

การเขียนโปรแกรมด้วยภาษาไพธอน  
**Special Lecture 1: Python Programming**

ชวณัฐ นาคะสันต์

ศูนย์สื่อสารสนเทศ มหาวิทยาลัยคานาซาวา

2021-01-28

# ดาวนโหลดสไลด์

- Slides Available at <https://github.com/LunaticNeko/chandrakasem-lectures>



# หัวข้อการบรรยาย / Topics

ภาษาไพธอน  
Python Language

ข้อมูลและตัวแปร  
Data and Variables

การรับและแสดงข้อมูล  
แบบมาตรฐาน  
Standard Input/Output

ตัวดำเนินการ  
Operators

การตรวจสอบเงื่อนไข  
Conditions (if, if-else)

การวนซ้ำ  
Repetition (for, while)

หลังจากการบรรยายวันนี้: นักศึกษามีความรู้พื้นฐาน สามารถเขียนโปรแกรมภาษาไพธอนแบบง่ายๆ เพื่อแก้โจทย์ระดับพื้นฐาน เช่น การรับและแสดงข้อมูล, การคำนวณพื้นฐาน, การตรวจสอบเงื่อนไข, และการวนซ้ำได้

ก่อนจะเริ่ม

วันนี้เป็นชั่วโมงติว

This is a tutorial, not a lecture!

ก่อนจะเริ่ม

“คอมพิวเตอร์มันคิดเองไม่เป็น

เราจึงต้องสอนให้มันคิด”

-- ไม่รู้จะอ้างใครดี พุดกันมาหลายคนมาก

## Formal Language

“ไปซื้อล็อตเตอรี่มาให้หน่อย เอาเลข 63 ถ้าไม่มีก็เอาอะไรใกล้ๆ มา”



ภาพ: <https://pantip.com/topic/31092061>

## Formal Language

“ไปซื้อล็อตเตอรี่มาให้หน่อย เอาเลข 63 ถ้าไม่มีก็เอาอะไรใกล้ๆ มา”

... ที่แผงมันไม่มี 63 แต่ข้างๆ มีปาทองโก๋ขาย คิดว่าสุดท้ายผมซื้ออะไรมา?



ภาพ: <https://pantip.com/topic/31092061>



ภาพ: <https://www.edtguide.com/article/416929/>

# Formal Language

- ภาษามนุษย์เป็นสิ่งที่กำกวมง่าย วิบัติง่าย ตีความผิดง่าย
- แต่มนุษย์เป็นสิ่งมีชีวิตที่ยืดหยุ่น เราจึงใช้ชีวิตกับความผิดพลาดเหล่านี้ได้
- นั่นไง ผมเขียนว่า “ยืดหยุ่น” ไม่ใช่ “ยึดหยุ่น”
- เราจึงต้องพูดกับคอมพิวเตอร์ให้เข้าใจ



# Formal Language



ภาพ: [100yen / Wikipedia](#)



ภาพ: [Thai Seafarer Community](#)

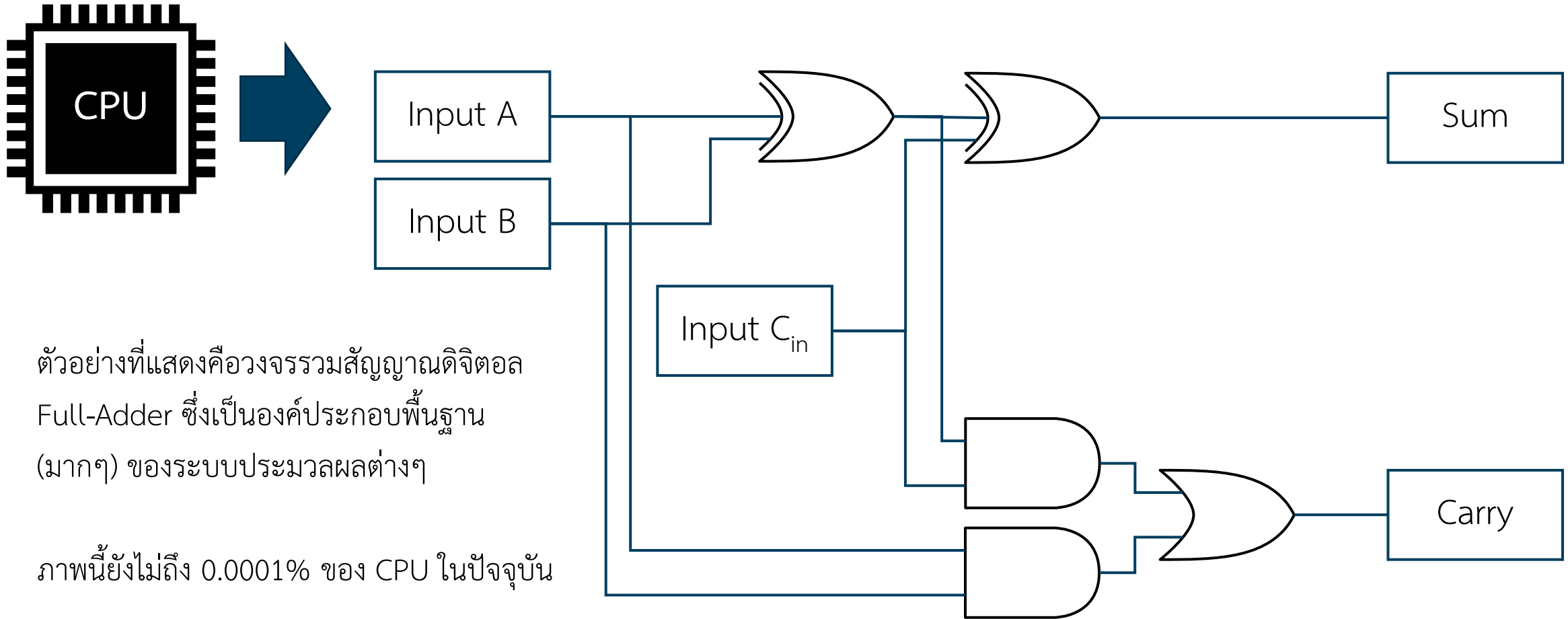


ภาพ: [US Air Force](#)

ตรงหน้าระวัง, วันทยา... วุธ!

ภาษาที่มีแบบแผน เข้าใจตรงกัน จะปฏิบัติได้เหมือนกัน ไม่ต้องคิดมาก  
ไม่ต้องตีความ เหมาะกับเรื่องที่มีกฎเกณฑ์ตายตัว

# CPU คือวงจรไฟฟ้า ที่แปลง input เป็น output



มันมีแค่ 0 กับ 1 เราจึงต้องสร้างชุดคำสั่งและภาษาให้มัน

สิ่งที่เราอยากทำ

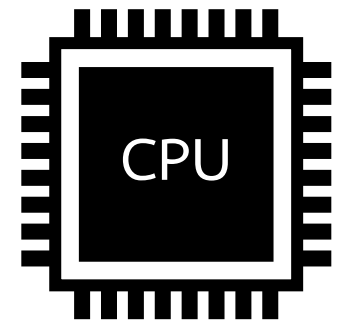
บวกเลขสองค่าในหน่วยความจำ (\$8, \$9)

บันทึกค่าลงในหน่วยความจำ (\$10)

สิ่งที่เราต้องสั่งคอมพิวเตอร์ (\*)

000000 01000 01001 01010 00000 100000

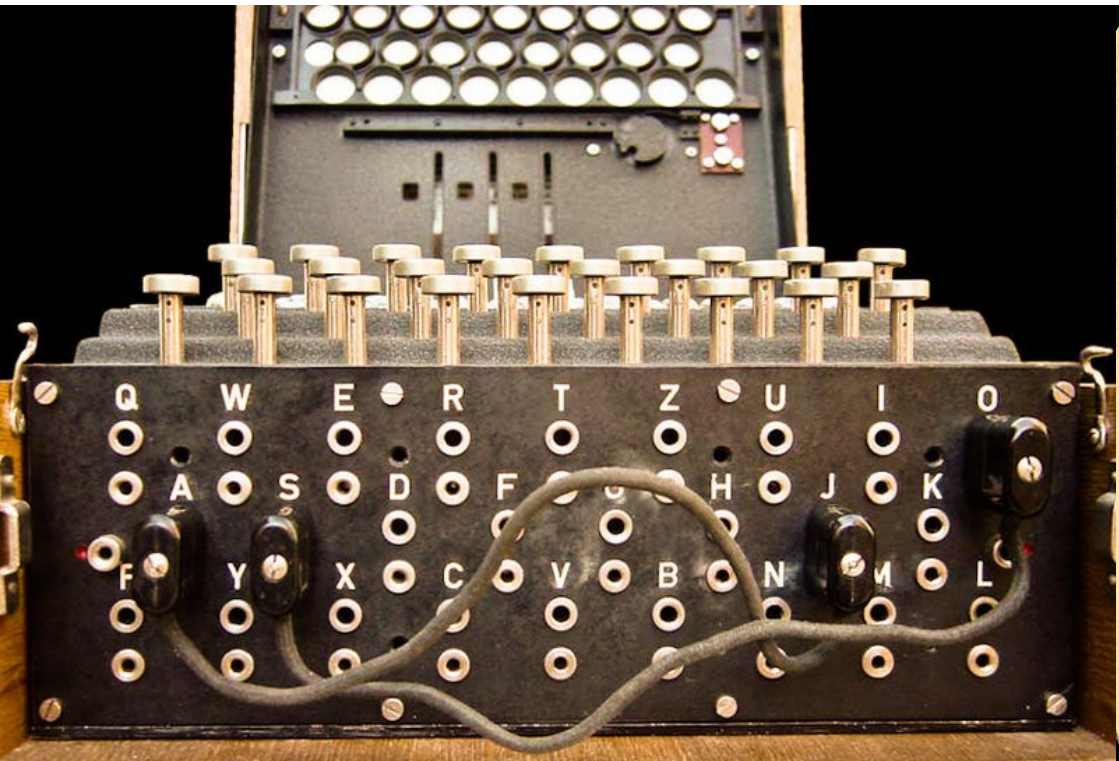
(สมมติว่าเรากำลังใช้ MIPS Instruction Set)



\* Price, C. 1995. MIPS IV Instruction Set, Revision 3.2. MIPS Technologies. (Whitepaper).



