

---

**멘토링 참여확인서 !!!**



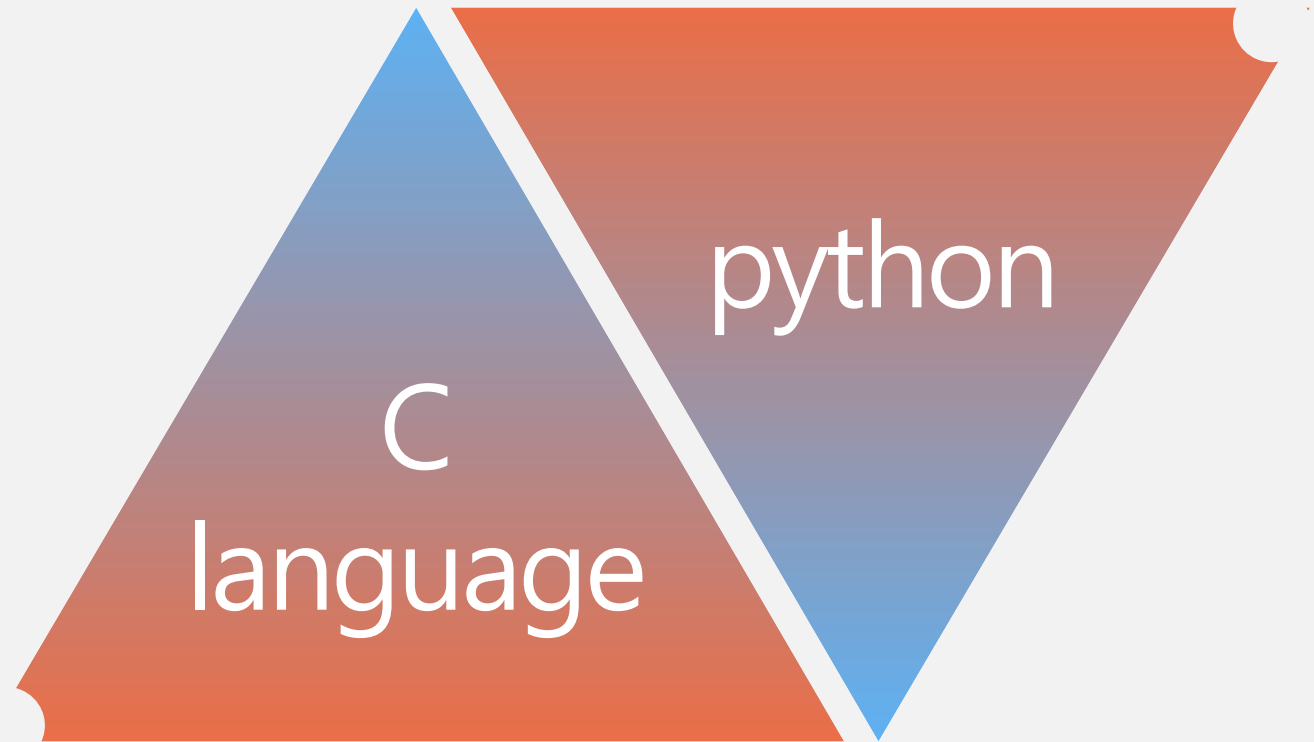
소입설  
멘토링 수업 ●  
(8주차)

# CONTENTS

---

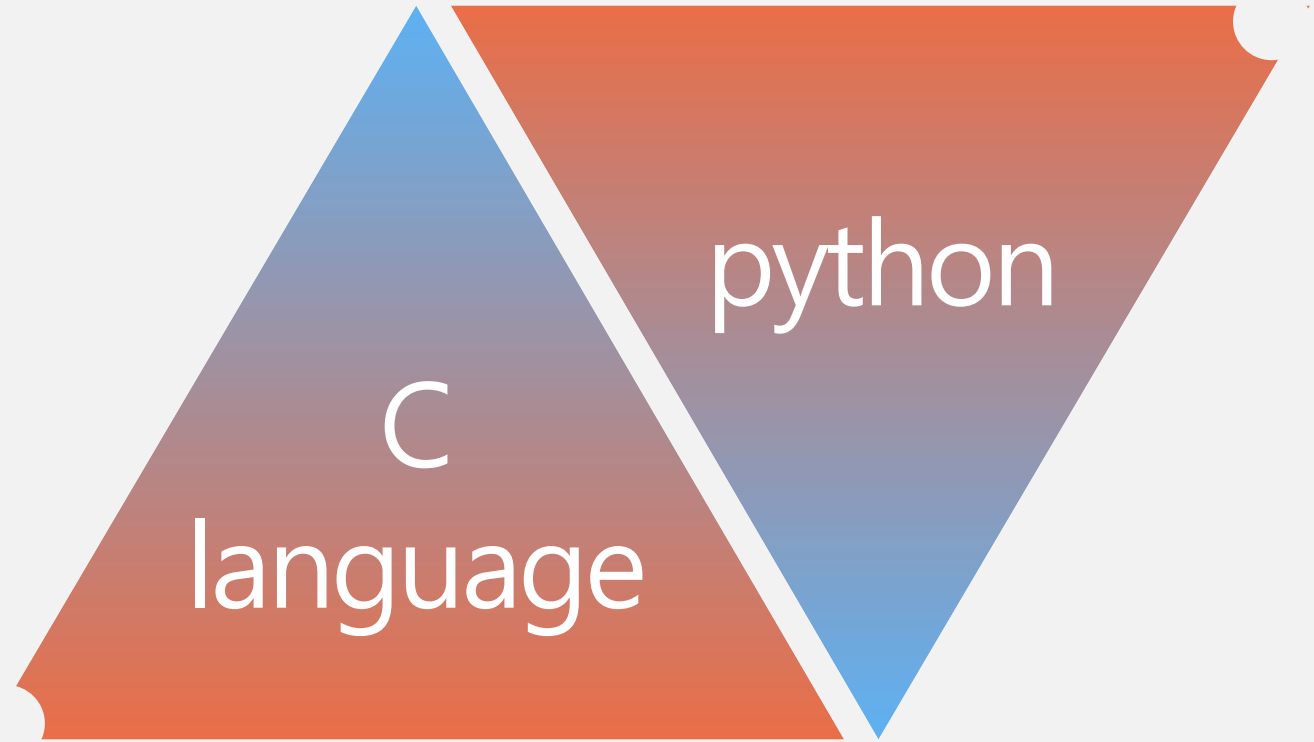
- 1 python vs C
  - 2 C Hello World!
  - 3 Pointer
  - 4 리눅스
-

