บจก.สยามถนัดแฮก

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
- ตัวอย่างโค้ด และรูปใน การบรรยาย นี้ เป็นระบบจำลองของทางผู้บรรยาย ไม่ใช่ระบบลูกค้า

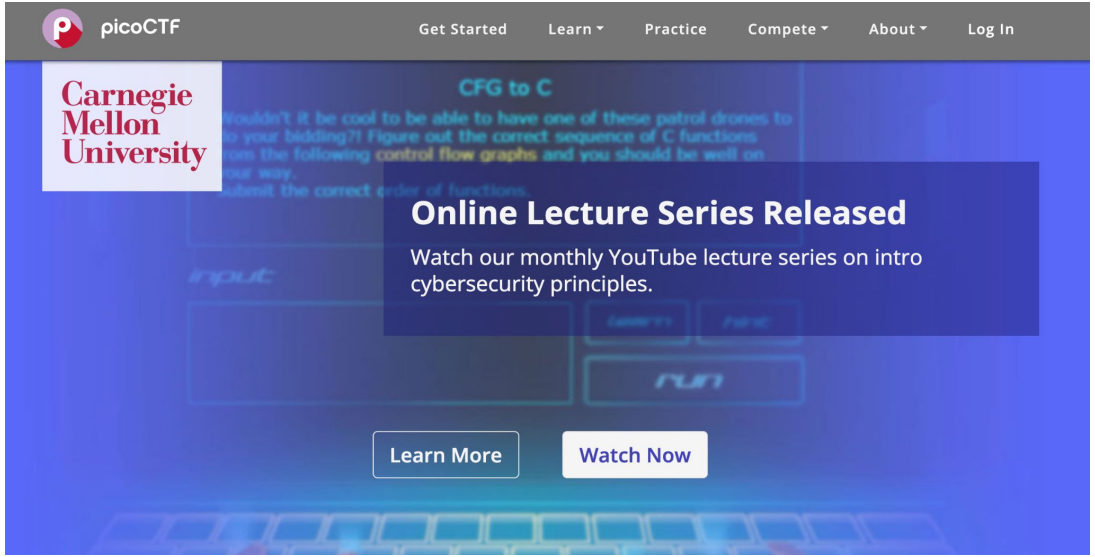
Agenda (Day 2)

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

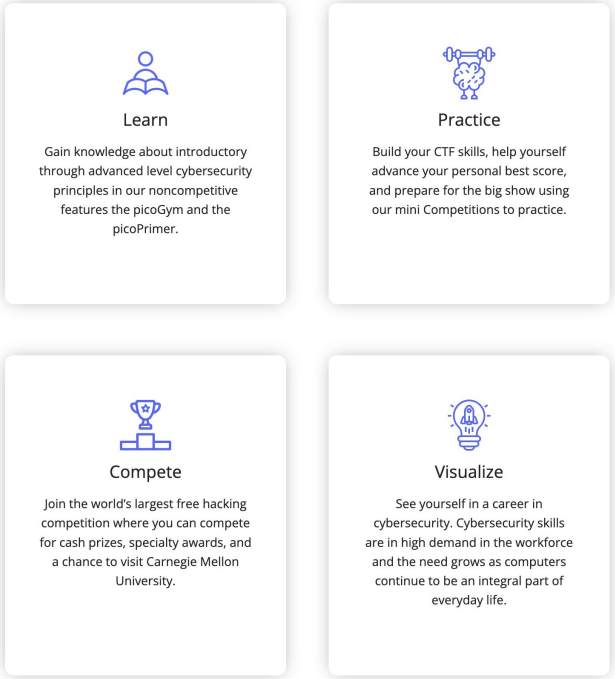
Content Overview

- 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



https://picoctf.org



Lab 1 - ลองทำ Socket อ่านข้อมูล

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

$ nc 127.0.0.1 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...
```

```
(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
```

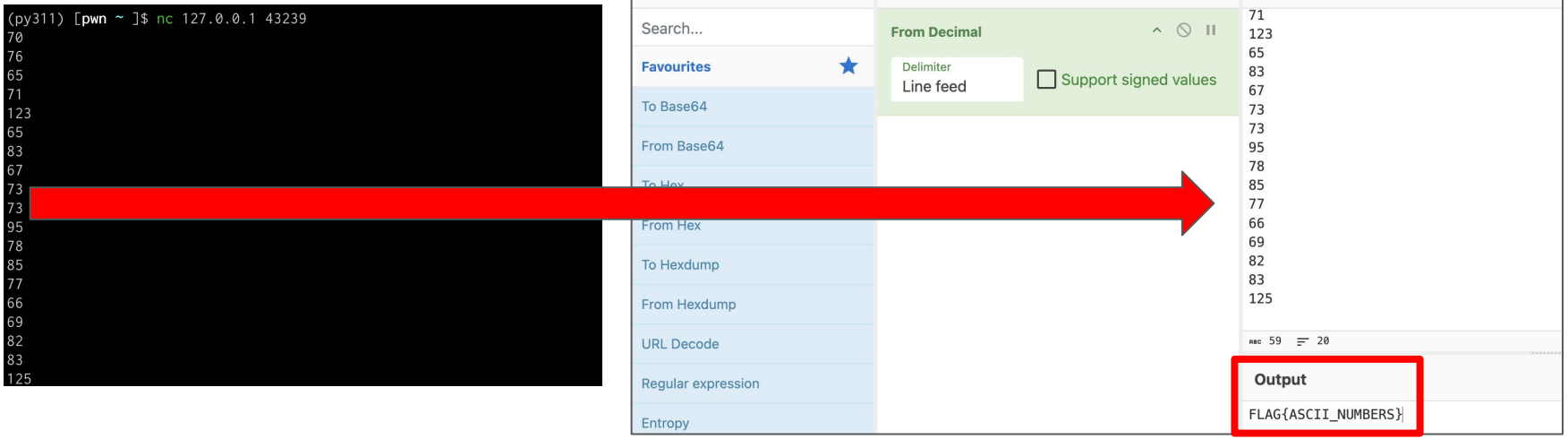
Lab 1 - ลองทำ Socket อ่านข้อมูล

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

https://www.cs.cmu.edu/~pattis/15-1XX/common/handouts/ascii.html

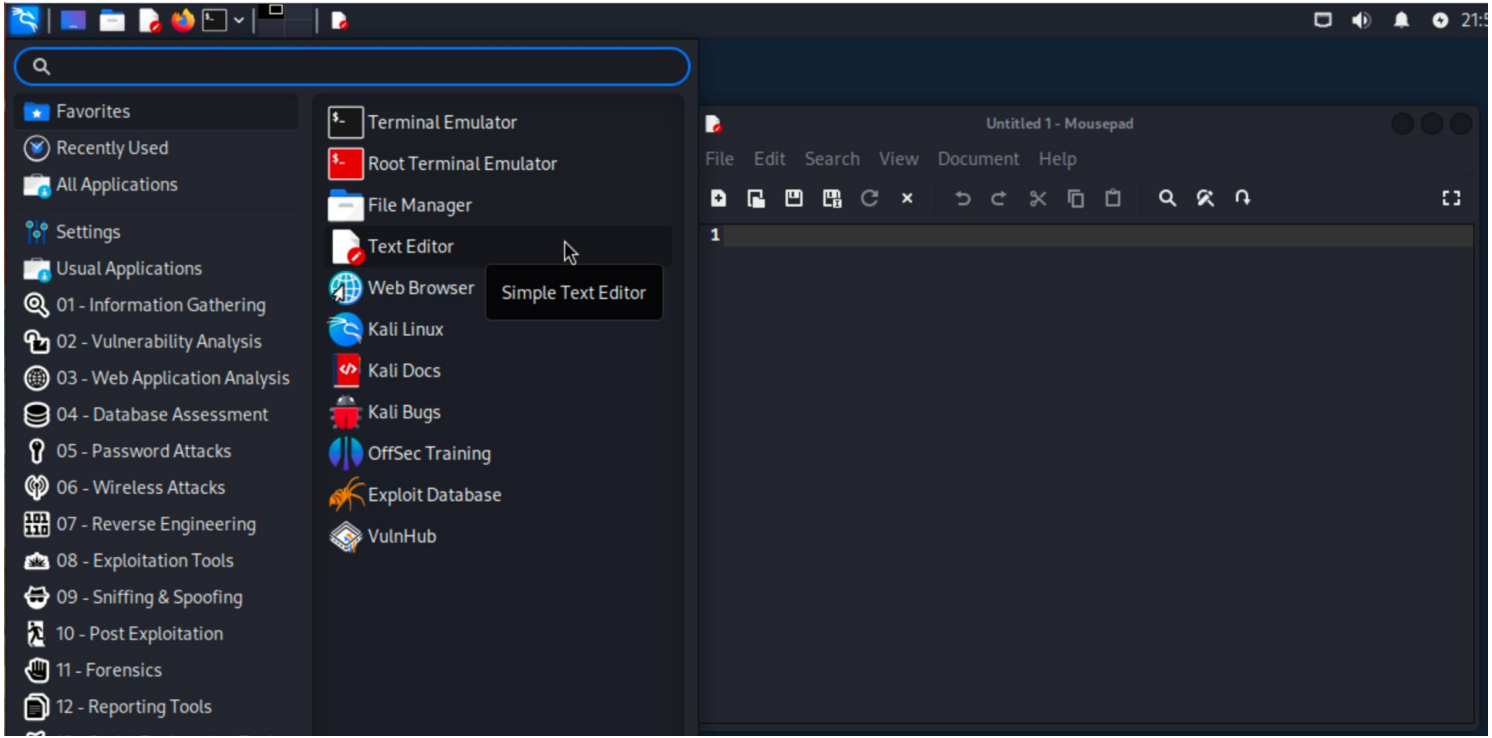
Lab 1 - ลองทำ Socket อ่านข้อมูล

$ nc 127.0.0.1 43239



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

```python
import socket

def main():
    # Server details
    host = '127.0.0.1'
    port = 43239

    # Create a socket and connect to the server
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((host, port))

    # Buffer to hold received data
    full_data = ""

    while True:
        # Receive data from the server in chunks of 1024 bytes
        data = client.recv(1024).decode()

        if not data:
            # No more data is being received, break the loop
            break