# สมัยก่อน เราต้องป้อนคำสั่งด้วยมือ หรือวิธีทางไฟฟ้า หรืออื่นๆ



แผงปลั๊ก (Plugboard) ของเครื่องเข้ารหัสอีนิกมา (Enigma)

ภาพ: [Bob Lord / Wikipedia](#)

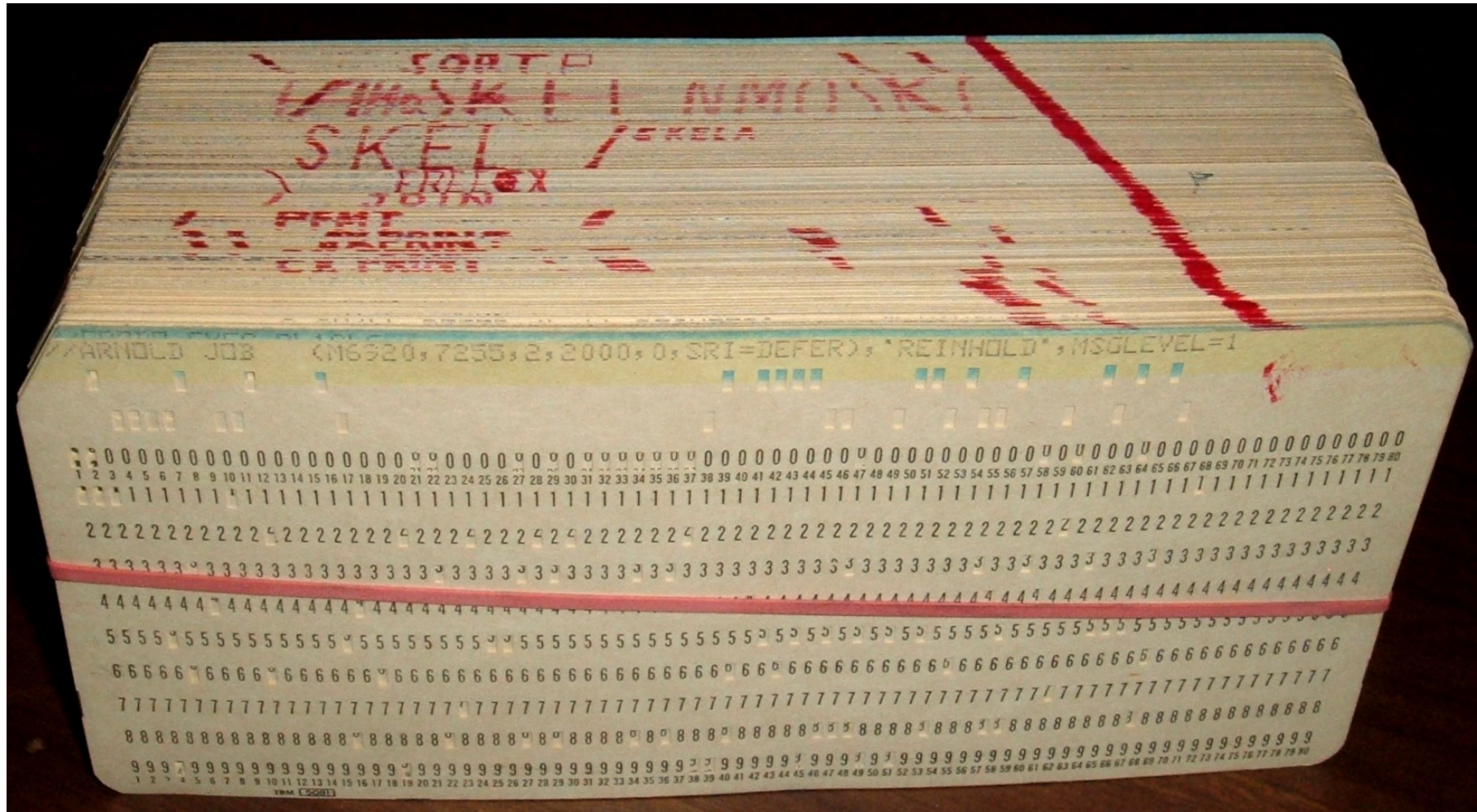


บัตรเจาะรู (punched card) เข้ารหัสโปรแกรมภาษาฟอร์แทรน (Fortran)

ภาพ: [Pete Birkinshaw / Flickr](#)



# เก็บคำสั่งต่างๆ ไว้บนคอมพิวเตอร์

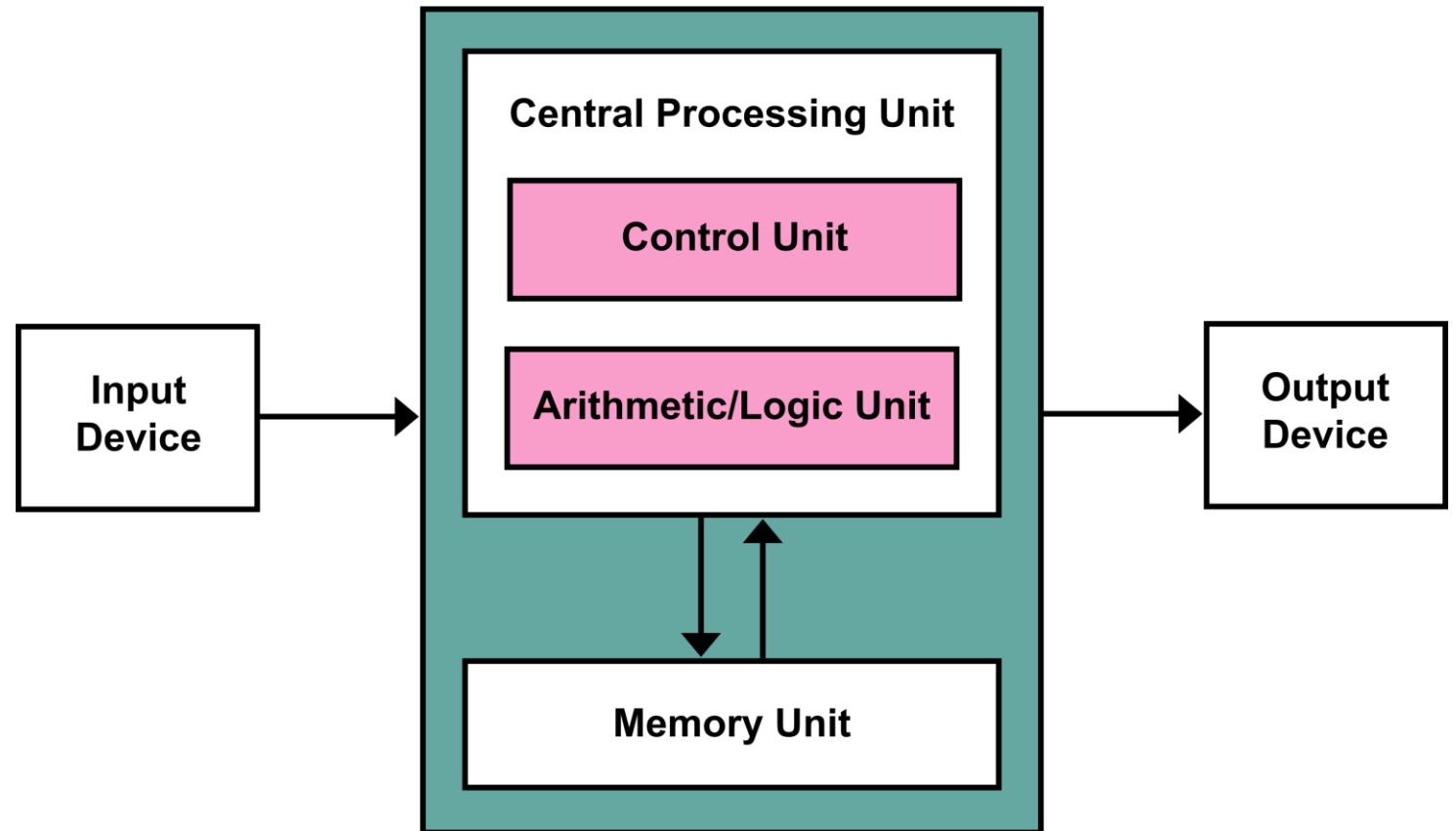


ภาพ: [Arnold Reinhold / Wikipedia](#)

