計算物理概論

Introduction to Computational Physics (PHYS290000)

Lecture 1: Introduction

Instructor: 潘國全

kuochuan.pan@gapp.nthu.edu.tw

Instructor

Kuo-Chuan Pan (潘國全)

Office: R506, General building II

Phone: 03-57(42563)

Office hours: by appointment

(kuochuan.pan@gapp.nthu.edu.tw)

Web: https://kuochuanpan.github.io

Teaching Assistant

Hao-Sheng Wang (王皓陞)

Office: R519, General building II

Phone: none

Email: whshoward@gmail.com

Office hours: by appointment (or in google classroom)

Computing courses in the Physics department

- PHYS290000 計算物理概論
- PHYS317000 數值分析
- PHYS401200 計算物理
- PHYS401300 計算物理實作
- PHYS597002 物理專題：計算物理
- PHYS597003 物理專題：凝態計算物理
- PHYS591000 物理與人工智慧(AI)實作導論
- ASTR660000 計算天文物理

Prerequisites

1. General physics, calculus
2. Access a computer (bring your laptop)
3. This course is designed as the first computational course for our physics students
4. All physics students without prior programming experience are highly encouraged to take this course
5. Spend > 4 hours per week

It takes times to be an experienced programmer

Do EXPECT that you can be a python Expert after taking this course.

“Researchers (Bloom (1985), Bryan & Harter (1899), Hayes (1989), Simmon & Chase (1973)) have shown it takes about ten years to develop expertise in any of a wide variety of areas, including chess playing, music composition, telegraph operation, painting, piano playing, swimming, tennis, and research in neuropsychology and topology. The key is deliberative practice: not just doing it again and again, but challenging yourself with a task that is just beyond your current ability, trying it, analyzing your performance while and after doing it, and correcting any mistakes. Then repeat. And repeat again.”

It is never too late to start learning programming.

REF: http://norvig.com/21-days.html#answers

Goal of this course

1. First course on Python programming
2. Learn basic python packages for scientific computing
3. Basic data analysis and visualization

Main topics

1. Python programming
2. Data analysis and visualization
3. Object-oriented programming with python
4. Using numpy/scipy for scientific computing

Google classroom

Course overview

Date	Description
2/13	Lecture 1
2/20	Lecture 2
2/27	**National Holiday (No lecture)**
3/6	Lecture 3
3/13	Lecture 4
3/20	Lecture 5
3/27	Lecture 6
4/3	**National Holiday (No lecture)**
4/10	Lecture 7
4/17	Lecture 8
4/24	Lecture 9
5/1	Lecture 10
5/8	Lecture 11
5/15	Lecture 12
5/22	Lecture 13
5/29	**Professor away (No lecutre)**
6/5	Lecture 14
6/12	**(no lecutre)**

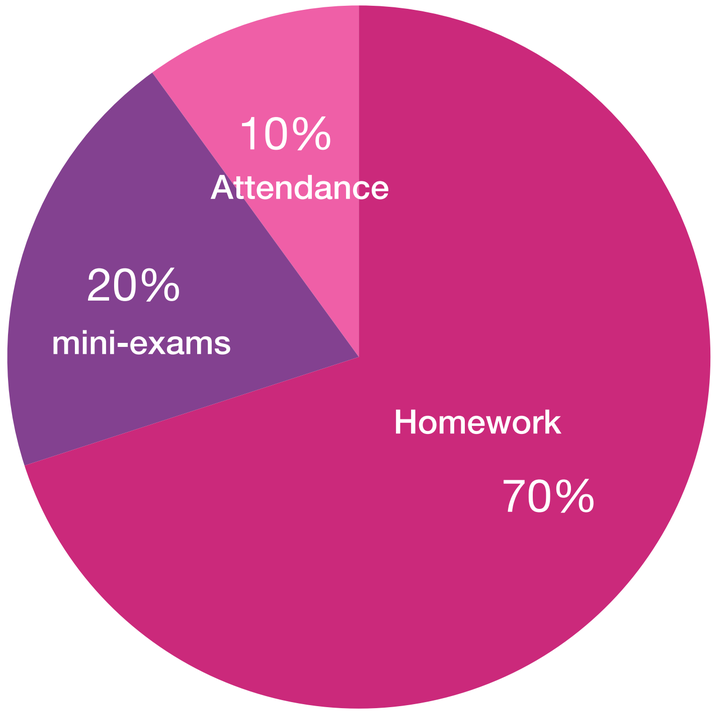
Weekly lectures on Monday
15:30 - ~~18:20~~ 18:00 (no break)

Topics
Programming with Python (from beginner to a junior computational physicist)
Object-oriented programming with Python
Precisions and floating point numbers
Command line interface and shells
Visual Studio Code and Jupyter notebook
Version control with `git`
`numpy` and `scipy` for scientific computing
Data frame and statistics
Data visualization with `matplotlib`
Speedup your codes
Using chatGPT to improve your codes

A “tentative” plan

Grade

Grade Breakdown



Bonus: Extra X %
(Asking questions & participation)

A+	> 90
A	85 - 89
A-	80 - 84
B+	77 - 79
B	73 - 76
B-	70 - 72
C	60 - 69
D	<60

Homework assignments

1. Roughly weekly or biweekly homework
2. Use .py or .ipynb for your solutions
3. “Must” have comments to explain your code
4. Submit all necessary codes/data to google classroom
5. There will be late penalty!!!

Mini-exams

1. There will be a few mini-exams at the end of a lecture.
2. You could check the lecture slides but do not discuss with your classmates or use internet.