        # Append the received data to the buffer
        full_data += data

    # Print the full result
    print("Data: ", full_data)

    # Close the connection
    client.close()

if __name__ == "__main__":
    main()
```

lab1_learn.py

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

1

การ นำเข้า
Module ชื่อ socket
เพื่อใช้ฟังก์ชันเพิ่มเติม

python lab1_learn.py

Lab 1 - ลองทำ Socket อ่านข้อมูล

```python
import socket

def main():
    # Server details
    host = '127.0.0.1'
    port = 43239

    # Create a socket and connect to the server
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((host, port))

    # Buffer to hold received data
    full_data = ""

    while True:
        # Receive data from the server in chunks of 1024 bytes
        data = client.recv(1024).decode()

        if not data:
            # No more data is being received, break the loop
            break

        # Append the received data to the buffer
        full_data += data

    # Print the full result
    print("Data: ", full_data)

    # Close the connection
    client.close()

if __name__ == "__main__":
    main()
```

Part 1 - อ่านข้อมูลจาก Server

lab1_learn.py

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

2

เริ่มโปรแกรม Python

python lab1_learn.py

Lab 1 - ลองทำ Socket อ่านข้อมูล

```python
import socket

def main():
    # Server details
    host = '127.0.0.1'
    port = 43239

    # Create a socket and connect to the server
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((host, port))

    # Buffer to hold received data
    full_data = ""

    while True:
        # Receive data from the server in chunks of 1024 bytes
        data = client.recv(1024).decode()

        if not data:
            # No more data is being received, break the loop
            break

        # Append the received data to the buffer
        full_data += data

    # Print the full result
    print("Data: ", full_data)

    # Close the connection
    client.close()

if __name__ == "__main__":
    main()
```

Part 1 - อ่านข้อมูลจาก Server

lab1_learn.py

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

3

ระบุเซิร์ฟเวอร์และ Port ที่ต้องการเชื่อมต่อ

python lab1_learn.py

Lab 1 - ลองทำ Socket อ่านข้อมูล

```python
import socket

def main():
    # Server details
    host = '127.0.0.1'
    port = 43239

    # Create a socket and connect to the server
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((host, port))

    # Buffer to hold received data
    full_data = ""

    while True:
        # Receive data from the server in chunks of 1024 bytes
        data = client.recv(1024).decode()

        if not data:
            # No more data is being received, break the loop
            break

        # Append the received data to the buffer
        full_data += data

    # Print the full result
    print("Data: ", full_data)

    # Close the connection
    client.close()

if __name__ == "__main__":
    main()
```

Part 1 - อ่านข้อมูลจาก Server

lab1_learn.py

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

4

connect()
ใช้เชื่อมต่อไปยังเซิร์ฟเวอร์และ Port ที่ระบุ

python lab1_learn.py

Lab 1 - ลองทำ Socket อ่านข้อมูล

```python
import socket

def main():
    # Server details
    host = '127.0.0.1'
    port = 43239

    # Create a socket and connect to the server
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((host, port))

    # Buffer to hold received data
    full_data = ""

    while True:
        # Receive data from the server in chu
        data = client.recv(1024).decode()

        if not data:
            # No more data is being received, break the loop
            break

        # Append the received data to the buffer
        full_data += data

    # Print the full result
    print("Data: ", full_data)

    # Close the connection
    client.close()

if __name__ == "__main__":
    main()
```

Part 1 - อ่านข้อมูลจาก Server

lab1_learn.py

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

5

วนลูป
(เพื่ออ่านข้อมูล
ที่เซิร์ฟเวอร์ตอบกลับมา)

python lab1_learn.py

Lab 1 - ลองทำ Socket อ่านข้อมูล

```python
import socket

def main():
    # Server details
    host = '127.0.0.1'
    port = 43239

    # Create a socket and connect to the server
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((host, port))

    # Buffer to hold received data
    full_data = ""

    while True:
        # Receive data from the server in chunks of 1024 bytes
        data = client.recv(1024).decode()

        if not data:
            # No more data is being received, break the loop
            break

        # Append the received data to the buffer
        full_data += data

    # Print the full result
    print("Data: ", full_data)

    # Close the connection
    client.close()

if __name__ == "__main__":
    main()
```

Part 1 - อ่านข้อมูลจาก Server

lab1_learn.py

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

6

อ่านข้อมูลที่เซิร์ฟเวอร์ตอกลับมาด้วย recv()

python lab1_learn.py

Lab 1 - ลองทำ Socket อ่านข้อมูล

```python
import socket

def main():
    # Server details
    host = '127.0.0.1'
    port = 43239

    # Create a socket and connect to the server
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((host, port))

    # Buffer to hold received data
    full_data = ""

    while True:
        # Receive data from the server in chunks of 1024 bytes
        data = client.recv(1024).decode()

        if not data:
            # No more data is being received, break the loop
            break

        # Append the received data to the buffer
        full_data += data

    # Print the full result
    print("Data: ", full_data)

    # Close the connection
    client.close()

if __name__ == "__main__":
    main()
```

Part 1 - อ่านข้อมูลจาก Server

lab1_learn.py

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

7

แสดงข้อมูลที่เซิร์ฟเวอร์ตอบกลับมา ออกมาทาง Console (Standard Input)

python lab1_learn.py

Lab 1 - ลองทำ Socket อ่านข้อมูล

```python
import socket

def main():
    # Server details
    host = '127.0.0.1'
    port = 43239

    # Create a socket and connect to the server
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((host, port))

    # Buffer to hold received data
    full_data = ""

    while True:
        # Receive data from the server in chunks of 1024 bytes
        data = client.recv(1024).decode()

        if not data:
            # No more data is being received, break the loop
            break

        # Append the received data to the buffer
        full_data += data

    # Print the full result
    print("Data: ", full_data)

    # Close the connection
    client.close()

if __name__ == "__main__":
    main()
```

Part 1 - อ่านข้อมูลจาก Server

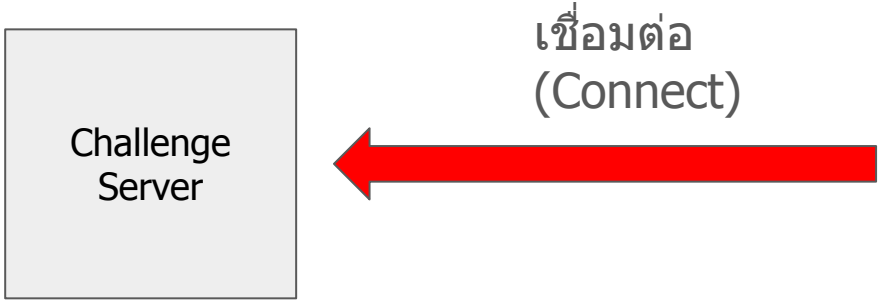
lab1_learn.py

1. การ นำเข้า
Module ชื่อ socket
เพื่อใช้ฟังก์ชันเพิ่มเติม
2. main() เริ่มโปรแกรม Python
3. ระบุเซิร์ฟเวอร์และ Port ที่ต้องการเชื่อมต่อ
4. **connect()**
ใช้เชื่อมต่อไปยัง เซิร์ฟเวอร์และ Port ที่ระบุ
5. วนลูป
(เพื่ออ่านข้อมูล ที่เซิร์ฟเวอร์ตอบกลับมา)
6. อ่านข้อมูล
ที่เซิร์ฟเวอร์ตอบกลับมา ด้วย **recv()**
7. แสดงข้อมูลที่เซิร์ฟเวอร์ตอบ
กลับมา ออกมาทาง 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
95
78
85
77
66
69
82
83
125
```

Lab 1 - ลองทำ Socket อ่านข้อมูล

```python
def ascii_to_string(data):
    # Split the data into individual ASCII numbers
    ascii_numbers = data.split()

    # Convert each ASCII number to its corresponding character
    # and join them into a string
    flag = ''.join(chr(int(num)) for num in ascii_numbers)

    return flag
```

```python
while True:
    # Receive data from the server in chunks of 1024 bytes
    data = client.recv(1024).decode()

    if not data:
        # No more data is being received, break the loop
        break

    # Append the received data to the buffer
    full_data += data

# Convert the received ASCII numbers to a flag string
flag = ascii_to_string(full_data)

# Print the flag
print("Flag: ", flag)
```

lab1_solve.py

1. สร้างฟังก์ชันเพื่อ Decode ค่า ตัวเลข ASCII เป็น ตัวอักษร
2. 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

Part 2 - แปลงข้อมูลเป็นค่า Flag

Lab 1 - ลองทำ Socket อ่านข้อมูล

สรุปการเรียนรู้ใน Lab 1

Part 1 - อ่านข้อมูลจาก Server

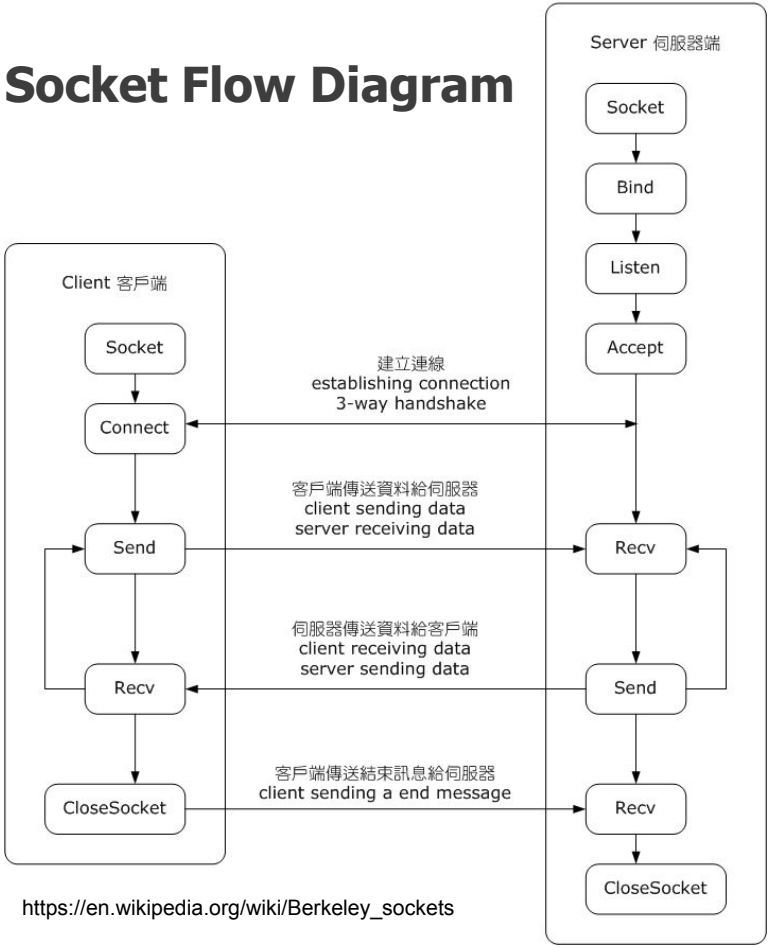
- connect()
- recv()

