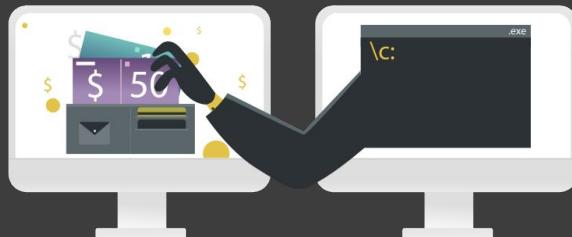


NCSA CTF Boot Camp #2

Programming for CTF

Responsible: Mr. Yasinthon Khemprakhon
Version (Date): 1.0 (2024-09-15)
Confidentiality class: Public



whoami



Pichaya (LongCat) Morimoto

Lead Penetration Tester
Siam Thanat Hack Co., Ltd.



Peeratach (Peter) Butto

Penetration Tester
Siam Thanat Hack Co., Ltd.



Yasinthon (Not) Khemprakhon

Penetration Tester
Siam Thanat Hack Co., Ltd.



Disclaimer

- จุดประสงค์ของการบรรยาย นี้เพื่อแบ่งปันความรู้ ทางด้านความปลอดภัยระบบสารสนเทศ
- ไม่สนับสนุนการนำความรู้ทางด้านความปลอดภัยฯ ไปใช้ในทางที่ผิดกฎหมายทั้งหมด
- ตัวอย่างໂຄດ และรูปใน การบรรยาย นี้ เป็นระบบจำลองของทางผู้บรรยาย ไม่ใช่ระบบลูกค้า



Agenda (Day 2)

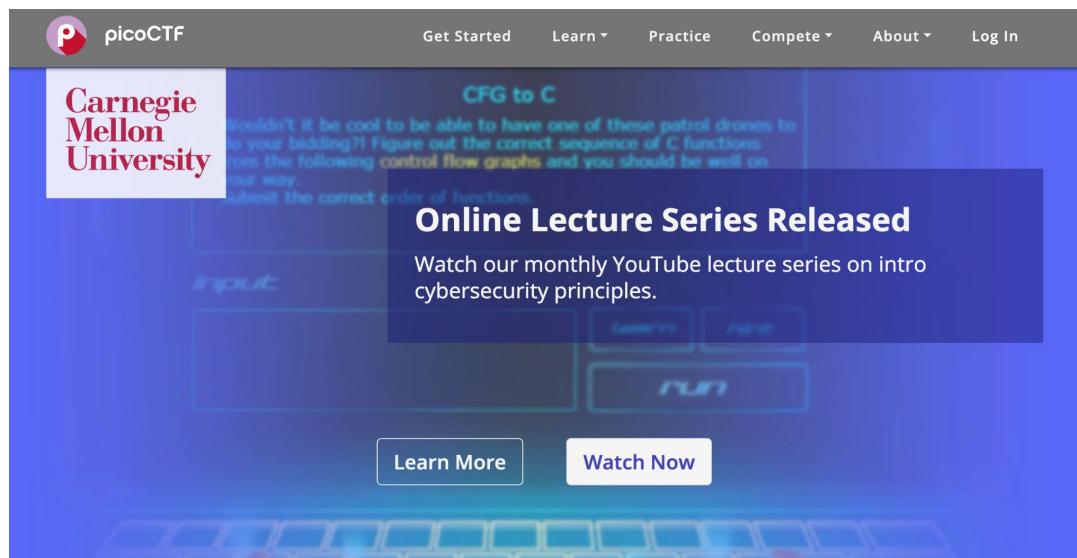
เวลา	รายละเอียด
09.00 - 10.30	Mobile Application Security
10.30 - 10.45	พักเบรก
10.45 - 12.00	Programming for CTF
12.00 - 13.00	พักรับประทาน อาหารกลางวัน
13.00 - 16.00	การแข่งขัน CTF (3 ชม.)
16.00 - 17.00	มอบรางวัลและพิธีปิด

Content Overview

- Lab 1 ลองทำ Socket อ่านข้อมูล
 - Part 1 - อ่านข้อมูลจาก Server
 - Part 2 - แปลงข้อมูลเป็นค่า Flag
- TCP Socket Flow Diagram
- Lab 2 ลองทำ Socket ตอบกลับข้อมูล
 - Part 1 - จำแนกข้อมูลจาก Server
 - Part 2 - คำนวณและส่งคำตอบ
- การทำ HTTP Client
 - ลองใช้ requests Module

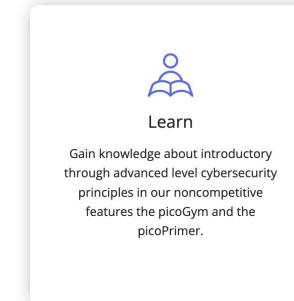


Credit for Challenges from PicoCTF

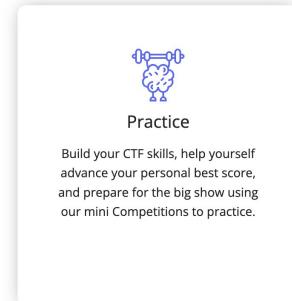


The screenshot shows the picoCTF website homepage. At the top, there's a navigation bar with links for "Get Started", "Learn", "Practice", "Compete", "About", and "Log In". A "picoCTF" logo is on the left. A banner for Carnegie Mellon University features the text "Online Lecture Series Released" and "Watch our monthly YouTube lecture series on intro cybersecurity principles". Below this, there's a section titled "CFG to C" with a challenge description. At the bottom, there are two buttons: "Learn More" and "Watch Now".

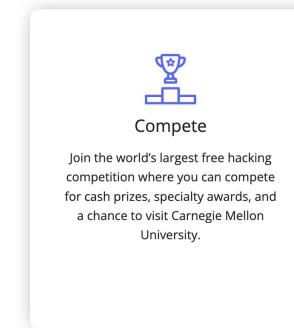
<https://picoctf.org>



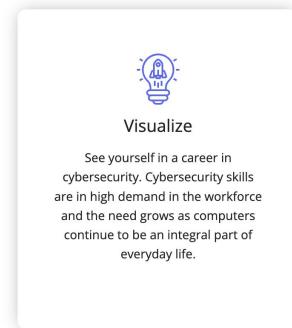
 **Learn**
Gain knowledge about introductory through advanced level cybersecurity principles in our noncompetitive features the picoGym and the picoPrimer.



 **Practice**
Build your CTF skills, help yourself advance your personal best score, and prepare for the big show using our mini Competitions to practice.



 **Compete**
Join the world's largest free hacking competition where you can compete for cash prizes, specialty awards, and a chance to visit Carnegie Mellon University.



 **Visualize**
See yourself in a career in cybersecurity. Cybersecurity skills are in high demand in the workforce and the need grows as computers continue to be an integral part of everyday life.

Lab 1 - ลองทำ Socket อ่านข้อมูล



\$ go run lab1_server.go
Server listening on port 43239...

```
(py311) [pwn 6_prog ]$  
(py311) [pwn 6_prog ]$  
(py311) [pwn 6_prog ]$  
(py311) [pwn 6_prog ]$  
(py311) [pwn 6_prog ]$ go run lab1_server.go  
Server listening on port 43239...
```

\$ nc 127.0.0.1 43239



```
(py311) [pwn ~ ]$ nc 127.0.0.1 43239  
79  
76  
65  
71  
123  
65  
83  
67  
73  
73  
95  
78  
85  
77  
66  
69  
82  
83  
125
```

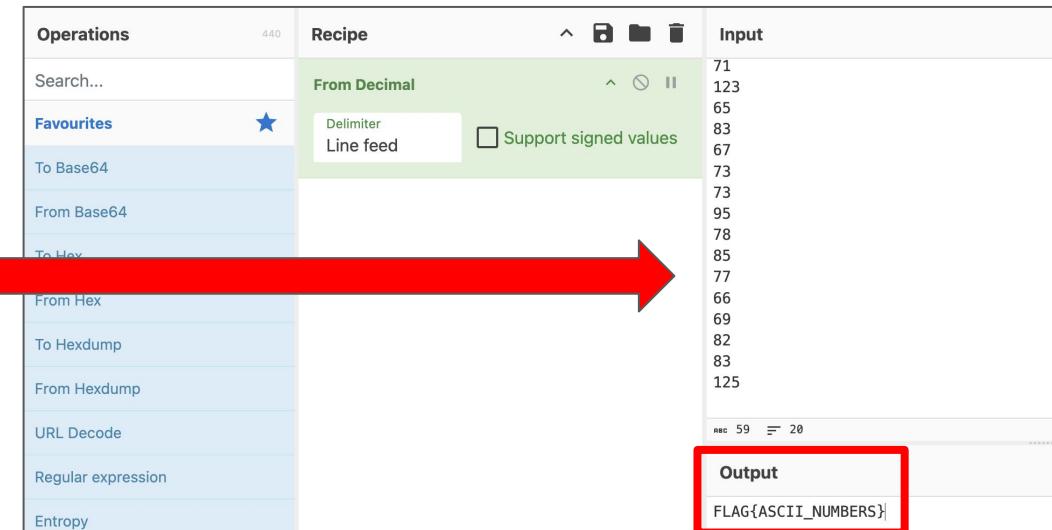
Lab 1 - ลองทำ Socket อ่านข้อมูล

Dec	Char	Dec	Char	Dec	Char	Dec	Char
0	NUL (null)	32	SPACE	64	@	96	`
1	SOH (start of heading)	33	!	65	A	97	a
2	STX (start of text)	34	"	66	B	98	b
3	ETX (end of text)	35	#	67	C	99	c
4	EOT (end of transmission)	36	\$	68	D	100	d
5	ENQ (enquiry)	37	%	69	E	101	e
6	ACK (acknowledge)	38	&	70	F	102	f
7	BEL (bell)	39	'	71	G	103	g
8	BS (backspace)	40	(72	H	104	h
9	TAB (horizontal tab)	41)	73	I	105	i
10	LF (NL line feed, new line)	42	*	74	J	106	j
11	VT (vertical tab)	43	+	75	K	107	k
12	FF (NP form feed, new page)	44	,	76	L	108	l
13	CR (carriage return)	45	-	77	M	109	m
14	SO (shift out)	46	.	78	N	110	n
15	SI (shift in)	47	/	79	O	111	o
16	DLE (data link escape)	48	0	80	P	112	p
17	DC1 (device control 1)	49	1	81	Q	113	q
18	DC2 (device control 2)	50	2	82	R	114	r
19	DC3 (device control 3)	51	3	83	S	115	s
20	DC4 (device control 4)	52	4	84	T	116	t
21	NAK (negative acknowledge)	53	5	85	U	117	u
22	SYN (synchronous idle)	54	6	86	V	118	v
23	ETB (end of trans. block)	55	7	87	W	119	w
24	CAN (cancel)	56	8	88	X	120	x
25	EM (end of medium)	57	9	89	Y	121	y
26	SUB (substitute)	58	:	90	Z	122	z
27	ESC (escape)	59	;	91	[123	{
28	FS (file separator)	60	<	92	\	124	
29	GS (group separator)	61	=	93]	125	}
30	RS (record separator)	62	>	94	^	126	~
31	US (unit separator)	63	?	95	_	127	DEL

Lab 1 - ลองทำ Socket อ่านข้อมูล

```
$ nc 127.0.0.1 43239
```

```
(py311) [pwn ~ ]$ nc 127.0.0.1 43239
70
76
65
71
123
65
83
67
73
73
95
78
85
77
66
69
82
83
125
```



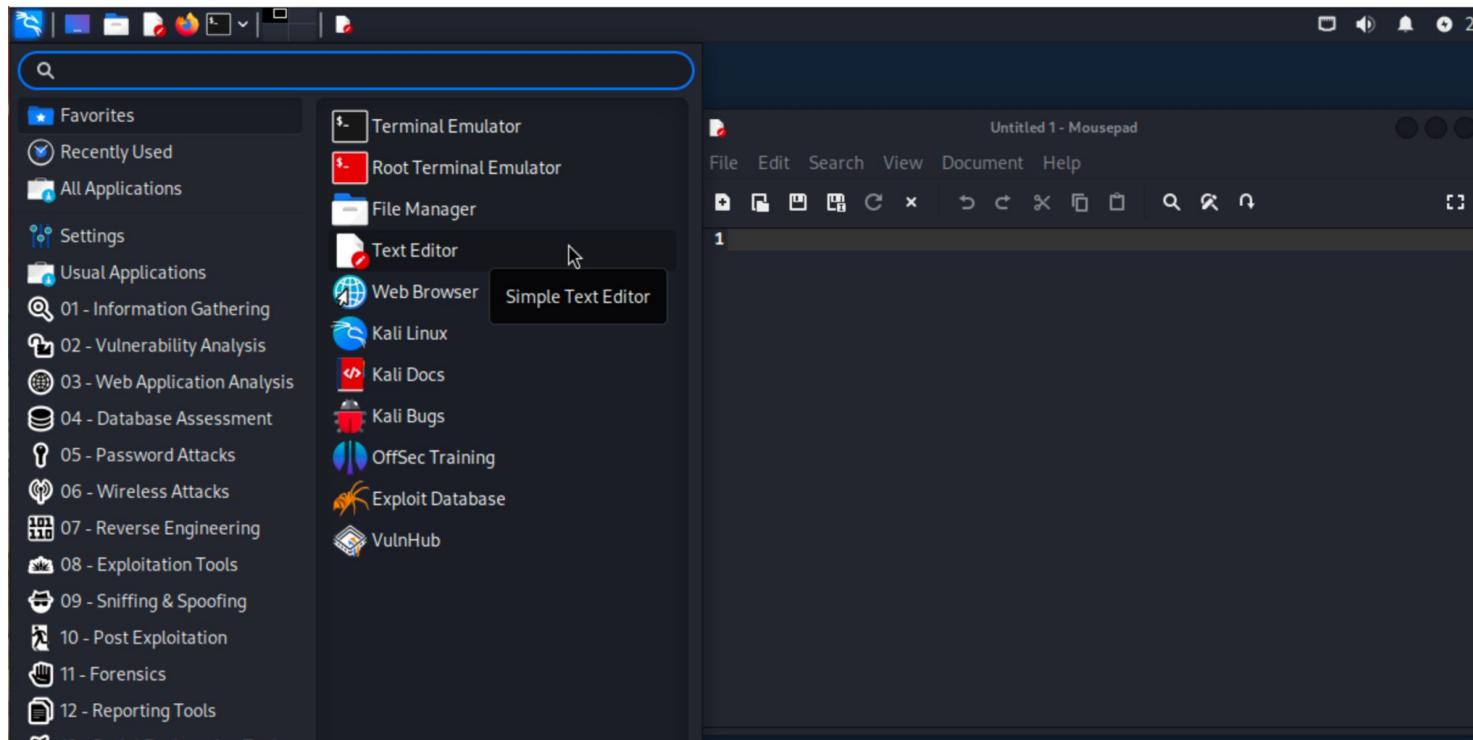
The screenshot shows the CyberChef interface with the following details:

- Operations:** A sidebar with various conversion options: Search..., Favourites (with a star icon), To Base64, From Base64, To Hex, From Hex, To Hexdump, From Hexdump, URL Decode, Regular expression, and Entropy.
- Recipe:** A central panel titled "From Decimal". It includes a "Delimiter" dropdown set to "Line feed" and a checkbox for "Support signed values".
- Input:** A list of ASCII values: 71, 123, 65, 83, 67, 73, 73, 95, 78, 85, 77, 66, 69, 82, 83, and 125.
- Output:** A red box highlights the output area which contains the string "FLAG{ASCII_NUMBERS}".