Honor Code

1. All work must be in your own words
2. You need to attach your source codes with your solutions
3. You may reference outside resources (for ideas or concepts), but include citations (e.g. urls) at the end of your answers. However, taking codes (or a minor function) from the internet are not allowed
4. You may ask questions to your friends/classmates. He/She may suggest strategies for solving the homework (or debugging), but he/she may not look at your code for you — not even if you sit together and look for the issues/bugs.
5. The purpose of homework assignments is to assess your understanding of concepts; you must demonstrate that understanding to get credit

AD: online learning

https://forms.gle/W8eqdNYAvYdGc55T7

#數位自學不斷電 #數位自學正式開賽 #搶先預約 #數位自學計畫

開催決定：

線上報名：即日起至2023年3月5日（週日）

公布通過名單：2023年3月13日（週一）

完成學習心得：2023年6月30日（週五）

最熱血報名：https://forms.gle/W8eqdNYAvYdGc55T7

初級駕照R：填寫線上申請報名表，教發中心組成駕訓班（數位自學審核小組）審查，擇優補助。通過申請階段之車手可獲得獎金新台幣100元或等值禮券。

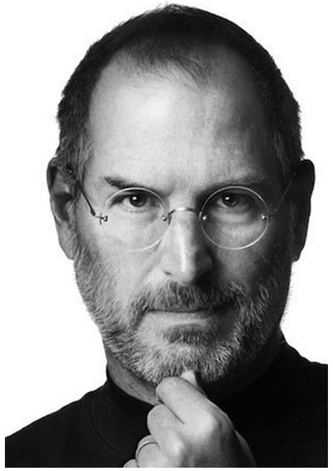
中級駕照L2：數位自學動起來，車手根據自訂日程數位自學。

PRO駕照：四個月內完成自學內容，提供學習心得（約500字，可包含：心得、照片、課程成果等）與修習證明（修課證明、課程完成證明、修課成績單或學習歷程截圖），經確認無誤後擇優予以核發獎金新台幣400元或等值禮券。

Why Python?

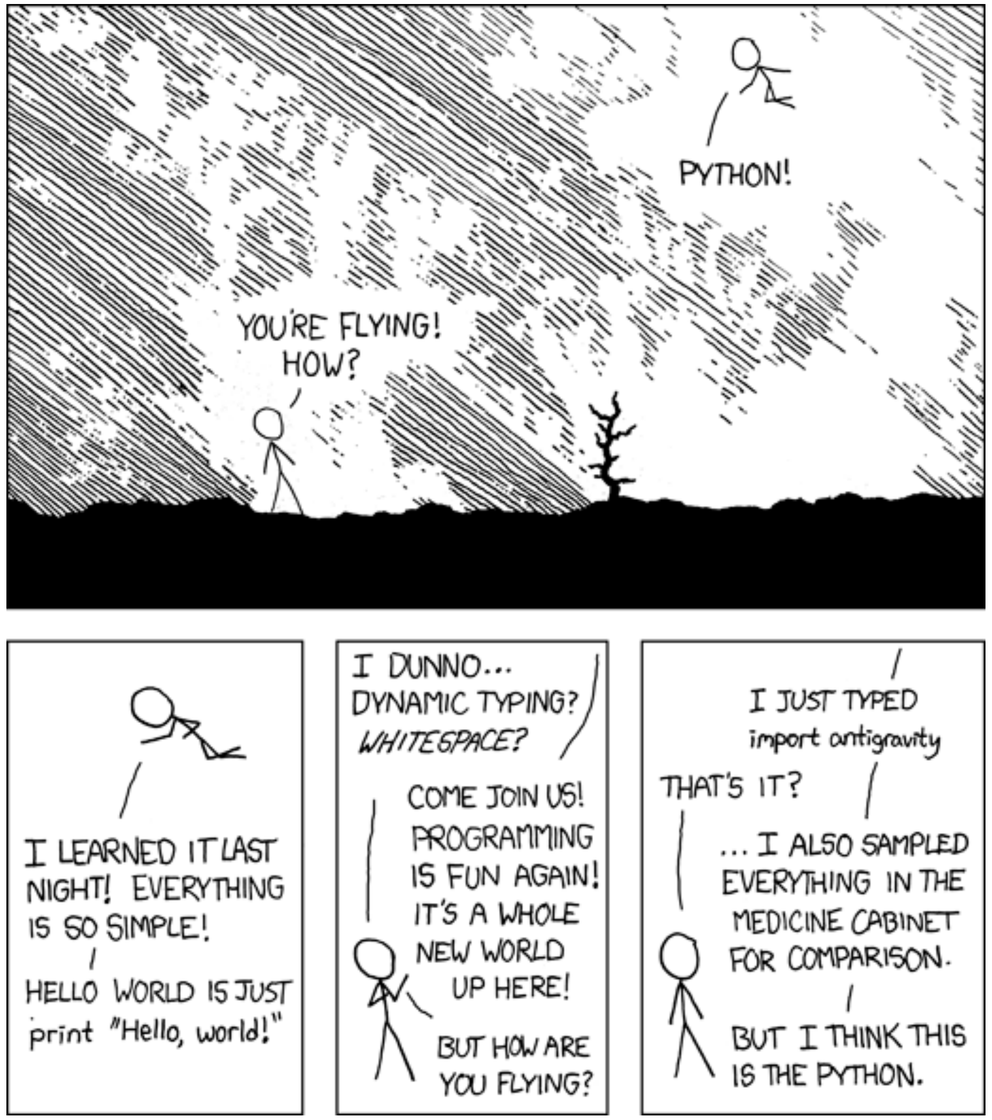
Course overview

"Everybody ~~in this country~~ should learn to program a computer, because it teaches you how to think"
— *Steve Jobs*



"Learning to write programs stretches your mind, and helps you think better, creates a way of thinking about things that I think is helpful in all domains."
—*Bill Gates*

Why Python?



xkcd.com

python™

- A modern, interpreted, high-level, general purpose programming language
- Easy to learn and use
- Expressive language: fewer codes
- Dynamically typed: No need to define the type of variables (disadvantage: slow)
- Interpreted: No need to compile (disadvantage: slow)
- Automatic memory management (disadvantage: memory leak)

Hello world program

C

```c
#include <stdio.h>

int main(void) {
   printf("Hello! World!\n");

   return 0;
}
```

```
$ gcc hello.c
$ ./a.out
```

Python

```python
print(“Hello! World!”)
```

```
$ python hello.py
```

What makes python good for scientific computing?