# Von Neumann Architecture (ใกล้จะได้เขียนโปรแกรมแล้วครับ)

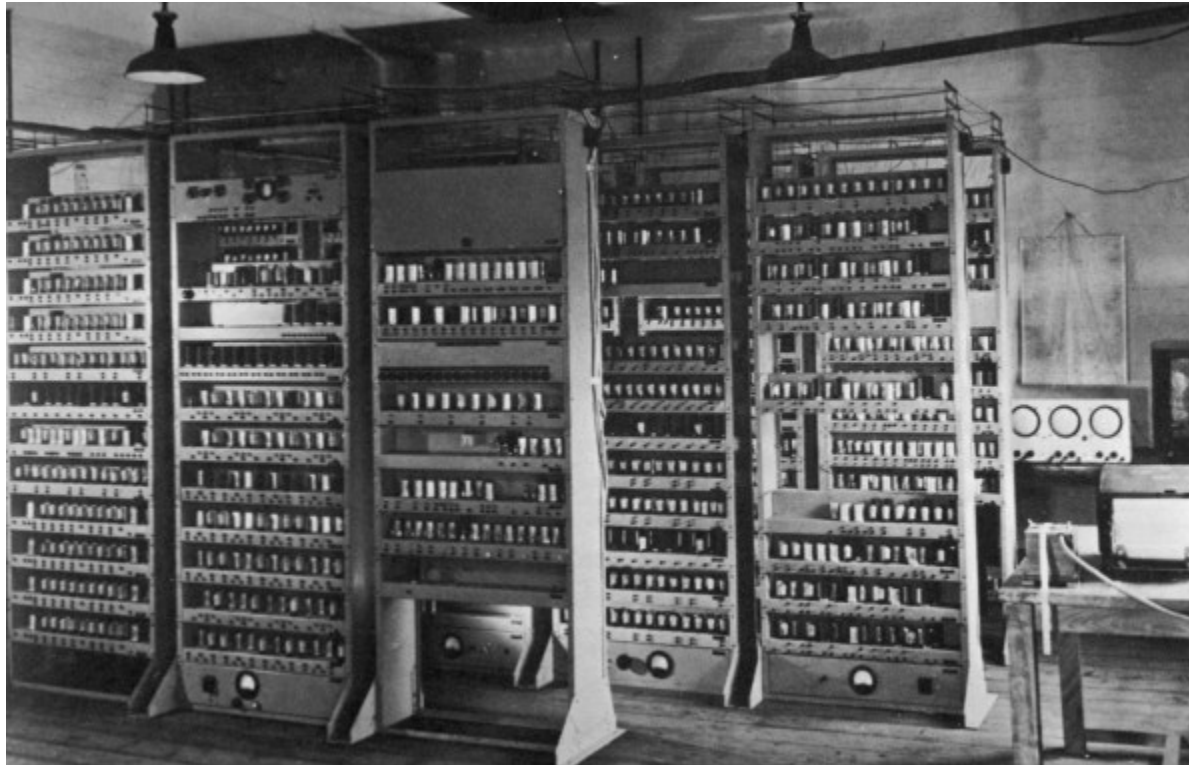


John von Neumann (1903-1957)



Von Neumann Architecture มีการเก็บข้อมูลโปรแกรมไว้ใน  
คอมพิวเตอร์ รวมกับระบบหน่วยความจำ

# Stored-program computer & Assembly Language



EDSAC (May 1949) at University of Cambridge

ภาพ: [University of Warwick](http://www.warwick.ac.uk)

Order bit pattern	Loc	Order	Meaning	Comment
00101 0 0000000000 0	0:	T0S	$m[0]=A$ ; $ABC=0$	
10101 0 0000000010 0	1:	H2S	$R=m[2]$	Put $10 \ll 11$ in R
00101 0 0000000000 0	2:	T0S	$m[0]=A$ ; $ABC=0$	
00011 0 0000000110 0	3:	E6S	goto 6	Jump to main loop
00000 0 0000000001 0	4:	P1S	data 2	The constant 2
00000 0 0000000101 0	5:	P5S	data 10	The constant 10
00101 0 0000000000 0	6:	T0S	$m[0]=A$ ; $ABC=0$	Start of the main loop
01000 0 0000000000 0	7:	I0S	$m[0]=rdch()$	Get operation code
11100 0 0000000000 0	8:	A0S	$A+=m[0]$	Put it in A
00100 0 0000010000 0	9:	R16S	$ABC \gg = 6$	Shift and store it
00101 0 0000000000 1	10:	T0L	$w[0]=AB$ ; $ABC=0$	so that it becomes the senior 5 bits of $m[0]$ $m[1]$ is now zero
01000 0 0000000010 0	11:	I2S	$m[2]=rdch()$	Put next ch in $m[2]$
11100 0 0000000010 0	12:	A2S	$A+=m[2]$	Put ch in A
01100 0 0000000101 0	13:	S5S	$A-=m[5]$	$A=ch-10$
00011 0 0000010101 0	14:	E21S	if $A \geq 0$ goto 21	Jump to 21, if $ch \geq 10$
00101 0 0000000011 0	15:	T3S	$m[3]=A$ ; $ABC=0$	Clear A, $m[3]$ is junk
11111 0 0000000001 0	16:	V1S	$AB+=m[1]*R$	$A = m[1]*(10 \ll 11)$
11001 0 0000001000 0	17:	L8S	$A \ll = 5$	Shift 5 more places
11100 0 0000000010 0	18:	A2S	$A+=m[2]$	Add the new digit
00101 0 0000000001 0	19:	T1S	$m[1]=A$ ; $ABC=0$	Store back in $m[1]$
00011 0 0000001011 0	20:	E11S	goto 11	Repeat from 11
00100 0 0000000100 0	21:	R4S	$ABC \gg = 4$	$A=2$ , if $ch='S' (=12)$ $A=15$ , if $ch='L' (=25)$
11100 0 0000000001 0	22:	A1S	$A+=m[1]$	lenbit=0, if $ch='S'$
11001 0 0000000000 1	23:	L0L	$ABC \ll = 1$	lenbit=1, if $ch='L'$ Add in the address Shift to correct position

ตัวอย่างคำสั่งที่ใช้กับ EDSAC

ภาพ: [University of Cambridge](http://www.cam.ac.uk)



# ชุมนุมนี้!

บิตที่อยู่ในโปรแกรม	บรรทัด	คำสั่ง	ความหมาย	คำอธิบาย
Order bit pattern	Loc	Order	Meaning	Comment
00101 0 0000000000 0	0:	T0S	$m[0]=A$ ; $ABC=0$	
10101 0 0000000010 0	1:	H2S	$R=m[2]$	Put $10 \ll 11$ in R
00101 0 0000000000 0	2:	T0S	$m[0]=A$ ; $ABC=0$	
00011 0 0000000110 0	3:	E6S	goto 6	Jump to main loop
00000 0 0000000001 0	4:	P1S	data 2	The constant 2
00000 0 0000000101 0	5:	P5S	data 10	The constant 10
00101 0 0000000000 0	6:	T0S	$m[0]=A$ ; $ABC=0$	Start of the main loop
01000 0 0000000000 0	7:	I0S	$m[0]=rdch()$	Get operation code
11100 0 0000000000 0	8:	A0S	$A+=m[0]$	Put it in A
00100 0 0000010000 0	9:	R16S	$ABC \gg= 6$	Shift and store it
00101 0 0000000000 1	10:	T0L	$w[0]=AB$ ; $ABC=0$	so that it becomes the senior 5 bits of $m[0]$ $m[1]$ is now zero
01000 0 0000000010 0	11:	I2S	$m[2]=rdch()$	Put next ch in $m[2]$
11100 0 0000000010 0	12:	A2S	$A+=m[2]$	Put ch in A
01100 0 0000000101 0	13:	S5S	$A-=m[5]$	$A=ch-10$
00011 0 0000010101 0	14:	E21S	if $A \geq 0$ goto 21	Jump to 21, if $ch \geq 10$
00101 0 0000000011 0	15:	T3S	$m[3]=A$ ; $ABC=0$	Clear A, $m[3]$ is junk
11111 0 0000000001 0	16:	V1S	$AB+=m[1]*R$	$A = m[1] * (10 \ll 11)$
11001 0 0000001000 0	17:	L8S	$A \ll= 5$	Shift 5 more places
11100 0 0000000010 0	18:	A2S	$A+=m[2]$	Add the new digit
00101 0 0000000001 0	19:	T1S	$m[1]=A$ ; $ABC=0$	Store back in $m[1]$
00011 0 0000001011 0	20:	E11S	goto 11	Repeat from 11