Part 2 - แปลงข้อมูลเป็นค่า Flag

- สร้างฟังก์ชัน ascii_to_string() เพื่อแปลงค่าตัวเลข ASCII เป็นตัวอักษร

```python
import socket

def ascii_to_string(data):
    # Split the data into individual ASCII numbers
    ascii_numbers = data.split()

    # Convert each ASCII number to its corresponding character
    # and join them into a string
    flag = ''.join(chr(int(num)) for num in ascii_numbers)

    return flag

def main():
    # Server details
    host = '127.0.0.1'
    port = 43239

    # Create a socket and connect to the server
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((host, port))

    # Buffer to hold received data
    full_data = ""

    while True:
        # Receive data from the server in chunks of 1024 bytes
        data = client.recv(1024).decode()

        if not data:
            # No more data is being received, break the loop
            break

        # Append the received data to the buffer
        full_data += data

    # Convert the received ASCII numbers to a flag string
    flag = ascii_to_string(full_data)

    # Print the flag
    print("Flag: ", flag)

    # Close the connection
    client.close()

if __name__ == "__main__":
    main()
```

TCP Socket Flow Diagram



https://en.wikipedia.org/wiki/Berkeley_sockets

lab1_server.go

```go
func main() {
    // Listen on port 43239
    listener, err := net.Listen("tcp", ":43239")
    if err != nil {
        fmt.Println("Error starting server:", err)
        return
    }
    defer listener.Close()
    fmt.Println("Server listening on port 43239...")

    for {
        // Accept connection
        conn, err := listener.Accept()
        if err != nil {
            fmt.Println("Error accepting connection:", err)
            continue
        }

        // Handle client connection in a goroutine
        go handleClient(conn)
    }
}

func handleClient(conn net.Conn) {
    defer conn.Close()

    // Convert the flag string to ASCII values
    asciiValues := convertToASCII(flag)

    // Send the ASCII values to the client, one per line
    for _, val := range asciiValues {
        conn.Write([]byte(strconv.Itoa(val) + "\n"))
    }
}
```

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

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!
```

1

2

Lab 2 - ลองทำ Socket ตอบกลับข้อมูล

โปรแกรมที่เราจะลองเขียน

Part 1 - จำแนกข้อมูลจาก Server
Part 2 - คำนวณและส่งคำตอบ

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: 10110111
Binary Number 2: 10001111

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

1

```python
def main():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(('127.0.0.1', 55816))

    print(client_socket.recv(1024).decode())

    binary1 = binary2 = None

    while True:
        # Receive and display server message, now using the delimiter function
        response = recv_until_delimiter(client_socket)
        print(f"Server response:\n{response}")

        operation = None  # Ensure operation is initialized for each loop

        # Parse binary numbers and operation from server's message
        for line in response.split('\n'):
            if 'Binary Number 1' in line and ': ' in line:
                binary1 = line.split(': ')[1].strip()
            if 'Binary Number 2' in line and ': ' in line:
                binary2 = line.split(': ')[1].strip()
            if 'Operation' in line and ': ' in line:
                operation = line.split(': ')[1].replace("'", "").strip()

```

2

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

Server response: Your task is to perform the uniqu
 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 - คำนวณและส่งคำตอบ

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
```

1

3

```
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/bit
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
```

2

Lab 2 - ลองทำ Socket ตอบกลับข้อมูล

```python
import socket

# Function to handle binary operations directly
def perform_operation(op_symbol, binary1, binary2=None):
    # Convert binary strings to integers
    a = int(binary1, 2)
    b = int(binary2, 2) if binary2 is not None else None

    # Define the possible operations using direct arithmetic/bitwise operators
    operations = {
        '<<': lambda x, y: bin(x << y)[2:].zfill(8),  # Left shift
        '>>': lambda x, y: bin(x >> y)[2:].zfill(8),  # Right shift
        '+': lambda x, y: bin(x + y)[2:].zfill(8),    # Addition
        '*': lambda x, y: bin(x * y)[2:].zfill(8),    # Multiplication
        '-': lambda x, y: bin(x - y)[2:].zfill(8),    # Subtraction
        '&': lambda x, y: bin(x & y)[2:].zfill(8),    # Bitwise AND
        '|': lambda x, y: bin(x | y)[2:].zfill(8),    # Bitwise OR
        '^': lambda x, y: bin(x ^ y)[2:].zfill(8)     # Bitwise XOR
    }

    # Perform the operation with two operands
    if op_symbol in operations:
        result = operations[op_symbol](a, b)
    else:
        raise ValueError(f"Unknown operation: {op_symbol}")

    return result

# Function to receive data until a delimiter is reached
def recv_until_delimiter(sock, delimiter='\n\n'):
    data = ""
    while delimiter not in data:
        packet = sock.recv(1024).decode()
        if not packet:
            break
        data += packet
    return data

def main():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(('127.0.0.1', 55816))

    print(client_socket.recv(1024).decode())

    binary1 = binary2 = None

    while True:
        # Receive and display server message, now using the delimiter function
        response = recv_until_delimiter(client_socket)
        print(f"Server response:\n{response}")

        operation = None  # Ensure operation is initialized for each loop

        # Parse binary numbers and operation from server's message
        for line in response.split('\n'):
            if 'Binary Number 1' in line and ': ' in line:
                binary1 = line.split(': ')[1].strip()
            if 'Binary Number 2' in line and ': ' in line:
                binary2 = line.split(': ')[1].strip()
            if 'Operation' in line and ': ' in line:
                operation = line.split(': ')[1].replace("'", "").strip()

        # Perform the operation if valid binaries and operation are received
        if binary1 and operation:
            if binary2:  # Handle binary operations
                result = perform_operation(operation, binary1, binary2)
            else:
                continue

            print(f"Result: {result}")
            client_socket.sendall(f"{result}\n".encode())

        # Check if challenge is completed
        if 'The flag is' in response:
            print("Challenge completed.")
            break

if __name__ == "__main__":
    main()
```

Lab 2 - ลองทำ Socket ตอบกลับข้อมูล

```
→ ykh (new_prog_lab2) $ python3 prog_lab2_solve.py
Welcome to the Binary Challenge!

Server response:
Your task is to perform the unique operations in the given order a

Binary Number 1: 10011101
Binary Number 2: 00101110

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


Result: 1001110100000000000000000000000000000000000000000000000000000000
Server response:
Enter the binary result: Correct!


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

Enter the binary result:
Result: 01101111
Server response:
Correct!

Question 3/6:
Operation 3: '&'
Perform the operation on Binary Number 1 and Binary Number 2.


Result: 00001100
Server response:
Enter the binary result: Correct!
```

```
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}


Challenge completed.
```

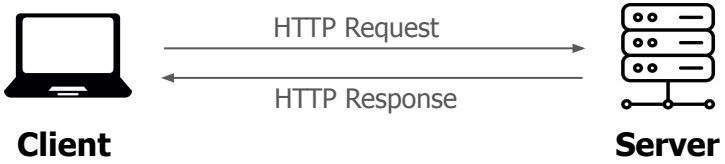
Lab 2 - ลองทำ Socket ตอบกลับข้อมูล

สรุปการเรียนรู้ใน Lab 2

Part 1 - จำแนกข้อมูลจาก Server
- in
- split

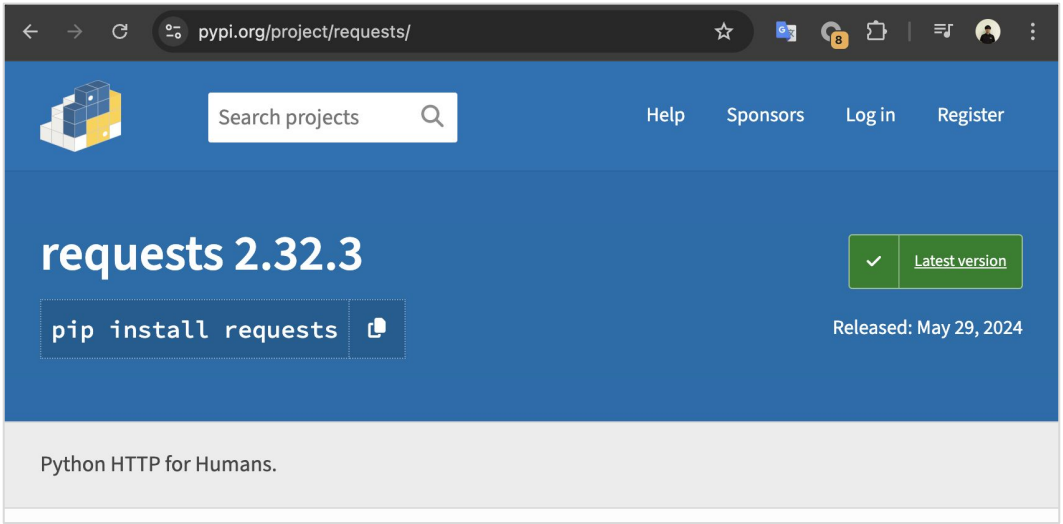
Part 2 - คำนวณและส่งคำตอบ
- สร้างฟังก์ชัน perform_operation() เพื่อคำนวณค่าตัวเลข Binary
- sendall()