A large red arrow points from the input list towards the output box.

[https://gchq.github.io/CyberChef/#recipe=From_Decimal\('Line%20feed',false\)&input=NzAKNzYKNjUKNzEKMTIzCjY1CjgzCjY3CjczCjczCjk1Cjc4Cjg1Cjc3CjY2CjY5CjgyCjgzCjEyNQo](https://gchq.github.io/CyberChef/#recipe=From_Decimal('Line%20feed',false)&input=NzAKNzYKNjUKNzEKMTIzCjY1CjgzCjY3CjczCjczCjk1Cjc4Cjg1Cjc3CjY2CjY5CjgyCjgzCjEyNQo)

Lab 1 - ลองทำ Socket อ่านข้อมูล



Lab 1 - ลองทำ Socket อ่านข้อมูล

โปรแกรมที่เราจะลองเขียน

Part 1 - อ่านข้อมูลจาก Server

Part 2 - แปลงข้อมูลเป็นค่า Flag

Lab 1 - ลองทำ Socket อ่านข้อมูล

```
1 import socket
2
3 def main():
4     # Server details
5     host = '127.0.0.1'
6     port = 43239
7
8     # Create a socket and connect to the server
9     client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10    client.connect((host, port))
11
12    # Buffer to hold received data
13    full_data = ""
14
15    while True:
16        # Receive data from the server in chunks of 1024 bytes
17        data = client.recv(1024).decode()
18
19        if not data:
20            # No more data is being received, break the loop
21            break
22
23        # Append the received data to the buffer
24        full_data += data
25
26    # Print the full result
27    print("Data: ", full_data)
28
29    # Close the connection
30    client.close()
31
32 if __name__ == "__main__":
33     main()
```

lab1_learn.py

(py311) [pwn 6_prog]\$ python lab1_learn.py

1

65
83
67
73
73
95
78
85
77
66
69
82
83
125

การ นำเข้า
Module ชื่อ socket
เพื่อใช้ฟังก์ชันเพิ่มเติม

python lab1_learn.py

Lab 1 - ลองทำ Socket อ่านข้อมูล

```
1 import socket
2
3 def main():
4     # Server details
5     host = '127.0.0.1'
6     port = 43239
7
8     # Create a socket and connect to the server
9     client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10    client.connect((host, port))
11
12    # Buffer to hold received data
13    full_data = ""
14
15    while True:
16        # Receive data from the server in chunks of 1024 bytes
17        data = client.recv(1024).decode()
18
19        if not data:
20            # No more data is being received, break the loop
21            break
22
23        # Append the received data to the buffer
24        full_data += data
25
26    # Print the full result
27    print("Data: ", full_data)
28
29    # Close the connection
30    client.close()
31
32 if __name__ == "__main__":
33     main()
```

Part 1 - อ่าน
ข้อมูลจาก Server

lab1_learn.py

(py311) [pwn 6_prog]\$ python lab1_learn.py

2

Data: 76
65
71
123
65
83
67
7
5
78
85
77
66
69
82
83
125

เริ่มโปรแกรม Python

python lab1_learn.py

Lab 1 - ลองทำ Socket อ่านข้อมูล

```
1 import socket
2
3 def main():
4     # Server details
5     host = '127.0.0.1'
6     port = 43239
7
8     # Create a socket and connect to the server
9     client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10    client.connect((host, port))
11
12    # Buffer to hold received data
13    full_data = ""
14
15    while True:
16        # Receive data from the server in chunks of 1024 bytes
17        data = client.recv(1024).decode()
18
19        if not data:
20            # No more data is being received, break the loop
21            break
22
23        # Append the received data to the buffer
24        full_data += data
25
26    # Print the full result
27    print("Data: ", full_data)
28
29    # Close the connection
30    client.close()
31
32 if __name__ == "__main__":
33     main()
```

Part 1 - อ่าน
ข้อมูลจาก Server

lab1_learn.py

(py311) [pwn 6_prog]\$ python lab1_learn.py

3

65
83
67
73
73
95
78
85
77
66
69
82
83
125

ระบุเซิร์ฟเวอร์และ Port
ที่ต้องการเชื่อมต่อ

python lab1_learn.py

Lab 1 - ลองทำ Socket อ่านข้อมูล

```
1 import socket
2
3 def main():
4     # Server details
5     host = '127.0.0.1'
6     port = 43239
7
8     # Create a socket and connect to the server
9     client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10    client.connect((host, port))
11
12    # Buffer to hold received data
13    full_data = ""
14
15    while True:
16        # Receive data from the server in chunks of 1024 bytes
17        data = client.recv(1024).decode()
18
19        if not data:
20            # No more data is being received, break the loop
21            break
22
23        # Append the received data to the buffer
24        full_data += data
25
26    # Print the full result
27    print("Data: ", full_data)
28
29    # Close the connection
30    client.close()
31
32 if __name__ == "__main__":
33     main()
```

Part 1 - อ่าน
ข้อมูลจาก Server

lab1_learn.py

```
(py311) [pwn 6_prog ]$ python lab1_learn.py
Data: 4
76
65
71
83
67
73
73
95
78
85
77
66
69
82
83
125
```

connect()
ใช้เชื่อมต่อไปยัง
เซิร์ฟเวอร์และ Port ที่ระบุ

python lab1_learn.py

Lab 1 - ลองทำ Socket อ่านข้อมูล

```
1 import socket
2
3 def main():
4     # Server details
5     host = '127.0.0.1'
6     port = 43239
7
8     # Create a socket and connect to the server
9     client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10    client.connect((host, port))
11
12    # Buffer to hold received data
13    full_data = ""
14
15    while True:
16        # Receive data from the server in chunks
17        data = client.recv(1024).decode()
18
19        if not data:
20            # No more data is being received, break the loop
21            break
22
23        # Append the received data to the buffer
24        full_data += data
25
26    # Print the full result
27    print("Data: ", full_data)
28
29    # Close the connection
30    client.close()
31
32 if __name__ == "__main__":
33     main()
```

Part 1 - อ่าน
ข้อมูลจาก Server

lab1_learn.py

(py311) [pwn 6_prog]\$ python lab1_learn.py

5

วนลูป
(เพื่ออ่านข้อมูล
ที่เซิร์ฟเวอร์ตอบกลับมา)

95
78
85
77
66
69
82
83
125

python lab1_learn.py

Lab 1 - ลองทำ Socket อ่านข้อมูล

```
1 import socket
2
3 def main():
4     # Server details
5     host = '127.0.0.1'
6     port = 43239
7
8     # Create a socket and connect to the server
9     client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10    client.connect((host, port))
11
12    # Buffer to hold received data
13    full_data = ""
14
15    while True:
16        # Receive data from the server in chunks of 1024 bytes
17        data = client.recv(1024).decode()
18
19        if not data:
20            # No more data is being received, break the loop
21            break
22
23        # Append the received data to the buffer
24        full_data += data
25
26    # Print the full result
27    print("Data: ", full_data)
28
29    # Close the connection
30    client.close()
31
32 if __name__ == "__main__":
33     main()
```

Part 1 - อ่าน
ข้อมูลจาก Server

lab1_learn.py

(py311) [pwn 6_prog]\$ python lab1_learn.py

6

76

65

71

123

65

83

67

95

78

85

77

66

69

82

83

125

อ่านข้อมูล
ที่เซิร์ฟเวอร์ต่อไปนี้
ด้วย recv()

python lab1_learn.py

Lab 1 - ลองทำ Socket อ่านข้อมูล

```
1 import socket
2
3 def main():
4     # Server details
5     host = '127.0.0.1'
6     port = 43239
7
8     # Create a socket and connect to the server
9     client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10    client.connect((host, port))
11
12    # Buffer to hold received data
13    full_data = ""
14
15    while True:
16        # Receive data from the server in chunks of 1024 bytes
17        data = client.recv(1024).decode()
18
19        if not data:
20            # No more data is being received, break the loop
21            break
22
23        # Append the received data to the buffer
24        full_data += data
25
26    # Print the full result
27    print("Data: ", full_data)
28
29    # Close the connection
30    client.close()
31
32 if __name__ == "__main__":
33     main()
```

Part 1 - อ่าน
ข้อมูลจาก Server

lab1_learn.py

(py311) [pwn 6_prog]\$ python lab1_learn.py
Data: 70
76
65
71
123
65
83
67
73
73
95

7

แสดงข้อมูลที่เซิร์ฟเวอร์ตอบ
กลับมา ออกมายัง Console
(Standard Input)

//
66
69
82
83
125

python lab1_learn.py

Lab 1 - ลองทำ Socket อ่านข้อมูล

```
1 import socket ←  
2  
3 def main():  
4     # Server details  
5     host = '127.0.0.1'  
6     port = 43239 ←  
7  
8     # Create a socket and connect to the server  
9     client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
10    client.connect((host, port)) ←  
11  
12    # Buffer to hold received data  
13    full_data = ""  
14  
15    while True: ←  
16        # Receive data from the server in chunks of 1024 bytes  
17        data = client.recv(1024).decode() ←  
18  
19        if not data:  
20            # No more data is being received, break the loop  
21            break  
22  
23        # Append the received data to the buffer  
24        full_data += data  
25  
26    # Print the full result  
27    print("Data: ", full_data) ←  
28  
29    # Close the connection  
30    client.close()  
31  
32    if __name__ == "__main__":  
33        main() ←
```

Part 1 - อ่านข้อมูลจาก Server

1. การนำเข้า Module ชื่อ socket เพื่อใช้งานฟังก์ชันเพิ่มเติม
2. main() เริ่มโปรแกรม Python
3. ระบุเซิร์ฟเวอร์และ Port ที่ต้องการเชื่อมต่อ
4. **connect()**
ใช้เชื่อมต่อไปยังเซิร์ฟเวอร์และ Port ที่ระบุ
5. วนลูป
(เพื่ออ่านข้อมูล ที่เซิร์ฟเวอร์ตอบกลับมา)
6. อ่านข้อมูล
ที่เซิร์ฟเวอร์ตอบกลับมา ด้วย **recv()**
7. แสดงข้อมูลที่เซิร์ฟเวอร์ตอบกลับมา ออกมายัง Console (Standard Input)

Lab 1 - ลองทำ Socket อ่านข้อมูล

โปรแกรมที่เราจะลองเขียน

Part 1 - อ่านข้อมูลจาก Server

Part 2 - แปลงข้อมูลเป็นค่า Flag



```
(py311) [pwn 6_prog ]$ python lab1_learn.py
Data: 70
76
65
71
123
65
83
67
73
73
73
95
78
85
77
66
69
82
83
125
```

Lab 1 - ลองทำ Socket อ่านข้อมูล

```
3 ▼ def ascii_to_string(data): ←  
4     # Split the data into individual ASCII numbers  
5     ascii_numbers = data.split()  
6  
7     # Convert each ASCII number to its corresponding character  
8     # and join them into a string  
9     flag = ''.join(chr(int(num)) for num in ascii_numbers)  
10  
11    return flag  
  
25 ▼     while True:  
26         # Receive data from the server in chunks of 1024 bytes  
27         data = client.recv(1024).decode()  
28  
29         if not data:  
30             # No more data is being received, break the loop  
31             break  
32  
33         # Append the received data to the buffer  
34         full_data += data  
35  
36         # Convert the received ASCII numbers to a flag string  
37         flag = ascii_to_string(full_data) ←  
38  
39         # Print the flag  
40         print("Flag: ", flag)
```

lab1_solve.py

Part 2 - แปลง
ข้อมูลเป็นค่า Flag

- สร้างฟังก์ชันเพื่อ Decode
ค่า ตัวเลข ASCII เป็น ตัวอักษร
- เรียกฟังก์ชัน ascii_to_string()
เพื่อแสดงค่า Flag

```
(py311) [pwn 6_prog ]$ python lab1_solve.py  
Flag: FLAG{ASCII_NUMBERS}  
(py311) [pwn 6_prog ]$ █
```