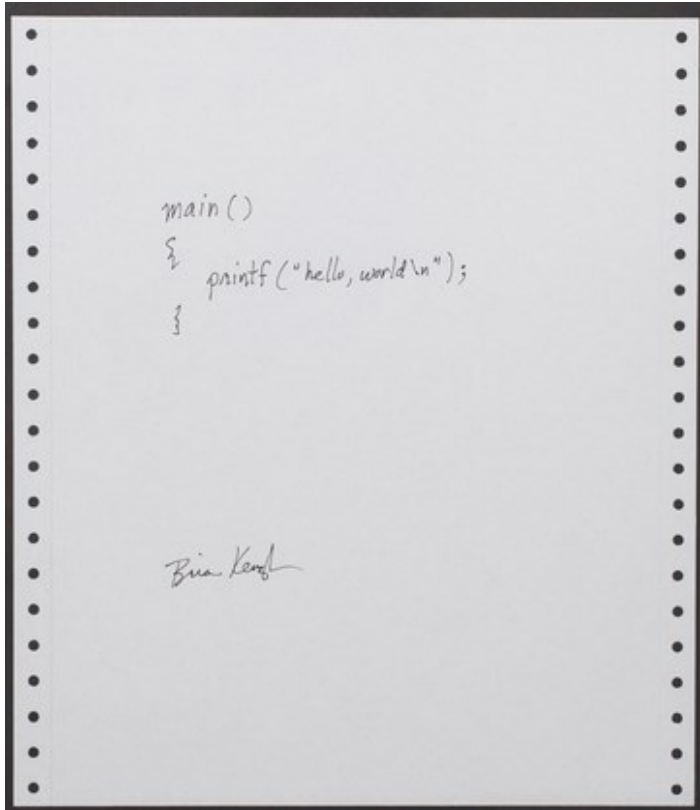
EDSAC เป็นคอมพิวเตอร์  
เครื่องที่สองที่เก็บทุกอย่างไว้  
ในระบบหน่วยความจำ ตาม  
หลักของ von Neumann (ใช้  
เทปกระดาษ) และเป็นเครื่อง  
แรกที่มีภาษา assembly เป็น  
ของตัวเอง

ตัวอย่างคำสั่งที่ใช้กับ EDSAC

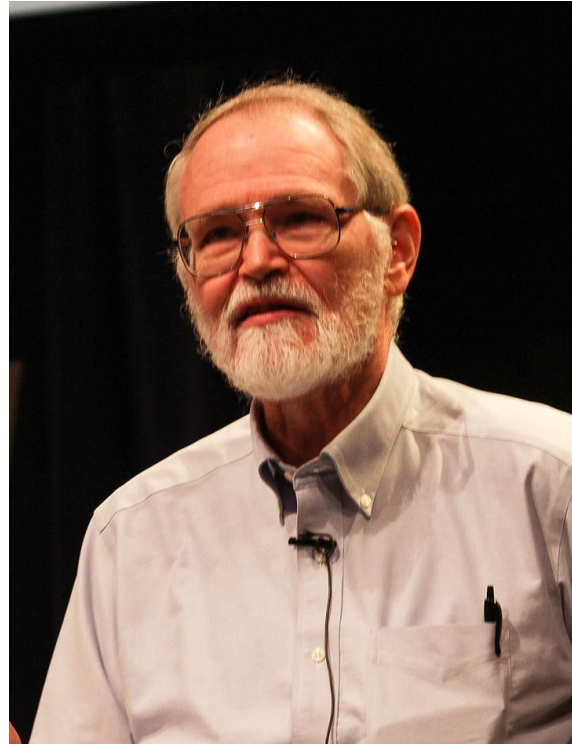
ภาพ: [University of Cambridge](https://www.youtube.com/watch?v=Kd8e0j8e0j8)



# หลังจาก Assembly เราก็พัฒนาภาษาอื่นๆ ตามมา

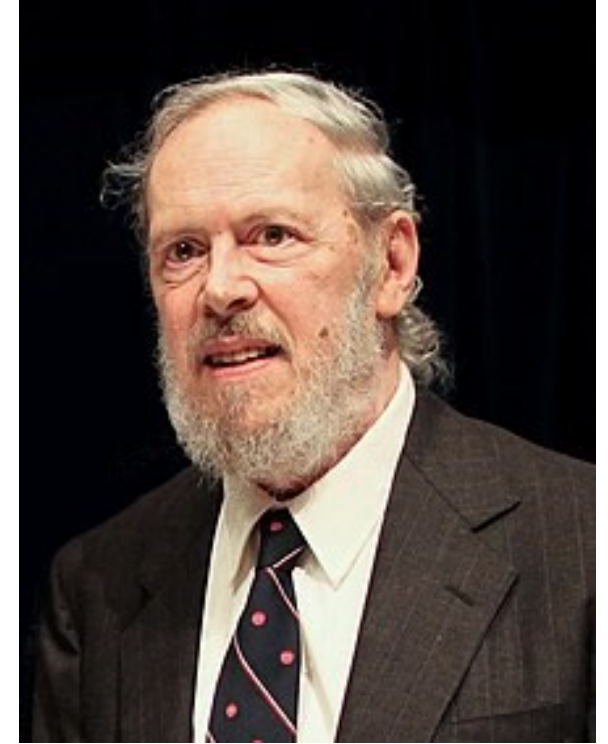


“hello, world” โดย Brian Kernighan.  
ปัจจุบัน “Hello, World” เป็นพิธีกรรม  
สากลในการเรียนเขียนโปรแกรมทุกภาษา  
ภาพ: [The Algorithm Auction](#)



Brian Kernighan (1942-)

ภาพ: [Ben Lowe / Flickr](#)



Dennis Ritchie (1941-2011)

ภาพ: [Denise Panyik-Dale / Flickr](#)

ภาษาไพธอน

Python Language

หลักการของภาษา  
(อ้างอิง: PEP 20, The  
Zen of Python)



เน้นความเรียบง่าย

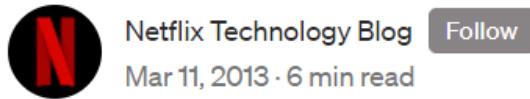


ทำอะไรให้ตรงไปตรงมา

# ภาษาไพธอนใช้ทำอะไรได้บ้าง



## Python at Netflix



Mar 11, 2013 · 6 min read



by Roy Rapoport, Brian Moyles, Jim Cistaro, and Corey Bertram

We've blogged a lot about how we use Java here at Netflix, but Python footprint in our environment continues to increase. In honor of our sponsorship of PyCon, we wanted to highlight our many uses of Python at Netflix.

