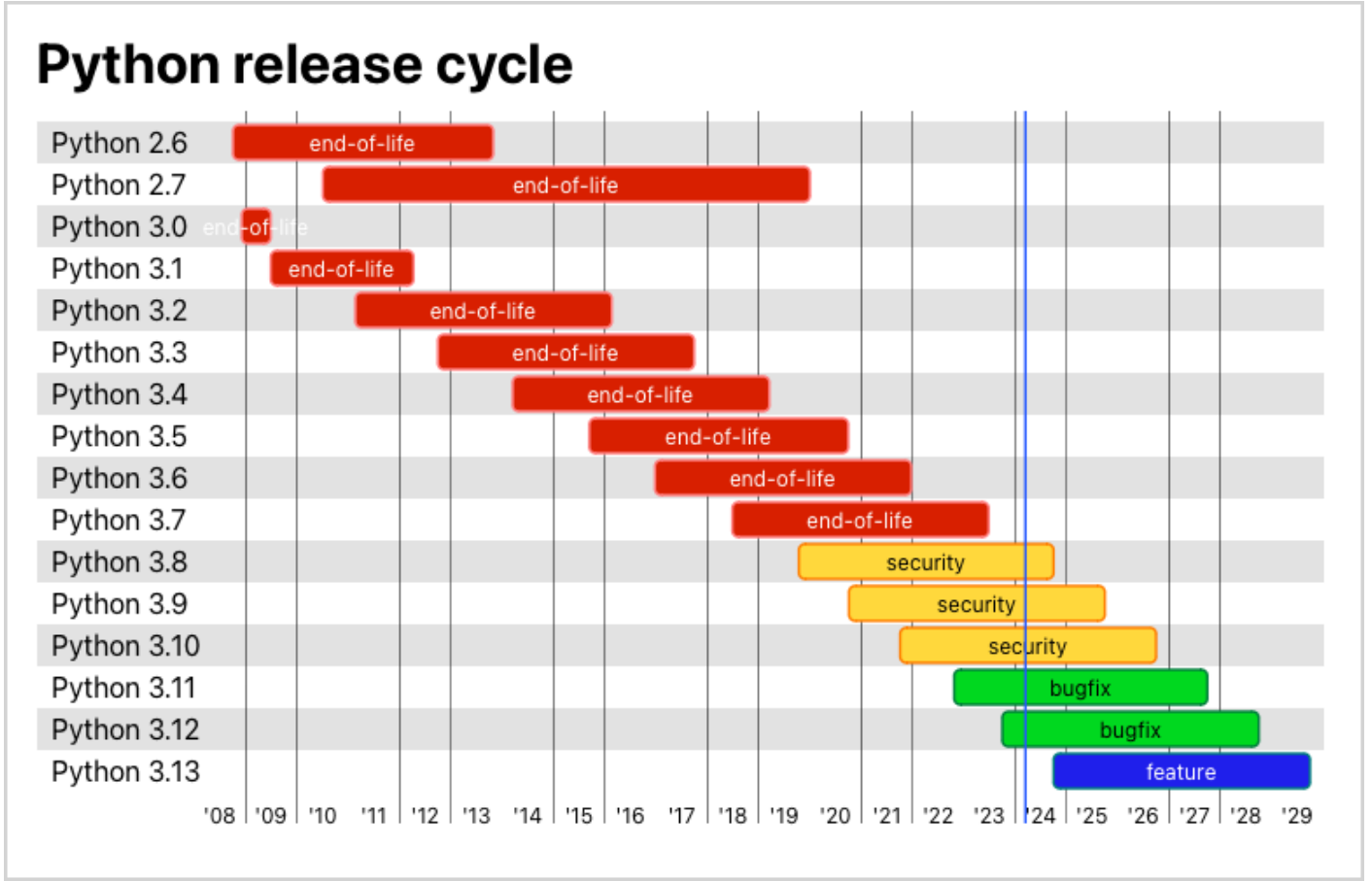
Some previous versions of the documentation remain available online. Use the list below to select a version to view.

For unreleased (in development) documentation, see In Development Versions.