\$ python lab1_solve.py

Lab 1 - ลองทำ Socket อ่านข้อมูล

สรุปการเรียนรู้ใน Lab 1

Part 1 - อ่านข้อมูลจาก Server

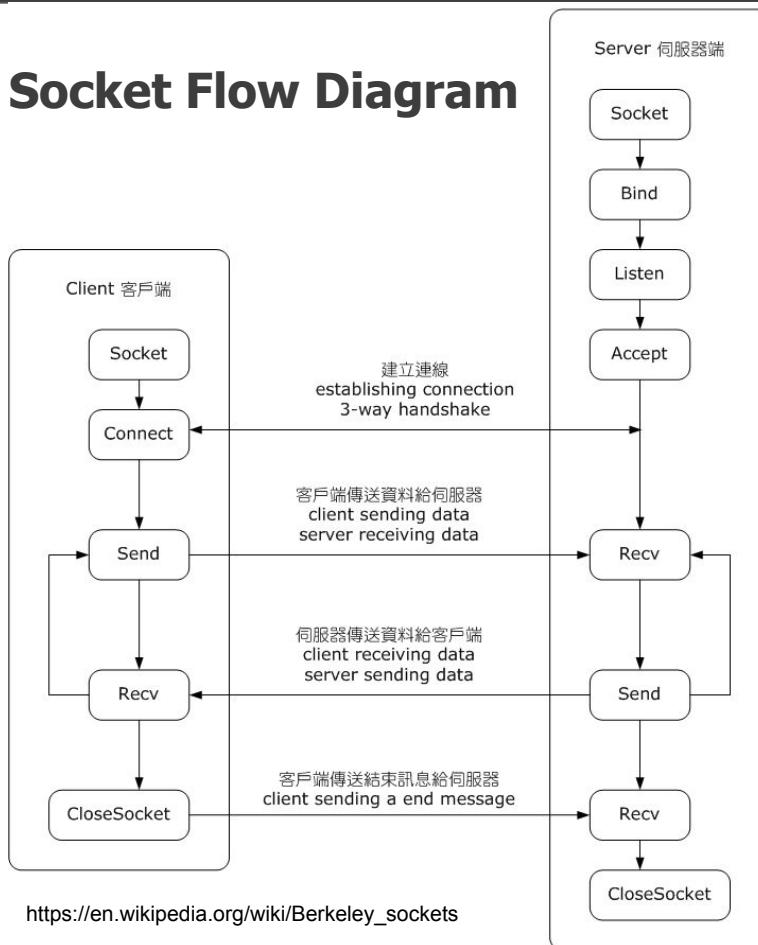
- connect()
- recv()

Part 2 - แปลงข้อมูลเป็นค่า Flag

- สร้างฟังก์ชัน ascii_to_string()
เพื่อแปลงค่าตัวเลข ASCII เป็นตัวอักษร

```
1 import socket
2
3 def ascii_to_string(data):
4     # Split the data into individual ASCII numbers
5     ascii_numbers = data.split()
6
7     # Convert each ASCII number to its corresponding character
8     # and join them into a string
9     flag = ''.join(chr(int(num)) for num in ascii_numbers)
10
11     return flag
12
13 def main():
14     # Server details
15     host = '127.0.0.1'
16     port = 43239
17
18     # Create a socket and connect to the server
19     client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
20     client.connect((host, port))
21
22     # Buffer to hold received data
23     full_data = ""
24
25     while True:
26         # Receive data from the server in chunks of 1024 bytes
27         data = client.recv(1024).decode()
28
29         if not data:
30             # No more data is being received, break the loop
31             break
32
33         # Append the received data to the buffer
34         full_data += data
35
36     # Convert the received ASCII numbers to a flag string
37     flag = ascii_to_string(full_data)
38
39     # Print the flag
40     print("Flag: ", flag)
41
42     # Close the connection
43     client.close()
44
45 if __name__ == "__main__":
46     main()
```

TCP Socket Flow Diagram



lab1_server.go

```

13 ▼ func main() {
14     // Listen on port 43239
15     listener, err := net.Listen("tcp", ":43239")
16     ▼ if err != nil {
17         fmt.Println("Error starting server:", err)
18         return
19     }
20     defer listener.Close()
21     fmt.Println("Server listening on port 43239...")
22
23     for {
24         // Accept connection
25         conn, err := listener.Accept()
26         ▼ if err != nil {
27             fmt.Println("Error accepting connection:", err)
28             continue
29         }
30
31         // Handle client connection in a goroutine
32         go handleClient(conn)
33     }
34 }
35
36 ▼ func handleClient(conn net.Conn) {
37     defer conn.Close()
38
39     // Convert the flag string to ASCII values
40     asciiValues := convertToASCII(flag)
41
42     // Send the ASCII values to the client, one per line
43     ▼ for _, val := range asciiValues {
44         conn.Write([]byte(strconv.Itoa(val) + "\n"))
45     }
46 }
47
  
```

Lab 2 - ลองทำ Socket ตอบกลับข้อมูล

1 ดาวน์โหลดไฟล์ lab2_server.py

2 \$ python3 server.py

```
(py3120) → ykh (p02) $ python3 server.py
Server listening on port 55816...
Accepted connection from ('127.0.0.1', 60624)
Accepted connection from ('127.0.0.1', 60625)
```

3

\$ nc 127.0.0.1 55816

→ ykh (p02) \$ nc 127.0.0.1 55816

Welcome to the Binary Challenge!

Your task is to perform the unique operations in the given order and find the final result in hexadecimal that yields the flag.

Binary Number 1: 10110111

Binary Number 2: 10001111

1

Question 1/5:

Operation 1: '<<'

Perform a left shift of Binary Number 1 by 1 bits.

Enter the binary result: 101101110

Correct!

Question 2/5:

Operation 2: '+'

Perform the operation on Binary Number 1 & 2.

Enter the binary result: 101000110

Correct!

2

Lab 2 - ลองทำ Socket ตอบกลับข้อมูล

โปรแกรมที่เราจะลองเขียน

Part 1 - จำแนกข้อมูลจาก Server

Part 2 - คำนวนและส่งคำตอบ

Lab 2 - ลองทำ Socket ตอบกลับข้อมูล

Basic binary operations

```
→ ykh (p02) $ nc 127.0.0.1 55816
Welcome to the Binary Challenge!
Your task is to perform the unique operations in the given order and find the final result
in hexadecimal that yields the flag.
```

Binary Number 1:

Binary Number 2:

Question 1/5:

Operation 1:

Perform a left shift of Binary Number 1 by 1 bits.

Enter the binary result:

ตัวอย่างข้อมูลที่เราต้องการ

1. 10110111
2. 10001111
3. <<

แล้วจะเขียนโปรแกรมยังไง ?

Lab 2 - ลองทำ Socket ตอบกลับข้อมูล

Part 1 : จำแนกข้อมูลจาก Server

```
39 def main():
40     client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
41     client_socket.connect(('127.0.0.1', 55816))
42
43     print(client_socket.recv(1024).decode())
44
45     binary1 = binary2 = None
46
47     while True:
48         # Receive and display server message, now using the delimiter function
49         response = recv_until_delimiter(client_socket)
50         print(f"Server response:\n{response}")
51
52         operation = None # Ensure operation is initialized for each loop
53
54         # Parse binary numbers and operation from server's message
55         for line in response.split('\n'):
56             if 'Binary Number 1' in line and ':' in line:
57                 binary1 = line.split(':')[1].strip()
58             if 'Binary Number 2' in line and ':' in line:
59                 binary2 = line.split(':')[1].strip()
60             if 'Operation' in line and ':' in line:
61                 operation = line.split(':')[1].replace(" ", "").strip()
62
```

1

2

```
→ ykh (prog_lab2) $ python3 prog_lab2_solve.py
Server response: Welcome to the Binary Challenge!
```

```
Server response: Your task is to perform the unique
order and find the final result in binary.
```

```
Binary Number 1: 11110101
Binary Number 2: 01111110
```

```
Server response: Question 1/6:
Operation 1: '|'
```

Lab 2 - ลองทำ Socket ตอบกลับข้อมูล

โปรแกรมที่เราจะลองเขียน

Part 1 - จำแนกข้อมูลจาก Server

Part 2 - คำนวนและส่งคำตอบ

Lab 2 - ลองทำ Socket ตอบกลับข้อมูล

Basic binary operations

1. + (Addition)
2. - (Subtraction)
3. * (Multiplication)
4. & (AND)
5. | (OR)
6. ^ (XOR (exclusive OR))
7. << (Left shift)
8. >> (Right shift)

Lab 2 - ลองทำ Socket ตอบกลับข้อมูล

Part 2 : คำนวณและส่งคำตอบ

```

47     while True:
48         # Receive and display server message, now using the delimiter function
49         response = recv_until_delimiter(client_socket)
50         print(f"Server response:\n{response}")
51
52         operation = None # Ensure operation is initialized for each loop
53
54         # Parse binary numbers and operation from server's message
55         for line in response.split('\n'):
56             if 'Binary Number 1' in line and ':' in line:
57                 binary1 = line.split(':')[1].strip()
58             if 'Binary Number 2' in line and ':' in line:
59                 binary2 = line.split(':')[1].strip()
60             if 'Operation' in line and ':' in line:
61                 operation = line.split(':')[1].replace("''", "").strip()
62
63         # Perform the operation if valid binaries and operation are received
64         if binary1 and operation:
65             if binary2: # Handle binary operations
66                 result = perform_operation(operation, binary1, binary2)
67             else:
68                 continue
69
70             print(f"Result: {result}")
71             client_socket.sendall(f"{result}\n".encode())
72
73         # Check if challenge is completed
74         if 'The flag is' in response:
75             print("Challenge completed.")
76             break
77

```

3

1

```

3     # Function to handle binary operations directly
4     def perform_operation(op_symbol, binary1, binary2=None):
5         # Convert binary strings to integers
6         a = int(binary1, 2)
7         b = int(binary2, 2) if binary2 is not None else None
8
9         # Define the possible operations using direct arithmetic/bitwise
10        operations = {
11             '<<': lambda x, y: bin(x << y)[2:].zfill(8), # Left shift
12             '>>': lambda x, y: bin(x >> y)[2:].zfill(8), # Right shift
13             '+': lambda x, y: bin(x + y)[2:].zfill(8), # Addition
14             '*': lambda x, y: bin(x * y)[2:].zfill(8), # Multiplication
15             '-': lambda x, y: bin(x - y)[2:].zfill(8), # Subtraction
16             '&': lambda x, y: bin(x & y)[2:].zfill(8), # Bitwise AND
17             '|': lambda x, y: bin(x | y)[2:].zfill(8), # Bitwise OR
18             '^': lambda x, y: bin(x ^ y)[2:].zfill(8) # Bitwise XOR
19         }
20
21
22         # Perform the operation with two operands
23         if op_symbol in operations:
24             result = operations[op_symbol](a, b)
25         else:
26             raise ValueError(f"Unknown operation: {op_symbol}")
27
28         return result

```

2

Lab 2 - ลองทำ Socket ตอบกลับข้อมูล

```

1 import socket
2
3 # Function to handle binary operations directly
4 def perform_operation(op_symbol, binary1, binary2=None):
5     # Convert binary strings to integers
6     a = int(binary1, 2)
7     b = int(binary2, 2) if binary2 is not None else None
8
9     # Define the possible operations using direct arithmetic/bitwise operators
10    operations = {
11        '<<': lambda x, y: bin(x << y)[2:].zfill(8), # Left shift
12        '>>': lambda x, y: bin(x >> y)[2:].zfill(8), # Right shift
13        '+': lambda x, y: bin(x + y)[2:].zfill(8), # Addition
14        '*': lambda x, y: bin(x * y)[2:].zfill(8), # Multiplication
15        '-': lambda x, y: bin(x - y)[2:].zfill(8), # Subtraction
16        '&': lambda x, y: bin(x & y)[2:].zfill(8), # Bitwise AND
17        '|': lambda x, y: bin(x | y)[2:].zfill(8), # Bitwise OR
18        '^': lambda x, y: bin(x ^ y)[2:].zfill(8) # Bitwise XOR
19    }
20
21    # Perform the operation with two operands
22    if op_symbol in operations:
23        result = operations[op_symbol](a, b)
24    else:
25        raise ValueError(f"Unknown operation: {op_symbol}")
26
27    return result
28
29 # Function to receive data until a delimiter is reached
30 def recv_until_delimiter(sock, delimiter='\n\n'):
31     data = ""
32     while delimiter not in data:
33         packet = sock.recv(1024).decode()
34         if not packet:
35             break
36         data += packet
37     return data
38
39 def main():
40     client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
41     client_socket.connect(('127.0.0.1', 55816))
42
43     print(client_socket.recv(1024).decode())
44
45     binary1 = binary2 = None
46
47     while True:
48         # Receive and display server message, now using the delimiter function
49         response = recv_until_delimiter(client_socket)
50         print(f"Server response:\n{response}")
51
52         operation = None # Ensure operation is initialized for each loop
53
54         # Parse binary numbers and operation from server's message
55         for line in response.split('\n'):
56             if 'Binary Number 1' in line and ':' in line:
57                 binary1 = line.split(':')[1].strip()
58             if 'Binary Number 2' in line and ':' in line:
59                 binary2 = line.split(':')[1].strip()
60             if 'Operation' in line and ':' in line:
61                 operation = line.split(':')[1].replace(" ", "").strip()
62
63         # Perform the operation if valid binaries and operation are received
64         if binary1 and operation:
65             if binary2: # Handle binary operations
66                 result = perform_operation(operation, binary1, binary2)
67             else:
68                 continue
69
70             print(f"Result: {result}")
71             client_socket.sendall(f"{result}\n".encode())
72
73         # Check if challenge is completed
74         if 'The flag is' in response:
75             print("Challenge completed.")
76             break
77
78     if __name__ == "__main__":
79         main()
80

```

Lab 2 - ลองทำ Socket ตอบกลับข้อมูล

Server response:
Question 4/6:
Operation 4: '*'
Perform the operation on Binary Number 1 and Binary Number 2.

Enter the binary result:
Result: 1110000110110
Server response:
Correct!

Server response:
Question 5/6:
Operation 5: '^'
Perform the operation on Binary Number 1 and Binary Number 2.

Enter the binary result:
Result: 10110011
Server response:
Correct!

