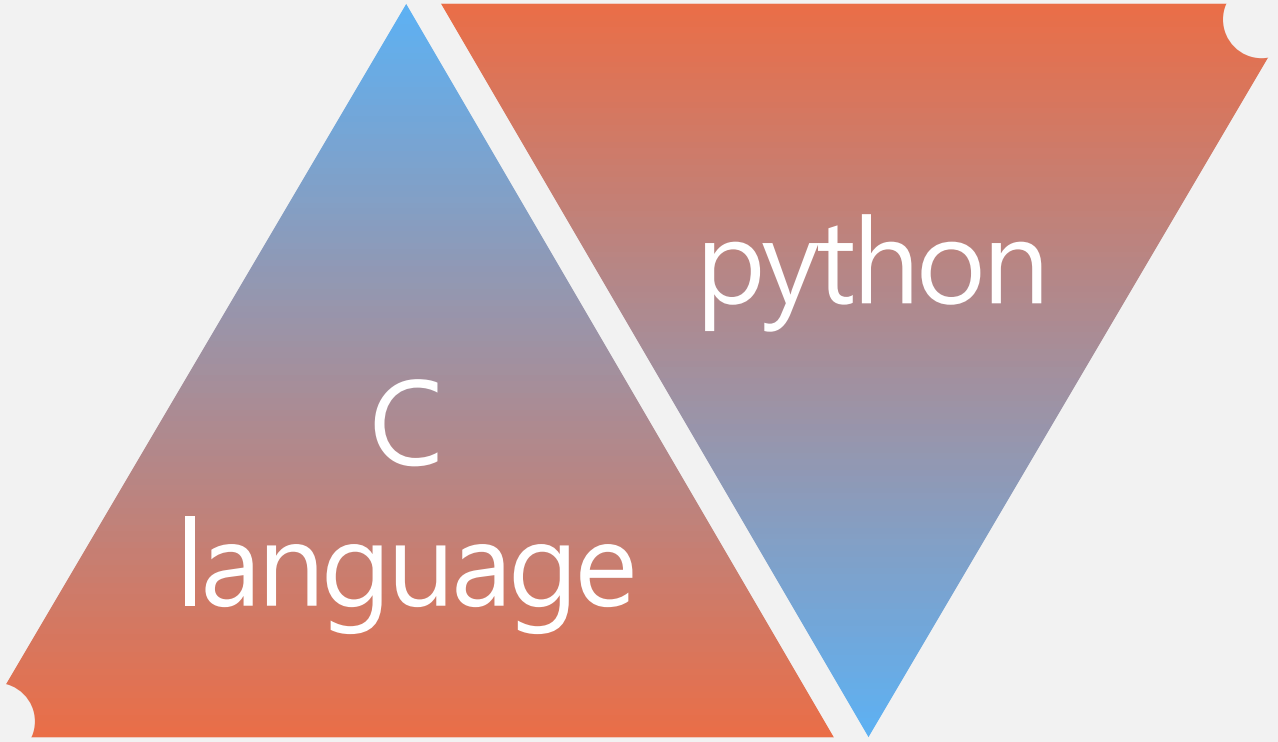
소입설
멘토링 수업
(5주차)