การทำ HTTP Client



requests module

```
$ pip install requests
```



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

Basic GET Request

```python
import requests

url = "http://sdh.local:8000/index.php"

response = requests.get(url)

print("URL:", response.url)
print(f"{response.status_code}: {response.reason}")
print(response.text)

```

get()

```
→ 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>
```

HTTP Request:

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

Basic POST Request

```python
import requests

url = "http://sdh.local:8000/post.php"

data = {
    "id": 1
}

response = requests.post(url, json=data)

print("URL:", response.url)
print(f"{response.status_code}: {response.reason}")
print(response.json())
```

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

```python
import requests

url = "http://sdh.local:8000/admin.php"

token = "e62a54f019f8b70fc0b77a50e827f86d"

headers = {
    "Authorization": f"Bearer {token}"
}

response = requests.get(url, headers=headers)

print("URL:", response.url)
print(f"{response.status_code}: {response.reason}")
print(response.text)
```

```
→ 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

```python
import requests

data = {'username': 'admin_user', 'password': 'admin_pass'}
response = requests.post('http://sdh.local:8000/login.php', json=data)
print(f"{response.status_code}: {response.reason}")
print(response.headers)
print(response.json())

```

```
→ ykh (request) $ python3 request_post_login.py
200: OK
{'Host': 'sdh.local:8000', 'Date': 'Sun, 08 Sep 2024 15:14:23 GMT', 'Connectio
n': '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

```python
import requests
from colorama import init, Fore, Style

init(autoreset=True)

urls = [
    'http://sdh.local:8000/index.php',
    'http://sdh.local:8000/admin_dashboard.php'
]

tokens = {
    'admin': 'admin_token_a035ada8cc57de7fad3e93fb2d1ccce2',
    'employee': 'employee_token_a0f70ab7b6ad7fbfed3465c4f59c0fd1',
    'invalid': 'invalid_token_99999'
}

def test_access(url, token, role):
    headers = {
        "Authorization": f"Bearer {token}"
    }

    response = requests.get(url, headers=headers)

    print(f"{Fore.CYAN}Testing {Fore.YELLOW}{role.capitalize()} Token: {Fore.WHITE}{token}")
    print(f"{Fore.CYAN}Status Code: {Fore.GREEN}{response.status_code} {Fore.WHITE}({response.reason})\n")

for url in urls:
    print(f"{Fore.MAGENTA}" + "=" * 70)
    print(f"{Fore.CYAN}Target URL: {Fore.GREEN}{url}")
    print(f"{Fore.MAGENTA}" + "=" * 70)
    for role, token in tokens.items():
        test_access(url, token, role)

```

```
→ 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

```python
import requests
from colorama import Fore, Style, init

init(autoreset=True)

for i in range(1, 4):
    response = requests.get(f'http://sdh.local:8000/profile_user.php?id={i}',
                            headers={"Authorization": "Bearer employee_token_67890"})

    print(f"{Fore.YELLOW}URL: {response.url}")

    if response.status_code == 200:
        print(f"{Fore.GREEN}ID: {i}, Status Code: {response.status_code} (OK)")
    else:
        print(f"{Fore.RED}ID: {i}, Status Code: {response.status_code}")

    print(f"{Fore.CYAN}{response.json()}")
    print(Style.RESET_ALL)

```

```
→ 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)

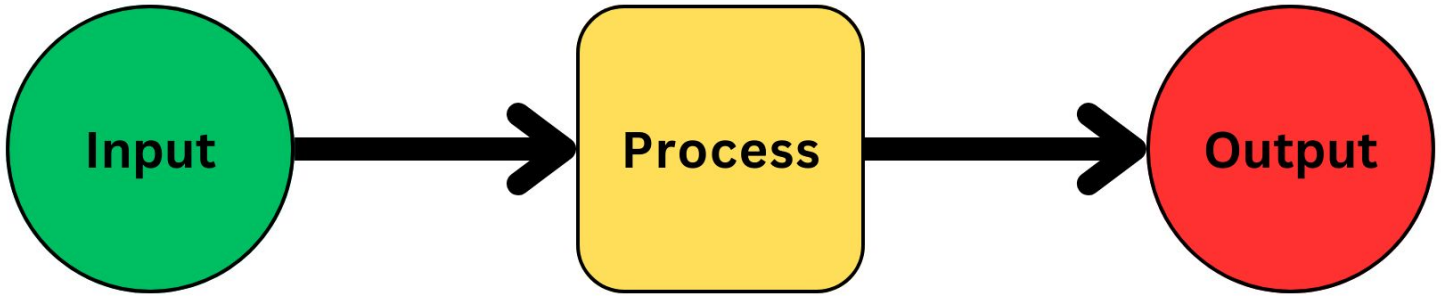
เวลา	รายละเอียด
09.00 - 10.30	Mobile Application Security
10.30 - 10.45	พักเบรก
10.45 - 12.00	Programming for CTF
12.00 - 13.00	พักรับประทาน อาหารกลางวัน
13.00 - 16.00	การแข่งขัน CTF (3 ชม.)
16.00 - 17.00	มอบรางวัลและพิธีปิด

Thank you !!

