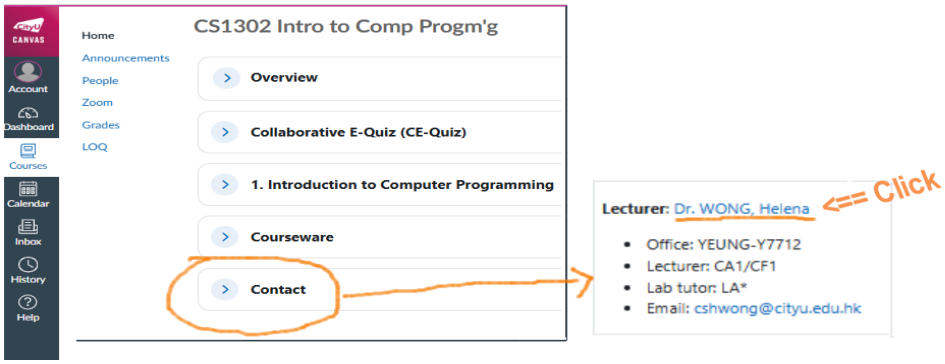
CS1302 - Lecture 1

By Dr Helena WONG

==========

Please get **Supplementary files** from
Canvas => CS1302 => Home => **Contact => Dr Helena WONG**
(e.g. Slides, Helena's cheat sheets)



==========

*.ipynb

a Jupyter Notebook file

- It can contain text, program code, etc.
- It is the format of our lecture and lab files
- popular among people doing *Data Science* projects etc. using **Python**

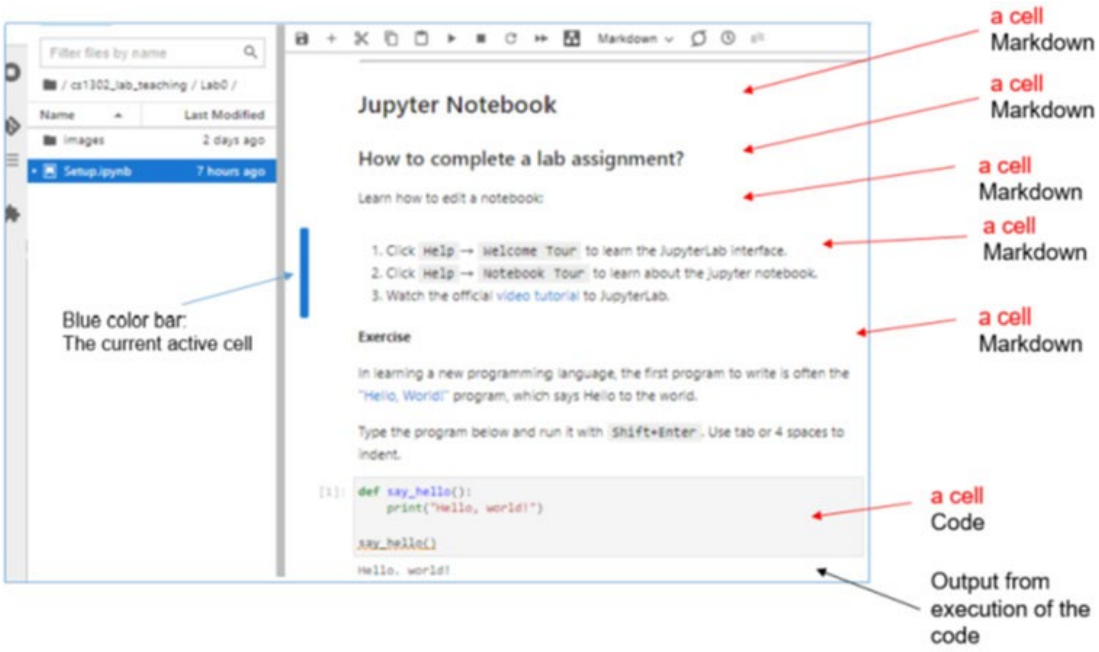
JupyterHub (~JupyterLab)

a platform that we open the jupyter notebook files
(like a virtual computer, has storage space)