- Python 3.12.5, documentation released on 6 August 2024.
- Python 3.12.4, documentation released on 6 June 2024.
- Python 3.12.3, documentation released on 9 April 2024.
- Python 3.12.2, documentation released on 6 February 2024.
- Python 3.12.1, documentation released on 8 December 2023.
- Python 3.12.0, documentation released on 2 October 2023.
- Python 3.11.9, documentation released on 2 April 2024.
- Python 3.11.8, documentation released on 6 February 2024.
- Python 3.11.7, documentation released on 4 December 2023.
- Python 3.11.6, documentation released on 2 October 2023.
- Python 3.11.5, documentation released on 24 August 2023.
- Python 3.11.4, documentation released on 6 June 2023.
- Python 3.11.3, documentation released on 5 April 2023.
- Python 3.11.2, documentation released on 8 February 2023.
- Python 3.11.1, documentation released on 6 December 2022.
- Python 3.11.0, documentation released on 24 October 2022.
- Python 3.10.14, documentation released on 19 March 2024.
- Python 3.10.13, documentation released on 24 August 2023.
- Python 3.10.12, documentation released on 6 June 2023.
- Python 3.10.11, documentation released on 5 April 2023.
- Python 3.10.10, documentation released on 8 February 2023.
- Python 3.10.9, documentation released on 6 December 2022.
- Python 3.10.8, documentation released on 8 October 2022.

Python (1991)

- **Python is evolving …**
- **The goal of this course:**
  - ~~Keep track of recent updates~~ ❌
  - Provide you a comprehensive knowledge base of programming via Python ✓
  - Our students have very diverse backgrounds …



Language efficiency v.s. development efficiency

- High level languages cannot be executed directly
- High level languages must be converted into low level languages first
- Lower level languages have higher language efficiency (they are faster to run on a computer)
- Higher level languages have higher development efficiency (it is easier to write programs in these languages)

Operating Systems

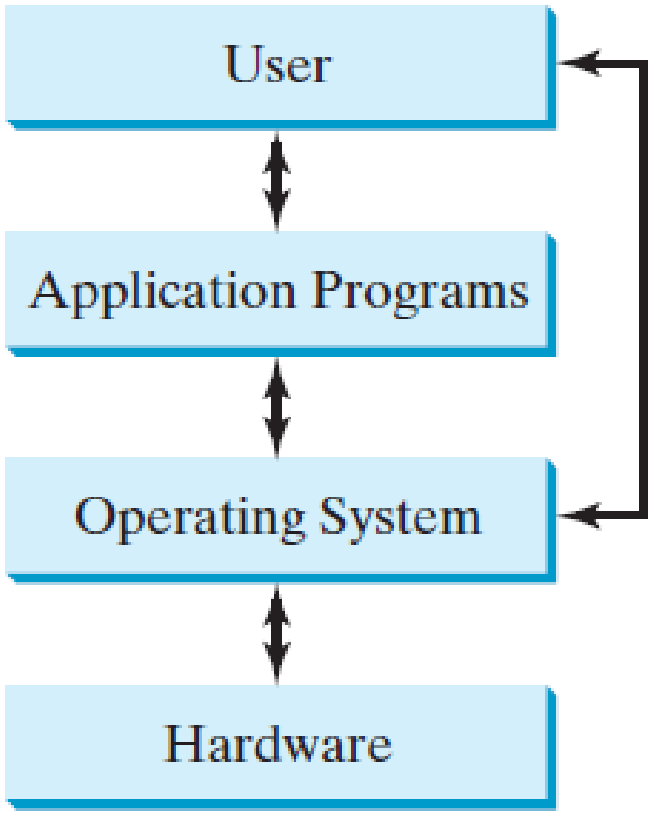
- The operating system (OS) is a low level program, which provides all basic services for managing and controlling a computer's activities

- Applications are programs which are built based upon an OS

- Main functions of an OS:

✓ Controlling and monitoring system activities
✓ Allocating and assigning system resources
✓ Scheduling operations

- Popular OS: Windows, Mac OS, Linux, iOS, Android…



Slogan for Python

Life is short, use Python

Data Representation and Conversion

- We use positional notation (进位记数法) to represent or encode numbers in a computer
- Data are stored essentially as binary numbers in a computer
- In practice, we usually represent data using either binary (二进制), decimal (十进制), octal (八进制) or hexadecimal (十六进制) number systems
- We may need to convert data between different number systems