<https://netflixtechblog.com/python-at-netflix-86b6028b3b3e>



Sign in

Get started



MACHINE LEARNING

ANDROID

IOS

INFRASTRUCTURE

DATA

OPEN SOL

## Web Service Efficiency at Instagram with Python



Instagram Engineering

Follow

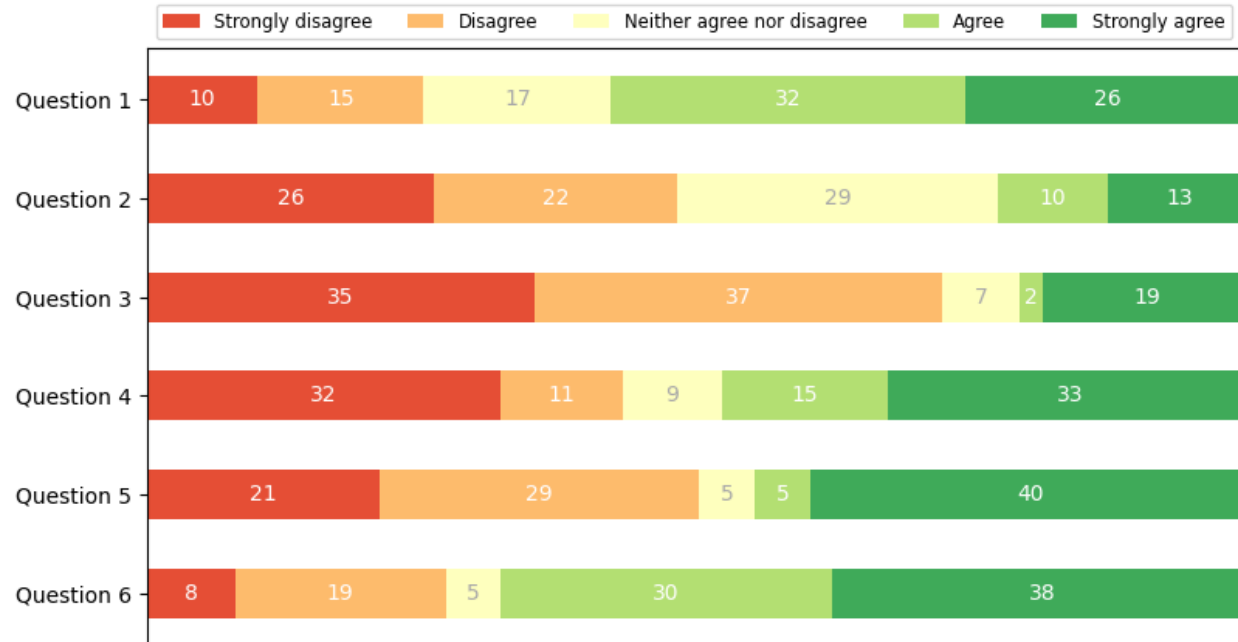
Jun 21, 2016 · 6 min read



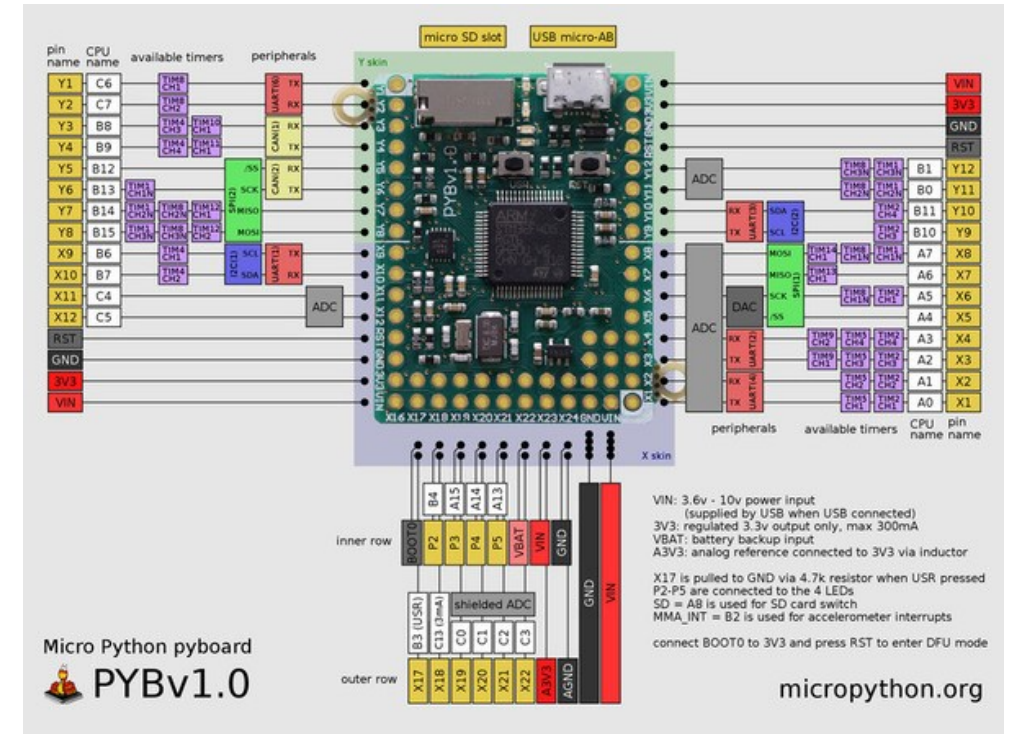
Instagram currently features the world's largest deployment of the Django web framework, which is written entirely in Python. We initially chose to use Python because of its reputation for simplicity and practicality, which aligns well with our philosophy of "do the simple thing first." But simplicity can come with a tradeoff: efficiency. Instagram has doubled in size over the

<https://instagram-engineering.com/web-service-efficiency-at-instagram-with-python-4976d078e366>

# ภาษาไพธอนใช้ทำอะไรได้อีก



[https://matplotlib.org/gallery/lines\\_bars\\_and\\_markers/horizontal\\_barchart\\_distribution.html](https://matplotlib.org/gallery/lines_bars_and_markers/horizontal_barchart_distribution.html)



<https://micropython.org/>

# ผู้สอนก็หากินกับไพธอน

## อัลกอริทึมสำหรับหาเส้นทางในเครือข่าย

**Algorithm 1** Algorithm to find a path set to route from S1 to S2 in network graph G

**Require:** graph  $G(V, E)$

**Require:**  $S1, S2 \in \text{switches}$

$G(V, E) \leftarrow \text{NetworkTopology}(\text{switches}, \text{links})$

$\text{PrimaryPath} \leftarrow \text{shortest\_path}(G, S1, S2)$

$\text{AltPaths} \leftarrow \text{all\_simple\_paths}(G, S1, S2) - \text{PrimaryPath}$

$\text{AltPaths} \leftarrow \text{AltPaths}$  sorted by number of edges shared with PrimaryPath ascending, by length of path ascending

$\text{PathSet} \leftarrow \text{PrimaryPath} + \text{AltPaths}$

**return** PathSet

([Nakasan et al., 2017](#))

เขียนเป็นภาษาไพธอน

<https://github.com/LunaticNeko/smoc/blob/master/overseer/overseer.py>

```
def get_path(self, from_dpid, to_dpid, packet):
    # TODO: Support IPv6

    tcp_packet = packet.find("tcp")
    udp_packet = packet.find("udp")
    ip_packet = packet.find("ipv4")

    """
    PATH RULES
    For ordinary packets and MP_CAPABLE (primary) connections:
        - Use *the* shortest path
    For MP_JOIN (secondary subflows) connections:
        - Use least-conflicting, shortest path that's not used above
    """

    # get shortest paths
    shortest_path = nx.shortest_path(core.overseer_topology.graph, from_dpid, to_dpid)

    if tcp_packet is None:
        return shortest_path

    # get all paths
    alt_paths = list(nx.all_simple_paths(core.overseer_topology.graph, from_dpid, to_dpid))
    alt_paths.remove(shortest_path)

    # prevents path assignment error on single path (retain backwards-compat)
    if alt_paths == []:
        alt_paths.append(shortest_path)

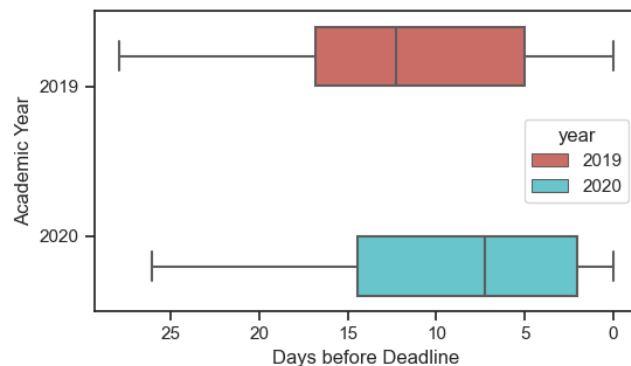
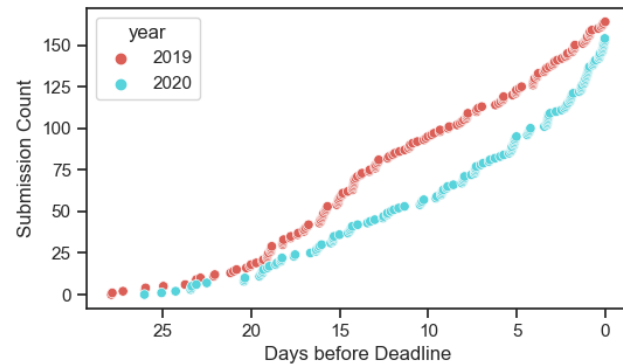
    alt_paths = path_utils.sort_path_list(shortest_path, alt_paths)

    # if MP_JOIN packet detected
    if tcp_packet is not None:
        for option in tcp_packet.options:
            # self.log.debug('%s %s' % (option.type, type(option.val)))
            if option.type == TCP_OPTION_KIND_MPTCP:
                mptcp_subtype = option.subtype
                self.log.info('MPTCP opt: %s' % MPTCP_SUBTYPES[mptcp_subtype])
                if mptcp_subtype == 1:
                    # pick the first Alt. Path
                    self.log.info("MP_JOIN => Alt. Path")
                    self.log.info("PATH: %s" % (alt_paths[0]))
                    return alt_paths[0]
                elif mptcp_subtype == 0:
                    self.log.info("MPTCP %s => Primary Path" % (MPTCP_SUBTYPES[mptcp_subtype]))
                    self.log.info("PATH: %s" % (shortest_path))
                    return shortest_path
                else:
                    return shortest_path

    # if all else fails, use default Overseer behavior
    return nx.shortest_path(core.overseer_topology.graph, from_dpid, to_dpid)
```

# ข้อมูลการศึกษา ก็ทำได้

นักศึกษาส่งการบ้านช้าลงในปี 2020!



กราฟ (บน) แสดงเวลาการส่งการบ้านของนักศึกษา วิชาการประมวลผลข้อมูลพื้นฐาน มหาวิทยาลัยคานาซาวะ ตามจำนวนวันก่อนถึงกำหนดส่ง (สองหมู่เรียน สองปี การศึกษา) และ (ล่าง) แสดงควอไทล์ (งานของผู้สอน ยังไม่ได้ตีพิมพ์)

## คะแนนสอบ “ภาคคอม” #TCAS63

เกษตรรับเยอะ เข้าไม่ยาก แต่ตัวท็อปก็ไม่แพ้จุฬา!



กราฟแสดงคะแนนสอบและจำนวนสมัครและรับเข้าภาควิชาหรือ สาขาวิชาเกี่ยวกับคอมพิวเตอร์ของมหาวิทยาลัยในระบบ TCAS63 (บางส่วน) (งานของผู้สอน ยังไม่ได้ตีพิมพ์)

# Demo: เริ่มเขียนภาษาไพธอน

- PyCharm Configuration
- Hello, World [000-helloworld.py]
  - Basic print() and comments



# เราถึงไหนแล้วนะ ... ?

ภาษาไพธอน  
Python Language

ข้อมูลและตัวแปร  
Data and Variables

การรับและแสดงข้อมูล  
แบบมาตรฐาน  
Standard Input/Output

ตัวดำเนินการ  
Operators

การตรวจสอบเงื่อนไข  
Conditions (if, if-else)