Server response:
Question 6/6:
Operation 6: '>>'
Perform the operation on Binary Number 1 and Binary Number 2.

Enter the binary result:
Result: 00000000
Server response:
Correct!

Server response:
Congratulations! You have completed all the questions.
The flag is: FLAG{B1N4RY_0P3R4TI0NS}

Lab 2 - ลองทำ Socket ตอบกลับข้อมูล

สรุปการเรียนรู้ใน Lab 2

Part 1 - จำแนกข้อมูลจาก Server

- in
- split

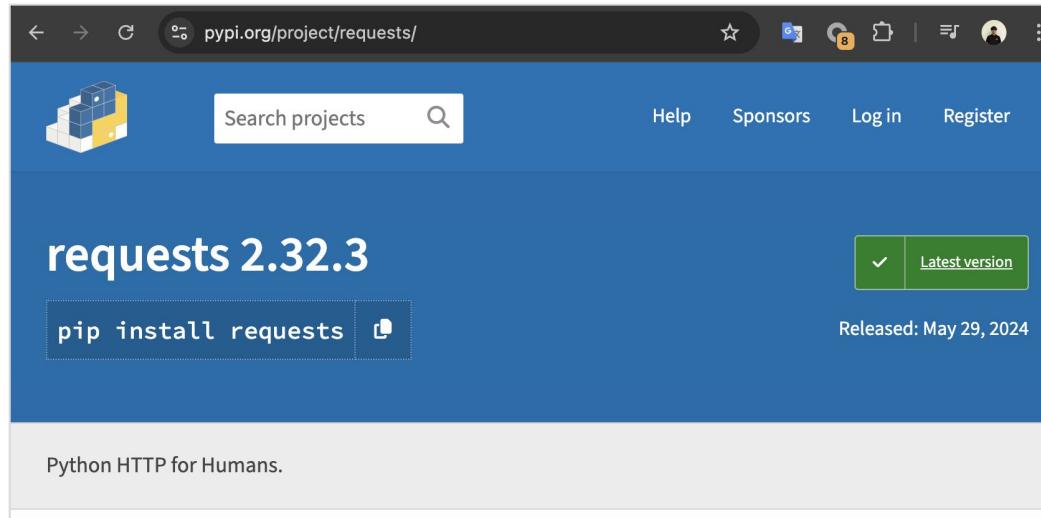
Part 2 - คำนวนและส่งคำตอบ

- สร้างฟังก์ชัน perform_operation()
เพื่อคำนวนค่าตัวเลข Binary
- sendall()

การทำ HTTP Client

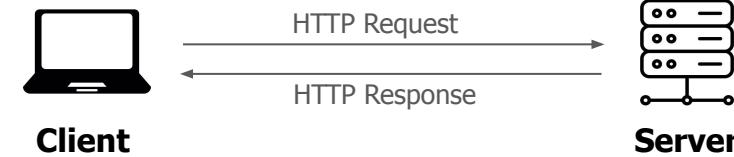
requests module

```
$ pip install requests
```



The screenshot shows the PyPI project page for the 'requests' module. The URL in the address bar is 'pypi.org/project/requests/'. The page title is 'requests 2.32.3'. Below the title, there is a green button with a checkmark icon and the text 'Latest version'. To the right of the button, it says 'Released: May 29, 2024'. At the bottom of the page, there is a footer with the text 'Python HTTP for Humans.'

<https://pypi.org/project/requests/>



Basic GET Request

```
1 import requests
2
3 url = "http://sdh.local:8000/index.php"
4
5 response = requests.get(url)
6
7 print("URL:", response.url)
8 print(f"{response.status_code}: {response.reason}")
9 print(response.text)
10
```

get()

HTTP Request:

```
GET /index.php HTTP/1.1
Host: sdh.local:8000
[...]
```

```
→ ykh (request) $ python3 request_get.py
URL: http://sdh.local:8000/index.php
200: OK
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome to SDH</title>
```

Basic POST Request

```
1 import requests
2
3 url = "http://sdh.local:8000/post.php"
4
5 data = {
6     "id": 1
7 }
8
9 response = requests.post(url, json=data)
10
11 print("URL:", response.url)
12 print(f"Status Code: {response.status_code} - {response.reason}")
13 print(response.json())
14
```

post()

```
→ ykh (request) $ python3 request_post.py
URL: http://sdh.local:8000/post.php
200 OK
{'status': 'success', 'message': 'Data received successfully', 'postId': 1}
```

HTTP Request:

```
GET /post.php HTTP/1.1
```

```
Host: sdh.local:8000
```

```
[...]
```

```
{
    "id": 1
}
```

Basic Header

```
1 import requests
2
3 url = "http://sdh.local:8000/admin.php"
4
5 token = "e62a54f019f8b70fc0b77a50e827f86d"
6
7 headers = {
8     "Authorization": f"Bearer {token}"
9 }
10
11 response = requests.get(url, headers=headers)
12
13 print("URL:", response.url)
14 print(f"{response.status_code}: {response.reason}")
15 print(response.text)
16
```

```
→ ykh (request) $ python3 request_header.py
URL: http://sdh.local:8000/admin.php
200: OK
{"message": "Welcome to the admin dashboard, admin_user"}
```

HTTP Request:

```
GET /admin.php HTTP/1.1
Host: sdh.local:8000
Authorization: Bearer e62a54f0[...]27f86d
[...]
```

Authentication POST Request

```
request_post_login.py x
1 import requests
2
3 data = {'username': 'admin_user', 'password': 'admin_pass'}
4 response = requests.post('http://sdh.local:8000/login.php', json=data)
5 print(f'{response.status_code}: {response.reason}')
6 print(response.headers)
7 print(response.json())
8
```

```
→ ykh (request) $ python3 request_post_login.py
200: OK
{'Host': 'sdh.local:8000', 'Date': 'Sun, 08 Sep 2024 15:14:23 GMT', 'Connection': 'close', 'X-Powered-By': 'PHP/8.3.9', 'Content-Type': 'application/json', 'Authorization': 'e62a54f019f8b70fc0b77a50e827f86d'}
{'message': 'Login successful', 'role': 'admin'}
```

HTTP Request:

POST /login.php HTTP/1.1

Host: sdh.local:8000

[...]

{

 "username": "admin_user",
 "password": "admin_pass"

}

Broken Access Control

```
1 import requests
2 from colorama import init, Fore, Style
3
4 init(autoreset=True)
5
6 urls = [
7     'http://sdh.local:8000/index.php',
8     'http://sdh.local:8000/admin_dashboard.php'
9 ]
10
11 tokens = {
12     'admin': 'admin_token_a035ada8cc57de7fad3e93fb2d1ccce2',
13     'employee': 'employee_token_a0f70ab7b6ad7fbfed3465c4f59c0fd1',
14     'invalid': 'invalid_token_99999'
15 }
16
17 def test_access(url, token, role):
18     headers = {
19         "Authorization": f"Bearer {token}"
20     }
21
22     response = requests.get(url, headers=headers)
23
24     print(f"{Fore.CYAN}Testing {Fore.YELLOW}{role.capitalize()} Token: {Fore.WHITE}{token}")
25     print(f"{Fore.CYAN}Status Code: {Fore.GREEN}{response.status_code} {Fore.WHITE}{response.reason}\n")
26
27 for url in urls:
28     print(f"{Fore.MAGENTA}" + "=" * 70)
29     print(f"{Fore.CYAN}Target URL: {Fore.GREEN}{url}")
30     print(f"{Fore.MAGENTA}" + "=" * 70)
31     for role, token in tokens.items():
32         test_access(url, token, role)
33
```

```
→ ykh (request) $ python3 request_BAC.py
=====
Target URL: http://sdh.local:8000/index.php
=====
Testing Admin Token: admin_token_a035ada8cc57de7fad3e93fb2d1ccce2
Status Code: 200 (OK)

Testing Employee Token: employee_token_a0f70ab7b6ad7fbfed3465c4f59c0fd1
Status Code: 200 (OK)

Testing Invalid Token: invalid_token_99999
Status Code: 200 (OK)

=====
Target URL: http://sdh.local:8000/admin_dashboard.php
=====
Testing Admin Token: admin_token_a035ada8cc57de7fad3e93fb2d1ccce2
Status Code: 200 (OK)

Testing Employee Token: employee_token_a0f70ab7b6ad7fbfed3465c4f59c0fd1
Status Code: 403 (Forbidden)

Testing Invalid Token: invalid_token_99999
Status Code: 401 (Unauthorized)
```

HTTP Request:

GET /**[path]** HTTP/1.1
Host: sdh.local:8000
Authorization: Bearer **[Token]**
[...]

IDOR

```
1 import requests
2 from colorama import Fore, Style, init
3
4 init(autoreset=True)
5
6 for i in range(1, 4):
7     response = requests.get(f'http://sdh.local:8000/profile_user.php?id={i}',
8                             headers={"Authorization": "Bearer employee_token_67890"})
9
10    print(f'{Fore.YELLOW}URL: {response.url}')
11
12    if response.status_code == 200:
13        print(f'{Fore.GREEN}ID: {i}, Status Code: {response.status_code} (OK)')
14    else:
15        print(f'{Fore.RED}ID: {i}, Status Code: {response.status_code}')
16
17    print(f'{Fore.CYAN}{response.json()}')
18    print(Style.RESET_ALL)
19
```

```
→ ykh (request) $ python3 request_IDOR.py
URL: http://sdh.local:8000/profile_user.php?id=1
ID: 1, Status Code: 200 (OK)
{'username': 'admin_user', 'email': 'admin@sdh.local', 'role': 'admin'}
```



```
URL: http://sdh.local:8000/profile_user.php?id=2
ID: 2, Status Code: 200 (OK)
{'username': 'employee1', 'email': 'employee1@sdh.local', 'role': 'employee'}
```



```
URL: http://sdh.local:8000/profile_user.php?id=3
ID: 3, Status Code: 200 (OK)
{'username': 'employee2', 'email': 'employee2@sdh.local', 'role': 'employee'}
```

```
GET /profile_user.php?id=1 HTTP/1.1
Host: sdh.local:8000
Authorization: Bearer [Token]
[...]
```

Agenda (Day 2)

เวลา	รายละเอียด
09.00 - 10.30	Mobile Application Security
10.30 - 10.45	พักเบรก
10.45 - 12.00	Programming for CTF
12.00 - 13.00	พักรับประทาน อาหารกลางวัน
13.00 - 16.00	การแข่งขัน CTF (3 ชม.)
16.00 - 17.00	มอบรางวัลและพิธีปิด



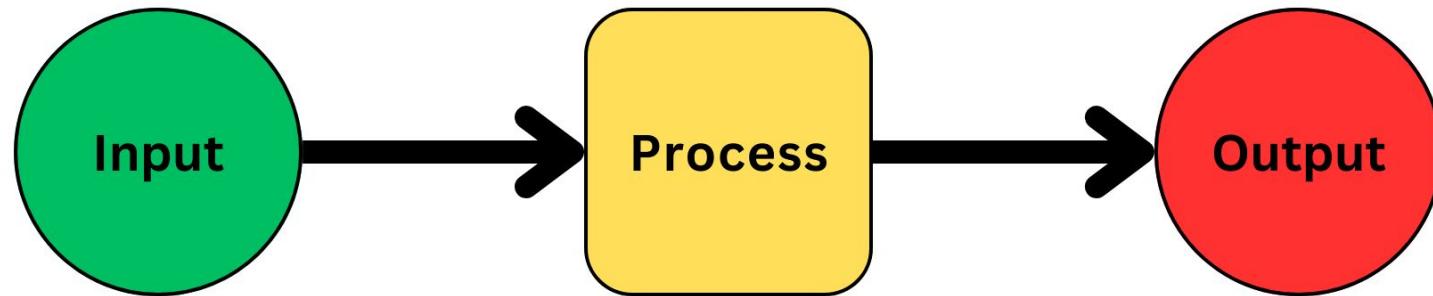
Thank you !!