The basic idea of positional notation

- Each positional number system contains two elements, a base (基数) and a set of symbols
- Using the decimal system (十进制系统) as an example, its base is 10, and the symbols are {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
- When a number “hits” 9, the next number will not be a different symbol, but a “1” followed by a “0” (逢十进一)

Decimal number system

- In the decimal number system, the base is 10, the symbols include 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Every number can be decomposed into the sum of a series of numbers, each is represented by a positional value times a weight
- $N = a_n \times 10^n + a_{n-1} \times 10^{n-1} + a_{n-2} \times 10^{n-2} \ \ldots\ldots + a_0 \times 10^0 + a_{-1} \times 10^{-1} + a_{-2} \times 10^{-2} \ldots$
- a_n is the positional value (ranging from 0 to 9), while 10^n represents the weight

Binary number system

- In the binary system, the base is 2, we use only two symbols 0 and 1
- “10” is used when we hit 2 (逢二进一)
- $N = a_n \times 2^n + a_{n-1} \times 2^{n-1} + a_{n-2} \times 2^{n-2} \ \ldots\ldots + a_0 \times 2^0 + a_{-1} \times 2^{-1} + a_{-2} \times 2^{-2} \ldots$
- a_n is the positional value (ranging from 0 to 1), while 2^n represents the weight

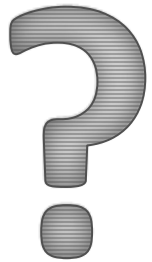
Why use binary number?

- Easy to implement physically
- Simple calculation rules
- Easy to combine arithmetic and logic operations
- Against noise (for analog signal)

Hexadecimal number system

- In the hexadecimal system, the base is 16, we use 16 symbols {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f}
- “10” is used when we hit 16 (逢十六进一)
- $N = a_n \times 16^n + a_{n-1} \times 16^{n-1} + a_{n-2} \times 16^{n-2} \ \ldots\ldots + a_0 \times 16^0 + a_{-1} \times 16^{-1} + a_{-2} \times 16^{-2} \ldots$
- a_n is the positional value (ranging from 0 to 15), while 16^n represents the weight

Octal number system



Converting binary number into decimal number

Example $(1101.01)_2$

$$=(1\times2^3+1\times2^2+0\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2})_{10}$$

$$=(13.25)_{10}$$

Practice $(10110.11)_2=(?)_{10}$

Converting binary number into decimal number

Answer

(10110.11)

$=(1\times2^4+0\times2^3+1\times2^2+1\times2^1+0\times2^0+1\times2^{-1}+1\times2^{-2})_{10}=(22.75)_{10}$.

Converting octal number into decimal number

Example $(24.67)_8=(2 \times 8^1+ 4 \times 8^0+6 \times 8^{-1}+7 \times 8^{-2})_{10}$

$=(20.859375)_{10}$

Practice $(35.7)_8= (?)_{10}$

Converting octal number into decimal number

Answer

$$(35.7)_8=(3\times8^1+5\times8^0+7\times8^{-1})_{10}$$
$$=(29.875)_{10}$$

Converting hexadecimal number into decimal number

Example $(2AB.C)_{16}$

$$=(2\times16^{2}+10\times16^{1}+11\times16^{0}+12\times16^{-1})_{10}$$

$$=(683.75)_{10}$$

Practice $(A7D.E)_{16}=(?)_{10}$

Converting hexadecimal number into decimal number

Answer

$$(A7D.E)_{16}=(10\times16^{2}+7\times16^{1}+13\times16^{0}+14\times16^{-1})_{10}$$
$$=(2685.875)_{10}$$

Converting other number system into decimal system

- Other number system can also be converted into decimal system in a similar way

- We just need to change the corresponding base

Some tests: converting into **decimal** system

- $(110110)\_2 = (?)\_10$
- $(101011.11)\_2 = (?)\_10$
- $(120)\_8 = (?)\_10$
- $(34.01)\_8 = (?)\_10$
- $(\mathrm{BCA})\_16 = (?)\_10$
- $(\mathrm{E05.C})\_8 = (?)\_10$