-이현민 멘토

CONTENTS

1. 지난주 수업 복습



이주호

5. 실행p

5L 수식p

임을규

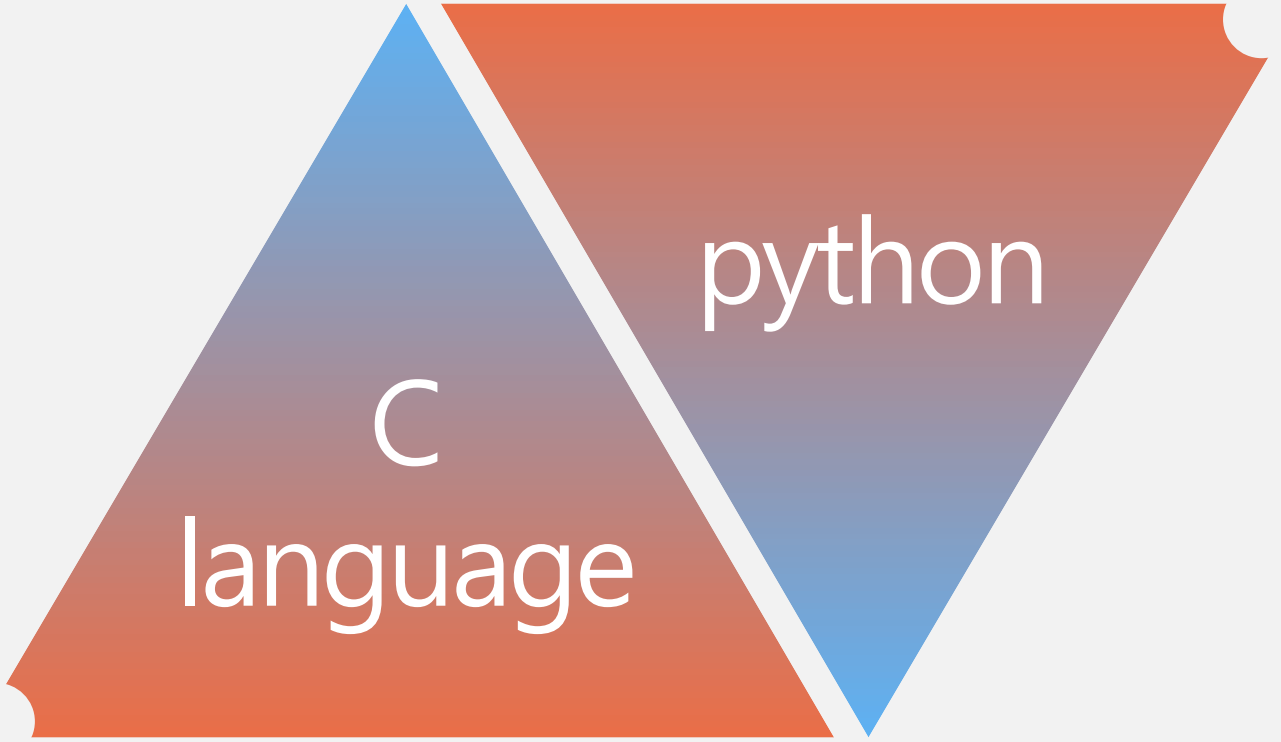