# 1. python vs C



Python	C language
Intepreter	Compiler
정의, 선언 동시에	정의, 선언 따로 (동시에)
string 자료형	char[], '\0'
코드가 짧다	코드가 길다
정해져있는 기능이 많음	정해져있는 기능이 적음
주소값을 다루기 어려움	주소값을 다룰 수 있음
main → global	main → function
White space 단위	; 단위, { } 단위

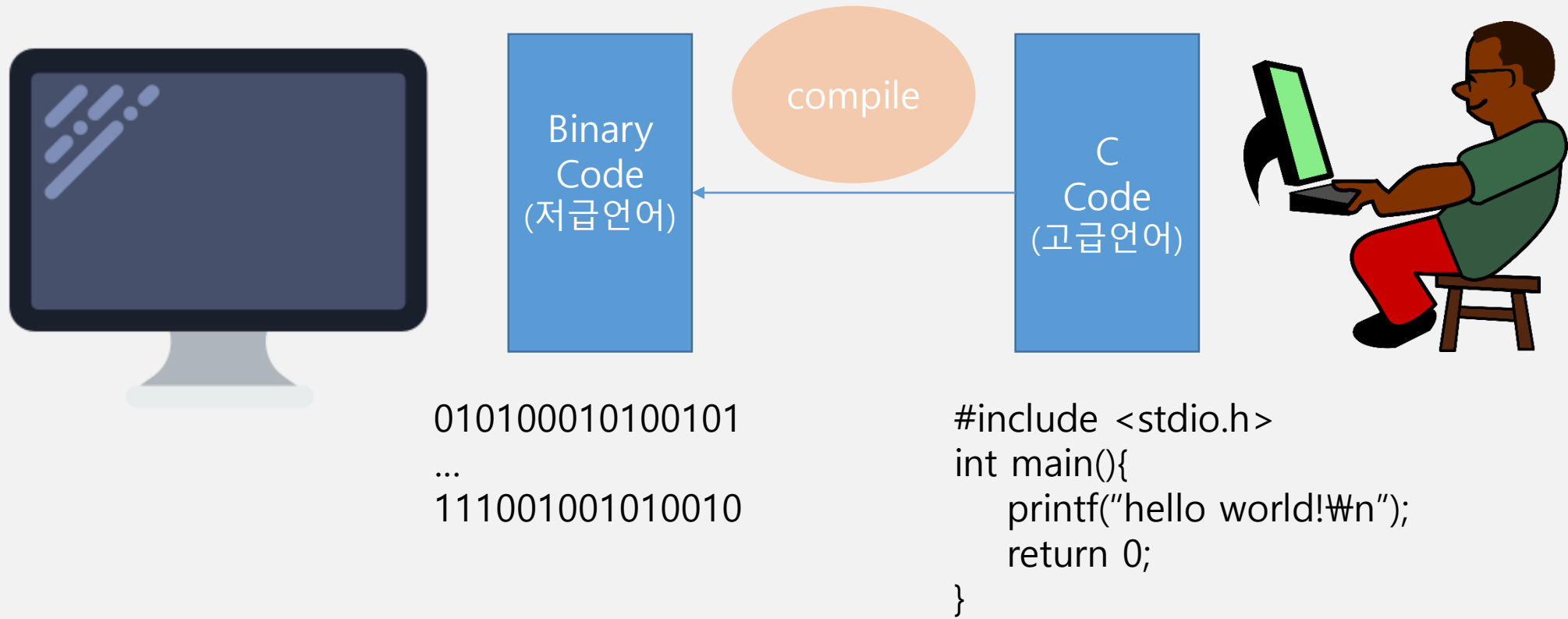
## 2. C Hello World!