==========

Working with *.ipynb

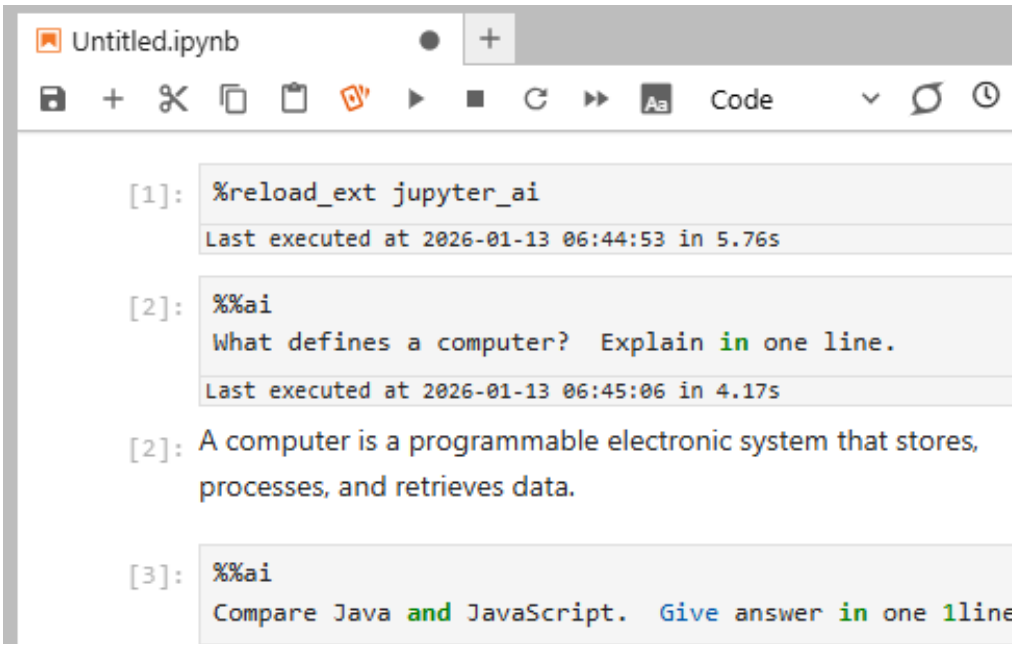
• *.ipynb contains "cells" for **code** and **Markdown** (narration).
• Select a cell: Click on it. A blue color bar appears at the left.



Generative AI is available. We use a cell magic named "ai".

1) Initialization:
   %reload_ext jupyter_ai

2) To ask a question, for example:
   %%ai
   What defines a computer? Explain in one line.



Agenda

(a) Course Introduction

(b) Lecture 1 Introduction to Computer Programming

1. Computer
1.1 What is a computer
1.2 What is the architecture of a computer?
1.2.1 Peripherals - Input and Output devices
1.2.2 Central Processing Unit

2. Programming
2.1 What is Programming? What is Machine Language?
2.2 How code and data is stored in a computer?
2.3 Why computer uses binary representation?

3. Different generations of programming languages

4. High-level Language
4.1 Compilation vs Interpretation
4.2 What programming language will you learn?

Q: Do I need to submit the answers inside the lecture notebook?
A: No. But you will have E-Quiz and CE-Quiz.

1. Computer

1.1 What is a computer

1.2 What is the architecture of a computer?

John von Neumann developed the computer model,
then called von Neumann architecture (in the 1940s)
which is still the most common computer model nowadays.