모듈

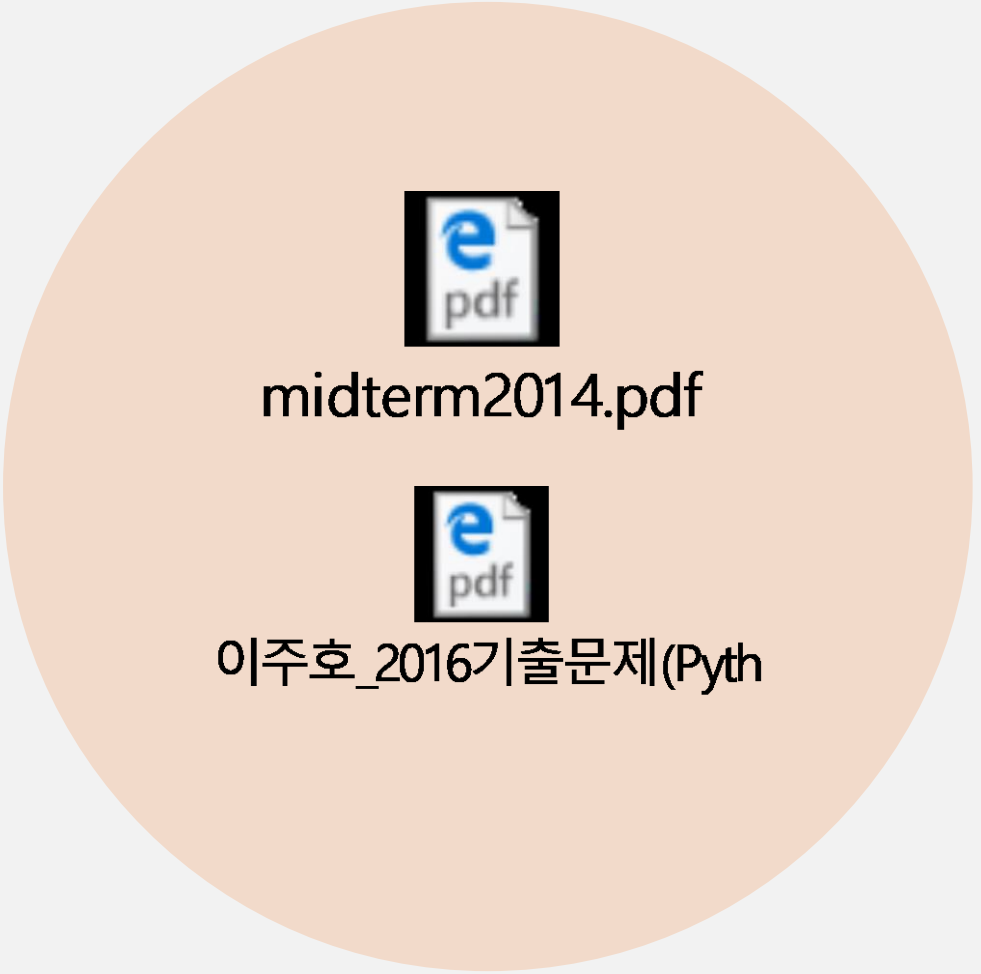
05. Hangman.pdf

소업설5-1.