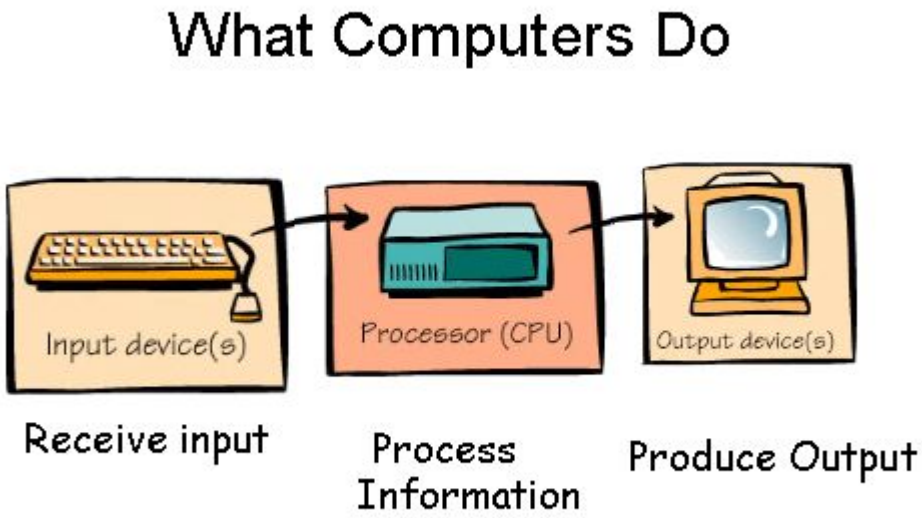
Appendix - Programming Basic

Programming คืออะไร

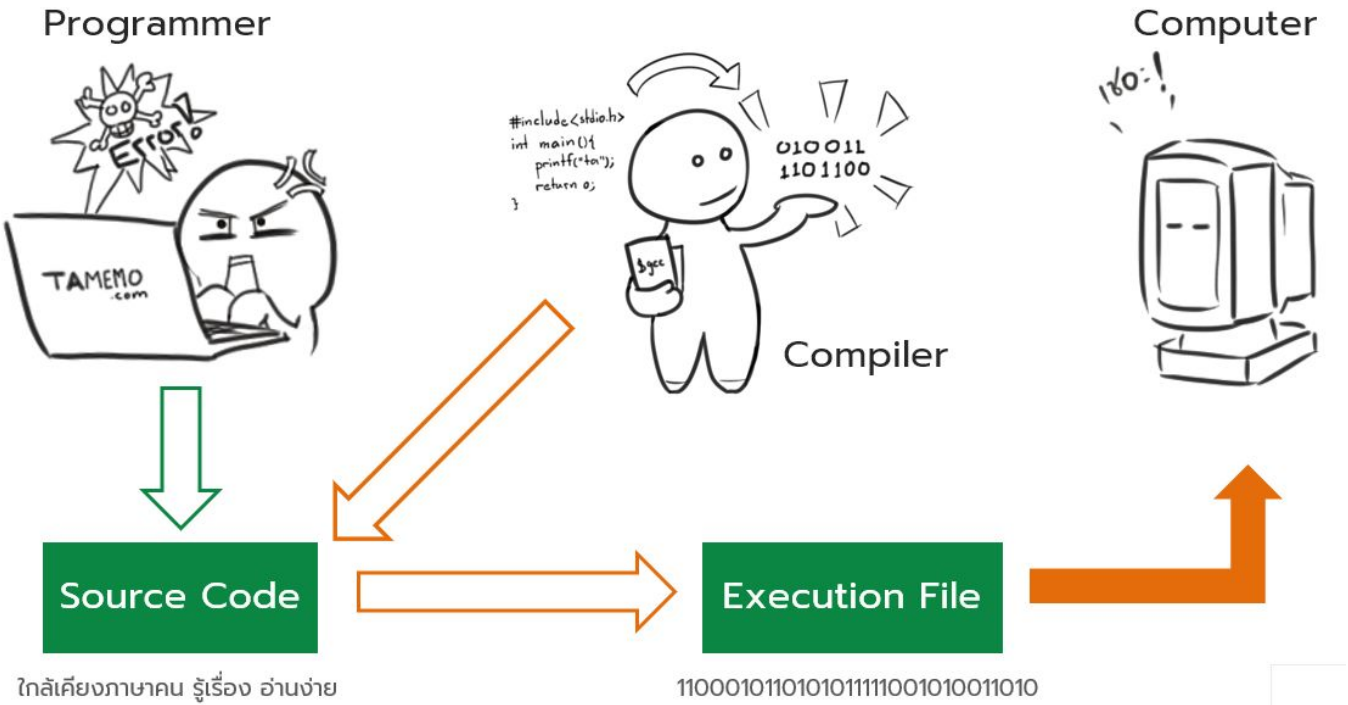
IPO model



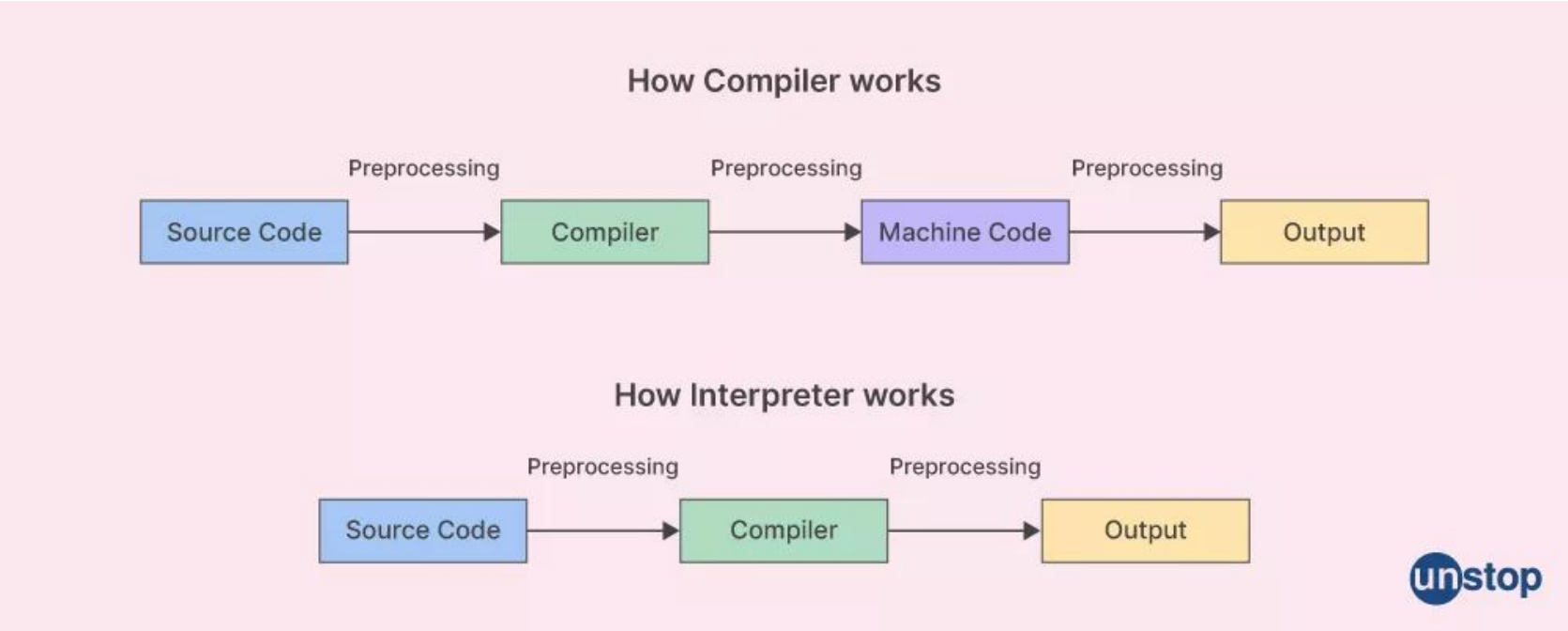
IPO model (ต่อ)


What Computers Do
Input device(s)
Processor (CPU)
Output device(s)
Receive input
Process Information
Produce Output