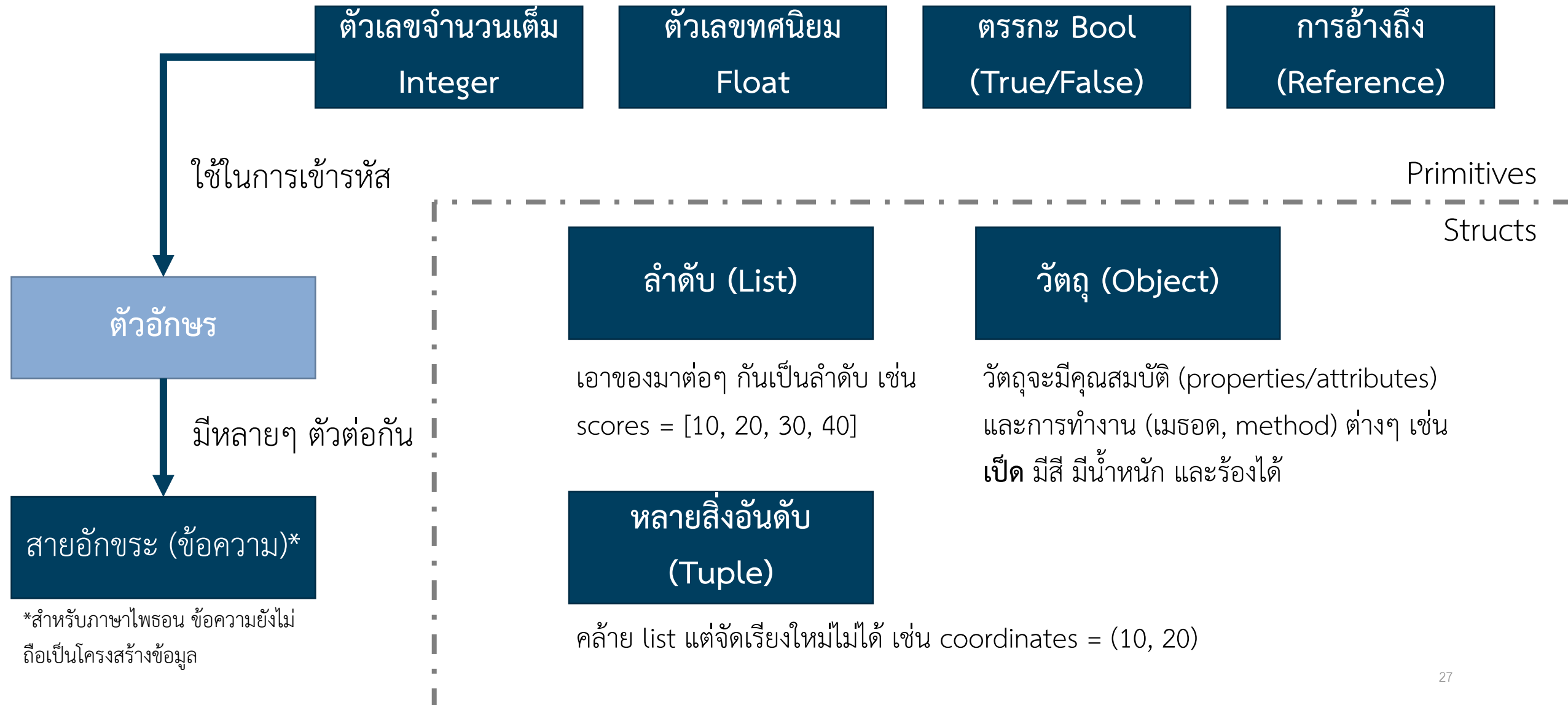
การวนซ้ำ  
Repetition (for, while)

หลังจากการบรรยายวันนี้: นักศึกษามีความรู้พื้นฐาน สามารถเขียนโปรแกรมภาษาไพธอนแบบง่ายๆ เพื่อแก้โจทย์ระดับพื้นฐาน เช่น การรับและแสดงข้อมูล, การคำนวณพื้นฐาน, การตรวจสอบเงื่อนไข, และการวนซ้ำได้

ข้อมูลและตัวแปร

Data and Variables

# ชนิดข้อมูลพื้นฐาน (Primitive Data Types) และโครงสร้างข้อมูล (Structs)




# Demo: Data Types [100-datatypes.py]

- Assigning variables
- Inspecting Data Types
- int, float, bool
- string
- list, tuples

# int

- Int พุดง่าย ๆ คือเอาเลขจำนวนเต็มแปลงเป็นฐานสองแล้วยัดลงไปหน่วยความจำ
- ไพธอนไม่แคร์เรื่องขนาดตัวแปร (ซึ่งดีแล้วสำหรับเรา)

32 bits หรืออาจเป็นขนาดอื่นๆ ก็ได้  
  
 00000000 00000000 00000000 10000001<sub>2</sub> = 129<sub>10</sub>

(สำหรับค่าติดลบ จะใช้ระบบ 2's Complement ซึ่งขอละไว้ในวันนี้)

# float

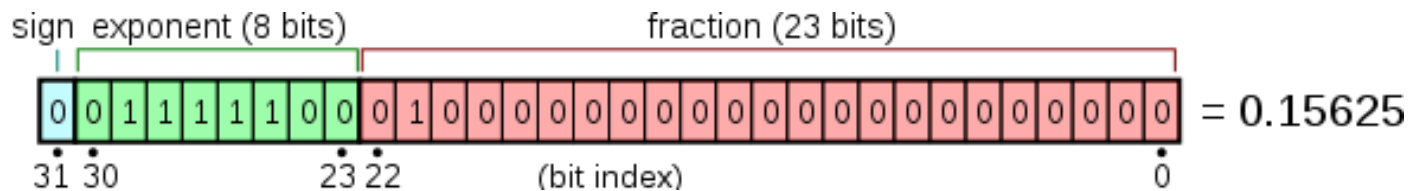
**Float** ใช้การเก็บค่าด้วยมาตรฐานการเข้ารหัสต่างๆ เช่น IEEE 754

ทำให้สามารถเก็บค่าได้กว้างมากเช่น ตัวแปร 32-bit เก็บได้สูงสุด

$$(2 - 2^{-23}) \times 2^{127} \approx 3.4028235 \times 10^{38}$$

หรือประมาณ 3,402,823,5xx,xxx,xxx,xxx,xxx,xxx,xxx,xxx,xxx,xxx

แต่มีความแม่นยำจำกัด และการบวกลบมีความยุ่งยากมากกว่าจำนวนเต็ม



[https://en.wikipedia.org/wiki/IEEE\\_754](https://en.wikipedia.org/wiki/IEEE_754)

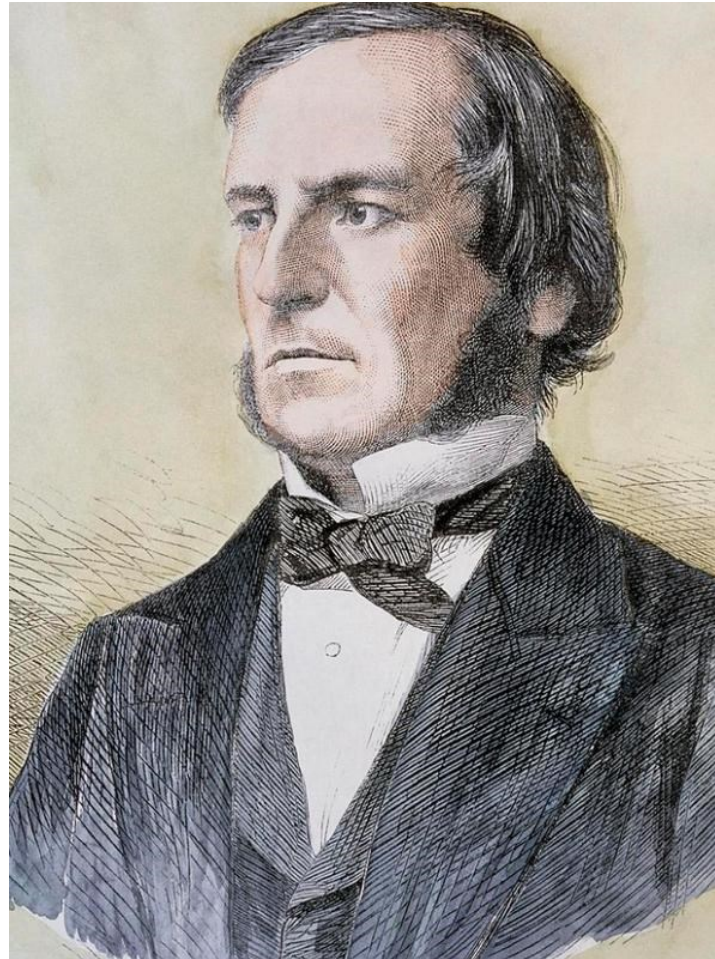
ลองแปลงค่าเล่นได้ที่ <https://www.h-schmidt.net/FloatConverter/IEEE754.html>

เบรกแป้นนะ ... แล้วในเมื่อ float เก็บค่าได้กว้างกว่า ทำไมเราไม่ใช่ float แทน int?