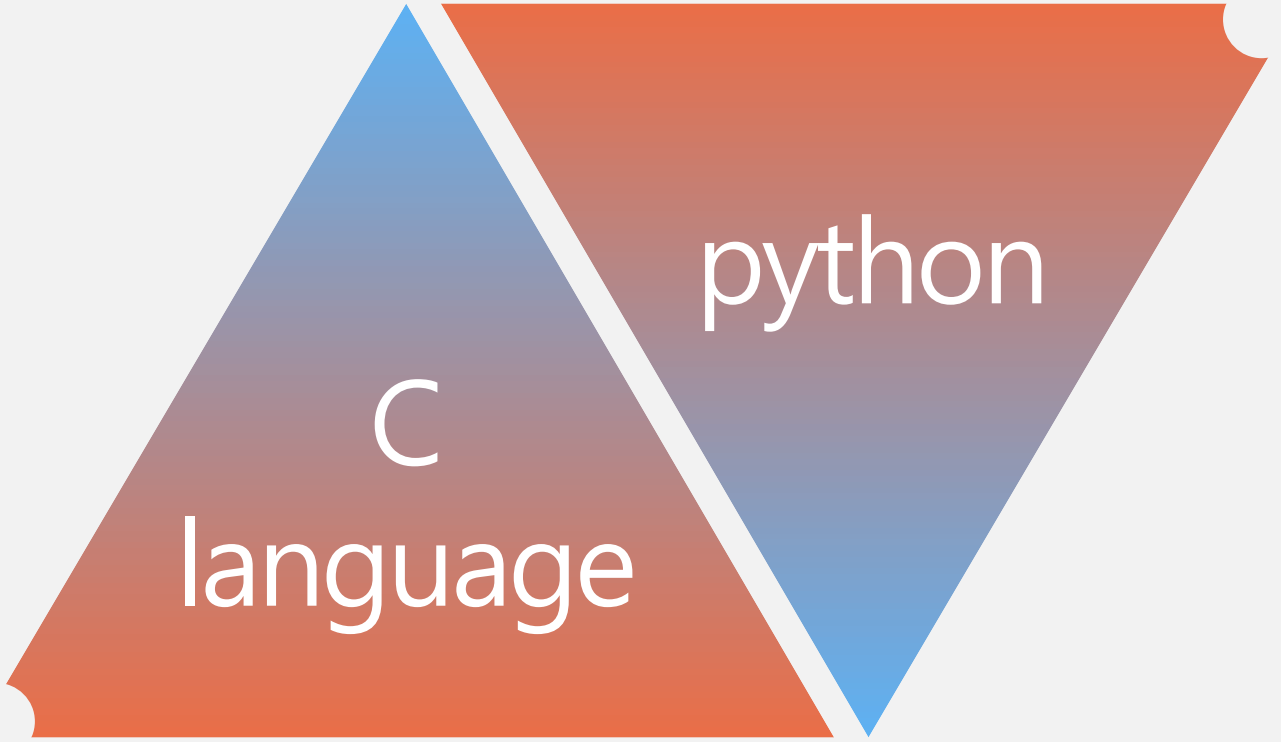
소업설5-2

2. Quiz




midterm2014.pdf
이주호_2016기출문제(Pyth

3. 변수와 자료형



변수란 무엇인가?

데이터를 저장할 수 있는 메모리 공간에 붙여진 이름

다양한 형태(자료형)의 변수

정수형 : char, int, long

실수형 : float, double

자료형(data type)

"선언할 변수의 특징을 나타내기 위한 키워드"

```
int val;
```

기본 자료형

기본적으로 제공이 되는 자료형

사용자 정의 자료형

사용자가 정의하는 자료형 : 구조체, 공용체

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	$3.4*10^{-37}$ ~ $3.4*10^{+38}$
	double	8 바이트	$1.7*10^{-307}$ ~ $1.7*10^{+308}$
	long double	8 바이트 혹은 그 이상	차이를 많이 보임

다양한 자료형이 제공되는 이유

데이터의 표현 방식이 다르기 때문

- 정수형 데이터를 표현하는 방식
- 실수형 데이터를 표현하는 방식

메모리 공간을 적절히 사용하기 위해서

- 데이터의 표현 범위를 고려해서 자료형 선택
- 작은 메모리 공간에 큰 데이터를 저장하는 경우 데이터 손실이 발생할 수 있음

sizeof 연산자

피연산자의 메모리 크기를 반환

피연산자로 자료형의 이름이 올 경우 괄호를 사용

그 이외의 경우 괄호의 사용은 선택적

```
int main(void)
{
    int val=10;
    printf("%d",  sizeof val  );     // 변수 val의 메모리 크기 출력
    printf("%d",  sizeof(int) );     // 자료형 int의 메모리 크기 출력
     . . . . .
```

자료형 선택의 기준

정수형 데이터를 처리하는 경우

- 컴퓨터는 내부적으로 int형 연산을 가장 빠르게 처리,
  따라서 정수형 변수는 int형으로 선언
- 범위가 int형 변수를 넘어가는 경우 long형으로 선언
- 값의 범위가 –128 ~ +127 사이라 할지라도 int형으로 선언

자료형 선택의 기준

실수형 데이터를 처리하는 경우

- 선택의 지표는 정밀도
- 정밀도란 오차 없이 표현 가능한 정도를 의미함
- 오늘날의 일반적 선택은 double!

자료형	정밀도
float	소수 이하 6자리
double	소수 이하 15자리
long double	double의 정밀도와 같거나 크다.

unsigned가 붙어서 달라지는 표현의 범위

MSB까지도 데이터의 크기를 표현하는데 사용

양의 정수로 인식

실수형 자료형에는 붙일 수 없다.