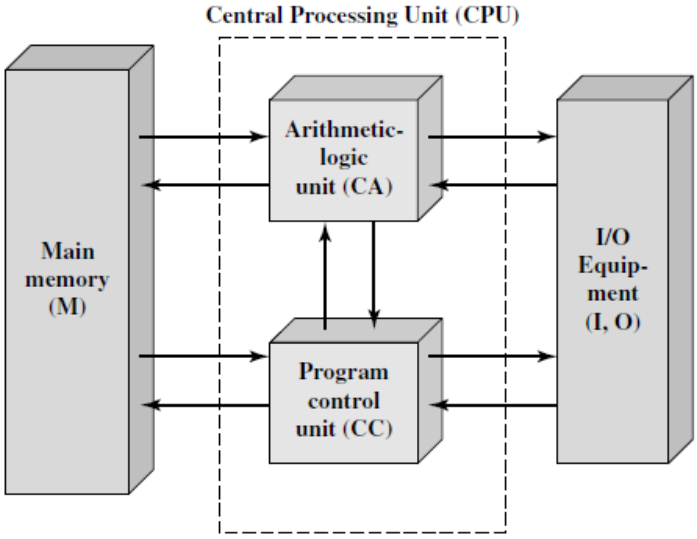
CA: Central Arithmetical Part; CC: Central Control
(https://nanopdf.com/download/ahmedabdelmailkmohammed1_pdf)

1.2.1 Input and Output devices

Input devices are typically devices which take aspects from the physical world and digitizes those sensations so that a computer can make use of that information, like a keyboard takes physical keystrokes and interprets those strokes to control a computer. A digital camera takes light from the physical world, digitizes that light, and generates a digital data through the CCD.

Output devices are typically devices that take computer signals and render them in a way that we can understand, like how a monitor is fed display information through the graphics card so we can physically see what the computer is doing. Since pretty much all digital cameras have a viewscreen, this would be considered the output device showing menu options and/or digital interpretations of what the camera is collecting through the lens.
(https://www.quora.com/How-can-a-digital-camera-be-considered-an-input-and-output-device)

1.2.2 Central Processing Unit



1.2.3 Memories

For reference:

RAM

SRAM : Static Random-Access Memory

to store static data,

faster, expensive, used as a cache memory

DRAM : Dynamic Random-Access Memory

For the dynamic storage of data,

smaller size, shorter life, needs to continuously refreshed to keep data

SRAM's data remains active as long as the computer system has a power supply. However, data is lost in SRAM when power failures have occurred.

DRAM can hold more data than an SRAM of the same size. However, the capacitor needs to be continuously refreshed to retain information because DRAM is volatile. If the power is switched off, the data store in memory is lost.

ROM

PROM : Programmable Read-Only Memory

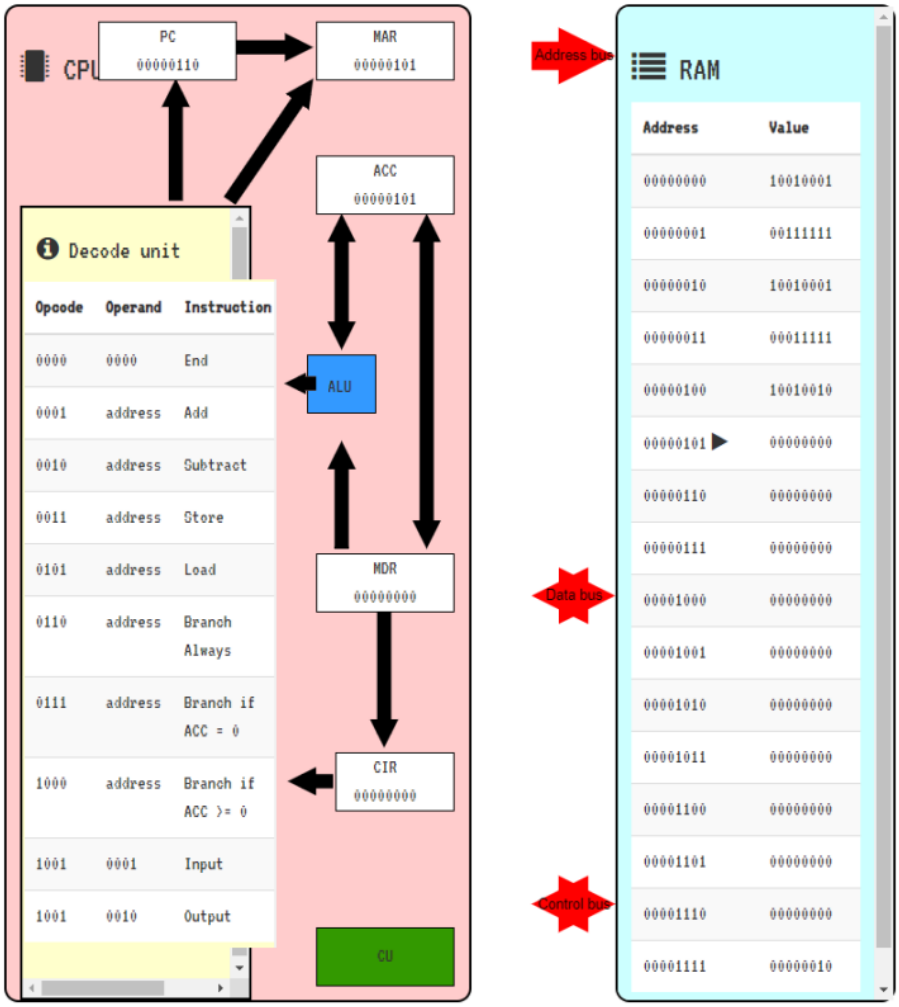
EPROM : Erasable Programmable Read-Only Memory