- Large community of users
- Plenty of scientific libraries and environments (ex. numpy, scipy, matplotlib, scikit-learn, astropy, …etc.)
- Good integration with highly optimized codes written in C and Fortran
- Good support for parallel programming (MPI) and GPU computing
- Open sourced

2021 Stackoverflow developer survey

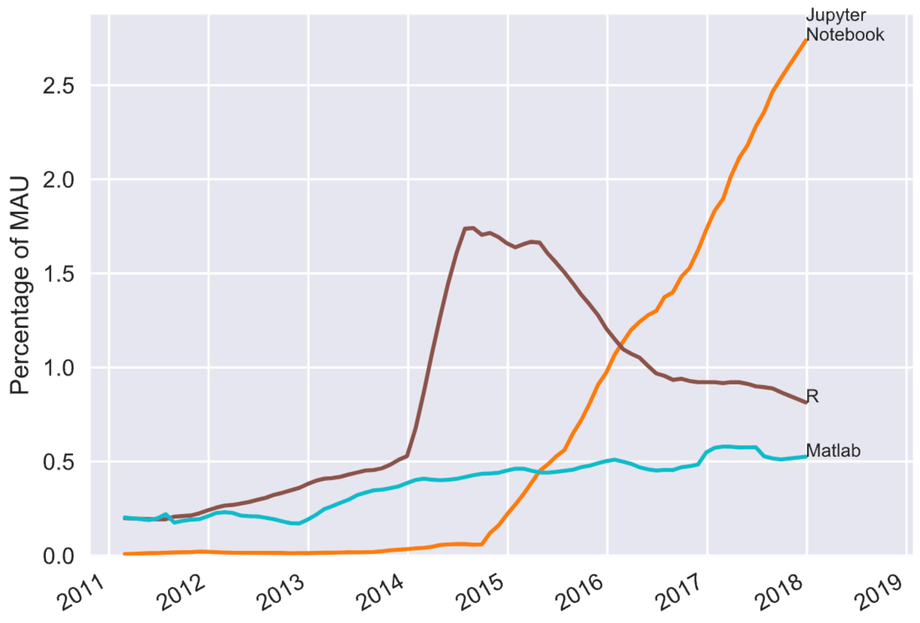
Mainly used for web development and web app.

https://insights.stackoverflow.com/survey/2021?fbclid=IwAR1Z1mL6gGopvbtpkprN-onJCtF25U_oTipDyMihEUQH4mPorRvl1WmDQIE#overview

Scientific programming

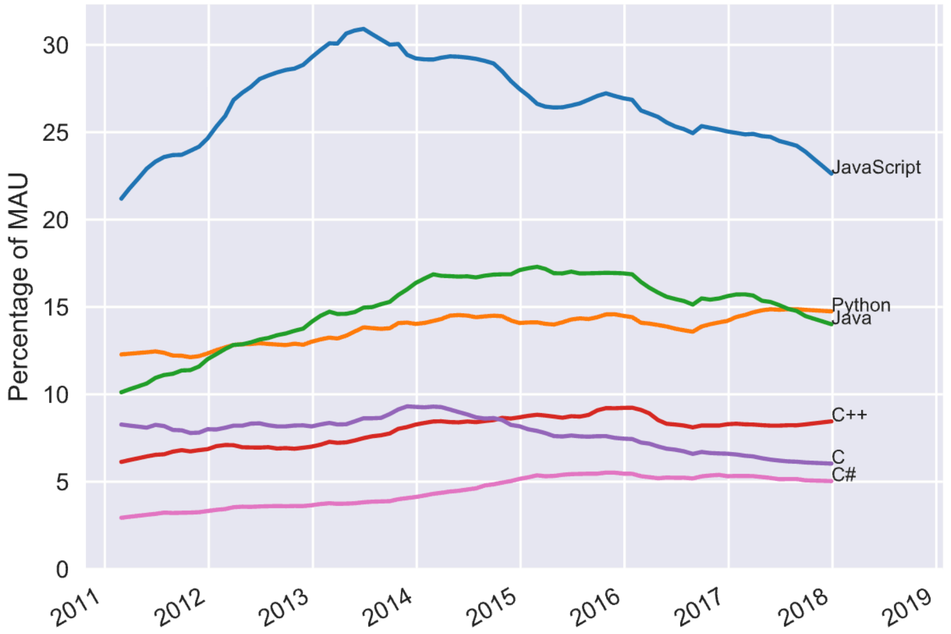
Scientific Languages

There was one other fast-growing ‘language’ included in the results that I purposefully left out:



Major Languages

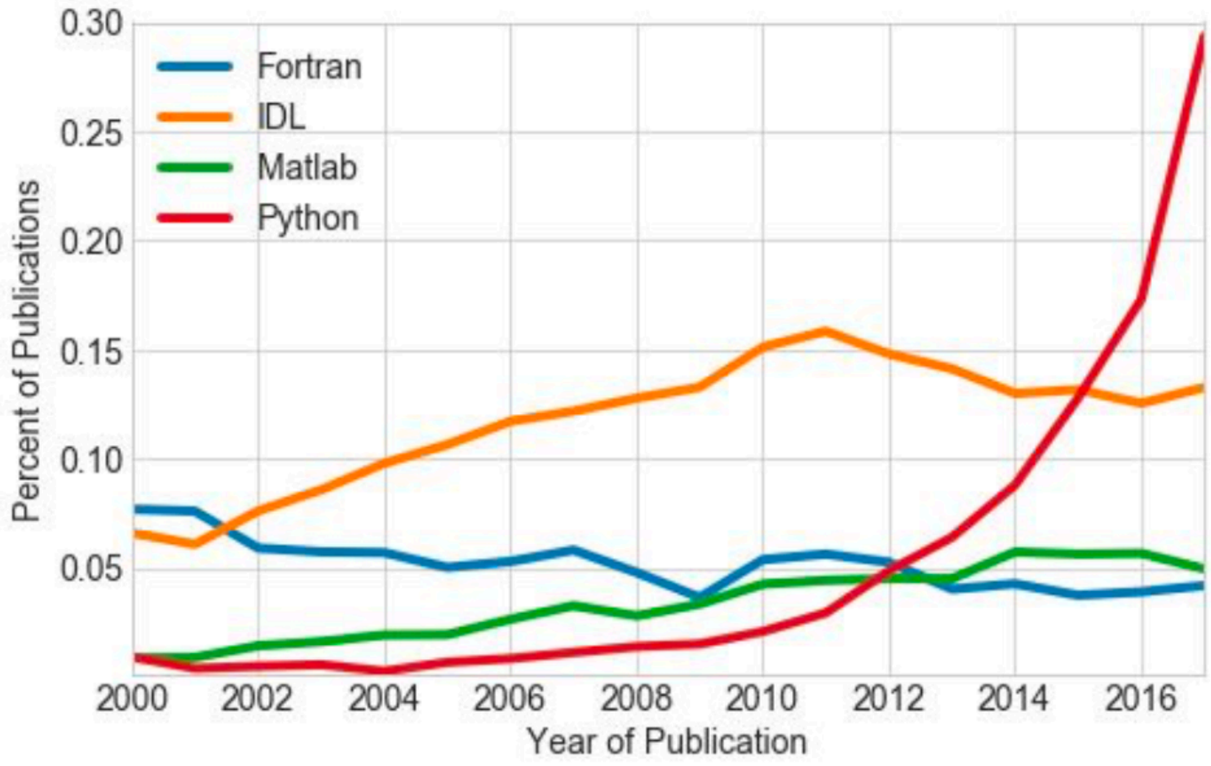
The major programming languages have relatively stable usage, and are mostly what you’d expect:



https://www.benfrederickson.com/ranking-programming-languages-by-github-users/

Scientific programming

Mentions of Software in Astronomy Publications:



Compiled from NASA ADS (code).

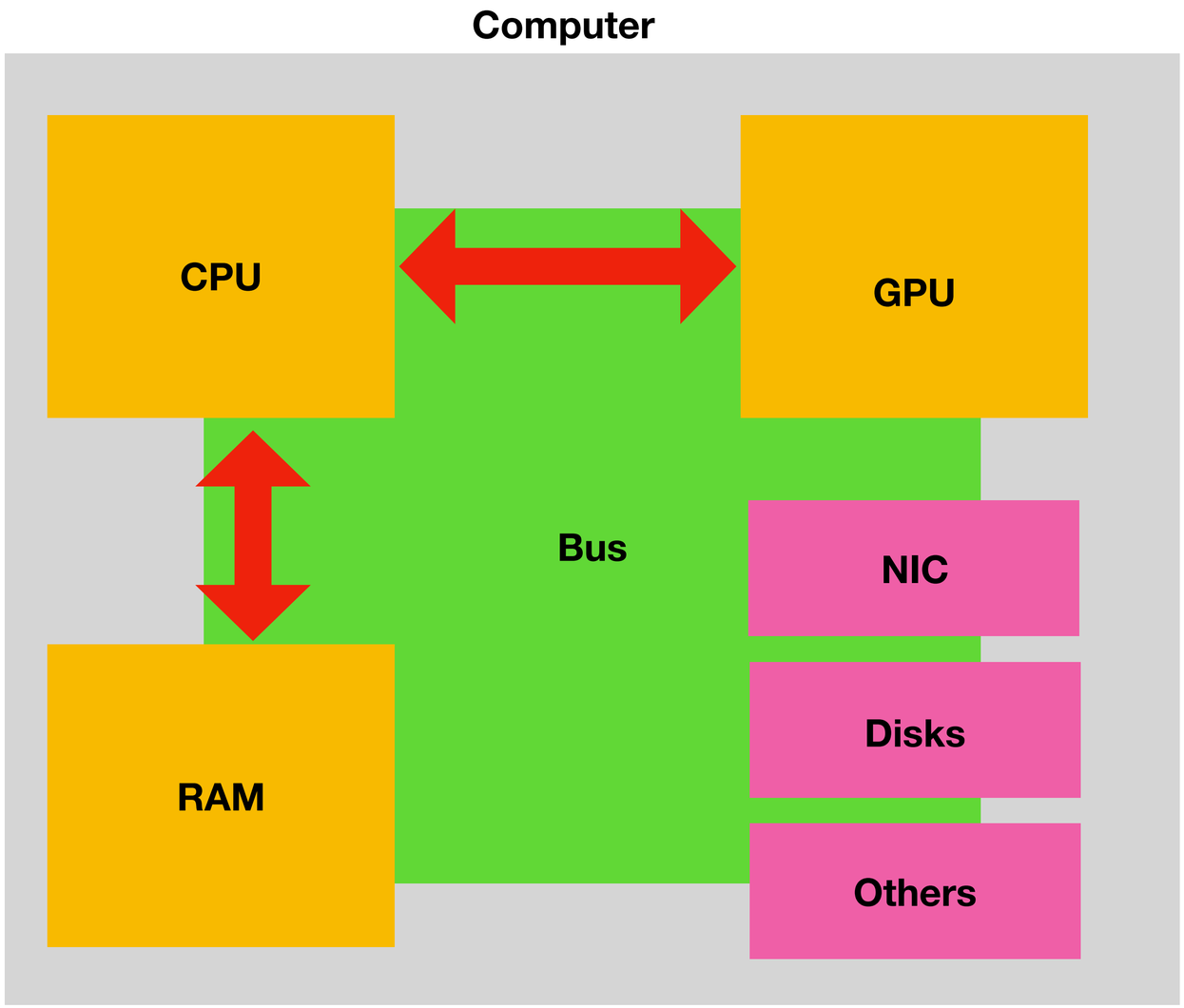
Thanks to Juan Nunez-Iglesias, Thomas P. Robitaille, and Chris Beaumont.

Python

- First released in 1991
- Purpose: improve the code readability
- Python 2.0 was released in 2000
- Python 3.0 was released in 2008 (not completely backward-compatible)
- Python 3.11 (current version) was released in 2022
- In this course, we will use python 3.10

Before we learn programming

Computer Architecture



- CPU (Central Processing Unit)
- GPU (Graphics Processing Unit)
- RAM (Random Access Memory)
- Disks: for storage
- NIC (Network Interface Card): connect to network
- Others: keyboard, mouse, monitor, USB drivers, …etc.
- Bus (PCIe, SATA): The speed of bus determine how fast the components to exchange data.