자료형	메모리 크기	표현 가능한 데이터의 범위
char(signed char)	1바이트	−128 ~ +127
unsigned char	1바이트	0 ~ (127 + 128)
short(signed short)	2바이트	−32768 ~ +32767
unsigned short	2바이트	0 ~ (32767 + 32768)
int(signed int)	4바이트	−2147483648 ~ +2147483647
unsigned int	4바이트	0 ~ (2147483647 + 2147483648)
long(signed long)	4바이트	−2147483648 ~ +2147483647
unsigned long	4바이트	0 ~ (2147483647 + 2147483648)

문자 표현을 위한 ASCII 코드의 등장

- 미국 표준 협회(ANSI)에 의해 정의
- 컴퓨터를 통해서 문자를 표현하기 위한 표준
  - –컴퓨터는 문자를 표현하지 못함
- 문자와 숫자의 연결 관계를 정의
  - –문자 A는 숫자 65, 문자 B는 숫자 66…

ASCII 코드표

DEC	HEX	OCT	Char	DEC	HEX	OCT	Char	DEC	HEX	OCT	Char
0	00	000	Ctrl-@ NUL	43	2B	053	+	86	56	126	V
1	01	001	Ctrl-A SOH	44	2C	054	,	87	57	127	W
2	02	002	Ctrl-B STX	45	2D	055	-	88	58	130	X
3	03	003	Ctrl-C ETX	46	2E	056	.	89	59	131	Y
4	04	004	Ctrl-D EOT	47	2F	057	/	90	5A	132	Z
5	05	005	Ctrl-E ENQ	48	30	060	0	91	5B	133	[
6	06	006	Ctrl-F ACK	49	31	061	1	92	5C	134	₩
7	07	007	Ctrl-G BEL	50	32	062	2	93	5D	135	]
8	08	010	Ctrl-H BS	51	33	063	3	94	5E	136	^
9	09	011	Ctrl-I HT	52	34	064	4	95	5F	137	_
10	0A	012	Ctrl-J LF	53	35	065	5	96	60	140	`
11	0B	013	Ctrl-K VT	54	36	066	6	97	61	141	a
12	0C	014	Ctrl-L FF	55	37	067	7	98	62	142	b
13	0D	015	Ctrl-M CR	56	38	070	8	99	63	143	c
14	0E	016	Ctrl-N SO	57	39	071	9	100	64	144	d
15	0F	017	Ctrl-O SI	58	3A	072	:	101	65	145	e
16	10	020	Ctrl-P DLE	59	3B	073	;	102	66	146	f
17	11	021	Ctrl-Q DCI	60	3C	074	<	103	67	147	g
18	12	022	Ctrl-R DC2	61	3D	075	=	104	68	150	h
19	13	023	Ctrl-S DC3	62	3E	076	>	105	69	151	i
20	14	024	Ctrl-T DC4	63	3F	077	?	106	6A	152	j
21	15	025	Ctrl-U NAK	64	40	100	@	107	6B	153	k
22	16	026	Ctrl-V SYN	65	41	101	A	108	6C	154	l
23	17	027	Ctrl-W ETB	66	42	102	B	109	6D	155	m
24	18	030	Ctrl-X CAN	67	43	103	C	110	6E	156	n
25	19	031	Ctrl-Y EM	68	44	104	D	111	6F	157	o
26	1A	032	Ctrl-Z SUB	69	45	105	E	112	70	160	p
27	1B	033	Ctrl-[ESC	70	46	106	F	113	71	161	q
28	1C	034	Ctrl-₩ FS	71	47	107	G	114	72	162	r
29	1D	035	Ctrl-] GS	72	48	110	H	115	73	163	s
30	1E	036	Ctrl-^ RS	73	49	111	I	116	74	164	t
31	1F	037	Ctrl_ US	74	4A	112	J	117	75	165	u
32	20	040	Space	75	4B	113	K	118	76	166	v
33	21	041	!	76	4C	114	L	119	77	167	w
34	22	042	"	77	4D	115	M	120	78	170	x
35	23	043	#	78	4E	116	N	121	79	171	y
36	24	044	$	79	4F	117	O	122	7A	172	z
37	25	045	%	80	50	120	P	123	7B	173	{
38	26	046	&	81	51	121	Q	124	7C	174	|
39	27	047	'	82	52	122	R	125	7D	175	}
40	28	050	(	83	53	123	S	126	7E	176	~
41	29	051	)	84	54	124	T	127	7F	177	DEL
42	2A	052	*	85	55	125	U	made by Lee Jae-wook			