It is the type of read only memory in which stored data can be erased and re-programmed only once in the EPROM memory. It is a non-volatile memory chip that holds data when there is no power supply and can also store data for a minimum of 10 to 20 years. In EPROM, if we want to erase any stored data and re-programmed it, first, we need to pass the ultraviolet light for 40 minutes to erase the data; after that, the data is re-created in EPROM.

EEPROM : Electrically Erasable Programmable Read-Only Memory

https://www.javatpoint.com/classification-of-memory

CPU Simulation (https://tools.withcode.uk/cpu)
Under Settings, click Examples: Add two numbers. Observe that the values in the RAM have changed. Click Run at the bottom right-hand corner.



Explanation of doing 2+3

Instruction Cycles: fetch, decode, execute for 2+3

Can you find these in the simulation?

Memory (RAM)
ALU (Arithmetic Logic Unit)
CU (Control unit)
Decode unit (Opcode, Operand) => Instruction
PC (Program Counter)
MAR (Memory Address Register)
MDR (Memory Data Register)
ACC (Accumulator Register)
CIR (Current Instruction Register)

* Both **code** and **data** are in RAM
The demo actually has 5 main steps:
1. fetch, decode, execute (input into the Accumulator register)
2. fetch, decode, execute (store the value in the Acc into memory)
3. fetch, decode, execute (input into the Accumulator register)
4. fetch, decode, execute (add value in ACC to the data at the memory location (brought to MDR), Result saved in ACC)
5. fetch, decode, execute (output from the Accumulator register)
[6. end]

Recall:

* **ALU** performs arithmetics like a calculator,
* **CU** directs the operations of the processor in executing a program.

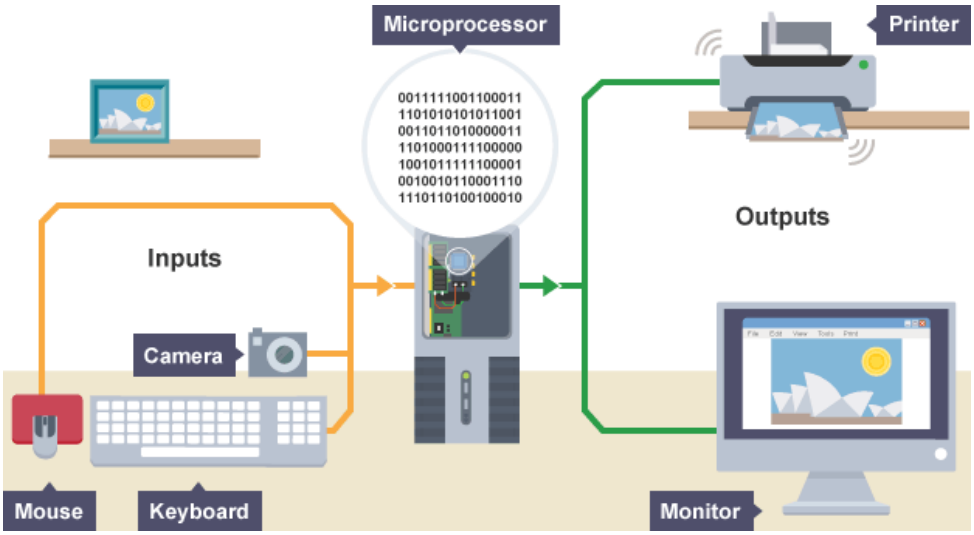
Although CS1302 mainly focuses on programming with a high level programming language (Python), the learning of how the CPU works could give a good idea (though not most detail) of how actually the translated machine instructions run step by step.