Appendix - Programming Basic

Programming คืออะไร

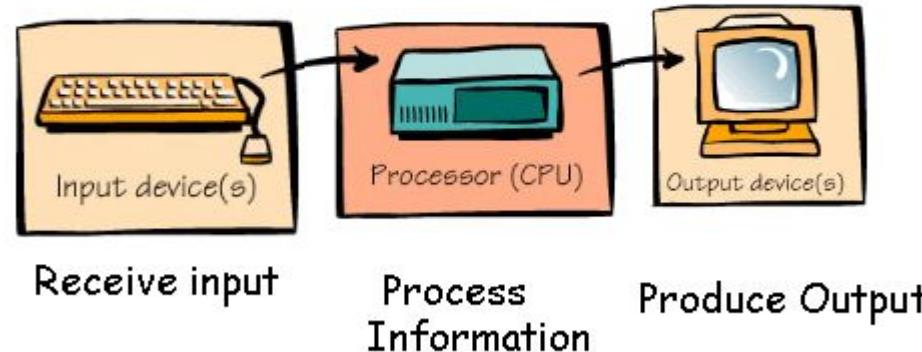


IPO model

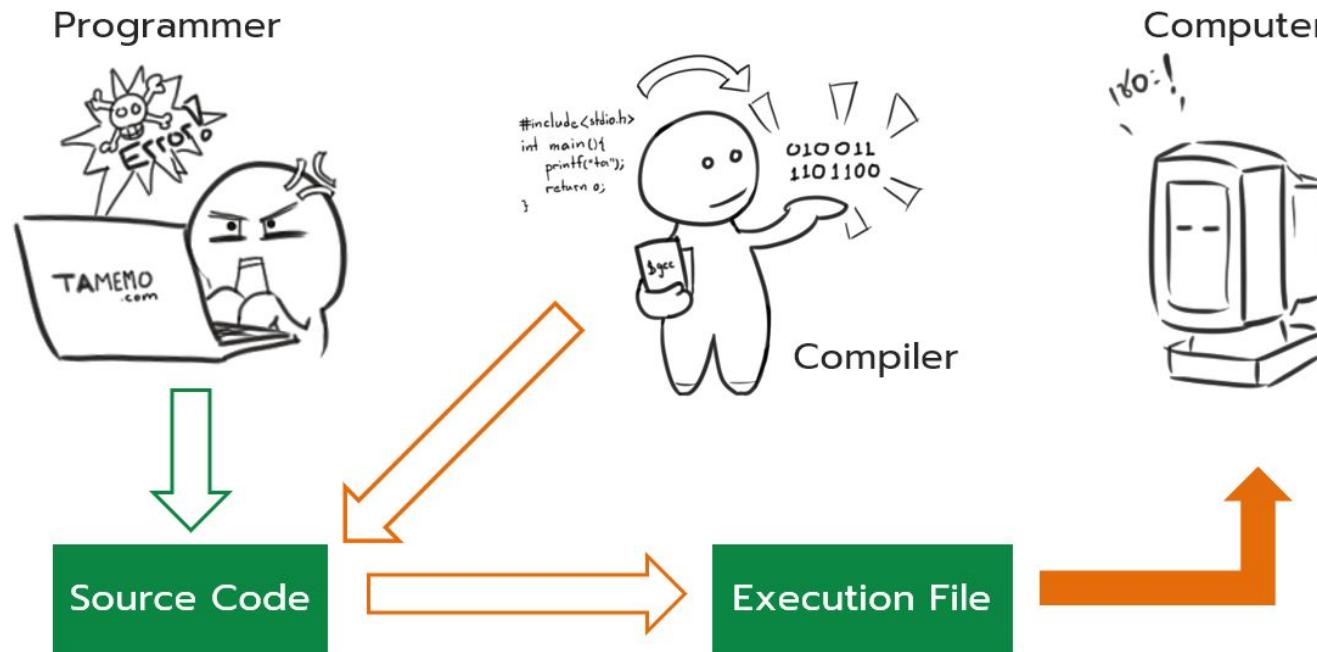


IPO model (ຕົວ)

What Computers Do



Computer เข้าใจได้อย่างไร

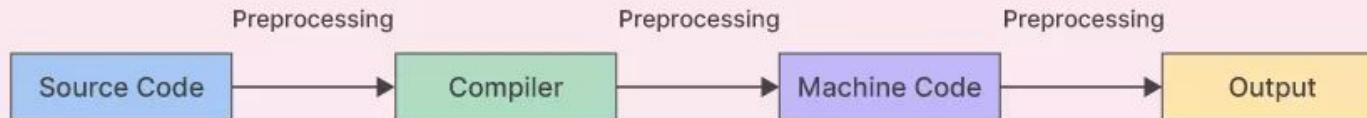


ใกล้เคียงภาษาคน รู้เรื่อง อ่านง่าย

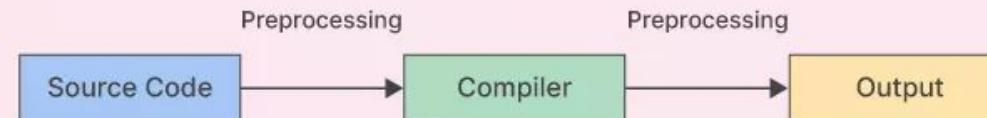
110001011010101111001010011010

Translators Programming

How Compiler works

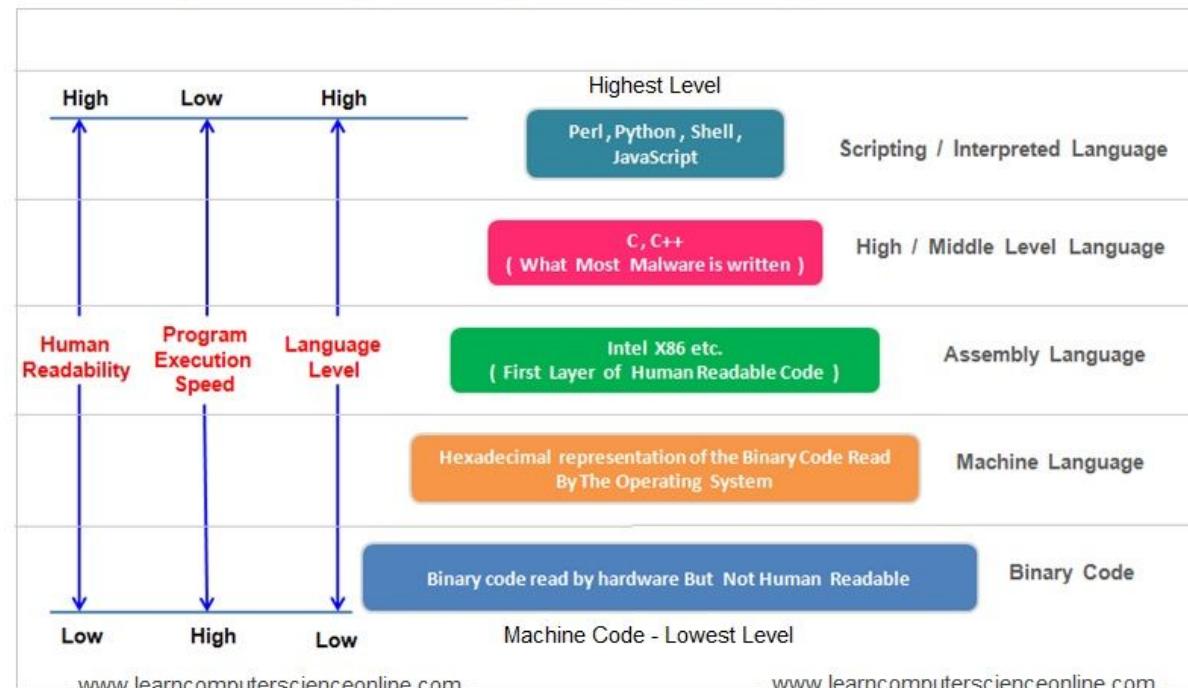
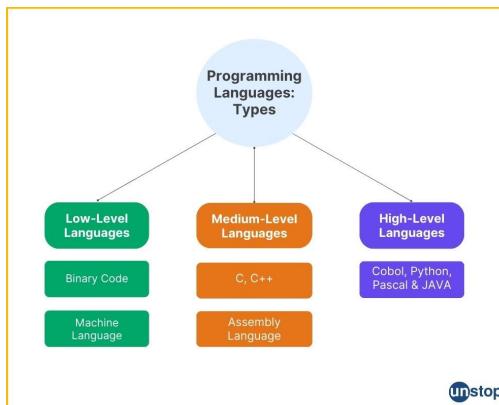


How Interpreter works



Levels of Programming Language

Computer Programming Language - Types And Levels

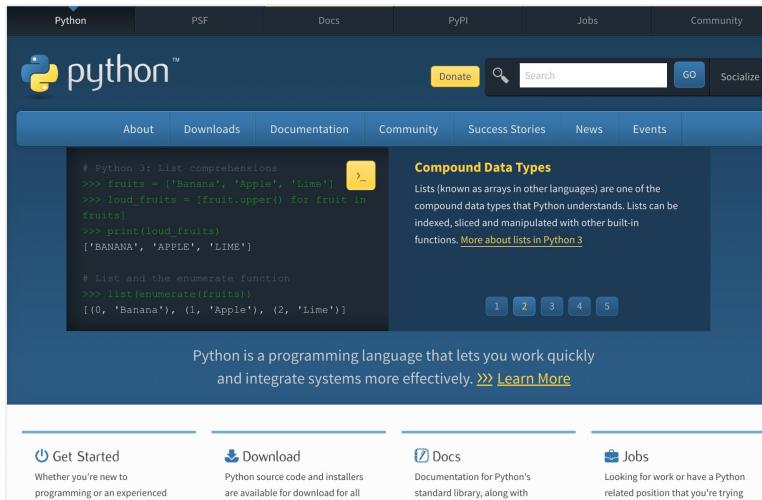


Programming Languages

Aug 2024		Aug 2023		Change	Programming Language	Ratings	Change	Aug 2024		Aug 2023		Change	Programming Language	Ratings	Change
1	1	2	3	▲	Python	18.04%	+4.71%	11	13	12	23	▲	MATLAB	1.72%	+0.67%
2	3	3	2	▼	C++	10.04%	-0.59%	13	10	14	19	▲	Delphi/Object Pascal	1.63%	+0.83%
3	2	4	4	▼	C	9.17%	-2.24%	15	17	15	18	▲	PHP	1.46%	+0.19%
4	4	5	5	▲	Java	9.16%	-1.16%	16	18	16	27	▲	Rust	1.28%	+0.39%
5	5	6	6	▲	C#	6.39%	-0.65%	17	9	17	11	▼	Ruby	1.28%	+0.37%
6	6	7	7	▲	JavaScript	3.91%	+0.62%	18	27	18	16	▲	Swift	1.28%	+0.37%
7	8	8	8	▲	SQL	2.21%	+0.68%	19	16	19	11	▼	Assembly language	1.21%	-0.13%
8	7	9	7	▼	Visual Basic	2.18%	-0.45%	20	11	20	11	▼	Kotlin	1.13%	+0.44%
9	12	12	14	▲	Go	2.03%	+0.87%	21	16	21	11	▼	R	1.11%	+0.19%
10	14	14	14	▲	Fortran	1.79%	+0.75%	22	11	22	11	▼	Scratch	1.09%	-0.13%

ສໍາດັບໂດຍ www.tiobe.com

Programming Languages



The screenshot shows the Python.org homepage. At the top, there are navigation links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the header, there's a search bar and a "Socialize" button. The main content area features a "python™" logo and two code snippets. The first snippet demonstrates list comprehensions:# Python 3: List comprehensions

```
>>> fruits = ['Banana', 'Apple', 'Lime']
>>> loud_fruits = [fruit.upper() for fruit in
fruits]
>>> print(loud_fruits)
['BANANA', 'APPLE', 'LIME']
```

The second snippet shows the use of the enumerate function:# List and the enumerate function

```
>>> list(enumerate(fruits))
[(0, 'Banana'), (1, 'Apple'), (2, 'Lime')]
```

Below the code, a section titled "Compound Data Types" explains lists. A "Learn More" link is present. At the bottom, a quote states: "Python is a programming language that lets you work quickly and integrate systems more effectively." followed by a "Learn More" link.

Get Started
Whether you're new to programming or an experienced

Download
Python source code and installers are available for download for all

Docs
Documentation for Python's standard library, along with

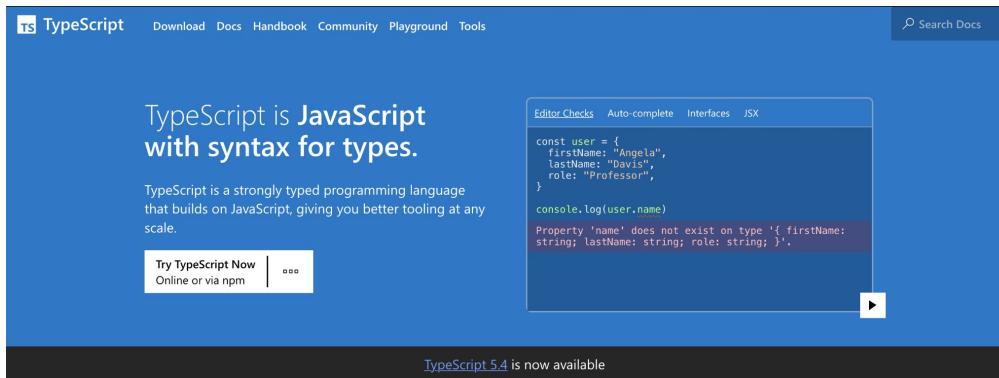
Jobs
Looking for work or have a Python related position that you're trying

นิยมใช้กับงาน

- Data Science
- Web Development
- Network System
- Etc
- AI & Machine Learning
- Desktop Application
- Games



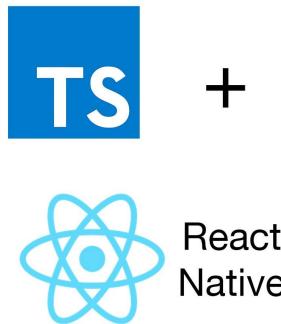
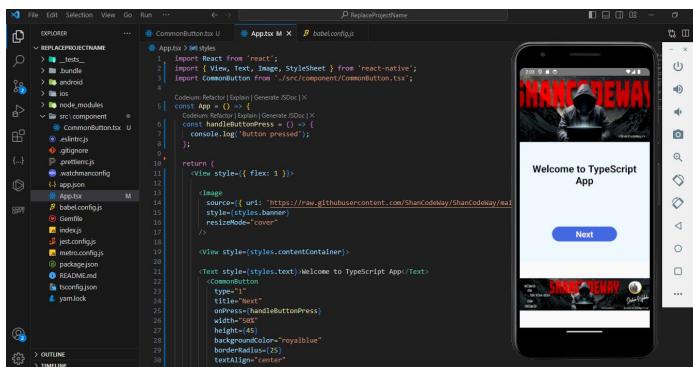
Programming Languages



TypeScript is **JavaScript with syntax for types**.
TypeScript is a strongly typed programming language that builds on JavaScript, giving you better tooling at any scale.