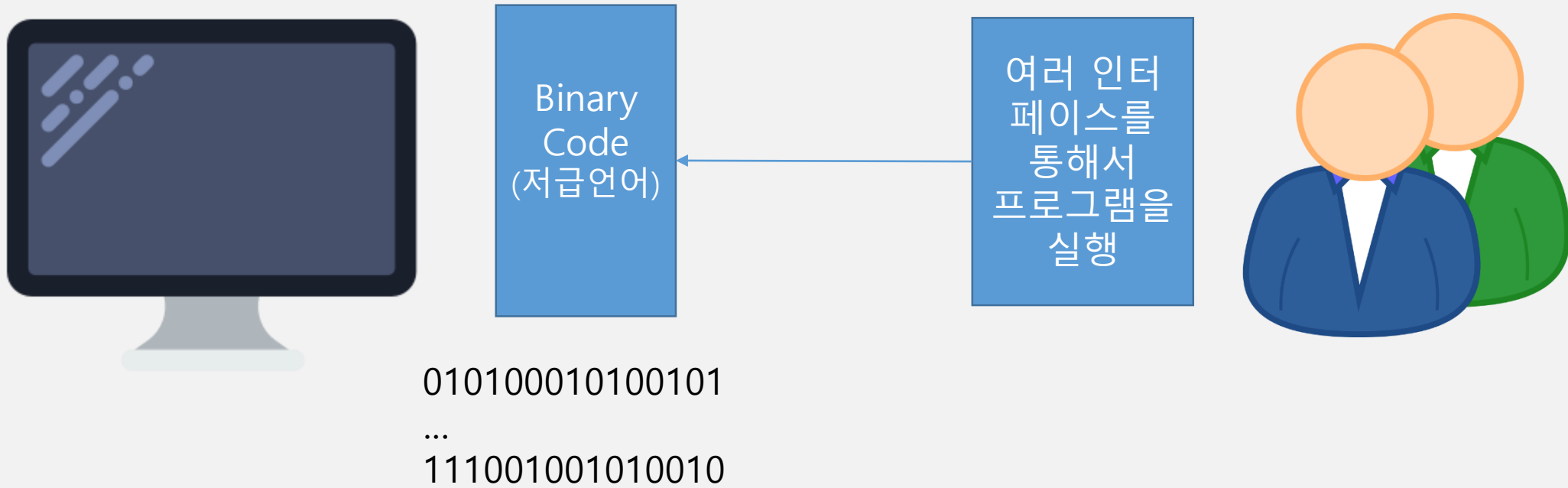


1. operators, value, expression, statement (8)
  2. data types (8)
  3. printf, scanf (8)
  4. comment (8)
  5. function (8)
  6. #include (8)
  7. while statement, if statement, for statement (9)
  8. condition and Boolean (9)
  9. #define (9)
-

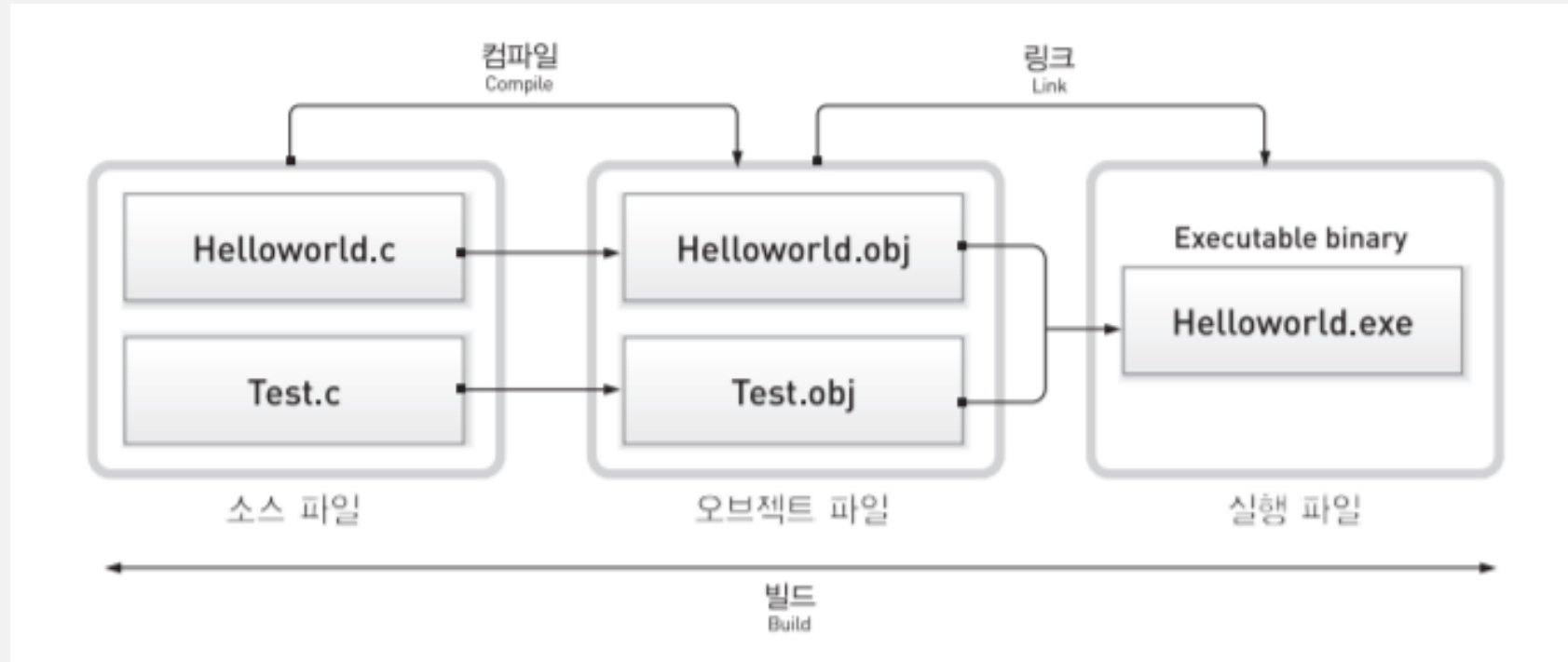
## 2 C Hello World!