Well, you **don't need to** dictate every details in the simulation (like what is the complete name of ACC), and you **don't need to worry** about writing down the detail instructions (5 steps) of adding 2 numbers, and you **don't need to explain** precisely how the fetch-decode-execute cycle runs with using the registers and the bus etc.. **It is sufficient if you understand:**

i) how program and data are stored in the main memory (RAM)
ii) for carrying out each instruction, the CU directs the operations in a fetch-decode-execute cycle.

2. What is Programming?

2.1 How code and data is stored in a computer?



2.2 Why computer uses binary representation?

https://www.youtube.com/embed/Xpk67YzOn5w (7 minutes)
0:00 Intro
0:25 What's binary
2:40 Transisters
3:05 Bit vs Byte
3:45 Text representation and ASCII
5:00 Byte, 8-bits vs 16-bits, etc..

the ENIAC was not binary, but decimal! It used 10 vacuum tubes to represent the digits 0–9.
Let's consider the number 128 . Here is how ENIAC stored this number (= the vacuum tube is on):

https://cs.calvin.edu/activities/books/rit/chapter2/history/electronic.htm

___decimals can be represented by 10 bits?

1 bit represents 2 values: 0 or 1
2 bits represent 4 values: 00, 01, 10, 11
3 bits represent 8 values: 000,001,010,011,100,101,110,111
10 bits represents __ values?

10 bits represents 2**10 = 1024 values?
30 bits represents 2**30 = 1024x1024x1024

= ~1,000,000,000 values!

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Why Hexadecimal is often used?

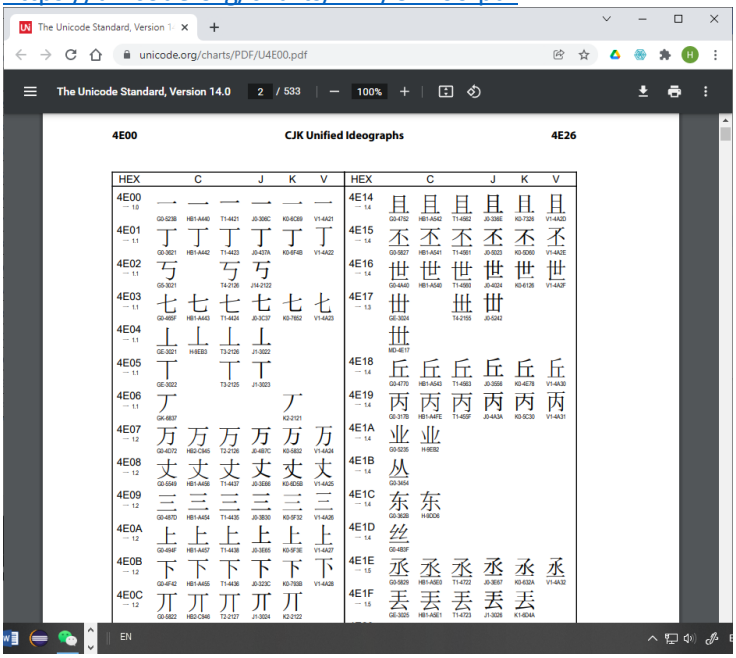
Reason: It is easy convert between $FDFC_{(16)}$ and $\_\_\_\_{(2)}$