Try TypeScript Now | Online or via npm

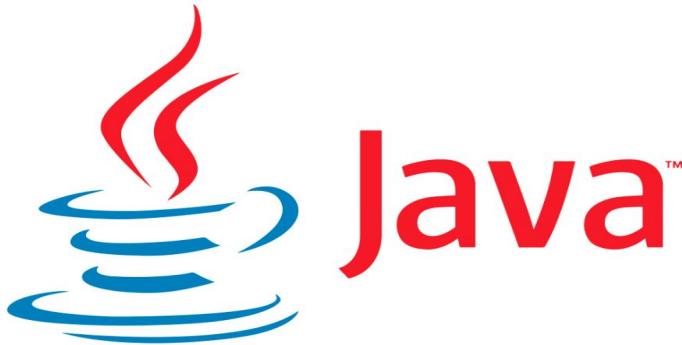
TypeScript 5.4 is now available



นิยมใช้กับงาน

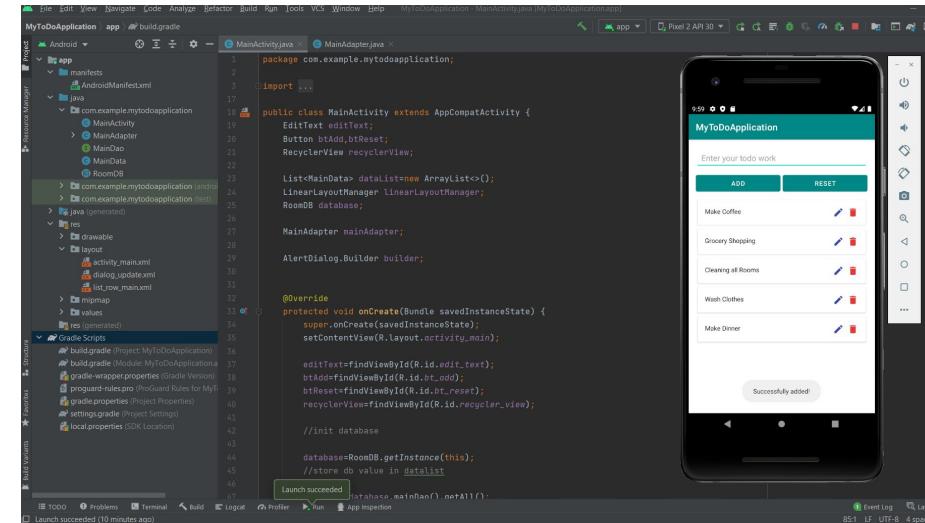
- Web Application Development (ทั้ง Front-end และ Back-end)
- Desktop Application Development (ด้วยเทคโนโลยี Electron)
- Mobile Application Development (ด้วยเทคโนโลยี React Native)
- Game Development
- Machine Learning และ Data Science
- การเขียน libraries หรือ utilities ต่าง ๆ

Programming Languages

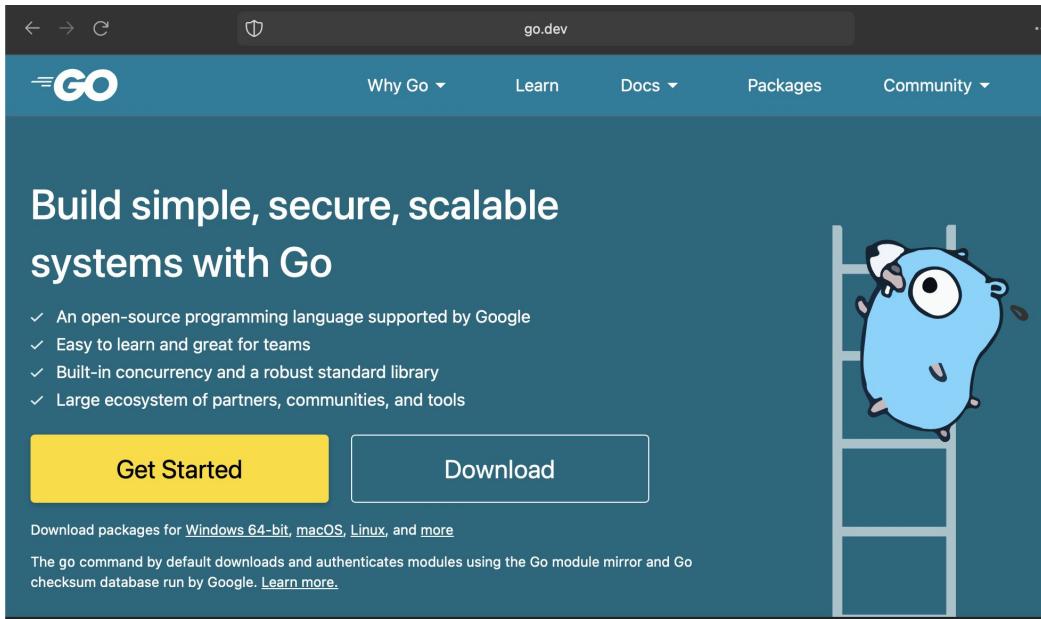


นิยมใช้กับงาน

- Web Development
- Mobile Application (Android)
- Desktop Application
- Etc



Programming Languages



The screenshot shows the official Go website at go.dev. The header includes navigation links for Why Go, Learn, Docs, Packages, and Community. The main content features a large blue cartoon gopher climbing a ladder, with the text "Build simple, secure, scalable systems with Go". Below this, a list of benefits includes being an open-source language supported by Google, easy to learn, having a robust standard library, and a large ecosystem. Two prominent buttons are "Get Started" and "Download". A note at the bottom explains the go command's function.

Build simple, secure, scalable systems with Go

- ✓ An open-source programming language supported by Google
- ✓ Easy to learn and great for teams
- ✓ Built-in concurrency and a robust standard library
- ✓ Large ecosystem of partners, communities, and tools

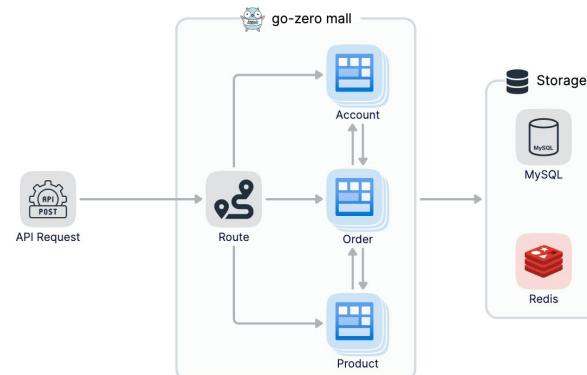
Get Started Download

Download packages for Windows 64-bit, macOS, Linux, and more

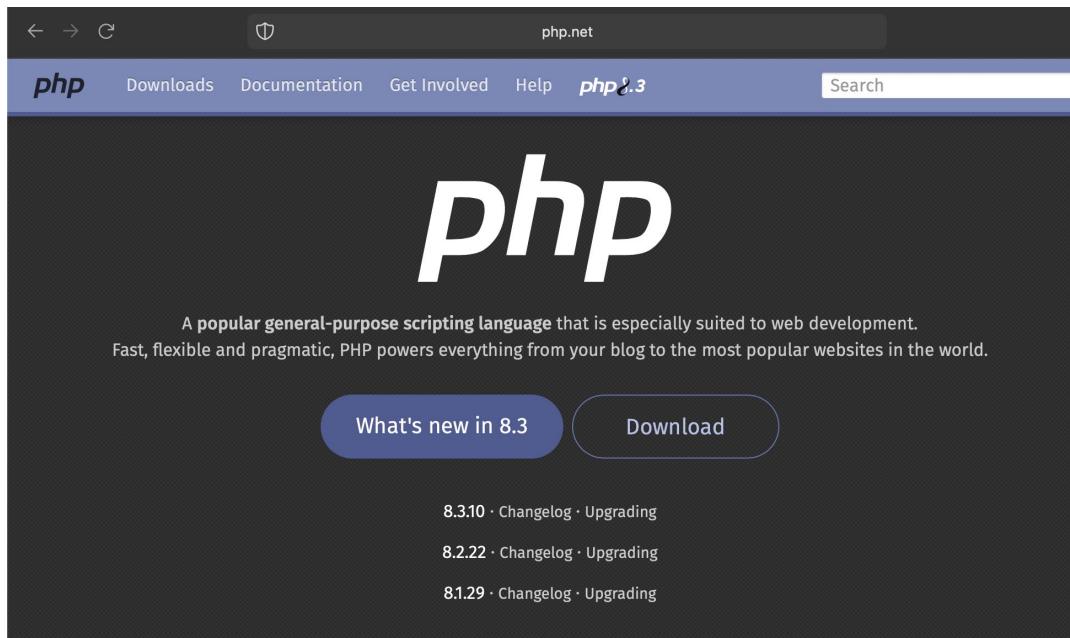
The go command by default downloads and authenticates modules using the Go module mirror and Go checksum database run by Google. [Learn more.](#)

นิยมใช้กันงาน

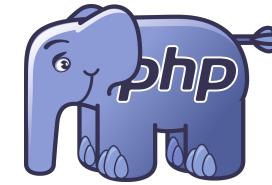
- Web Development (web APIs, web services)
- Network System
- Etc



Programming Languages



The screenshot shows the homepage of the official PHP website (php.net). The header includes navigation links for 'Downloads', 'Documentation', 'Get Involved', 'Help', and the current version 'php 8.3'. A search bar is also present. The main visual is a large, stylized white 'PHP' logo on a dark background. Below it, text reads: 'A popular general-purpose scripting language that is especially suited to web development. Fast, flexible and pragmatic, PHP powers everything from your blog to the most popular websites in the world.' Two prominent buttons at the bottom are 'What's new in 8.3' and 'Download'. Smaller links for '8.3.10 · Changelog · Upgrading', '8.2.22 · Changelog · Upgrading', and '8.1.29 · Changelog · Upgrading' are also visible.



นิยมใช้กันงาน

- Web Development
ร่วมกับ Laravel Web framework



The screenshot shows the homepage of the Laravel framework (laravel.com). The top navigation bar includes links for 'Forge', 'Vapor', 'Ecosystem', 'News', 'Partners', and 'DOCUMENTATION'. The main title is 'The PHP Framework for Web Artisans' with a subtitle 'Deploy Laravel with the infinite scale of serverless using Laravel Vapor'. Below the title, there is a brief description: 'Laravel is a web application framework with expressive, elegant syntax. We've already laid the foundation — freeing you to create without sweating the small things.' At the bottom, there are two buttons: 'GET STARTED' and 'WATCH LABCASTS'.