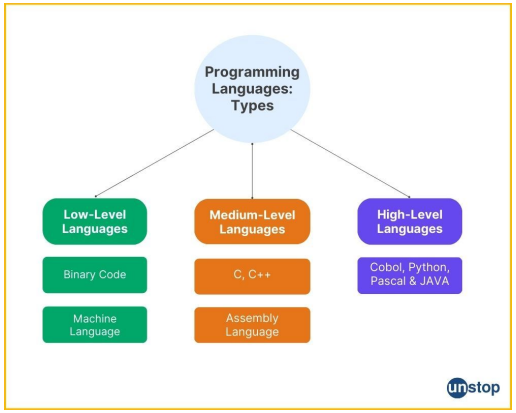
Computer เข้าใจได้อย่างไร

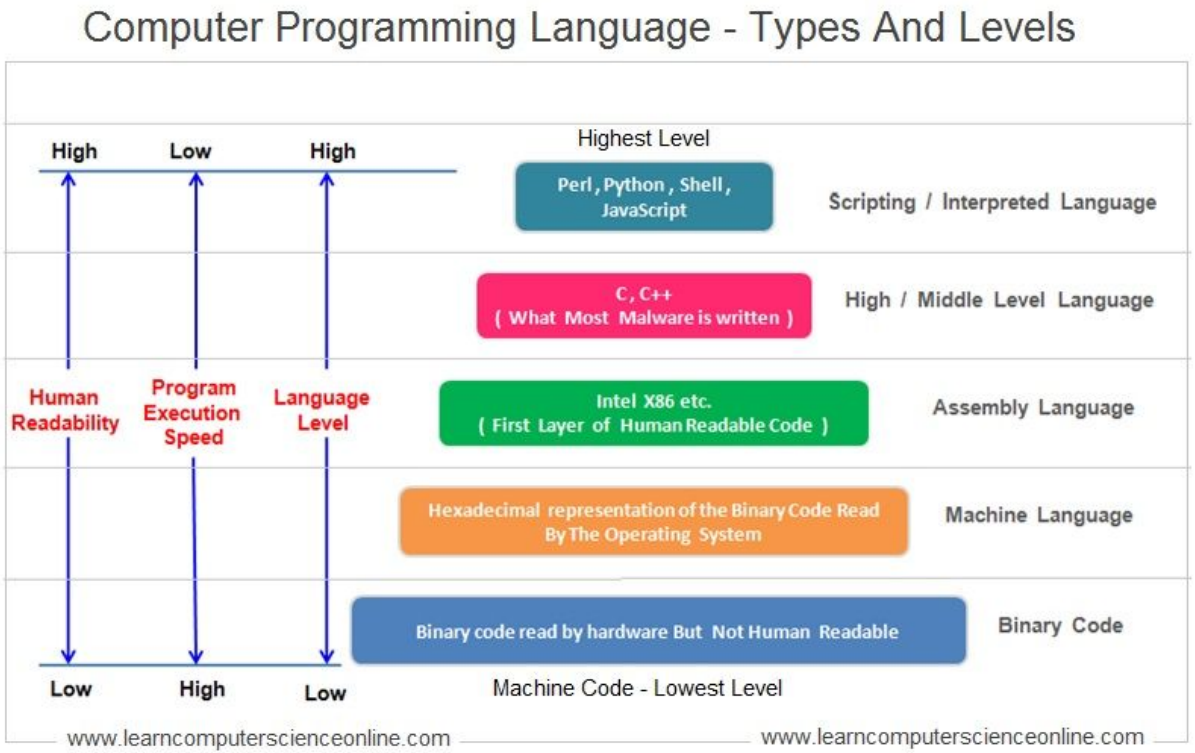


Translators Programming



Levels of Programming Language





Programming Languages

tiobe.com

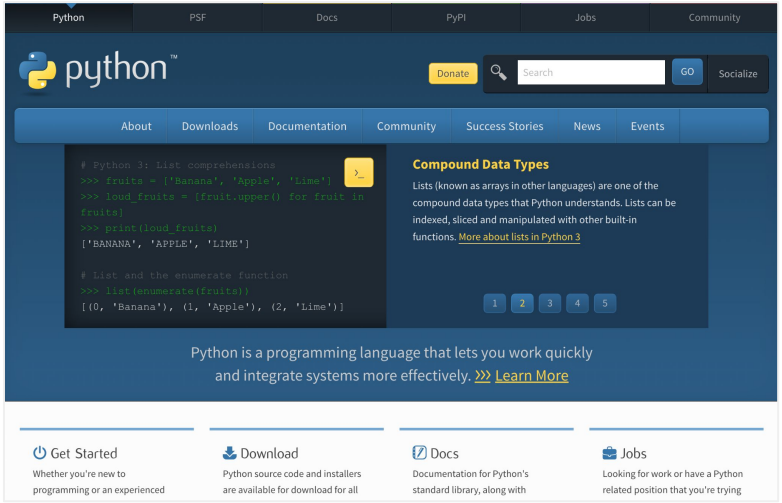
TIOBE

Schedule a demo

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

ลำดับโดย www.tiobe.com

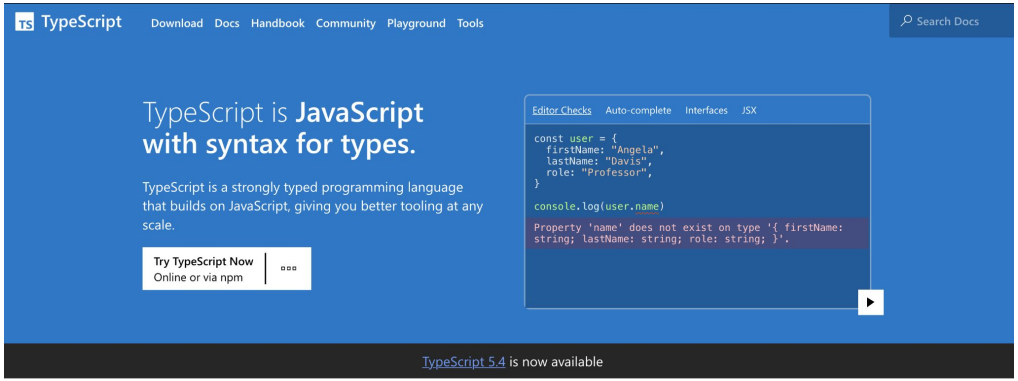
Programming Languages

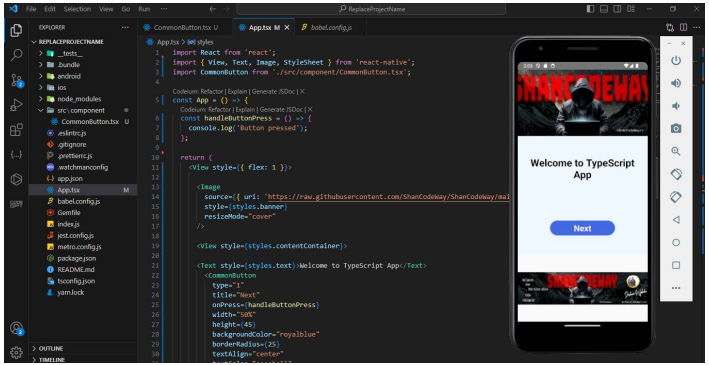


นิยมใช้กับงาน

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

Programming Languages



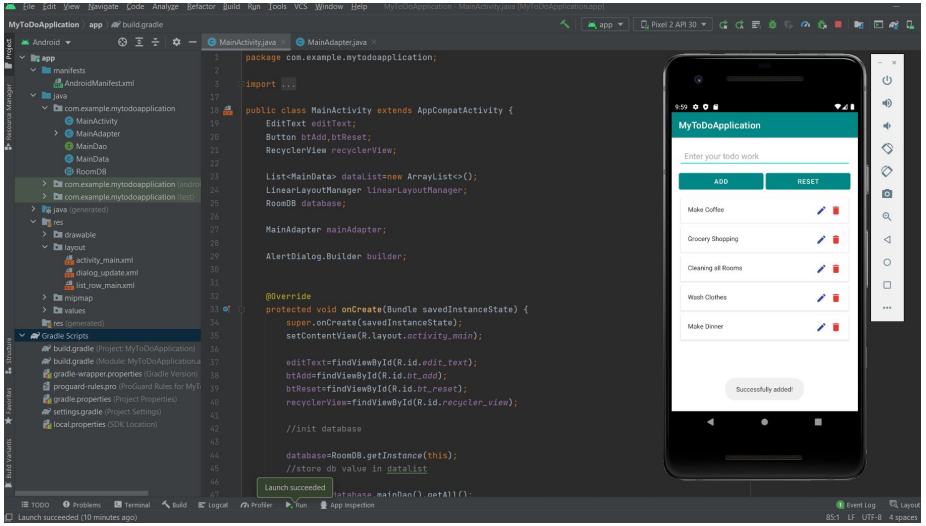


TS + React Native

นิยมใช้กับงาน

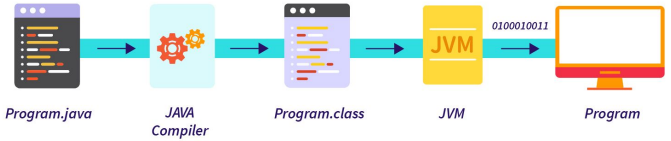
- Web Application Development (ทั้ง Front-end และ Back-end)
- Desktop Application Development (ด้วยเทคโนโลยี Electron)
- Mobile Application Development (ด้วยเทคโนโลยี React Native)
- Game Development
- Machine Learning และ Data Science
- การเขียน libraries หรือ utilities ต่าง ๆ