- เพราะ float ไม่สามารถเก็บ “จำนวนเป๊ะ” ได้ เนื่องจากมันไม่ได้ “เก็บเลขทุกหลัก” เหมือน int
- เดี่ยวทำให้ดูใน demo ตอนนี้อนผ่านไปก่อน

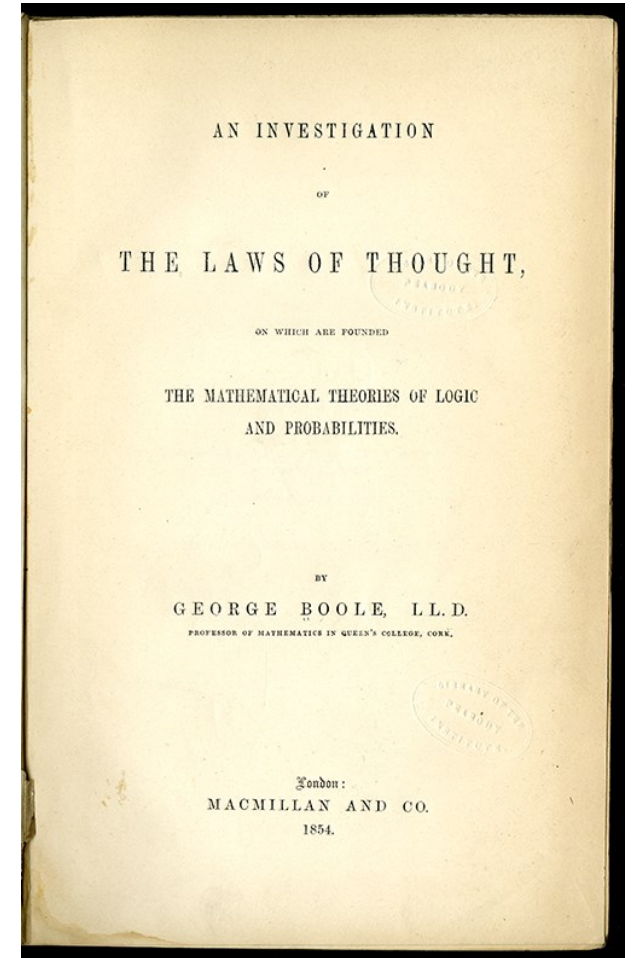
# bool

- มีค่าแค่ True กับ False (ขึ้นต้นด้วยตัวใหญ่)
- ใช้เวลาทำการเปรียบเทียบตรรกะต่างๆ หรือแสดงความเป็นจริงไม่จริงของสิ่งต่างๆ
- เป็นสิ่งที่ใช้เยอะอย่างมหาศาลในการเขียนโปรแกรม แม้ที่เราอาจไม่ได้ใช้คำว่า True หรือ False อยู่ตลอดเวลาก็ตาม



George Boole (1815-1864)

ภาพ: [ที่มาไม่ชัดเจน / Wikipedia](#)



ภาพ: [MAA](#)



string



ดิอิมพอสซิเบิล



รอยัลสไปรท์ส

โอเค ขอโทษที่รับ string (จริงๆ ละ)

ต้องรอบด้วยเครื่องหมายคำพูด (อัญประกาศ) เดี่ยวหรือคู่ก็ได้

S1 = "This is a sentence."  
S2 = 'This is a sentence.'

000000000011111111

0123456789012345678

อักขระแต่ละตัวมี “ตำแหน่ง” ของมัน เริ่มจากศูนย์

string

**S1 = "This is a sentence."**

0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

S1[0] == ?

S1[0:5] == ?

S1[9] == ?

S1[7:] == ?

S1[16] == ?

S1[:12] == ?

# list & tuple

- คือการเก็บของหลายๆ อย่างเป็นลำดับ
- List เป็นลำดับของสิ่งที่สลับไปมาได้ หรือมีการเพิ่มลดได้ เช่น  $L = [\text{หมา}, \text{แมว}, \text{กระต่าย}]$ 
  - สามารถอ้างถึงสิ่งของด้วยวงเล็บเหลี่ยมได้เหมือนสตริง  $L[2]$  คือ กระต่าย
  - สามารถใช้ของข้างในที่มีชนิดข้อมูลต่างกันได้ เช่น  $LL = [1, \text{"HAHAHA"}, 1.222222]$
- Tuple คล้าย list แต่แก้ไข เพิ่มลดไม่ได้ ตายตัวกว่า เหมาะกับค่าที่ต้องไปด้วยกันตลอด เช่น คู่อันดับ  $(x,y)$

# Demo: Data Types (Advanced) [101-datatypes-hard.py]

- Int vs Float
- Addressing
- Mutability

# การรับและแสดงข้อมูลแบบมาตรฐาน

## Standard Input/Output

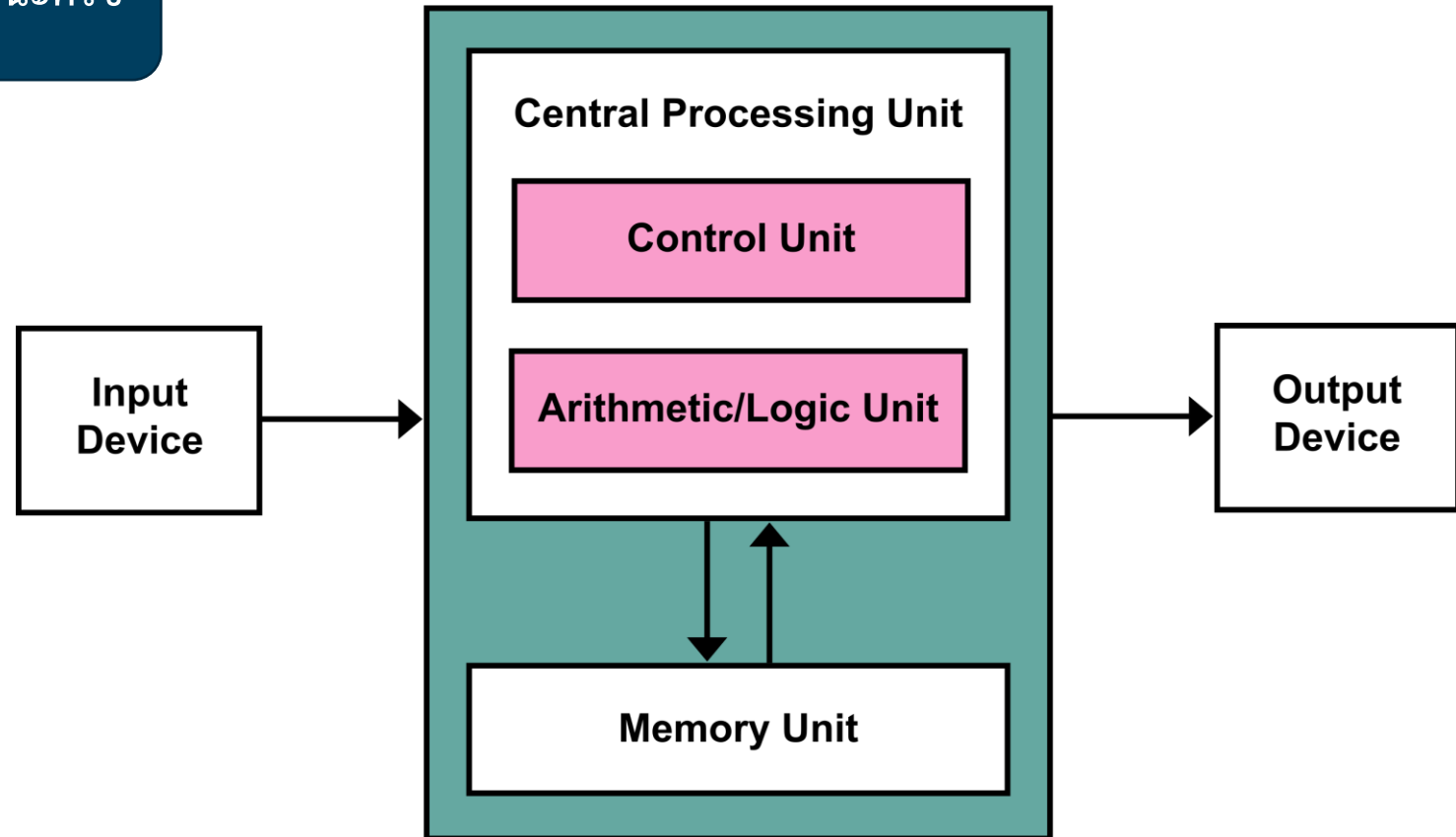
# ระบบคอมพิวเตอร์มี input และ output

กลับมาเจอกันอีกแล้วนะครัช



John von Neumann  
(1903-1957)

ภาพ: [Los Alamos National Laboratory](#)