---









Ctrl+F5

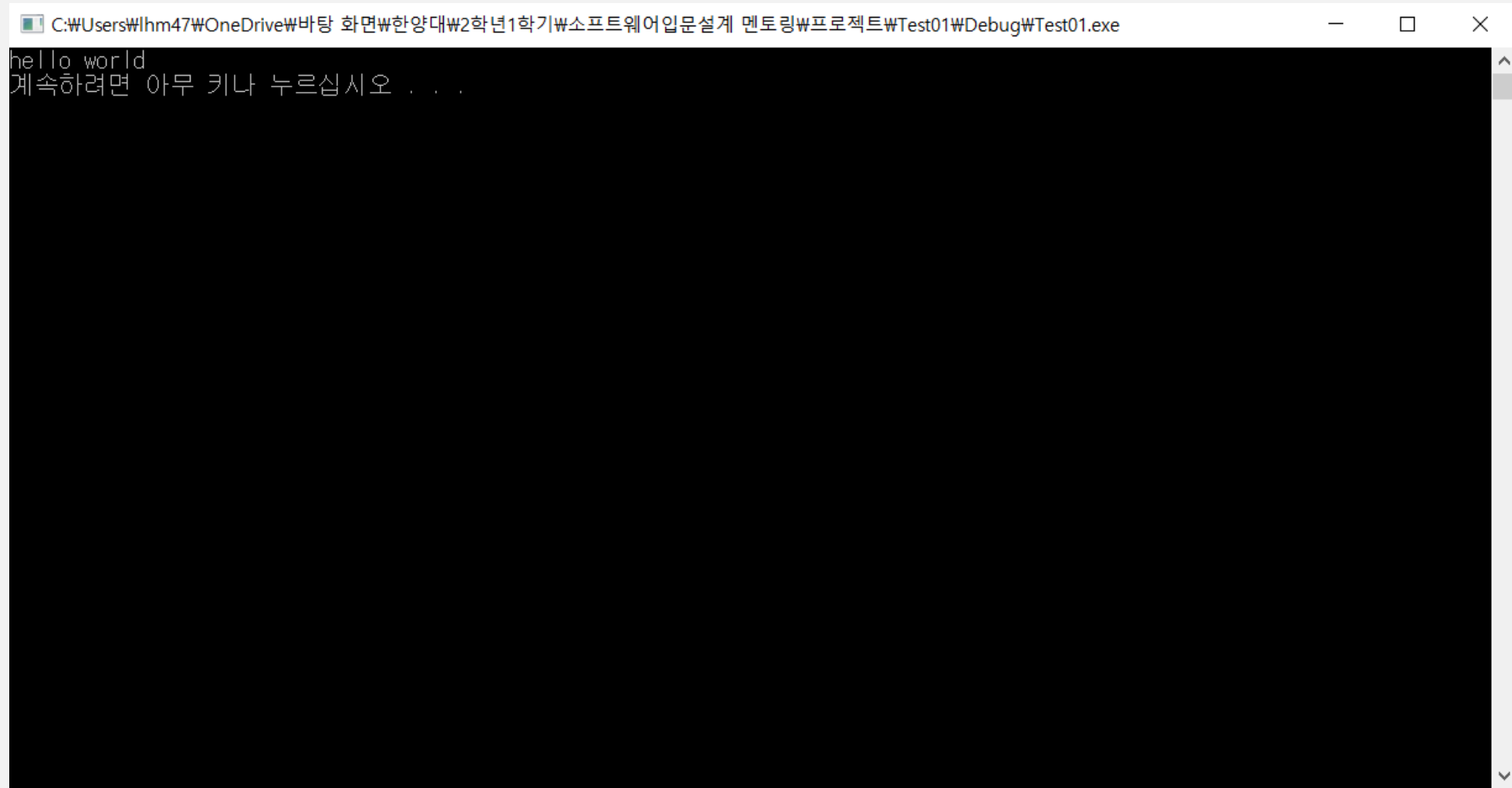
gcc

---

```
1  #include<stdio.h>
2  int main() {
3      printf("hello world\n");
4
5      system("pause");
6      return 0;
7  }
```