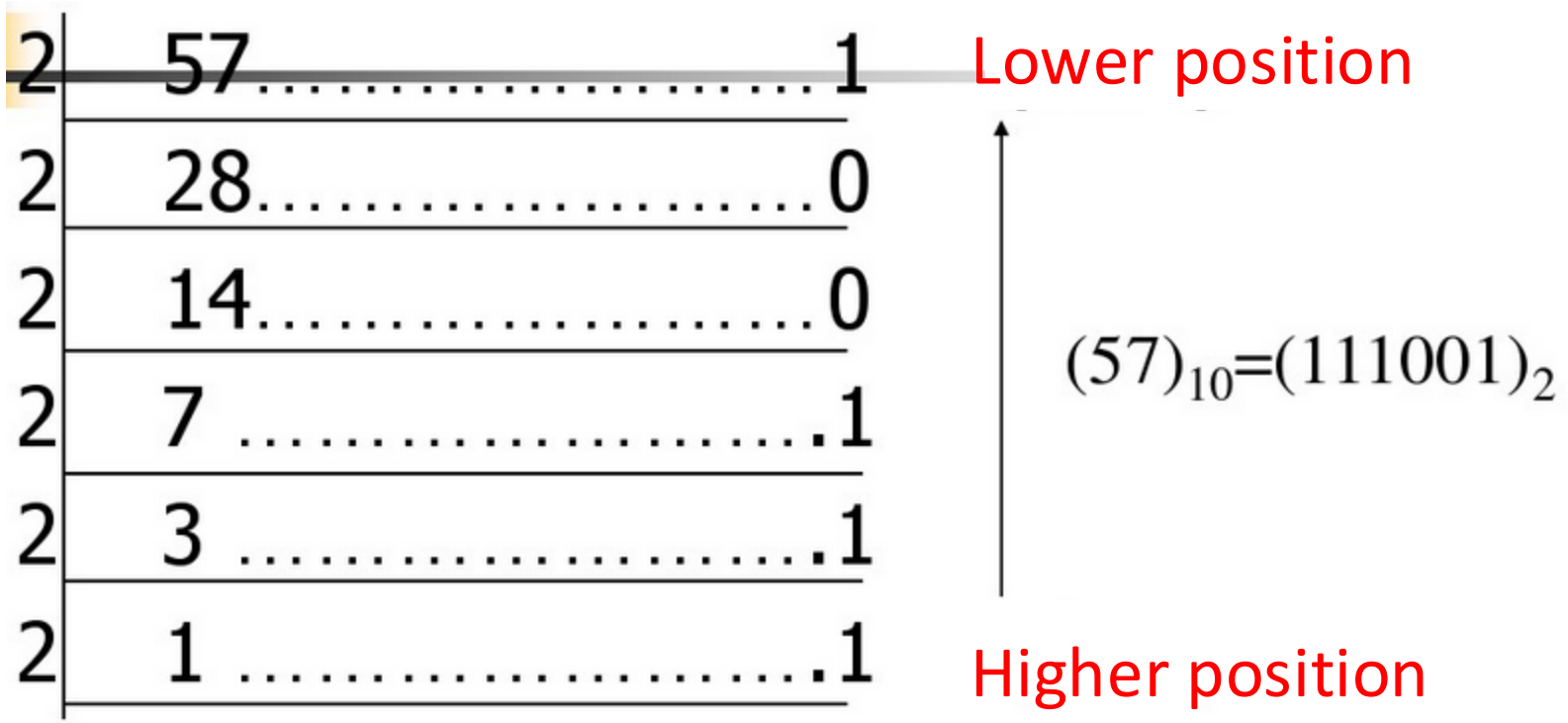
Some tests: converting into **decimal** system

- $(110110)\_2 = (\mathbf{118})\_10$
- $(101011.11)\_2 = (\boldsymbol{43.75})\_10$
- $(120)\_8 = (\mathbf{80})\_10$
- $(34.01)\_8 = (\boldsymbol{28.015625})\_10$
- $(\mathrm{BCA})\_16 = (\boldsymbol{3018})\_10$
- $(\mathrm{E05.C})\_8 = (\boldsymbol{3589.75})\_10$

https://www.rapidtables.com/convert/number/hex-to-decimal.html

Converting decimal integer into binary integer

Example: $(57)_{10} = (?)_2$



Converting decimal fraction into binary fraction

How to convert fractions to binary?

STEP 1: Take a decimal fraction and start multiplying by two the decimal part.

STEP 2: Every time the result is smaller than 1 , add a 0 to the binary representation. If the result is greater or equal to 1 , add a 1 to the binary representation and subtract 1 from the multiplication result.