Hexadecimal	<=>	Binary (4 bits)
0		0000
1		0001
2		0010
3		
4		
5		
6		
7		
8		
9		
A		
B		
C		
D		
E		
F		1111
FDFC		1111 1101 1111 1100

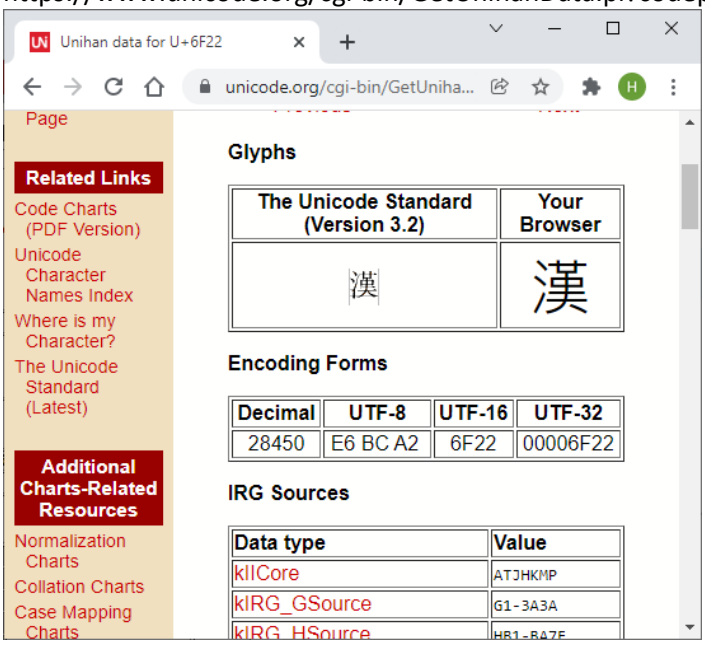
Graphic character symbol	Hexadecimal character value

0020	0 0030	@ 0040	P 0050	` 0060	p 0070	00A0	° 00B0	À 00C0	Ð 00D0	à 00E0	ð 00F0
! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071	¡ 00A1	± 00B1	Á 00C1	Ñ 00D1	á 00E1	ñ 00F1
" 0022	2 0032	B 0042	R 0052	b 0062	r 0072	¢ 00A2	² 00B2	Â 00C2	Ò 00D2	â 00E2	ò 00F2
# 0023	3 0033	C 0043	S 0053	c 0063	s 0073	£ 00A3	³ 00B3	Ã 00C3	Ó 00D3	ã 00E3	ó 00F3
$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074	¤ 00A4	´ 00B4	Ä 00C4	Ô 00D4	ä 00E4	ô 00F4
% 0025	5 0035	E 0045	U 0055	e 0065	u 0075	¥ 00A5	µ 00B5	Å 00C5	Õ 00D5	å 00E5	õ 00F5
& 0026	6 0036	F 0046	V 0056	f 0066	v 0076	¦ 00A6	¶ 00B6	Æ 00C6	Ö 00D6	æ 00E6	ö 00F6
' 0027	7 0037	G 0047	W 0057	g 0067	w 0077	§ 00A7	· 00B7	Ç 00C7	× 00D7	ç 00E7	÷ 00F7
(0028	8 0038	H 0048	X 0058	h 0068	x 0078	¨ 00A8	¸ 00B8	È 00C8	Ø 00D8	è 00E8	ø 00F8
) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079	© 00A9	¹ 00B9	É 00C9	Ù 00D9	é 00E9	ù 00F9
* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A	ª 00AA	º 00BA	Ê 00CA	Ú 00DA	ê 00EA	ú 00FA
+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B	« 00AB	» 00BB	Ë 00CB	Û 00DB	ë 00EB	û 00FB
, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C	¬ 00AC	¼ 00BC	Ì 00CC	Ü 00DC	ì 00EC	ü 00FC
- 002D	= 003D	M 004D	] 005D	m 006D	} 007D	00AD	½ 00BD	Í 00CD	Ý 00DD	í 00ED	ý 00FD
. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E	® 00AE	¾ 00BE	Î 00CE	Þ 00DE	î 00EE	þ 00FE
/ 002F	? 003F	O 004F	_ 005F	o 006F	007F	¯ 00AF	¿ 00BF	Ï 00CF	ß 00DF	ï 00EF	ÿ 00FF