## 2 C Hello World!

---



A screenshot of a Windows command prompt window. The title bar at the top reads "C:\Users\lhm47\OneDrive\바탕 화면\한양대\2학년1학기\소프트웨어입문설계 멘토링\프로젝트\Test01\Debug\Test01.exe". The window contains two lines of text: "hello world" on the first line and "계속하려면 아무 키나 누르십시오 . . ." on the second line. The rest of the window is black.

```
C:\Users\lhm47\OneDrive\바탕 화면\한양대\2학년1학기\소프트웨어입문설계 멘토링\프로젝트\Test01\Debug>Test01.exe
hello world
계속하려면 아무 키나 누르십시오 . . .
```

```
/* Hello.c */
#include <stdio.h>

int main(void)
{
    printf("Hello, World! \n");
    return 0;
}
```

## 자료형(data type)

"선언할 변수의 특징을 나타내기 위한 키워드"

## 기본 자료형

기본적으로 제공이 되는 자료형

```
int val;
```

## 사용자 정의 자료형

사용자가 정의하는 자료형 : 구조체, 공용체

---

## 자료형 (data type)

### 1. 정수

- char, short, int, long

### 2. 실수

- double, float

### 3. 배열

- char[], int[], ...

---

## 자료형 (data type)

### 4. 포인터

- `char*`, `int*`, ...

### 5. 구조체

- `struct`

---

## 기본 자료형 종류와 데이터의 표현 범위

자료형(data type)		할당되는 메모리 크기	표현 가능한 데이터의 범위
정수형	char	1 바이트	-128 ~ +127
	short	2 바이트	-32768 ~ +32767
	int	4 바이트	-2147483648 ~ +2147483647
	long	4 바이트	-2147483648 ~ +2147483647
실수형	float	4 바이트	$3.4 \times 10^{-37} \sim 3.4 \times 10^{+38}$
	double	8 바이트	$1.7 \times 10^{-307} \sim 1.7 \times 10^{+308}$
	long double	8 바이트 혹은 그 이상	차이를 많이 보임



```
/* printf */
#include <stdio.h>

int main(void)
{
    int a=3, b=4;
    printf("a+b=%d\n",a+b);
    return 0;
}
```

**printf는 문자열을 출력하는 함수이다.**

**printf는 특수 문자 출력이 가능하다.**

특수 문자	의 미
\a	경고음 소리 발생
\b	백스페이스(backspace)
\f	폼 피드(form feed)
\n	개행
\r	캐리지 리턴(carriage return)
\t	수평 탭
\v	수직 탭
\\	백슬래시(\)
\'	작은 따옴표
\"	큰 따옴표

## 서식 문자의 종류와 그 의미

서식 문자	출력 형태
<b>%c</b>	단일 문자
<b>%d</b>	부호 있는 10진 정수
<b>%i</b>	부호 있는 10진 정수, %d와 같음
<b>%f</b>	부호 있는 10진 실수
<b>%s</b>	문자열
<b>%o</b>	부호 없는 8진 정수
<b>%u</b>	부호 없는 10진 정수
<b>%x, %X</b>	부호 없는 16진 정수, 소,대문자 사용
<b>%p, %P</b>	주소값(16진 정수), 소,대문자 사용
<b>%e</b>	e 표기법에 의한 실수
<b>%E</b>	E 표기법에 의한 실수
<b>%g</b>	값에 따라서 %f, %e 둘 중 하나를 선택
<b>%G</b>	값에 따라서 %f, %E 둘 중 하나를 선택
<b>%%</b>	% 기호 출력

```
/* scanf */  
#include <stdio.h>  
  
int main(void)  
{  
    int a, b;  
    scanf("%d %d",&a,&b);  
    printf("a+b=%d\n",a+b);  
    return 0;  
}
```

## 함수의 형태



## 함수의 형태 (정의)

```
[data type of return value] [name] ([argument,parameter]) {  
    ...  
    ...  
    return [return value];  
}
```

## 함수(정의, 선언, 호출)

### 정의 (함수를 직접 만들 때)

```
int FirstFunction(int num)
{
    printf("hello world.");
    return num+3;
}
```

## 함수(정의, 선언, 호출)

선언 (함수가 메인 함수 아래에 정의되어 있을 때)

```
int FirstFunction(int num);  
int main(void) {  
    ...  
    ...  
}  
int FirstFunction(int num){  
    ...  
}
```