Converting decimal fraction into binary fraction

Example: $(0.875)_{10} = (?)_2$

Higher position

$0.875 \times 2 = \mathbf{1}.75$ Integer part: **1**

$0.75 \times 2 = \mathbf{1}.5$ Integer part: **1**

$0.5 \times 2 = \mathbf{1}$ Integer part: **1**

Lower position

Answer: $(0.875)_{10} = (0.111)_2$

Practice: $(0.6875)_{10} = (?)_2$

Converting decimal fraction into binary fraction

Answer:

$0.6875 \times 2 = \mathbf{1}.375$ Integer part: **1**

$0.375 \times 2 = \mathbf{0}.75$ Integer part: **0**

$0.75 \times 2 = \mathbf{1}.5$ Integer part: **1**

$0.5 \times 2 = \mathbf{1}$ Integer part: **1**

Higher position

Lower position

So, $(0.6875)_{10} = (0.1011)_2$

Converting decimal number into binary number

- For a decimal number that has both integer and fractional parts
- Convert the integer and fractional parts separately
- Example: $(215.3125)_{10} = (?)_2$

Converting decimal number into binary number

Answer:

$$(215)_{10} = (11010111)_2$$
$$(0.3125)_{10} = (0.0101)_2$$
$$(215.3125)_{10} = (11010111.0101)_2$$

The one-to-one relationship between binary and octal numbers

There is a “one-to-one” (一一对应) relationship between three digits binary number and one digit octal number

$$
\begin{aligned}
(0)_8 &= (000)_2 \\
(1)_8 &= (001)_2 \\
(2)_8 &= (010)_2 \\
(3)_8 &= (011)_2 \\
(4)_8 &= (100)_2 \\
(5)_8 &= (101)_2 \\
(6)_8 &= (110)_2 \\
(7)_8 &= (111)_2
\end{aligned}
$$

Converting octal number into binary number

- Convert each octal digit into binary number of three digits
- Keep the digit order unchanged
- Example: $(0.754)_8 = (?)_2$

$$(0.754)_8 = (\underline{000}.\underline{111}\ \underline{101}\ \underline{100})_2$$
$$= (0.1111011)_2$$

- Practice: $(16.327)_8 = (?)_2$

Converting octal number into binary number

Answer:

$$(16.327)_8$$
$$= (\underline{001\ 110}.\ \underline{011\ 010\ 111})_2$$
$$= (1110.011010111)_2$$

Converting hexadecimal number into binary number

- Convert each hexadecimal digit into binary number of four digits

- Keep the digit order unchanged

- Example: $(4C.2E)_{16} = (?)_2$

$$
\begin{aligned}
&(4C.2E)_{16} \\
&= (\underline{0100}\ \underline{1100}.\underline{0010}\ \underline{1110})_2 \\
&= (1001100.0010111)_2
\end{aligned}
$$

- Practice: $(AD.7F)_{16} = (?)_2$

Converting hexadecimal number into binary number

Answer:

$$(AD.7F)_{16}$$

$$= (\underline{1010}\ \underline{1101}.\underline{0111}\ \underline{1111})_{2}$$

$$= (10101101.01111111)_{2}$$

Converting binary number into octal number

- Starting from lower positions, convert every three digits of the integer part into an octal digit
- When there is not enough higher positions in the integer part, fill with 0
- Starting from higher positions, convert every three digits of the fractional part into an octal digit
- When there is not enough lower positions in the fractional part, fill with 0
- Keep the digit order unchanged

Converting binary number into octal number

Example:

$(0.10111)_2 = (\underline{000}.\ \underline{101}\ \underline{110})_2 = (0.56)_8$

$(11101.01)_2 = (\underline{011}\ \underline{101}.\ \underline{010})_2 = (35.2)_8$

Practice:

$(1101101.011)_2$

Converting binary number into octal number

Answer:

$$(1101101.011)_2 = (\underline{001}\ \underline{101}\ \underline{101}.\ \underline{011})_2$$
$$= (155.3)_8$$

Converting binary number into hexadecimal number

- Starting from lower positions, convert every four digits of the integer part into an octal digit
- When there is not enough higher positions in the integer part, fill with 0
- Starting from higher positions, convert every four digits of the fractional part into an octal digit
- When there is not enough lower positions in the fractional part, fill with 0
- Keep the digit order unchanged