Speed and capacity

CPU:

Intel i9-13900KF【24核/32緒】3.0G(↑5.8G)/36M/無內顯/無風扇/125W【代理盒裝】, $20300↘$19600

Core/thread **Speed** **Cache size** **Power**

MB:

技嘉 Z790 AORUS ELITE AX(ATX/Realtek2.5Gb+Wi-Fi 6E/註冊五年)16+1+2相電源, $9490 ◆ ★ ↓任搭100↓

CPU pins **MB size** **Network speed**

RAM:

金士頓 32GB(雙通16GB*2) DDR5-6400 FURY Renegade RGB(KF564C32RSAK2-32), $8700↘$5950 ◆ ★ 熱賣

Capacity **Speed: 6400 MT/s = 6400 MHz**

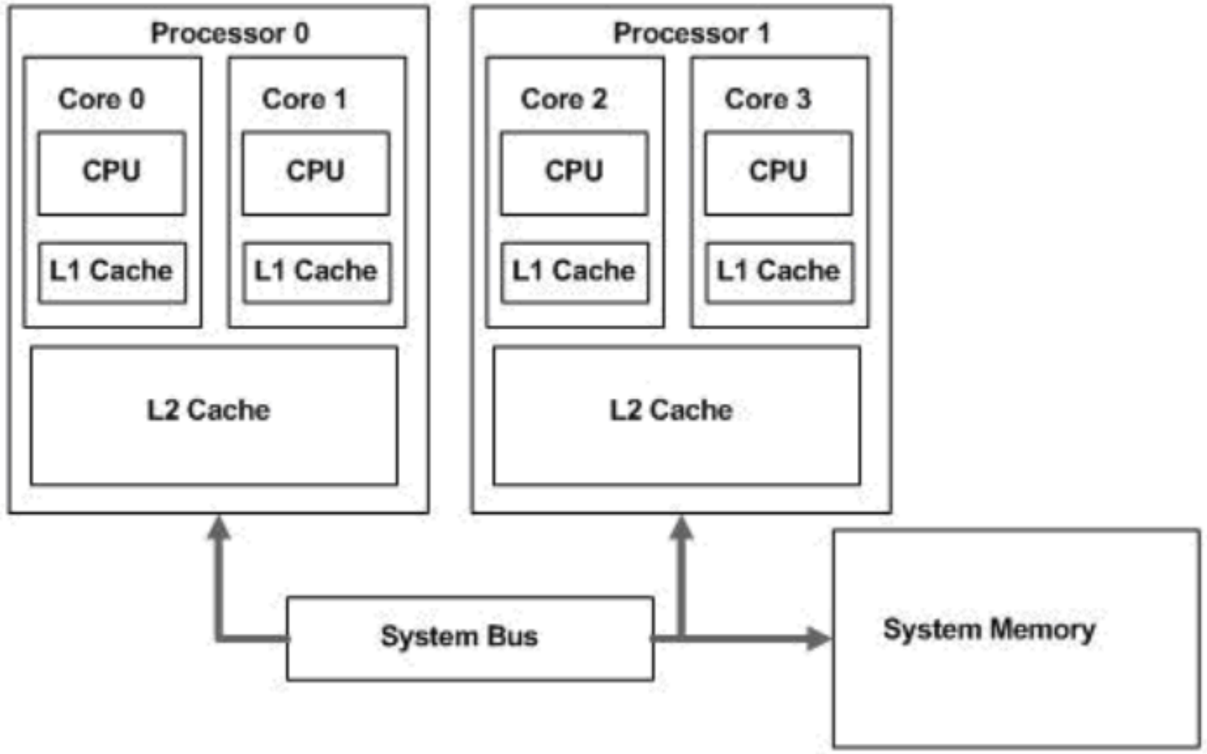
GPU:

技嘉 AORUS RTX4090 XTREME WATERFORCE 24G(2565MHz/23.8cm)封閉水冷*限購一片, $63990 ◆ ★

Capacity **Speed**

NOTE: Nowadays, the memory speed is usually slower than the cpu clock speed.
Will that impact the performance? Ans: usually not, we cpu has cache

Cache memory and cache latency



- CPU Caches
  L1: ~ 64 KB / core
  L2: ~ 256 KB / core
  L3: ~ 2.5 MB / core
  Memory: < TB
  Storage: ~ 10 TB or more

- Level 1 cache (L1): fastest but small
- Level 2 cache (L2): slower than L1 but bigger than L1
- Level 3 cache (L3): slower than L2 but bigger than L2
- Cache hit: find data in cache
- Cache miss: cannot find data in cache —> go to next level
- Cache performance: measured by hit ratio (= hit / (hit+miss))
- Cache latency: the delay from different levels
- System memory (RAM):
- Disk storage (DISK):