## 함수(정의, 선언, 호출)

### 호출 (함수를 실제 사용할 때)

```
int FirstFunction(int num);  
int main(void) {  
    int k=3;  
    k=FirstFunction(30);  
}  
int FirstFunction(int num){  
    ...  
}
```

```
// function

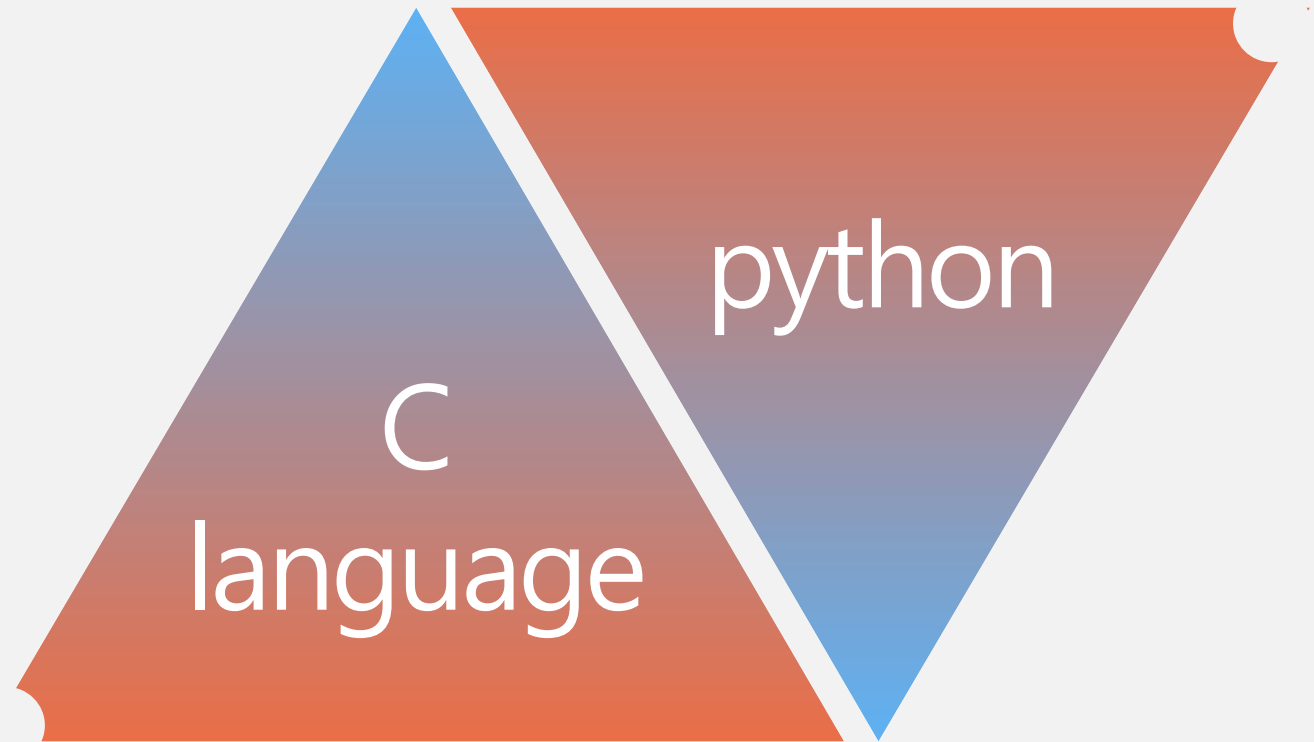
#include <stdio.h>

int function(int);

int main(void) {
    int num,result;
    scanf("%d",&num);
    result=function(num);
    printf("result: %d",result);
    return 0;
}
```

```
int function (int a){
    int temp;
    printf("function call\n");
    scanf("%d",&temp);
    return a+temp;
}
```

### 3. Pointer



Memory!!!

---

1 Byte = 8 bits  
00000000  
~  
11111111

0xF014	CC
0xF015	CC
0xF016	CC
0xF017	CC
0xF018	CC
0xF019	CC
0xF020	CC
0xF021	CC
0xF022	CC
0xF023	CC
0xF024	CC
0xF025	CC
0xF026	CC
0xF027	CC
0xF028	CC

---


int a;

int: 4Bytes(32bits)

0xF014	00
0xF015	00
0xF016	00
0xF017	00
0xF018	CC
0xF019	CC
0xF020	CC
0xF021	CC
0xF022	CC
0xF023	CC
0xF024	CC
0xF025	CC
0xF026	CC
0xF027	CC
0xF028	CC

a


```
int a;  
  
a=8;
```



A blue box containing the letter 'a' has a blue arrow pointing to the first row of the memory table (0xF014).

<b>0xF014</b>	08
<b>0xF015</b>	00
<b>0xF016</b>	00
<b>0xF017</b>	00
<b>0xF018</b>	CC
<b>0xF019</b>	CC
<b>0xF020</b>	CC
<b>0xF021</b>	CC
<b>0xF022</b>	CC
<b>0xF023</b>	CC
<b>0xF024</b>	CC
<b>0xF025</b>	CC
<b>0xF026</b>	CC
<b>0xF027</b>	CC
<b>0xF028</b>	CC

```
int a;  
  
a=8;  
  
a=0x12345678;
```



A blue box labeled 'a' has a blue arrow pointing to the first row of the memory table (0xF014).