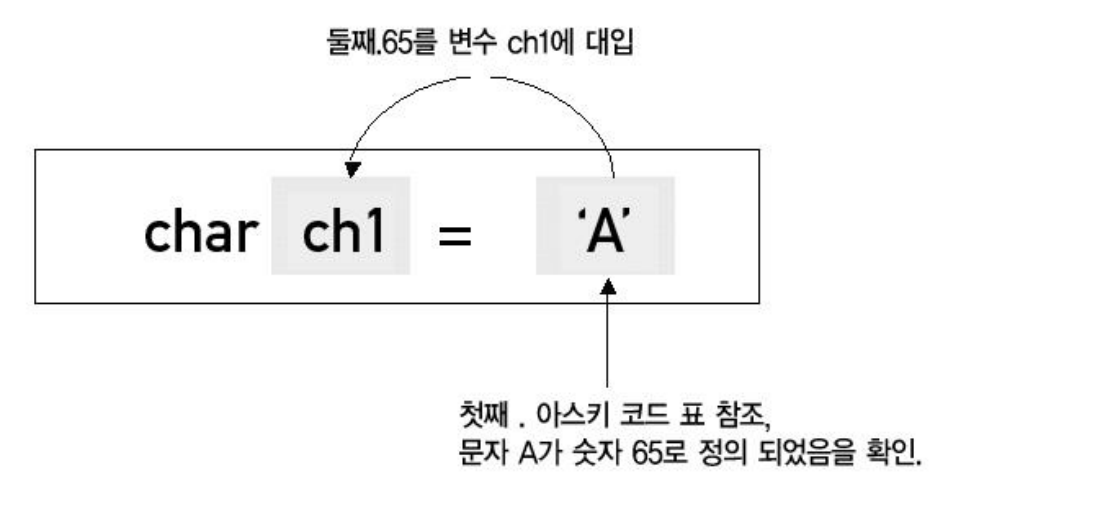
ASCII 코드의 범위

0이상 127이하, char형 변수로 처리 가능

char형으로 처리하는 것이 합리적

문자의 표현

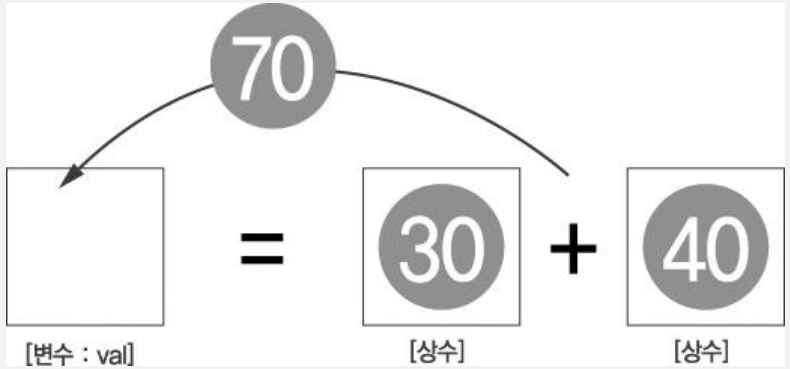
따옴표(' ')를 이용해서 표현



리터럴(literal) 상수

이름을 지니지 않는 상수

```
int main(void)
{
    int val = 30 + 40;
    . . . . .
```



리터럴 상수의 기본 자료형

상수도 메모리 공간에 저장되기 위해서 자료형이 결정된다.

```
int main(void)
{
    char c = 'A';           // 문자상수(char)
    int i = 5;              // 정수상수(int)
    double d= 3.15;         // 실수상수(double)
    . . . . .
```

리터럴 상수의 기본 자료형

```
int main(void)
{
    float f = 3.14;     // float f= 3.14f
    return 0;
}
```

warning C4305: 'initializing' : truncation from 'const double ' to 'float '

warning C4305: ' 초기화 중' : 'double'에서 float'(으)로 잘립니다.