Programming Languages



นิยมใช้กับงาน

- Operating System (OS)
- Embedded System
- IoT (Internet of Things)
- Etc

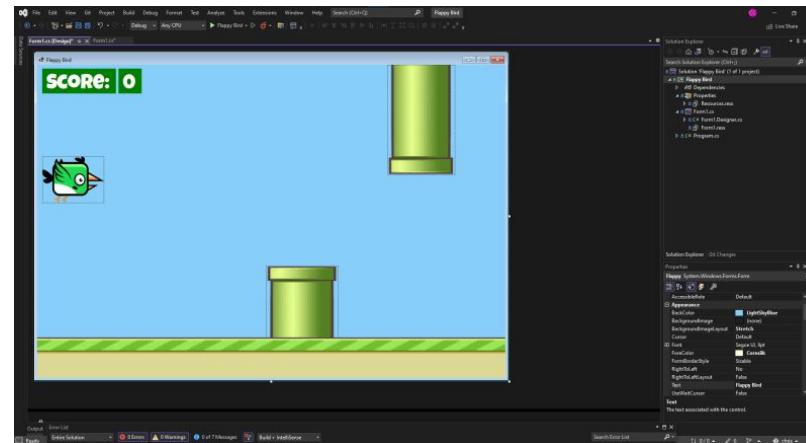


Programming Languages

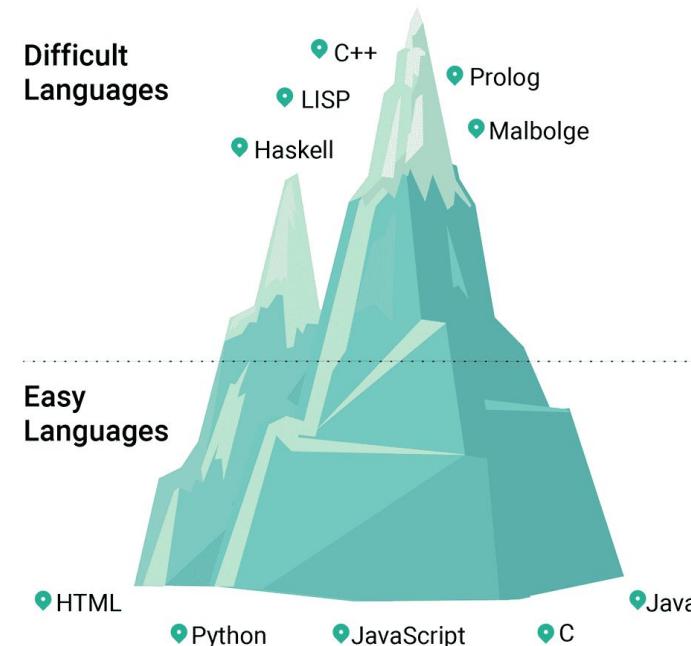


นิยมใช้กับงาน

- Embedded System
- IoT
- Games
- Browser Extension
- Etc

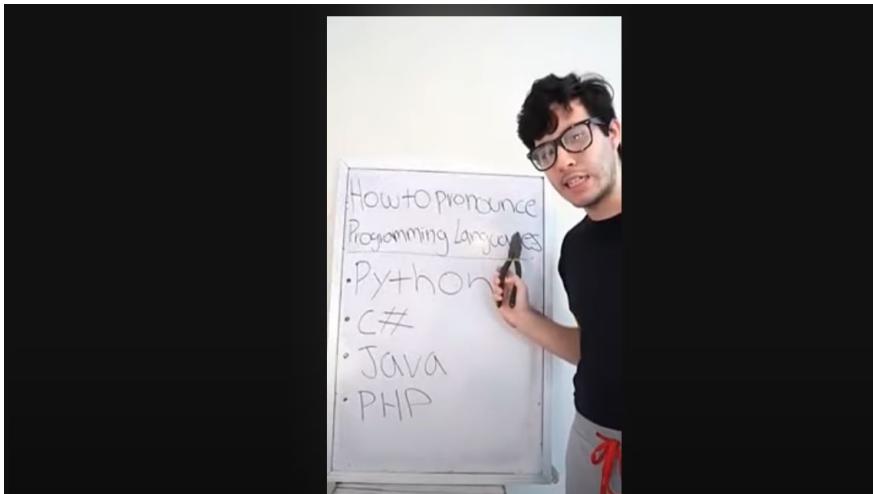


ภาษาโปรแกรมยาก-ง่าย



MEME Programming Language

https://www.youtube.com/watch?v=3WRbWdr_p9M

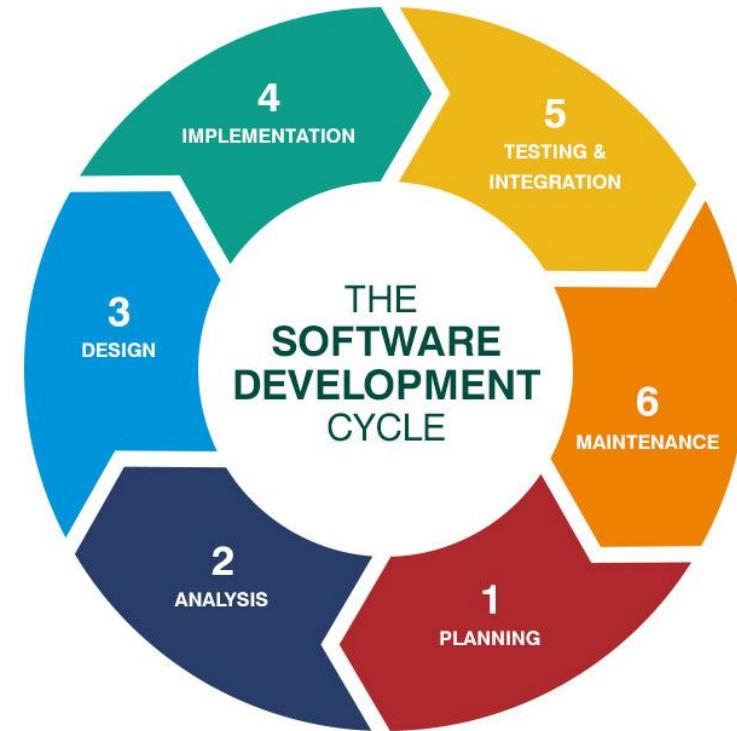


<https://youtube.com/shorts/M1jycCH2kL4?feature=shared>



SDLC คืออะไร

Software Development Life Cycle
“วงจรชีวิตซอฟต์แวร์ระยะพัฒนา”



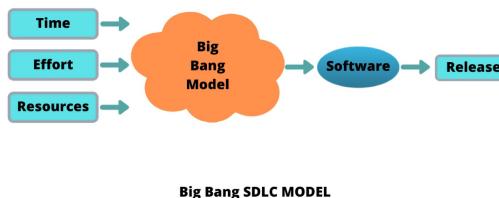
Software Development Life Cycle (SDLC) Phases

Software Development Life Cycle (SDLC) Phases

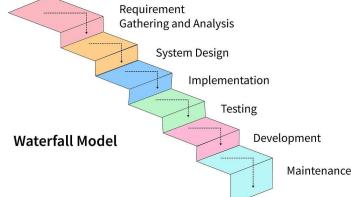


7 ประเภท SDLC Model

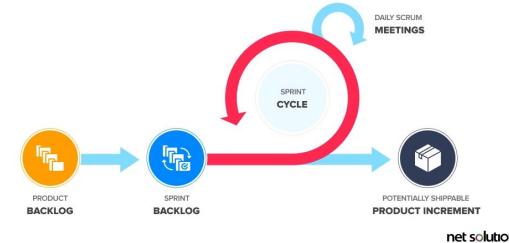
1. Big Bang Model



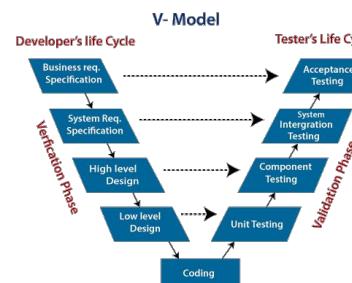
2. Waterfall Model



7. Agile Model



3. V-Shaped Model



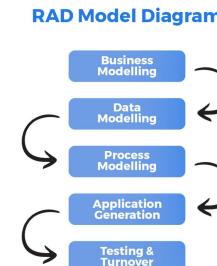
4. Iterative Model



6. Spiral Model



5. RAD Model



การเขียนโค้ดด้วยภาษา Python

ติดตั้ง Python

หลักภาษาและไวยกรณ์ทั่วของภาษา Python

- ตัวแปรและประเภทข้อมูลพื้นฐาน (Variables & Basic Data Types)
- การตั้งชื่อตัวแปร (Naming Variable)
- ตัวดำเนินการพื้นฐาน (Basic Operators)

การเขียนคำสั่งเกี่ยวกับทางเลือก/เงื่อนไข (Condition)

- if..else

การวนลูป (Loops)

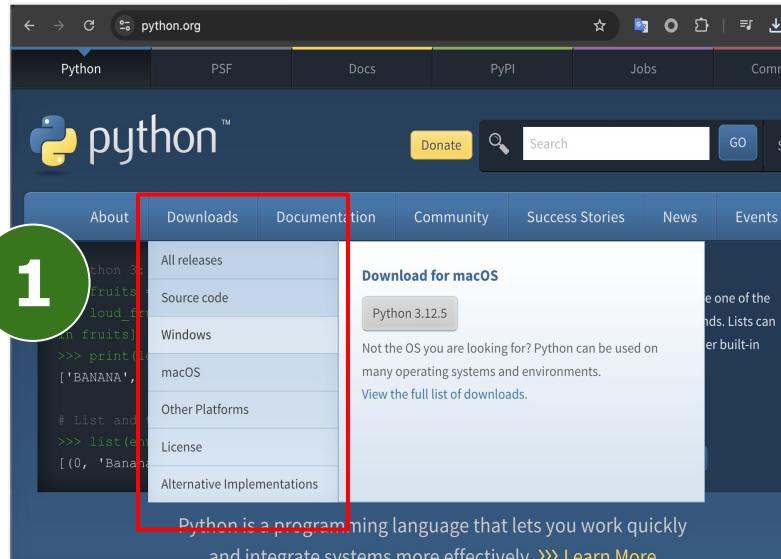
- For loops
- While loops
- Break and continue

การสร้างและใช้งานฟังก์ชัน (Function)



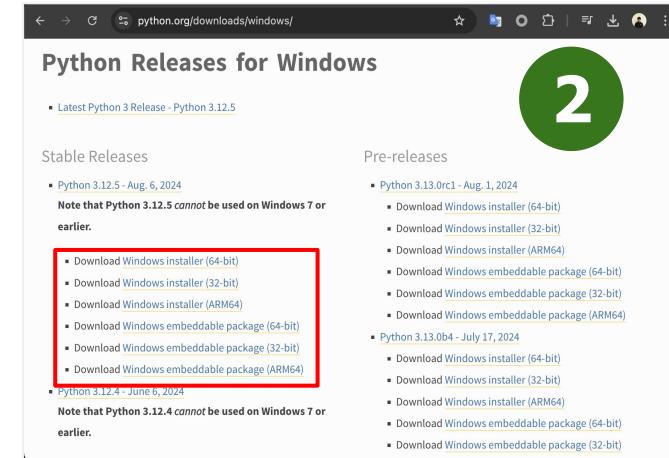
ติดตั้ง Python

<https://www.python.org>



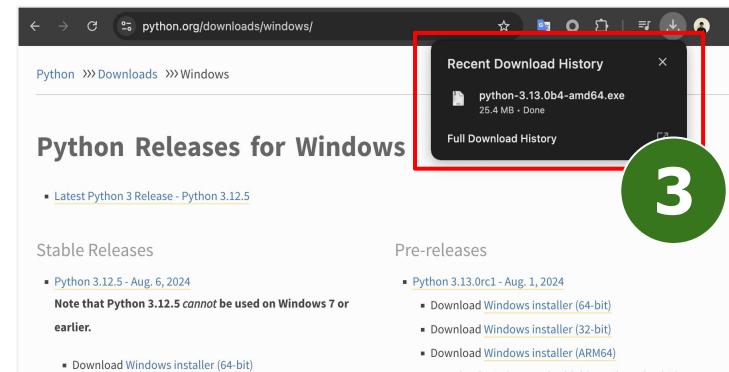
1

The screenshot shows the Python.org homepage. A red box highlights the 'Downloads' menu item in the top navigation bar. Below the menu, there is a sidebar with links for 'All releases', 'Source code', 'Windows', 'macOS', 'Other Platforms', 'License', and 'Alternative Implementations'. A large green circle with the number '1' is positioned to the left of the screenshot.



2

The screenshot shows the 'Python Releases for Windows' page. It displays the 'Stable Releases' section, which includes links for Python 3.12.5 (August 6, 2024) and Python 3.12.4 (June 6, 2024). A note states that Python 3.12.5 cannot be used on Windows 7 or earlier. A red box highlights the download links for Python 3.12.5, including 'Download Windows installer (64-bit)', 'Download Windows installer (32-bit)', 'Download Windows installer (ARM64)', 'Download Windows embeddable package (64-bit)', 'Download Windows embeddable package (32-bit)', and 'Download Windows embeddable package (ARM64)'. A large green circle with the number '2' is positioned to the right of the screenshot.

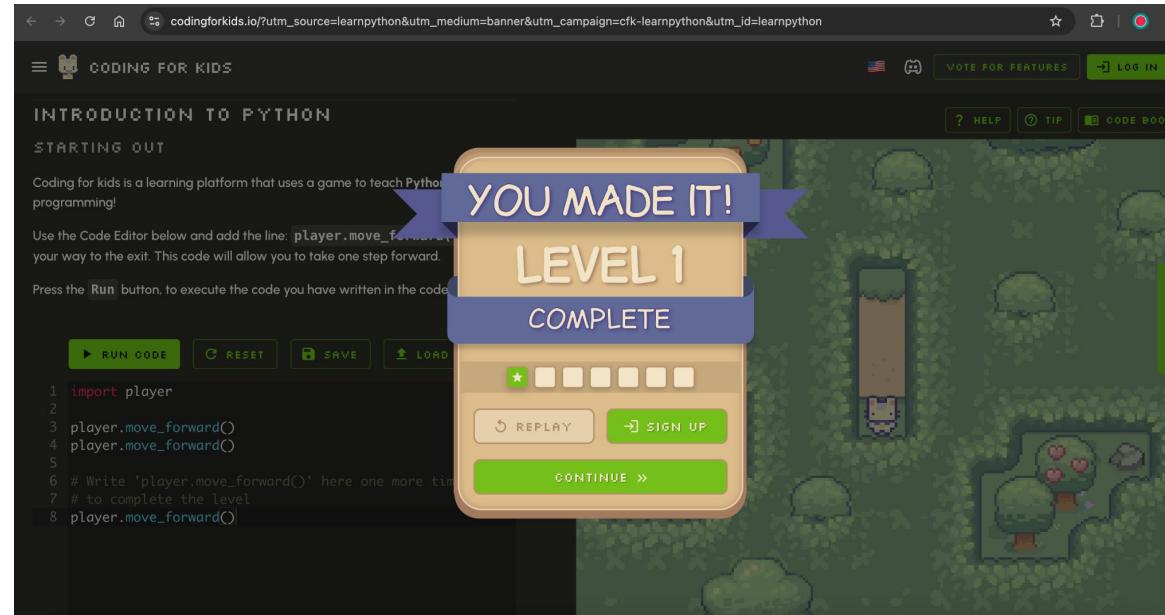


3

The screenshot shows the 'Python Releases for Windows' page again. A red box highlights the 'Recent Download History' window, which lists a single file: 'python-3.13.0b4-amd64.exe' (25.4 MB, Done). A large green circle with the number '3' is positioned to the right of the screenshot.

Coding for Kids

<https://codingforkids.io/>



ตัวแปรและประเภทข้อมูลพื้นฐาน (Variables & Basic Data Types)

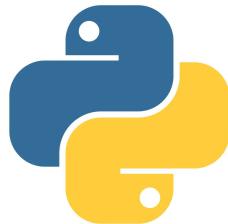
ตัวแปร (Variable)

```
age = 10  
total = 33.33  
name = "Bob"  
is_student = True
```

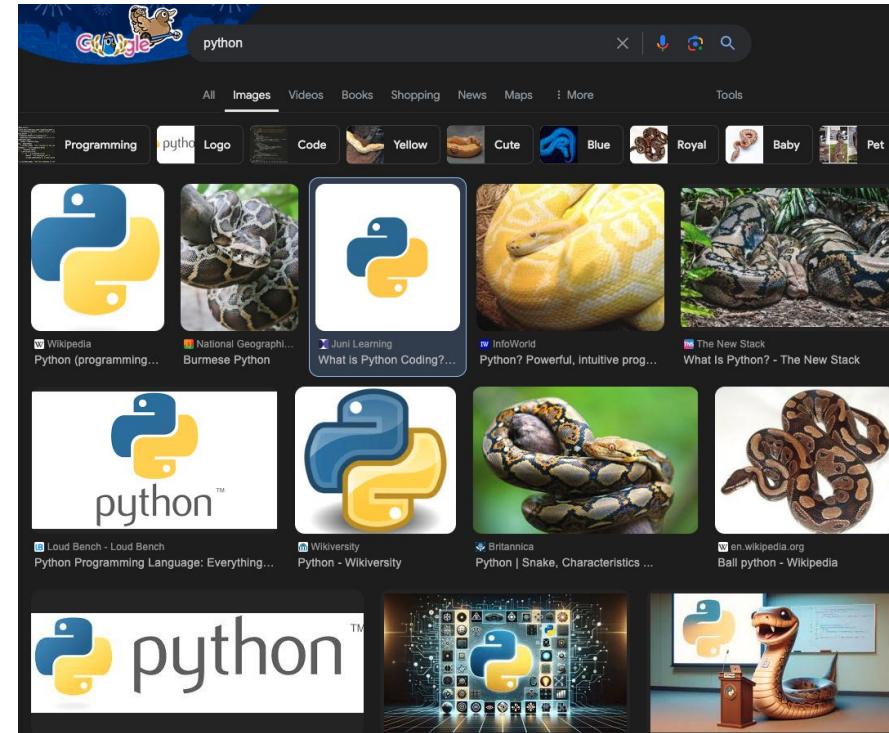
ประเภทข้อมูลพื้นฐาน (Basic Data Types)

int	เช่น 10, -5
float	เช่น 3.14, -0.001
str	เช่น "Hello", "123"
bool	True หรือ False

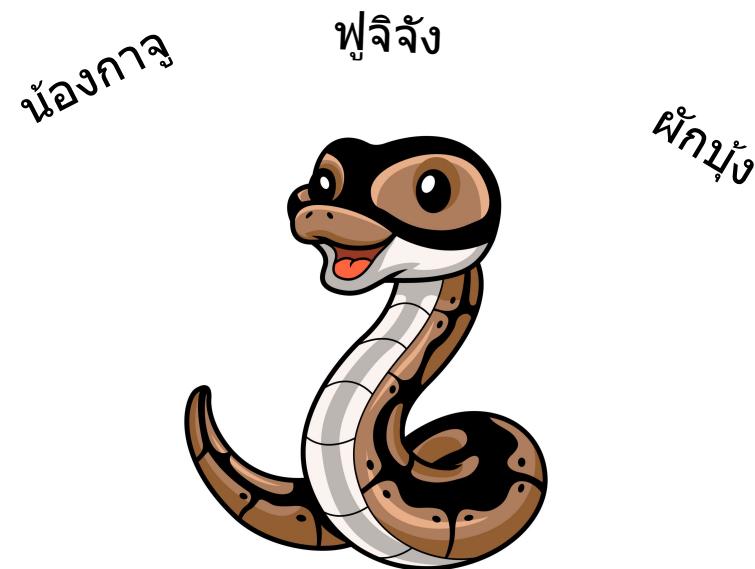
Python ???