<b>0xF014</b>	78
<b>0xF015</b>	56
<b>0xF016</b>	34
<b>0xF017</b>	12
<b>0xF018</b>	CC
<b>0xF019</b>	CC
<b>0xF020</b>	CC
<b>0xF021</b>	CC
<b>0xF022</b>	CC
<b>0xF023</b>	CC
<b>0xF024</b>	CC
<b>0xF025</b>	CC
<b>0xF026</b>	CC
<b>0xF027</b>	CC
<b>0xF028</b>	CC



```
int a;  
a=0x12345678;  
int* pa;
```

0xF014	78	← a
0xF015	56	
0xF016	34	
0xF017	12	
0xF018	cc	← pa
0xF019	cc	
0xF020	cc	
0xF021	00	
0xF022	00	
0xF023	00	
0xF024	00	
0xF025	00	
0xF026	00	
0xF027	00	
0xF028	00	

```
int a;  
a=0x12345678;  
int* pa;  
pa=&a;
```

0xF014	78	← a
0xF015	56	
0xF016	34	
0xF017	12	
0xF018	cc	← pa
0xF019	cc	
0xF020	cc	
0xF021	14	
0xF022	F0	
0xF023	00	
0xF024	00	
0xF025	00	
0xF026	00	
0xF027	00	
0xF028	00	

1 Byte = 8 bits  
00000000  
~  
11111111

0xF014	CC
0xF015	CC
0xF016	CC
0xF017	CC
0xF018	CC
0xF019	CC
0xF020	CC
0xF021	CC
0xF022	CC
0xF023	CC
0xF024	CC
0xF025	CC
0xF026	CC
0xF027	CC
0xF028	CC

---

```
char ch=0x30;
```

0xF014	CC
0xF015	CC
0xF016	CC
0xF017	CC
0xF018	CC
0xF019	CC
0xF020	CC
0xF021	CC
0xF022	CC
0xF023	CC
0xF024	CC
0xF025	CC
0xF026	30
0xF027	CC
0xF028	CC

ch



```
char ch=0x30;  
char* pch=&ch;
```

0xF014	cc	
0xF015	26	← pch
0xF016	F0	
0xF017	00	
0xF018	00	
0xF019	00	
0xF020	00	
0xF021	00	
0xF022	00	
0xF023	cc	
0xF024	cc	
0xF025	cc	
0xF026	30	← ch
0xF027	cc	
0xF028	cc	

1 Byte = 8 bits  
00000000  
~  
11111111

0xF014	CC
0xF015	CC
0xF016	CC
0xF017	CC
0xF018	CC
0xF019	CC
0xF020	CC
0xF021	CC
0xF022	CC
0xF023	CC
0xF024	CC
0xF025	CC
0xF026	CC
0xF027	CC
0xF028	CC

---

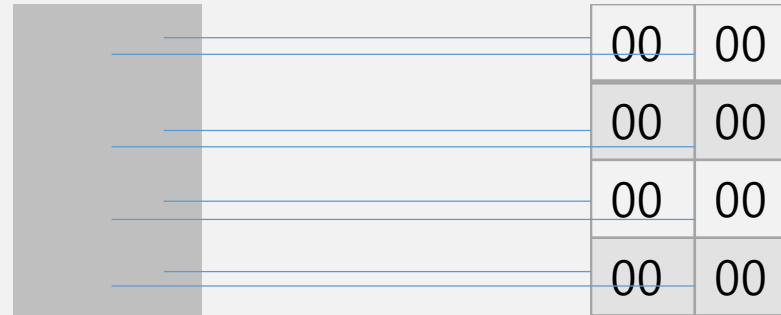
Double \*da,  
\*db;

0xF014	00	
0xF015	00	← da
0xF016	00	
0xF017	00	
0xF018	00	
0xF019	00	
0xF020	00	
0xF021	00	
0xF022	00	
0xF023	cc	
0xF024	cc	
0xF025	cc	
0xF026	00	← db
0xF027	00	
0xF028	00	↓

why?

CPU

RAM



$2^3$  Bytes

연결통로  $2^3$  개 필요

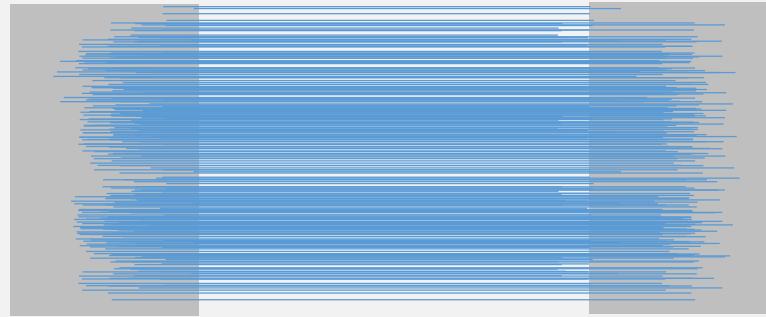
---



why?