접미사에 따른 다양한 상수의 표현

접미사	자료형	사용 예
u or U	unsigned int	304U
l or L	long	304L
ul or UL	unsigned long	304UL
f or F	float	3.15F
l or L	long double	3.15L

심볼릭(symbolic) 상수

이름을 지니는 상수

심볼릭 상수를 정의하는 방법

const 키워드를 통한 변수의 상수화

매크로를 이용한 상수의 정의

const 키워드에 의한 상수화

```
int main(void)
{
    const int MAX=100;
    const double PI=3.1415;
    . . . . .
}
```

- 잘못된 상수 선언

```
int main(void)
{
    const int MAX;
    MAX=100;
    . . . . .
}
```

자료형 변환의 두 가지 형태 (왜 casting??)

자동 형 변환

- 자동적으로 발생하는 형태의 변환을 의미한다.
- 묵시적 형 변환이라고도 표현한다.

강제 형 변환

- 프로그래머가 명시적으로 형 변환을 요청하는 형태의 변환
- 명시적 형 변환이라고도 표현한다.

자동 형 변환이 발생하는 상황 1

대입 연산 시 (무조건 변수의 자료형으로)

```
int main(void)
{
    int n=5.25;       // 소수부의 손실
    double d=3;       // 값의 표현이 넓은 범위로의 변환
    char c=129;       // 상위 비트의 손실
```

자동 형 변환이 발생하는 상황 2

정수의 승격에 의해(int형 연산이 빠른 이유)

정수형 연산 자체를 단일화시킨 결과

```
int main(void)
{
    char c1=10, c2=20;
    char c3=c1+c2;
    . . . . .
```

자동 형 변환이 발생하는 상황 3

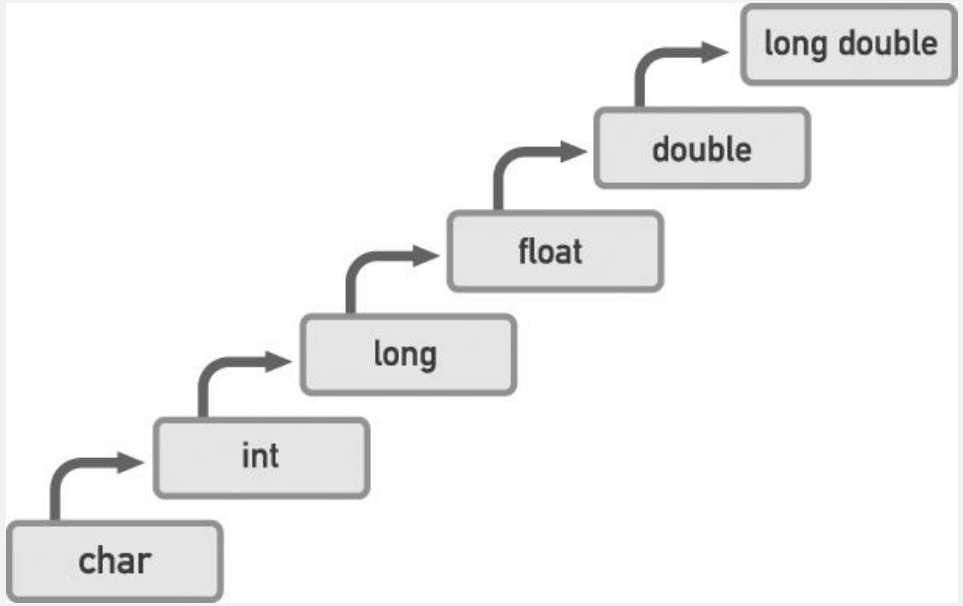
산술 연산 과정에 의해 (더 큰 공간을 차지하는 쪽으로)

```
int main(void)
{
    double e1 = 5.5 + 7;              // double + int
    double e2 = 3.14f+5.25; // float + double
    . . . . .
```