ICON



การตั้งชื่อตัวแปร (Naming Variable) ของภาษา Python



การตั้งชื่อตัวแปร (Naming Variable) ของภาษา Python

- ชื่อควรสื่อความหมาย เช่น age แทนที่จะใช้ a
- ใช้ตัวอักษรเล็ก-ใหญ่ (**case-sensitive**) เช่น Age และ age เป็นคุณลักษณะตัวแปร
- ห้ามใช้คำสงวนของ Python เป็นชื่อตัวแปร เช่น def, class
- ต้องเริ่มต้นด้วยตัวอักษรเล็ก (a-z), ตัวอักษรใหญ่ (A-Z), หรือ _ (Underscore) เท่านั้น
หลังจากตัวแรก (a-z), ตัวอักษรใหญ่ (A-Z), ตัวเลข (0-9), และ _ (Underscore)

```
my age
Name_01
FirstName
_result_status_1
```

การประกาศตัวแปร (Variable declare)

ชื่อตัวแปร = ค่า

ตัวอย่าง : การประกาศตัวแปร

```
name = "Bob"          # ข้อความ
age = 25              # จำนวนเต็ม
height = 1.75         # จำนวนทศนิยม
is student = True     # Boolean
result = None          # None
```

```
print(type(name))      # <class 'str'>
print(type(age))       # <class 'int'>
print(type(height))    # <class 'float'>
print(type(is student))# <class 'bool'>
print(type(result))    # <class 'NoneType'>
```

```
x, y, z = 10, 20, 30
x = y = z = 10, 20, 30
```

ຕຳດໍາເນີນການພື້ນຖານ (Basic Operators)

- **Arithmetic operators** (+, -, *, /, %, **, //)
- **Assignment operators** (=, +=, -=, *=, /=, %=, **=, etc.)
- **Comparison operators** (==, !=, >, <, >=, <=)
- **Logical operators** (and, or, not)

ຕ້ວອຍ່າງ ການໃຊ້ຕົວດຳເນີນການພື້ນຖານ (Basic Operators)

ຕ້ວອຍ່າງ : Arithmetic operators (+, -, *, /, %, **, //)

```
print(5 + 10)    #15
print(8 - 1)     #7
print(4 * 5)     #20
print(5 / 2)      #2.5
print(9 % 2)      #1
print(3 ** 2)     #9
print(9 // 2)     #4
```

ຕ້ວອຍ່າງ : Comparison operators (==, !=, >, <, >=, <=)

```
x = 5
print(3 < x)      #True
print(x > 1)      #True
print(x == 5)      #True
print(x >= 6)     #False
print(x <= 5)      #True
print(x != 10)     #True
```

```
a = "Hello" + " " + "World!"
print(a) #Hello World!
b = "Hi" * 3
print(b) #HiHiHi
```

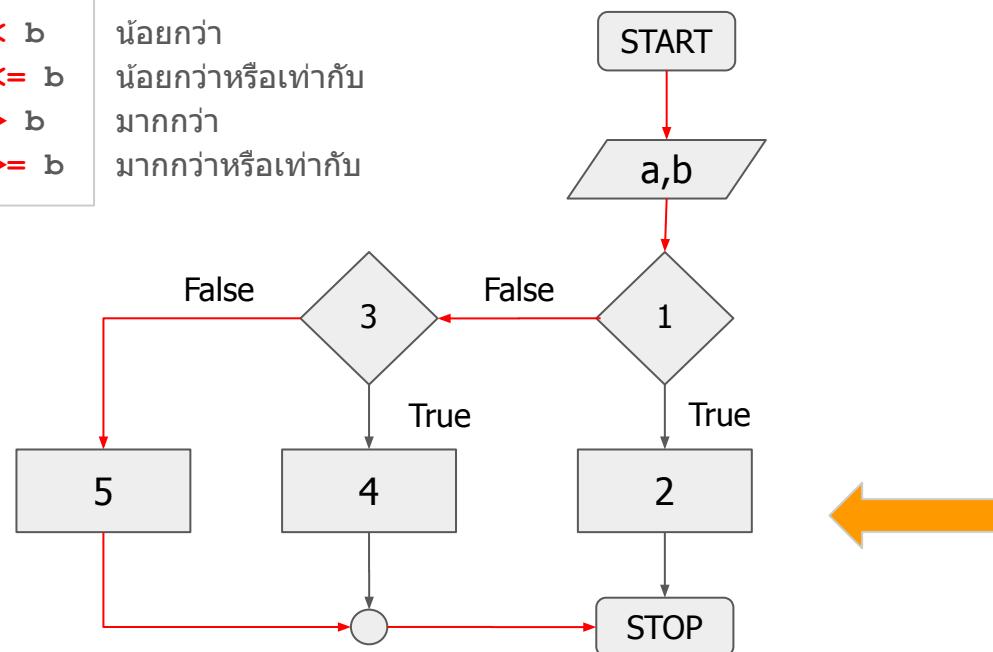
ຕ້ວອຍ່າງ : Logical operators (and, or, not)

```
is_true = (5 > 3) and (3 > 1)          #True
is_false = (5 < 3) or (3 < 1)           #False
is_not_true = not (5 > 3)                #False
```

เงื่อนไข (Condition) - If..Else

<code>a == b</code>	เท่ากัน
<code>a != b</code>	ไม่เท่ากัน
<code>a < b</code>	น้อยกว่า
<code>a <= b</code>	น้อยกว่าหรือเท่ากัน
<code>a > b</code>	มากกว่า
<code>a >= b</code>	มากกว่าหรือเท่ากัน

Result: 1-3-5



If

```

a = 20
b = 10
if b > a:
    print("b มากกว่า a")
  
```

Elif

```

a = 20
b = 10
if b > a:
    print("b มากกว่า a")
elif a == b:
    print("a และ b เท่ากัน")
  
```

Else

```

a = 20
b = 10
if b > a:
    print("b มากกว่า a") #1
elif a == b:
    print("a และ b เท่ากัน") #3
else:
    print("a มากกว่า b") #5
  
```

ลูป (Loops) - While loop

if

```
i = 1
while i < 6:
    print(i)
    i += 1
```

ผลลัพธ์

```
1
2
3
4
5
```

break

```
i = 1
while i < 6:
    print(i)
    if i == 3:
        break
    i += 1
```

ผลลัพธ์

```
1
2
3
```

else

```
i = 1
while i < 6:
    print(i)
    i += 1
else:
    print("i ไม่ได้มีค่ามากกว่า 6")
```

ผลลัพธ์

```
1
2
3
4
5
6 ไม่ได้มีค่ามากกว่า 6
```

Continue

```
i = 0
while i < 6:
    i += 1
    if i == 3:
        continue
    print(i)
```

ผลลัพธ์

```
1
2
4
5
```

ลูป (Loops) - For loop

แสดงค่าของตัวแปร num

```
num = ["1", "2", "3"]
for x in num:
    print(x)
```

ผลลัพธ์

```
1
2
3
```

แสดงค่า strings

```
for x in "number":
    print(x)
```

ผลลัพธ์

```
n
u
m
b
e
r
```

```
for i in range(5):
    print(i)
```

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

Break

```
num = ["1", "2", "3"]
for x in num:
    print(x)
    if x == "2":
        break
```

ผลลัพธ์

```
1
2
```

Continue

```
num = ["1", "2", "3"]
for x in num:
    if x == "2":
        continue
    print(x)
```

ผลลัพธ์

```
1
3
```

การสร้างและเรียกใช้งานฟังก์ชัน (Function)

สร้างฟังก์ชัน

```
def my_function():
    print("Hello from a function")
```

ผลลัพธ์

(Empty box)

เรียกใช้งานฟังก์ชัน

```
def my_function():
    print("Hello from a function")

my_function()
```

ผลลัพธ์

Hello from a function

เรียกใช้งานฟังก์ชัน โดยส่งตัวแปร

```
def my_function(fname, lname):
    print(fname + " " + lname)

my_function("Bob", "Snow")
```

ผลลัพธ์

Bob Snow

ໂມດູລ (Module)

สร้าง Function ดังต่อไปนี้

File: mymodule.py

```
def greeting(name):
    print("Hello, " + name)
```

สร้างไฟล์สำหรับเรียกใช้งาน Module

File: main.py

```
import mymodule

mymodule.greeting("Jonathan")
```

```
# operations.py

# a method that adds two numbers and return the result
def add(a, b):
    result = a + b
    return result

# a method that multipy two numbers and return the result
def mul(a, b):
    result = a*b
    return result

# a method that subtracts two numbers and return the result
def sub(a, b):
    result = a-b
    return result
```

```
import operations.py as calculator

result = calculator.add(1,1)
print(result)
```

ເຄື່ອງຈັດການແພຶກເກີຈ (package manager)

ຕິດຕັ້ງ pip ໄດ້ທີ່

```
https://pypi.org/project/pip/
```

ຕິດຕັ້ງ Package ທີ່ຕ້ອງການໃຊ້

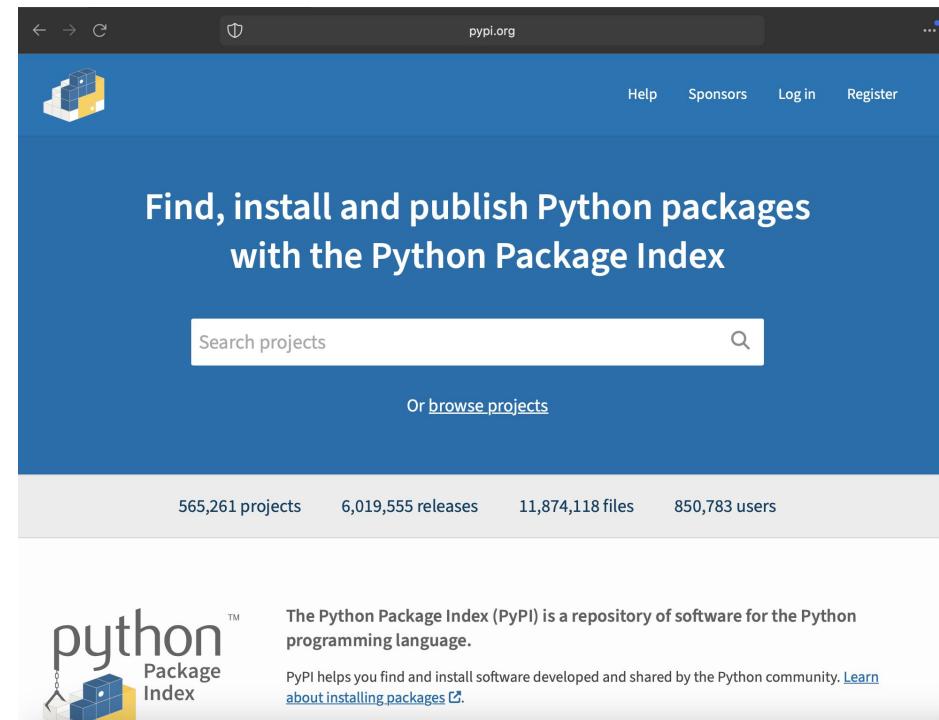
```
C:\>pip install camelcase
```

ເຮັດໃຫ້ງານໄດ້ສະດວກ

```
import camelcase

c = camelcase.CamelCase()
txt = "hello world"

print(c.hump(txt))
```



Socket methods

Server

```
socket.socket()  
.bind()  
.listen()  
.accept()  
.connect()  
.send()  
.recv()  
.close()
```

Send message request to server

Server



```
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('localhost', 12345))
s.listen(5)

print("Waiting for connect...")

while True:
    conn, addr = s.accept()
    print(f"Connected by {addr}")
    data = conn.recv(1024)
    print(f"Client {addr[0]}: {data.decode()}")
    conn.send(b'Hello!')
    conn.close()
```

Client



```
import socket

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('localhost', 12345))
client_socket.send(b'Hello!')
data = client_socket.recv(1024)
print(f"Server: {data.decode()}")
client_socket.close()
```