Cache memory and cache latency

Numbers everyone should know

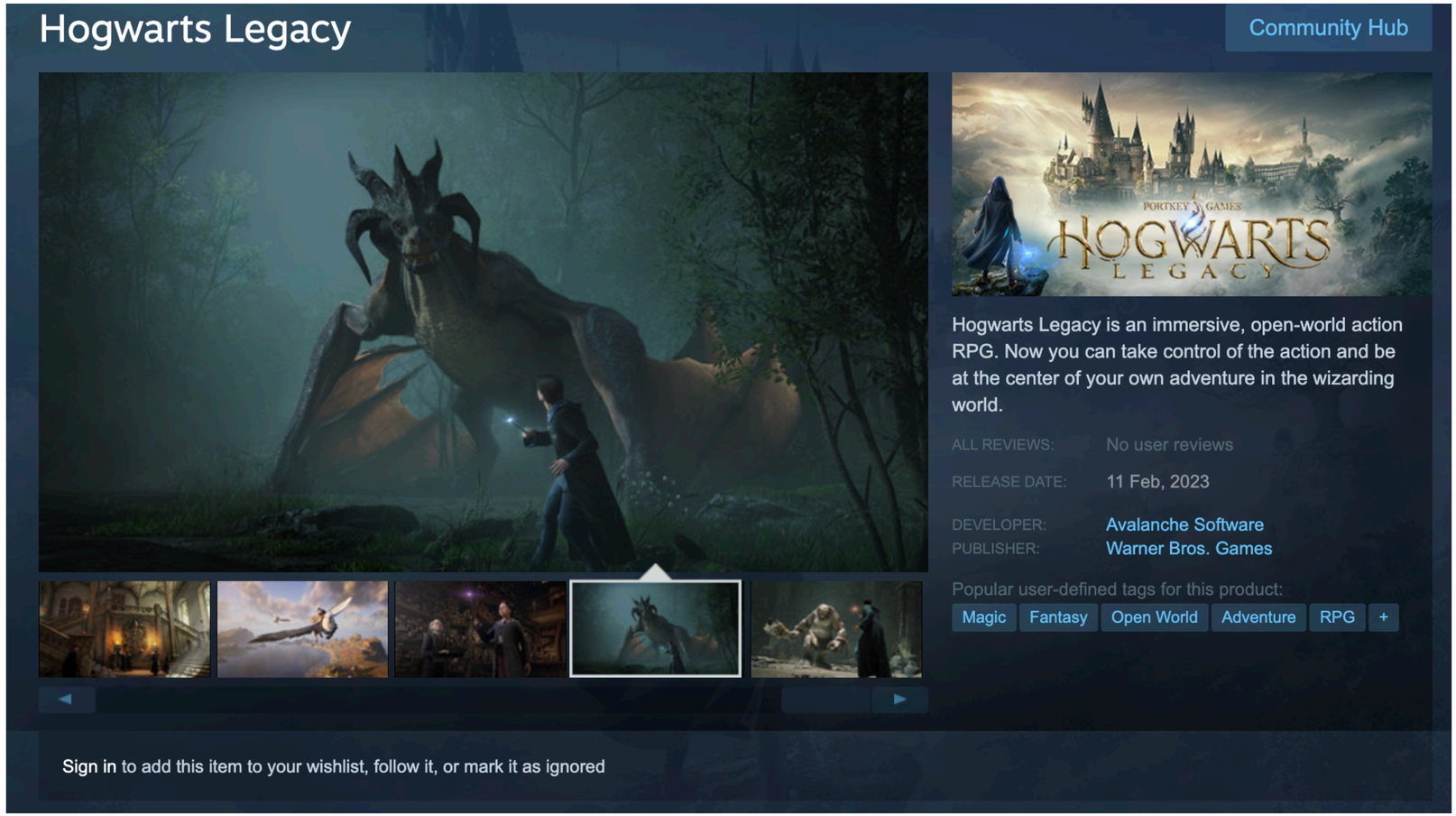
```
          0.5 ns - CPU L1 dCACHE reference
          1   ns - speed-of-light (a photon) travel a 1 ft (30.5cm) distance
          5   ns - CPU L1 iCACHE Branch mispredict
          7   ns - CPU L2  CACHE reference
         71   ns - CPU cross-QPI/NUMA best  case on XEON E5-46*
        100   ns - MUTEX lock/unlock
        100   ns - own DDR MEMORY reference
        135   ns - CPU cross-QPI/NUMA best  case on XEON E7-*
        202   ns - CPU cross-QPI/NUMA worst case on XEON E7-*
        325   ns - CPU cross-QPI/NUMA worst case on XEON E5-46*
     10,000   ns - Compress 1K bytes with Zippy PROCESS
     20,000   ns - Send 2K bytes over 1 Gbps NETWORK
    250,000   ns - Read 1 MB sequentially from MEMORY
    500,000   ns - Round trip within a same DataCenter
 10,000,000   ns - DISK seek
 10,000,000   ns - Read 1 MB sequentially from NETWORK
 30,000,000   ns - Read 1 MB sequentially from DISK
150,000,000   ns - Send a NETWORK packet CA -> Netherlands
|   |   |   |
|   |   | ns|
|   | us|
| ms|
```

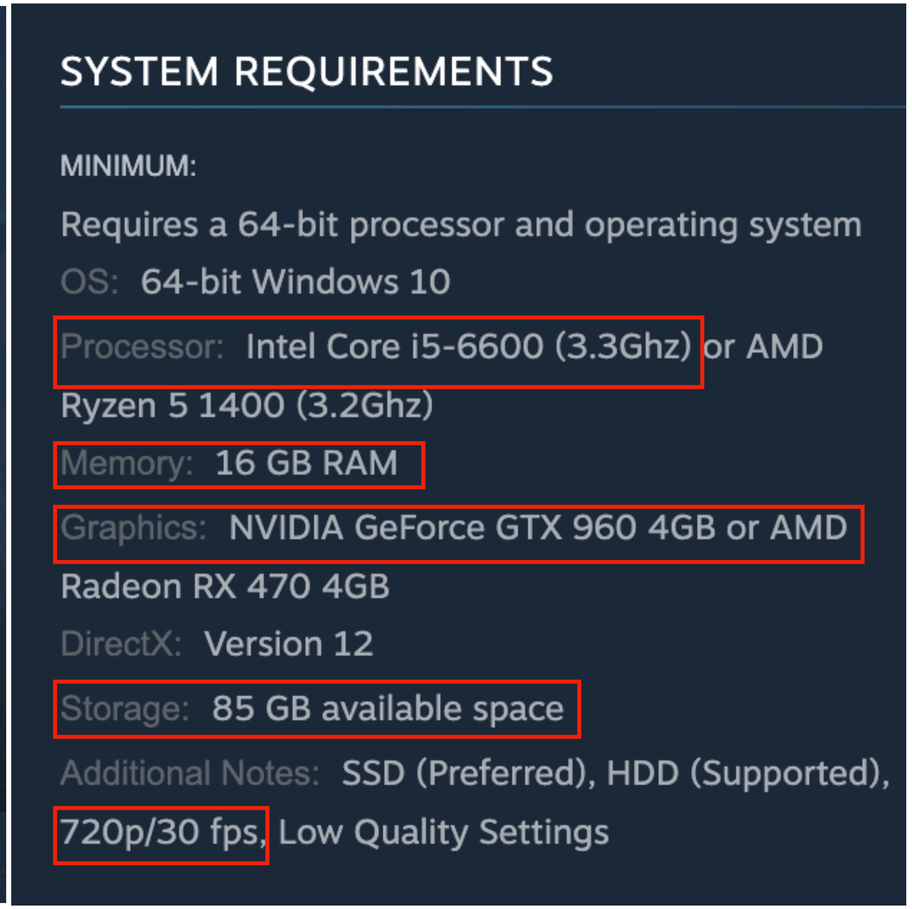
From: Originally by Peter Norvig:

- http://norvig.com/21-days.html#answers
- http://surana.wordpress.com/2009/01/01/numbers-everyone-should-know/,
- http://sites.google.com/site/io/building-scalable-web-applications-with-google-app-engine

https://stackoverflow.com/questions/4087280/approximate-cost-to-access-various-caches-and-main-memory

Know what you could do with your computer





You could find a similar system minimum requirement for your computational task.