CPU

RAM

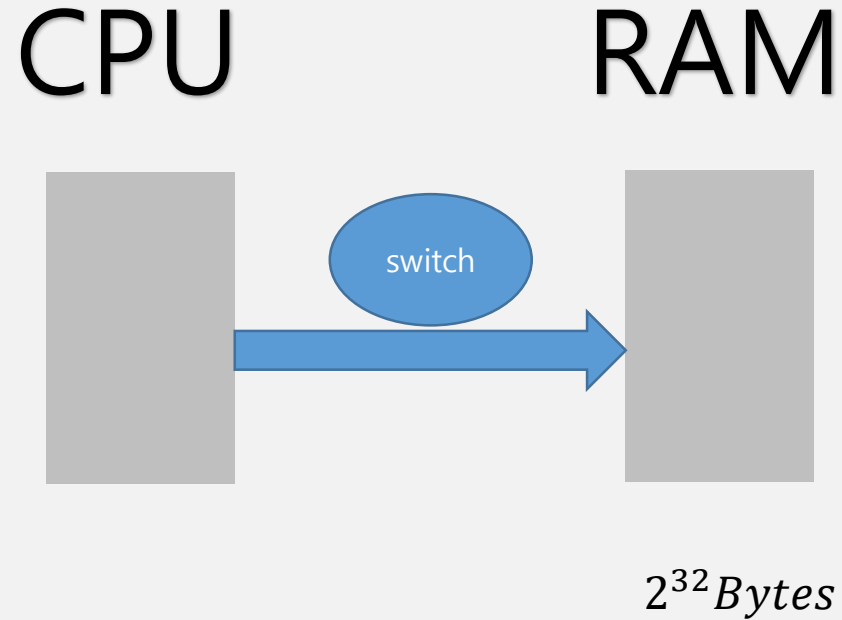


$2^{32}$  Bytes

연결통로  $2^{32}$  개 필요

---

why?



➔ 연결통로 32 개 필요

---

# CPU와 메모리의 원활한 소통

---

# 사용 예)

```
#include <stdio.h>

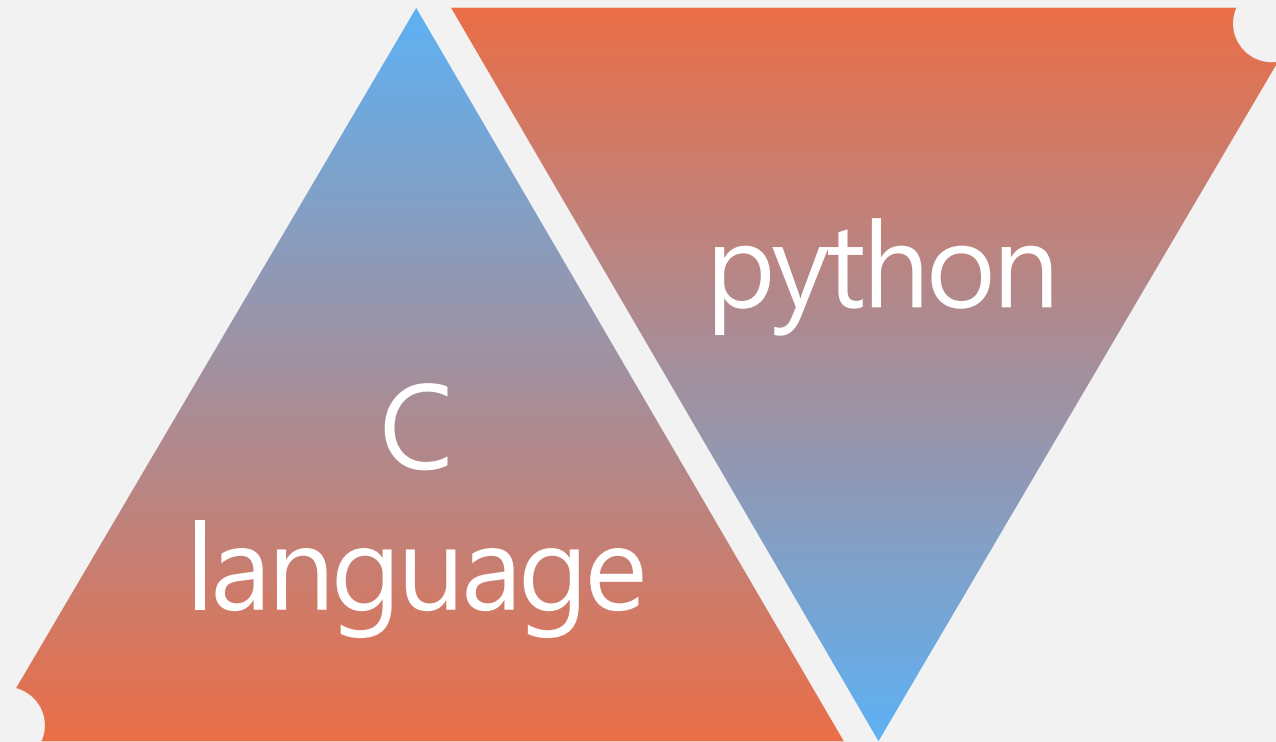
int function(int*,int*);

int main(void) {
    int num1=3, num2=5;
    function(&num1,&num2);
    printf("%d, %d\n",num1,num2);

    return 0;
}

void function(int* pa, int* pb){
    int temp=*pa;
    *pa=*pb;
    *pb=temp;
}
```

## 4. 리눅스





소입설1-1.



소입설1-2.

---

복습

---