https://unicode.org/charts/PDF/U4E00.pdf



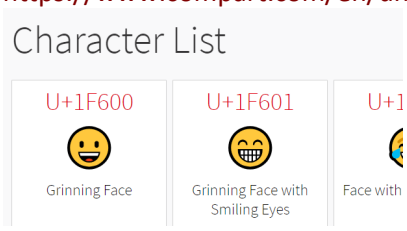
https://www.unicode.org/cgi-bin/GetUnihanData.pl?codepoint=%E6%BC%A2



UTF-8 takes 1 to 4 bytes
UTF-16 takes 2 or 4 bytes
UTF-32 takes 4 bytes

UTF-8/16: variable lengths
UTF-32: fixed length

https://www.compart.com/en/unicode/block/U+1F600



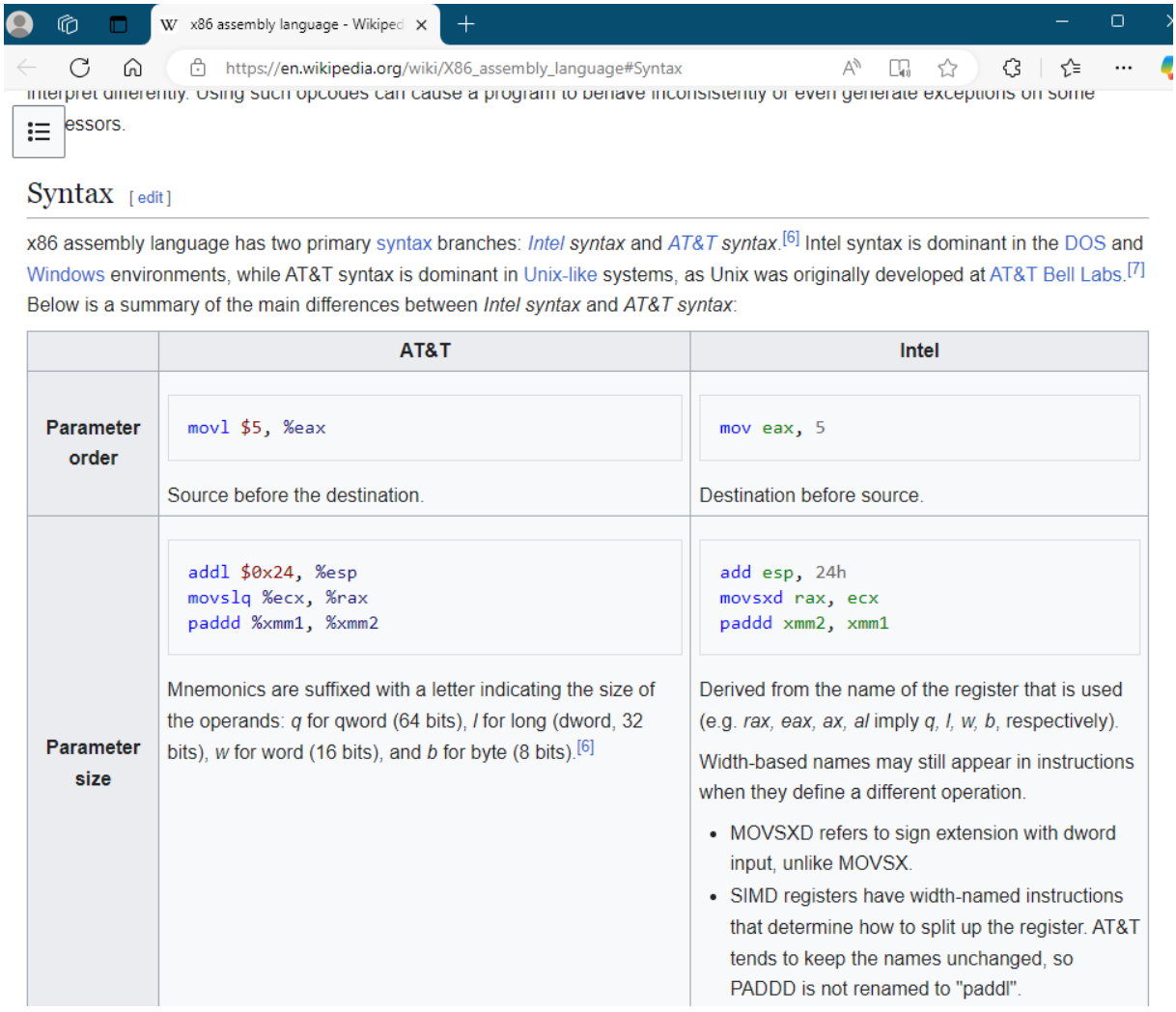
print("\U0001F602")