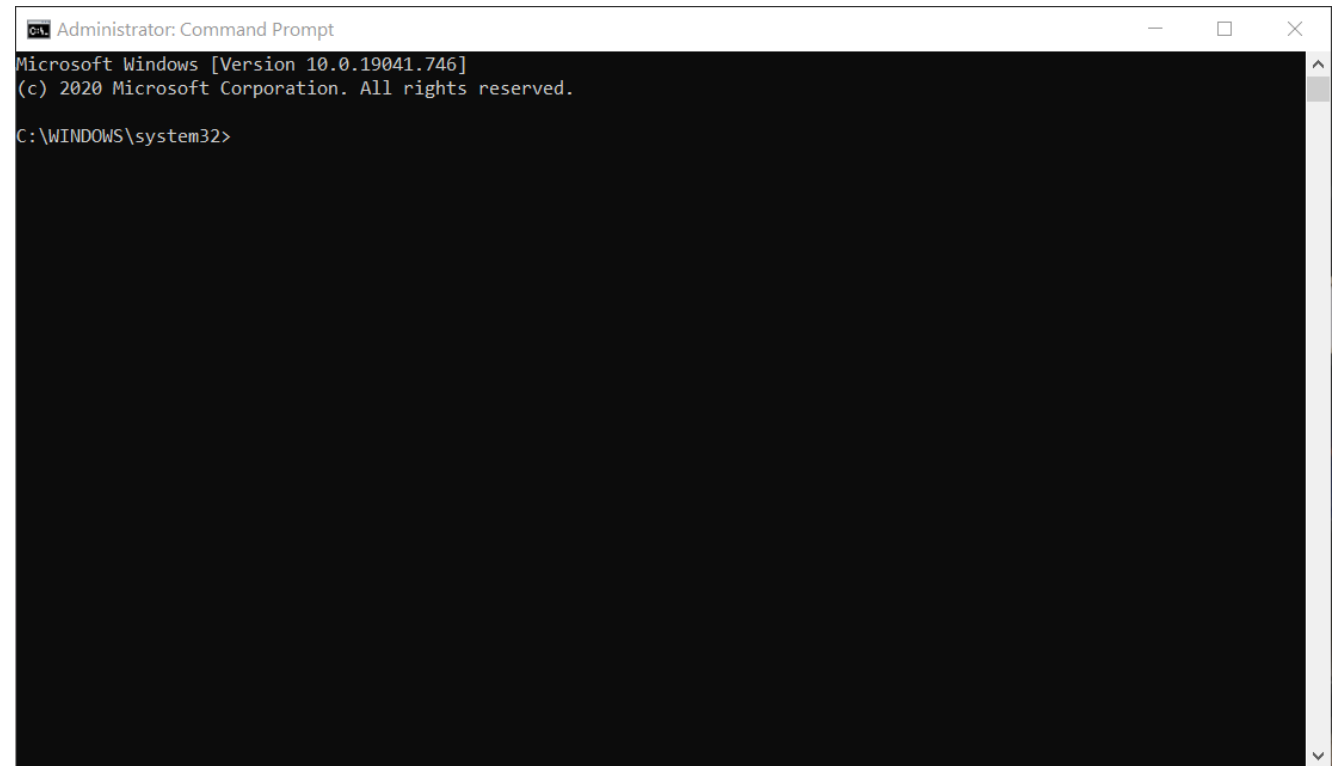


Von Neumann Architecture

ภาพ: [Kapooht / Wikipedia](#)

# Standard Input/Output

- Standard Input (stdin)  
แปลว่า การพิมพ์เข้าไปใน  
หน้าจอ cmd
- Standard Output (stdout)  
แปลว่า ข้อความออกมาทาง  
cmd
- stdin & stdout เหมือนกัน  
เกือบทุกระบบปฏิบัติการ และ  
เหมือนกันทุกภาษา





# คำสั่งพื้นฐานสำหรับ Standard Input/Output

- การพิมพ์อะไรออกทาง stdout ปกติใช้คำสั่ง print
- การรับข้อความ ใช้คำสั่ง input

## Demo: I/O ခြံး [200-stupidio.py]

- print()
- input()

## Demo: File I/O [201-fileio.py]

- การเปิดไฟล์ตามชื่อ ด้วยคำสั่ง `open` และการใช้บริบท `with` เพื่อให้เปิดปิดไฟล์อย่างปลอดภัย
- ข้อควรระวังเมื่อเปิดไฟล์
  - ต้องใส่ 'w' หากต้องการเขียนค่าด้วย (หรือ 'a' หากต้องการต่อท้ายไฟล์) หากจะอ่านอย่างเดียว ควรใส่ 'r'
  - หากเปิดไฟล์โดยไม่ใช้บริบท `with` ต้องปิดไฟล์เองด้วยคำสั่ง `f.close()` ไม่เช่นนั้นอาจเกิดข้อผิดพลาดได้
  - ระวังการเขียนทับไฟล์ระบบ
  - โหมด 'a' ที่ใช้ต่อท้ายไฟล์ เหมาะสำหรับการทำงานที่เขียนเพิ่มเติม ไม่ได้จะสร้างเนื้อหาทั้งหมดหรือทับเนื้อหาเก่า เช่น การเขียนข้อความลงไฟล์บันทึก (log) เป็นต้น (ไม่แสดงในตัวอย่างวันนี้)

# ตัวดำเนินการ

# Operators

# Arithmetic Operators (ตัวดำเนินการเลขคณิต)

- ตัวดำเนินการ คือ การบวกลบคูณหาร อะไรพวกนี้ ใช้เครื่องหมาย  $+$   $-$   $*$   $/$
- การยกกำลัง ใช้  $**$
- การหารแบบตัดเศษทิ้ง ใช้  $//$
- การหารเอาแต่เศษ ใช้  $\%$

# Comparison Operators (ตัวดำเนินการเปรียบเทียบ)

- $>$   $<$   $>=$   $<=$
- $==$  คือการเท่ากัน (= เดียว หมายถึงการกำหนดตัวแปร อย่าสับสน)
- $!=$  หมายถึง ไม่เท่ากัน

# Logical Operators (ตัวดำเนินการตรรกะ)

- มีแค่ and, or, not
- $x$  and  $y$  เป็น True เมื่อ  $x$  และ  $y$  เป็น True
- $x$  or  $y$  เป็น True เมื่อ  $x$  หรือ  $y$  อย่างน้อยตัวหนึ่ง เป็น True
- not  $x$  เป็น True เมื่อค่าของ  $x$  คือ False
  
- 0 มีความหมายเหมือน False
- 1 มีความหมายเหมือน True



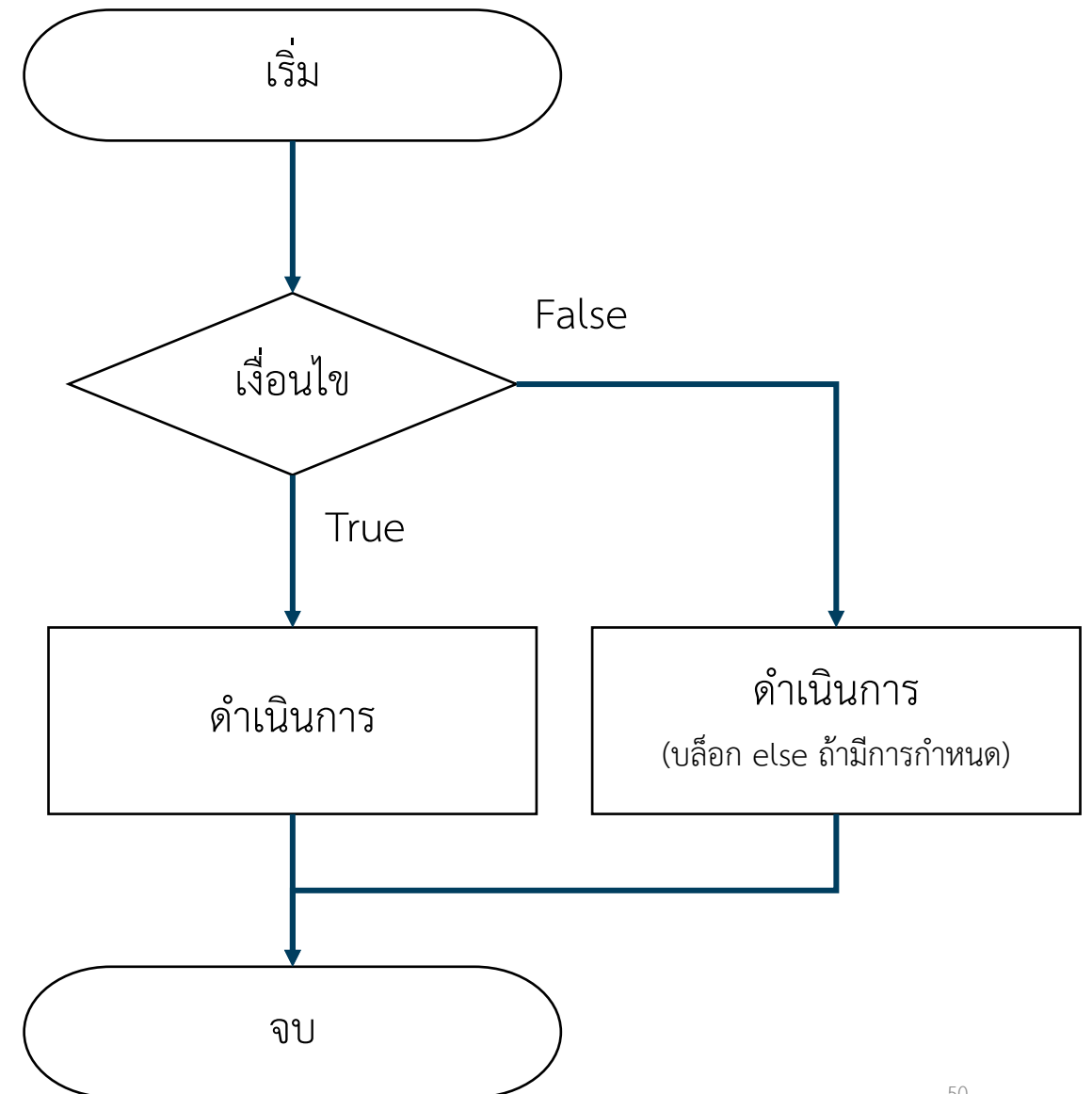
Demo [300-operators.py]

# การตรวจสอบเงื่อนไข

## Conditions (if, if-else)

# Conditions

- คำสั่ง if และ if else ใช้ในการเลือกการทำงาน  
ของโปรแกรมตามเงื่อนไขต่างๆ
- Demo ยาวๆ ไป
  - [400-if.py]
  - [401-ifelse.py]
  - [402-ifelifelse.py]



การวนซ้ำ

Repetition (for, while)

# Repetitions

- คือการให้โปรแกรมทำงานวนซ้ำ
  - for = ตามเงื่อนไขหรือรอบที่กำหนดแต่แรก
  - while = ตรรกะใดที่เงื่อนไขยังเป็นจริง
- Demo ยาวๆ ไป
  - [500-forrange.py]
  - [501-foreach.py]
  - [502-while.py]