If your laptop does't match your requirement or it takes too long to finish the calculation,

1) buy a new one, 2) use the work station from your advisor, 3) the cluster in your institute, 4) use national supercomputers

Storage

- To estimate how many storage (ram and disk) we need for a computing problem, we should understand the unit of storage first.
- Network bandwidth or hard drive size are usually described by ~ XXX Mb/s (megabits per second) or TB (terabytes)
- A bit is a binary digit (0 or 1)
- 8 bits = 1 byte
- For hard disks = 1 TB = 10^{12} bytes, but in the operating system, 1 TB = 1024 * 1024 * 1024 bytes = 931 GB
- How many bits of memory you need to storage one number?

Rabbit 

Rabbyte 

Integer and Floating point numbers

- In computer, you need finite size of memory to store an integer or a real number.
- Depending on the programming language and systems, there might be different systems.
- Integers usually takes 2-4 bytes.

Type	Signed?	Number of bits	Smallest value	Largest valu
`Int8`	✓	8	-2^7	2^7 - 1
`UInt8`		8	0	2^8 - 1
`Int16`	✓	16	-2^15	2^15 - 1
`UInt16`		16	0	2^16 - 1
`Int32`	✓	32	-2^31	2^31 - 1
`UInt32`		32	0	2^32 - 1
`Int64`	✓	64	-2^63	2^63 - 1
`UInt64`		64	0	2^64 - 1
`Int128`	✓	128	-2^127	2^127 - 1
`UInt128`		128	0	2^128 - 1

Integer types in Julia

Question: how to store 0 and the sign?

Floating point numbers

- Similar to scientific notation
- Floating point number system characterized by four integers

β	Base
p	Precision
$[L, U]$	Exponent range

Real number x is represented as

$$x = \pm \left(d_0 + \frac{d1}{\beta} + \frac{d_2}{\beta^2} + ... + \frac{d_{p-1}}{\beta^{p-1}} \right) \beta^E$$

Where $0 \leq d_i \leq \beta - 1$ and $L \leq E \leq U$

Example: Floating point numbers

$$x = \pm \left(d_0 + \frac{d1}{\beta} + \frac{d_2}{\beta^2} + ... + \frac{d_{p-1}}{\beta^{p-1}} \right) \beta^E$$

If $\beta = 10$

Dec. 362.5 = (3 + 6/10 + 2/100 + 5/1000) 10^2

If $\beta = 2$

Dec. 362.5 = 256 + 64 + 32 +8 +2 + 0.5
= (2^8 + 2^6 + 2^5 + 2^3 + 2^1 + 2^-1)
= 10110101.1 = 1.01101011 * 2^8