Programming Languages

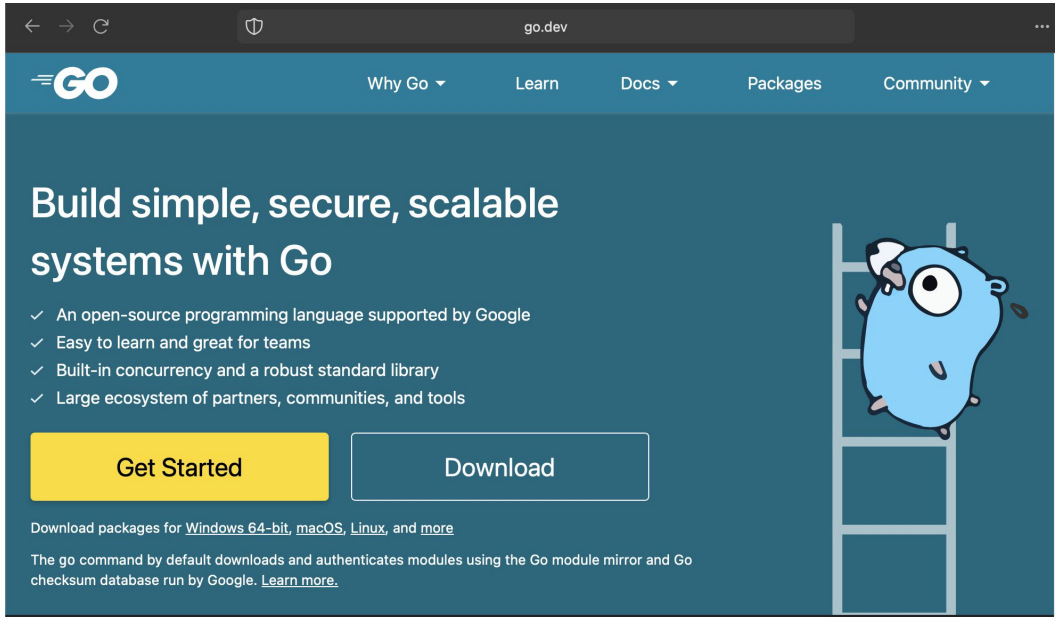


นิยมใช้กับงาน

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

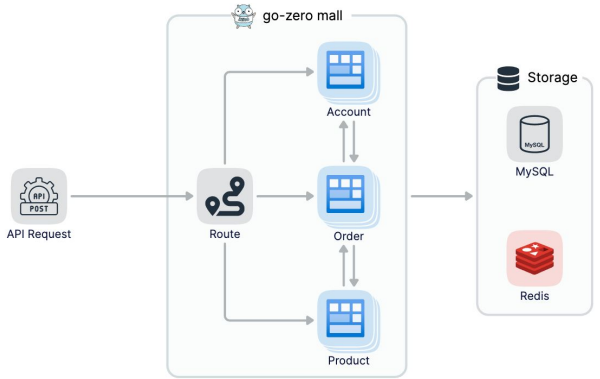


Programming Languages

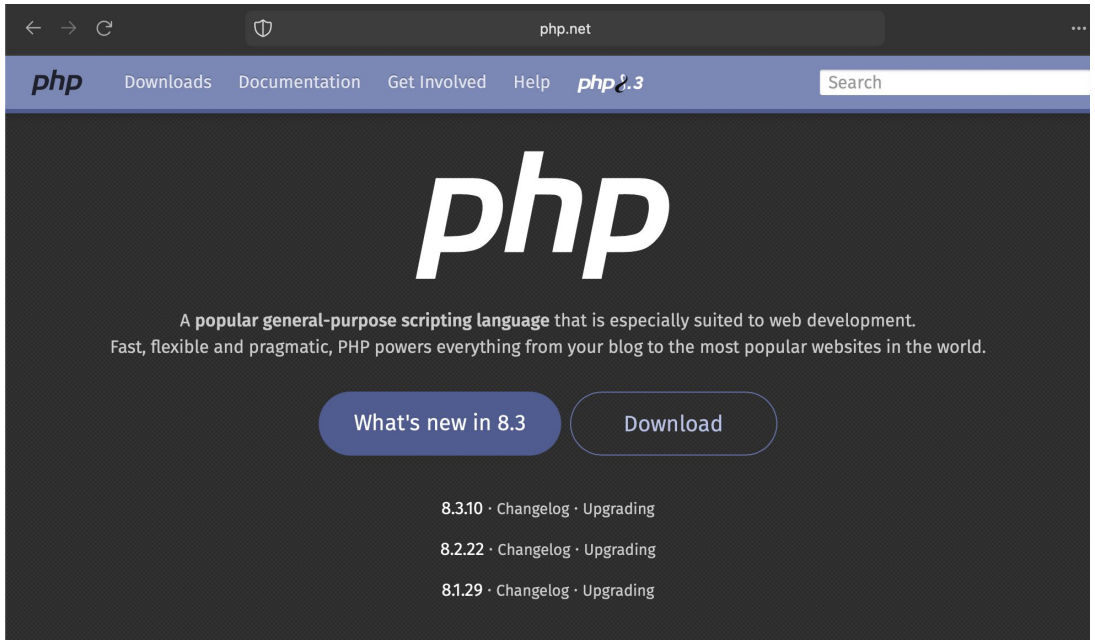


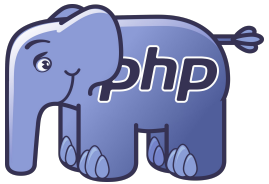
นิยมใช้กับงาน

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



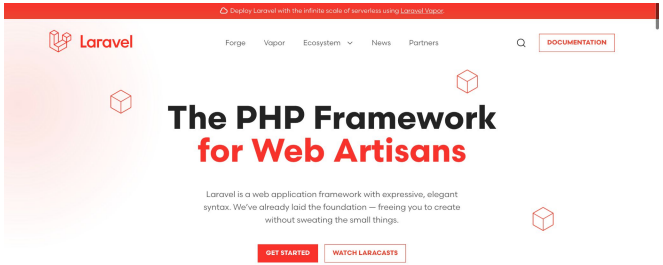
Programming Languages





นิยมใช้กับงาน

- Web Development
  ร่วมกับ Laravel Web framework



Programming Languages

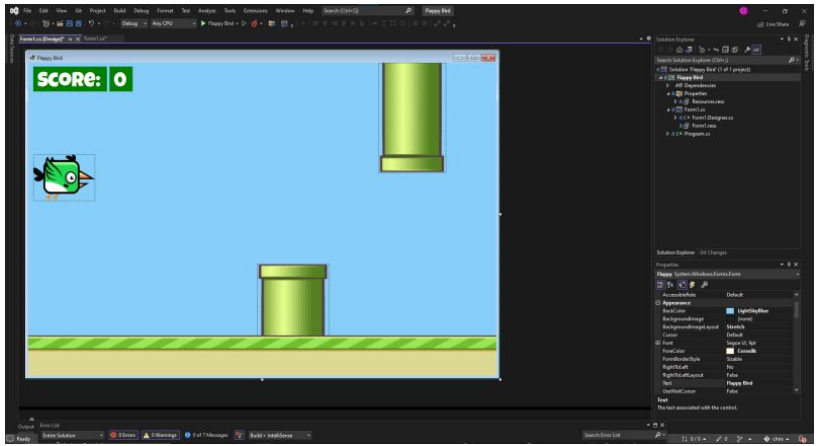
นิยมใช้กับงาน

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

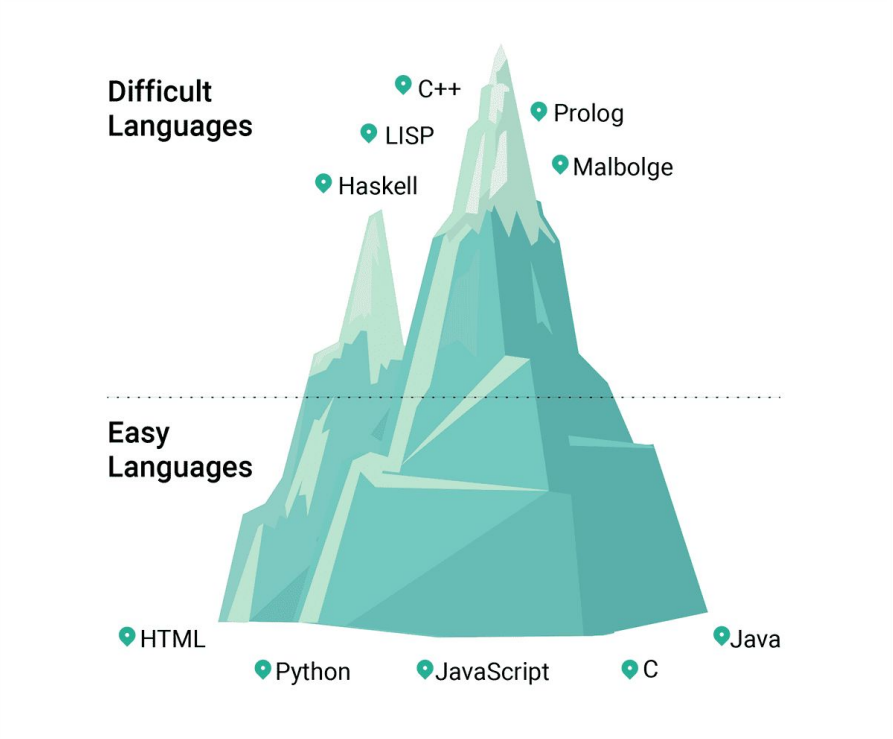
Programming Languages

นิยมใช้กับงาน

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

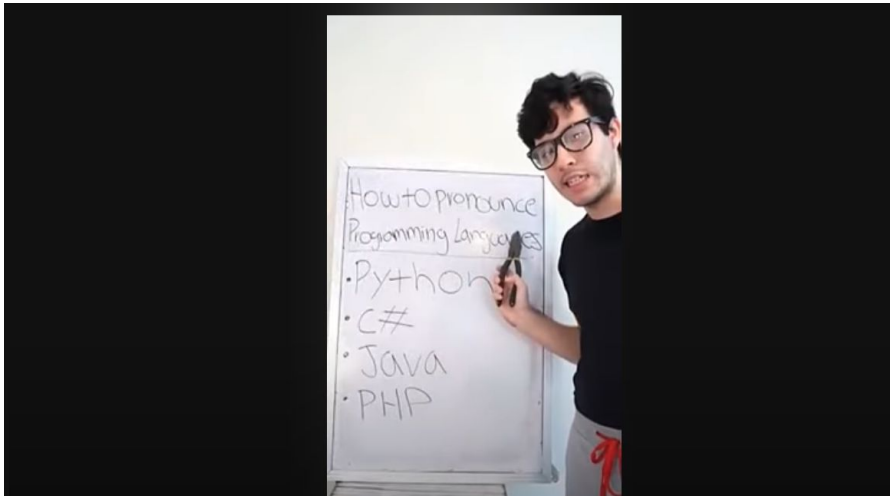


ภาษาโปรแกรมยาก-ง่าย



MEME Programming Language

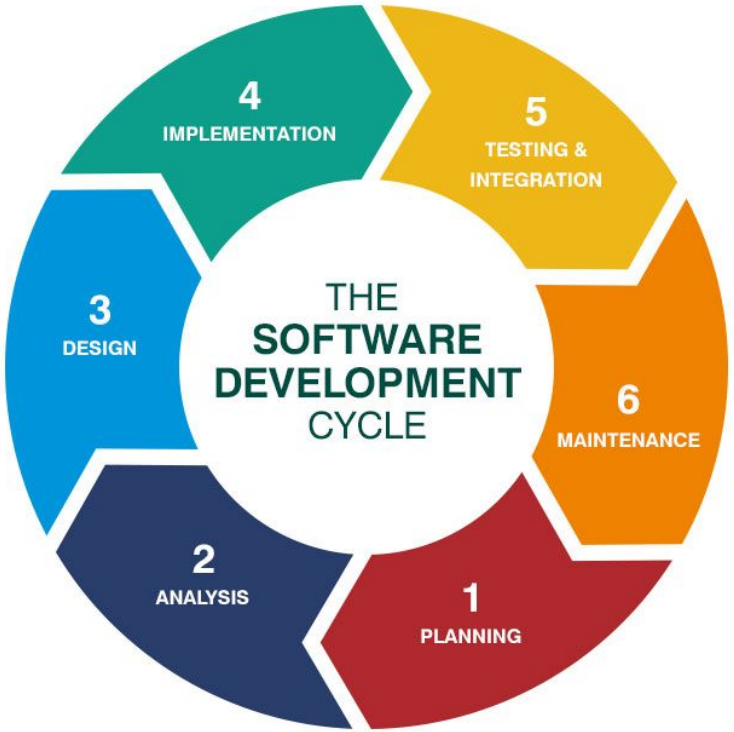
https://www.youtube.com/watch?v=3WRbWdr_p9M



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

SDLC คืออะไร

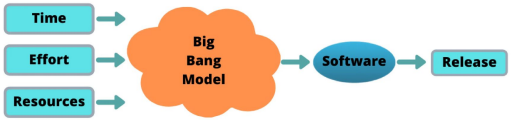
Software Development Life Cycle
“วงจรชีวิตซอฟต์แวร์ระยะพัฒนา”



Software Development Life Cycle (SDLC) Phases

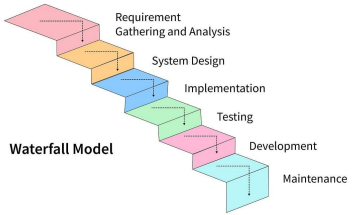
7 ประเภท SDLC Model

1. Big Bang Model

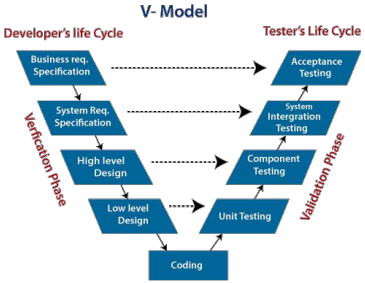


Big Bang SDLC MODEL

2. Waterfall Model

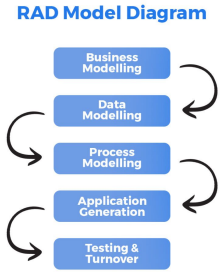


3. V-Shaped Model

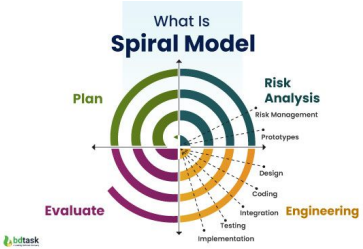


4. Iterative Model

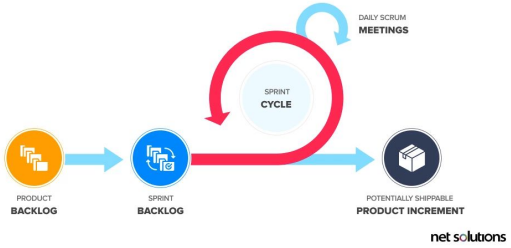
5. RAD Model



6. Spiral Model



7. Agile Model



การเขียนโค้ดด้วยภาษา Python

ติดตั้ง Python

หลักภาษาและไวยกรณ์ทั่วของภาษา Python

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

การเขียนคำสั่งเกี่ยวกับทางเลือก/เงื่อนไข (Condition)

- if..else

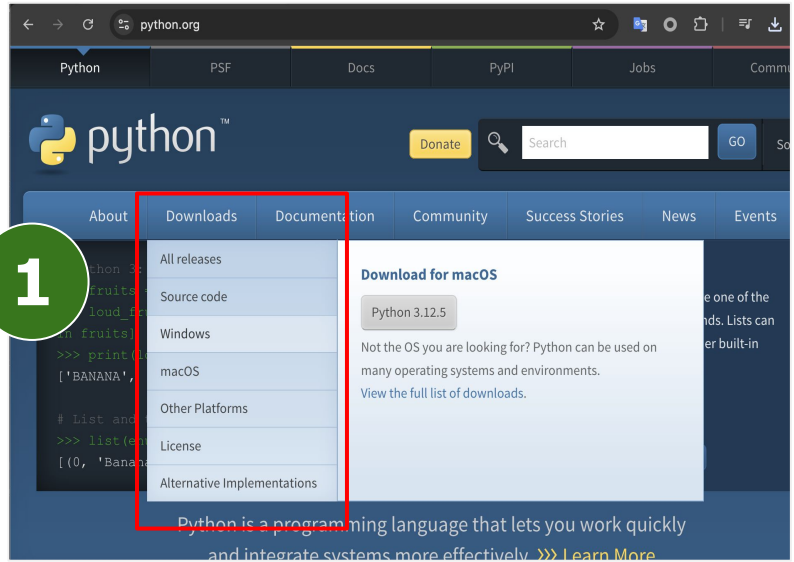
การวนลูป (Loops)

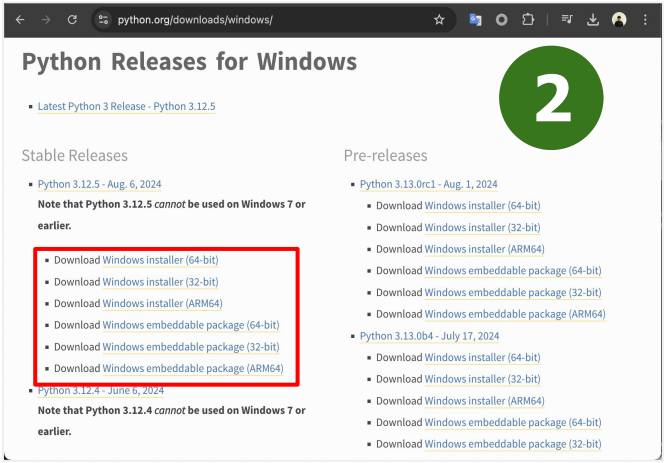
- For loops
- While loops
- Break and continue