3. Different generations of programming languages

Assembly language is platform-specific

Different instruction sets - examples:

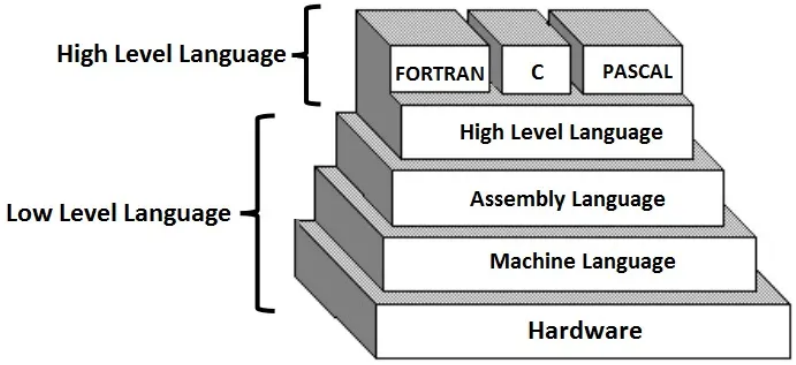
https://en.wikipedia.org/wiki/Motorola_68000_series#Architecture
https://en.wikipedia.org/wiki/X86_assembly_language#Syntax



The Art of Writing Software https://www.youtube.com/watch?v=QdVFvsCWXrA
skip to 2:11 why not machine/assembly language

4. High-level Language

4.1 What is a high-level language?



Computer Language and its Types

Ref: https://www.youtube.com/watch?v=aYjGXzktatA (1:18 - 3:05)

Compilation
We write program source code.
Then via Compilation: We will get a executable file.
Run the executable file (not the program source code)!

Interpretation
We write program source code.
Execute on the fly

4.2 What programming language will you learn?

Video
Why Python: 2:11

```
[11]: import datetime  # library to obtain current year

cohort = input("In which year did you join CityU? [e.g., 2020]")
year = datetime.datetime.now().year - int(cohort) + 1
print("So you are a year", year, "student.")
```

```
In which year did you join CityU? [e.g., 2020] 1988
So you are a year 36 student.
```

```
import datetime # library to obtain current year
cohort = input("In which year did you join CityU? ")
year = datetime.datetime.now().year - int(cohort) + 1
print("So you are a year", year, "student.")
```

In which year did you join CityU? 2020
So you are a year 4 student.

Python Turtle Example

Go https://pythonsandbox.com/turtle
Try the code below:

```
import turtle
t = turtle.Turtle()
t.color('red', 'yellow')
t.begin_fill()
i = 0
while i < 10:
    t.forward(200)
    t.left(170)
    i += 1
t.end_fill()
t.done()
```

4.3 What is the next generation language?

Donald Ervin Knuth (born January 10, 1938)

is an American computer scientist, mathematician, and professor emeritus at Stanford University. He is the 1974 recipient of the ACM Turing Award, informally considered the Nobel Prize of computer science.

Knuth has been called the "father of the analysis of algorithms".

Literate Programming (Literate: able to read and write)

a computer program is given an explanation of its logic in a natural language
"how could I write a program that people enjoy reading it; could be presentable, nice type-set"