If $\beta = 8$ Dec. 25 = Oct. 31

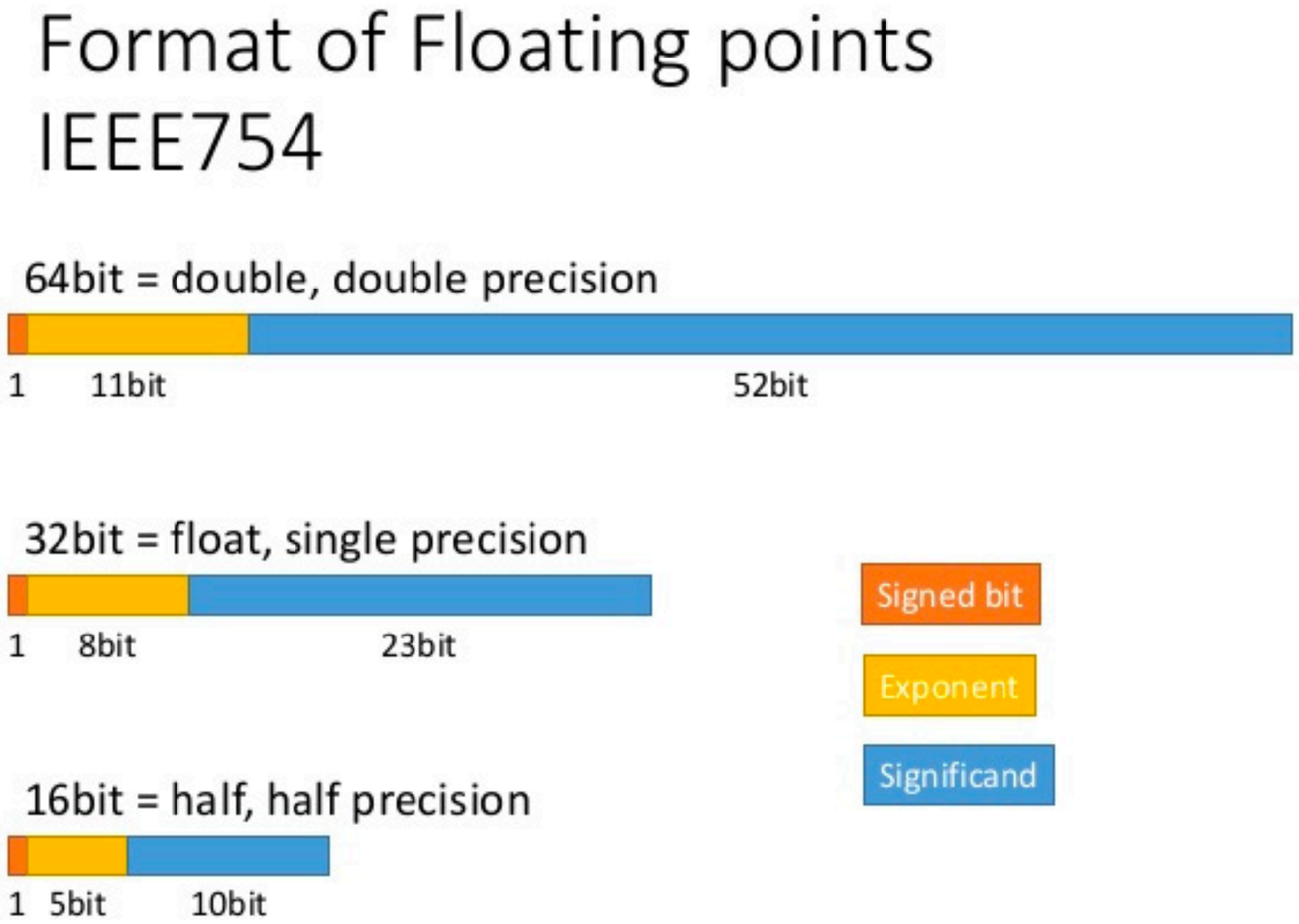
Type of Floating point numbers

Parameters for typical floating-point systems

system	β	p	L	U		
IEEE HP	2	11	−14	15	**2 bytes**	5 bits for L,U,0, sign, 11 bits for p
IEEE SP	2	24	−126	127	**4 bytes**	8 bits for L,U,0, sign, 24 bits for p
IEEE DP	2	53	−1022	1023	**8 bytes**	
IEEE QP	2	113	−16382	16383	**16 bytes**	
Cray-1	2	48	−16383	16384		
HP calculator	10	12	−499	499		
IBM mainframe	16	6	−64	63		

Modern computers use binary arithmetic (beta=2)

Memory Usage



Note that a Python float object takes a full 24 bytes of memory: 8 bytes for the underlying double precision value, 8 bytes for a pointer to the object type, a 8 bytes for a reference count (used for garbage collection)

A Python integer takes > 24 bytes of memory: 8 bytes for the value, 8 bytes for a pointer to the object type, a 8 bytes for a reference count, and at least 4 bytes to store the type. The larger the integer, the higher the memory are used

An empty string “” takes 49 bytes in Python.

We will talk about these more in later lectures.

Normalization

- Floating point number is normalized if leading digit is always non-zero unless number represented is zero
- In normalized system, mantissa m of nonzero floating point number always satisfies $1 \leq m < \beta$
- Reasons for normalization:
  - Each number is unique
  - No digits are wasted

Properties

$$x = \pm \left(d_0 + \frac{d1}{\beta} + \frac{d_2}{\beta^2} + ... + \frac{d_{p-1}}{\beta^{p-1}} \right) \beta^E$$

- Floating point numbers are finite and discrete
- Total number of normalized floating point number is

$$\underbrace{2}_{\text{signs}}\underbrace{(\beta - 1)}_{\text{1st digit}}\beta^{p-1}\underbrace{(U - L + 1)}_{\text{Exponents}} + \underbrace{1}_{\text{zero}}$$

- Smallest positive normalized number: $UFL = \beta^L$
- Largest floating point number: $OFL = \beta^{U+1}(1 - \beta^{-p})$
- Not all real numbers exactly representable

Rounding rules

- If a real number is not exactly representable, then it is approximated by "nearby" floating point number
- This process is called rounding, and error introduced is call "rounding error"
- In IEEE systems: round to nearest
- So IEEE single, double, and quadruple precision systems have about 7, 16, and 36 decimal digits of precision, respectively.

Machine precision

- Rounding to nearest,

$$\epsilon_{mach} = \frac{1}{2}\beta^{1-p}$$

Recall:

$$x = \pm\left(d_0 + \frac{d1}{\beta} + \frac{d_2}{\beta^2} + ... + \frac{d_{p-1}}{\beta^{p-1}}\right)\beta^E$$

- Maximum relative error in floating point system is

$$\left|\frac{fl(x) - x}{x}\right| \leq \epsilon_{mach}$$

Exceptional values

- IEEE floating point standard provides special values to indicate positive and negative “infinity” (+Inf/ -Inf) and “not a number” (NaN)
- Examples:

$$1/0 = \infty$$

$$\log(0) = -\infty$$

$$\sqrt{-1} = \mathrm{NaN}$$

Floating Point Arithmetic

- Addition or subtraction: shifting mantissa to make exponents match may cause loss of some digits of smaller number

$$1.23 \times 10^{14} + 1.01 \times 10^{-5}$$

- Multiplication: product of two p-digit mantissas contains up to 2p digits, so results may not be representable

$$1.2 \times 1.2 = 1.44$$

- Division: Quotient of two p-digit mantissas may contain more than p digits.

$$1/10$$

- Results of floating-floating-point arithmetic operation may differ from results of corresponding real arithmetic operation on same operands

Cancellation

- Subtraction between p-digit numbers having same sign and similar magnitudes yield result with fewer than p digits, so it is usually exactly representable

$$1.234 \times 10^3 - 1.223 \times 10^3 = 0.011 \times 10^3 = 11$$

- Which is correct and exactly representable but has only 2 significant digits
- Often implies serious loss of information

Example: Cancellation

- Total energy of helium atom is sum of kinetic and potential energies, which are computed separately and have opposite signs, so suffer cancellation

- Although computed values for kinetic and potential energies changed by only 6% or less, resulting estimate for total energy changed by 144%

Year	Kinetic	Potential	Total
1971	13.0	−14.0	−1.0
1977	12.76	−14.02	−1.26
1980	12.22	−14.35	−2.13
1985	12.28	−14.65	−2.37
1988	12.40	−14.84	−2.44

Example: Quadratic Formula

- Equation $ax^2 + bx + c = 0$

has two solutions $$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- Naive use of formula can suffer overflow or underflow or several cancellation
- Cancellation between -b and square root

$$x = \frac{2c}{-b \mp \sqrt{b^2 - 4ac}}$$

- Cancellation inside square root cannot be easily avoided without using higher precision

Example: Mean & Std

- Mean and standard deviation are given by

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \text{and} \quad \sigma = \left[\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2\right]^{\frac{1}{2}}$$

- Is mathematically equivalent to

$$\sigma = \left[\frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - n\bar{x}^2\right)\right]^{\frac{1}{2}}$$

Summary: floating point systems

- Real numbers are approximated by finite and discrete floating-point number, with sign, exponent, and mantissa
- Exponent field determines range of representable magnitudes
- Mantissa field determines precision and relative accuracy or floating point approximation
- Rounding error is loss of least significant, railing digits when approximating true real number by nearby floating point numbers
- Cancellation is loss of most significant, resulting in fewer significant digits in finite precision

Programming Environment

Environment

- Linux and Mac OS are recommended.
- Windows users are suggested to use WSL2
- Possible to use Windows
- Python 3.10 is recommended (via Anaconda)
- VS Code is recommended