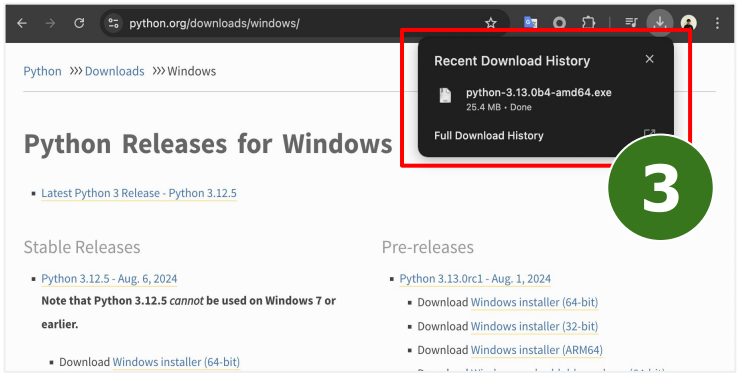
การสร้างและใช้งานฟังก์ชัน (Function)

ติดตั้ง Python

https://www.python.org

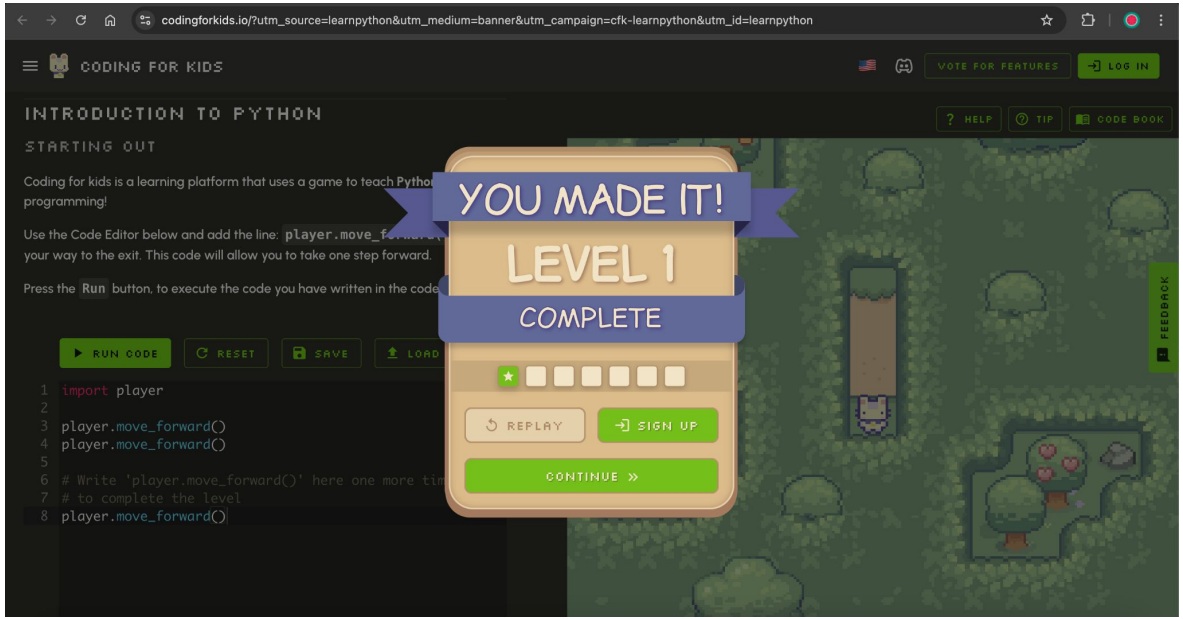






Coding for Kids

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


python
All
Images
Videos
Books
Shopping
News
Maps
More
Tools
Programming
Logo
Code
Yellow
Cute
Blue
Royal
Baby
Pet
Wikipedia
Python (programming...
National Geographi...
Burmese Python
Juni Learning
What is Python Coding?...
InfoWorld
Python? Powerful, intuitive prog...
The New Stack
What is Python? - The New Stack
Loud Bench - Loud Bench
Python Programming Language: Everything...
Wikiversity
Python - Wikiversity
Britannica
Python | Snake, Characteristics ...
en.wikipedia.org
Ball python - Wikipedia

การตั้งชื่อตัวแปร (Naming Variable) ของภาษา Python

การตั้งชื่อตัวแปร (Naming Variable) ของภาษา Python

- ชื่อควร**สื่อความหมาย** เช่น age แทนที่จะใช้ a
- ใช้ตัวอักษรเล็ก-ใหญ่ (**case-sensitive**) เช่น Age และ age เป็นคนละตัวแปร
- ห้ามใช้**คำสงวน**ของ 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
```

ตัวดำเนินการพื้นฐาน (Basic Operators)

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

ตัวอย่าง การใช้ตัวดำเนินการพื้นฐาน (Basic Operators)

ตัวอย่าง : 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
```

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

ตัวอย่าง : 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
```

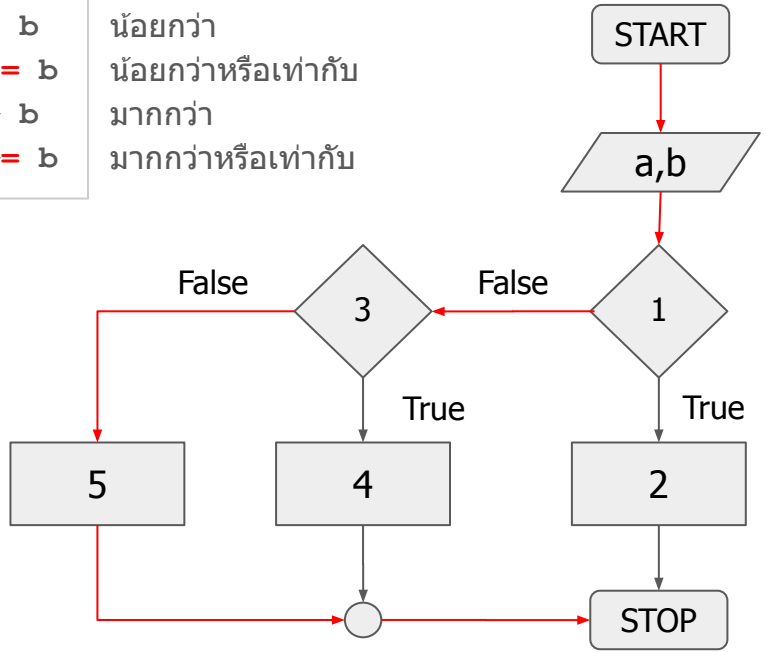
ตัวอย่าง : Logical operators (and, or, not)

```
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

`a == b`	เท่ากับ
`a != b`	ไม่เท่ากับ
`a < b`	น้อยกว่า
`a <= b`	น้อยกว่าหรือเท่ากับ
`a > b`	มากกว่า
`a >= b`	มากกว่าหรือเท่ากับ

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:                    #1
  print("b มากกว่า a")      #2
elif a == b:                 #3
  print("a และ b เท่ากัน")  #4
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
```

else

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

ผลลัพธ์

```
1
2
3
4
5
6 ไม่ได้มีค่าน้อยกว่า 6
```

break

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

ผลลัพธ์

```
1
2
3
```

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

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
```

โมดูล (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)
```

เครื่องจัดการแพ็คเก็จ (package manager)

ติดตั้ง pip ได้ที่

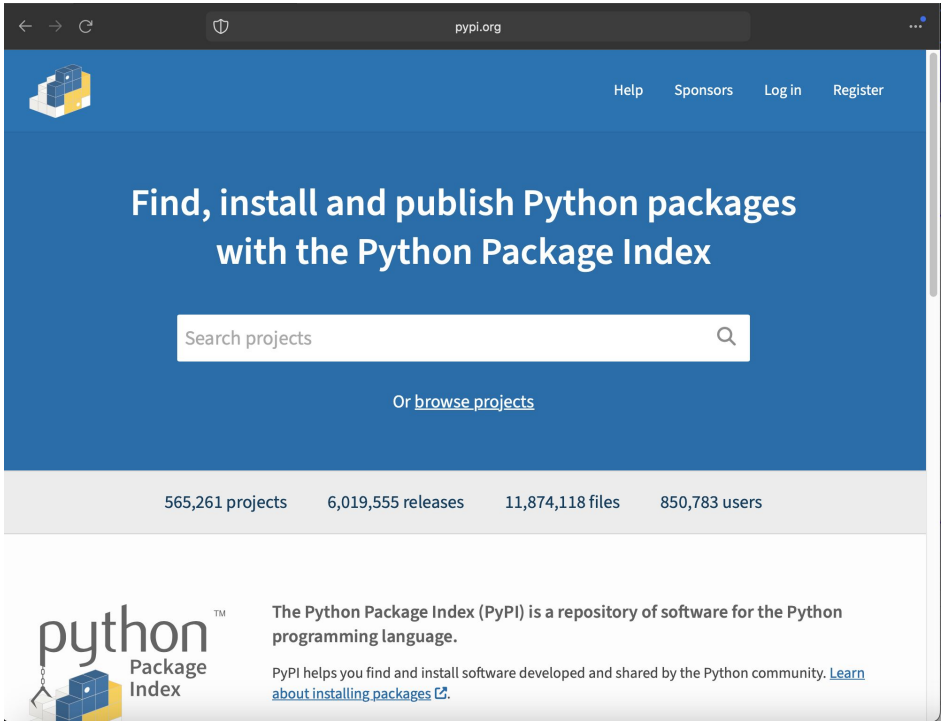
https://pypi.org/project/pip/

ติดตั้ง Package ที่ต้องการใช้

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

เรียกใช้งานได้สะดวก

```
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()
```