Converting binary number into hexadecimal number

Example:

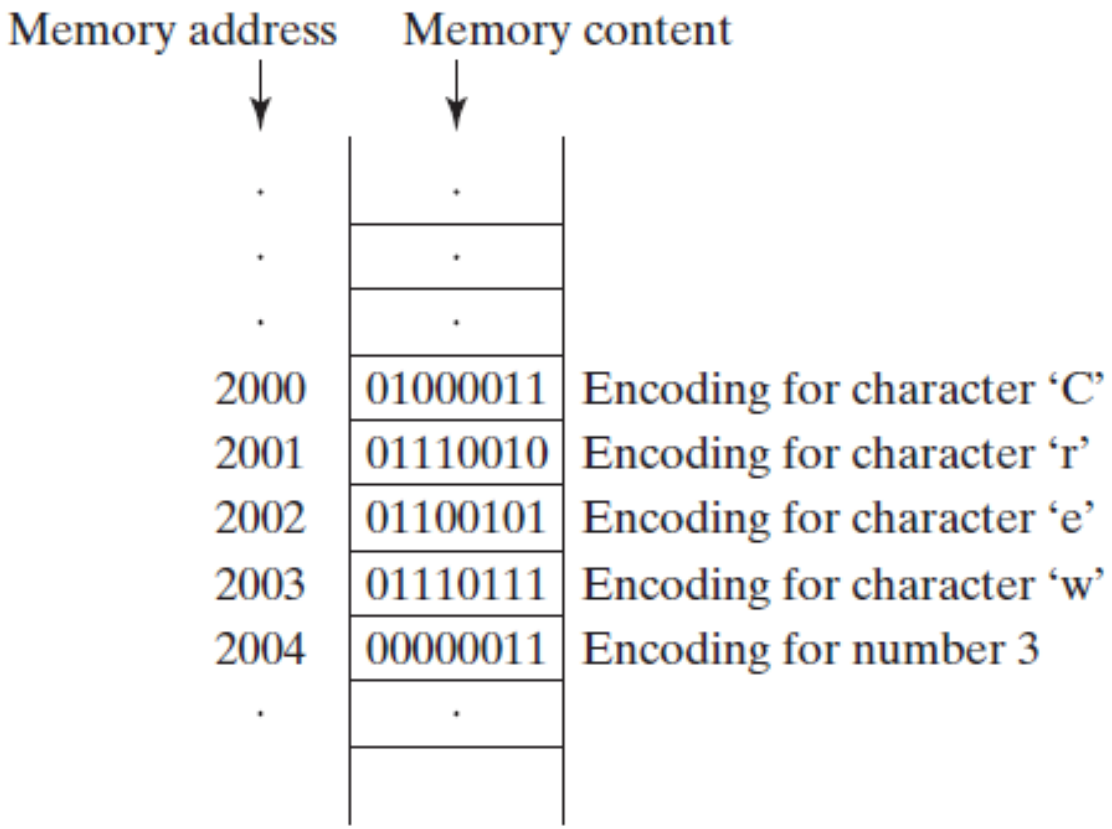
$$(11101.01)_2 = (\underline{0001}\ \underline{1101}.\ \underline{0100})_2$$
$$= (1D.4)_{16}$$

The units of information (data)

- Bit (比特/位): a binary digit which takes either 0 or 1
- Bit is the smallest information unit in computer programming
- Byte (字节): 1 byte = 8 bits, every English character is represented by 1 byte
- KB (千字节)：1 KB = 2^10 B = 1024 B
- MB (兆字节)：1MB = 2^20 B = 1024 KB
- GB (千兆字节)：1GB = 2^30 B = 1024 MB
- TB (兆兆字节)：1TB = 2^40 B = 1024 GB

Memory and addressing

- A computer's memory consists of an ordered sequence of bytes for storing data
- Every location in the memory has a unique address
- The key difference between high and low level programming languages is whether programmer needs to deal with memory addressing directly



Practice

- $(135.8125)_{10} = \ (10000111.1101)_{2}$
- $(1314.205)_{8} = \ (1\ 011\ 001\ 100.010\ 000\ 101)_{2}$
- $(0101010000.0010110011)_{2} = (520.1314)_{8}$
- $(0101010000.0010110011)_{2} = (150.2CC)_{16}$

Thanks