산술 연산 형 변환 규칙

데이터의 손실이 최소화되는 방향으로...

더 큰 공간으로...



강제 형 변환

프로그래머의 요청에 의한 형 변환

```
float f= (float)3.14;           // 3.14를 float 형으로 형 변환
double e1 = 3 + 3.14;           //정수 30이 double 형으로 자동 형 변환
double e2 = 3 + (int)3.14;      // 3.14가 int형으로 강제 형 변환
```

printf 함수는 서식 지정이 가능하다.

printf의 f는 "formatted"를 의미한다.

서식 지정 : 출력의 형태를 지정한다는 의미
(ex : 문자열 안에 숫자 삽입)

서식 지정의 예

```
#include <stdio.h>

int main(void)
{
    int age=12;
    printf("10진수로 %d살이고 16진수로 %x살 입니다.", age, age);
    return 0;
}
```

서식 문자의 종류와 그 의미

서식 문자	출력 형태
%c	단일 문자
%d	부호 있는 10진 정수
%i	부호 있는 10진 정수, %d와 같음
%f	부호 있는 10진 실수
%s	문자열
%o	부호 없는 8진 정수
%u	부호 없는 10진 정수
%x	부호 없는 16진 정수, 소문자 사용
%X	부호 없는 16진 정수, 대문자 사용
%e	e 표기법에 의한 실수
%E	E 표기법에 의한 실수
%g	값에 따라서 %f, %e 둘 중 하나를 선택
%G	값에 따라서 %f, %E 둘 중 하나를 선택
%%	% 기호 출력

%c, %d, %f, %s

가장 많이 쓰이는 서식 문자들

%o, %u, %x, %X

부호 없는 정수형 출력

%e, %E

'부동소수점 표현 방식'에 의한 출력

$3.1245e+2 \rightarrow 3.1245 \times 10^{+2}$

$2.45e-4 \rightarrow 2.45 \times 10^{-4}$

%g, %G

표현하고자 하는 실수의 값이 소수점 이하 6자리인 경우 %f의 형태로 출력

이 범위를 넘길 경우 %e의 형태로 출력

```
#include <stdio.h>

int main(void)
{
    printf("%g ₩n", 0.00123);          // 0.00123 출력
    printf("%G ₩n", 0.000123);         // 0.000123 출력
    printf("%g ₩n", 0.0000123);        // 1.23e-005 출력
    printf("%G ₩n", 0.00000123);       // 1.23E-006 출력

    return 0;
}
```

필드 폭을 지정하여 멋진 출력을!

서식 문자를 이용해서 출력의 폭 지정 가능

예제 field_wid.c 참조

서식 문자	출력의 형태
%8d	필드 폭을 8칸 확보하고 오른쪽 정렬해서 출력하라.
%-8d	필드 폭을 8칸 확보하고 왼쪽 정렬해서 출력하라.
%+8d	필드 폭을 8칸 확보하고 오른쪽 정렬한 상태에서 양수는 +, 음수는 -를 붙여서 출력하라.

scanf 함수의 입력 형태 정의

데이터를 입력받는 형태를 지정할 수 있다.
즉 입력 서식을 지정하는 것이다.

예 : "%d %o %x"

실수 입력에 있어서 주의사항

정밀도 생각!

소수 6자리 이하의 실수 입력 시 %f 사용

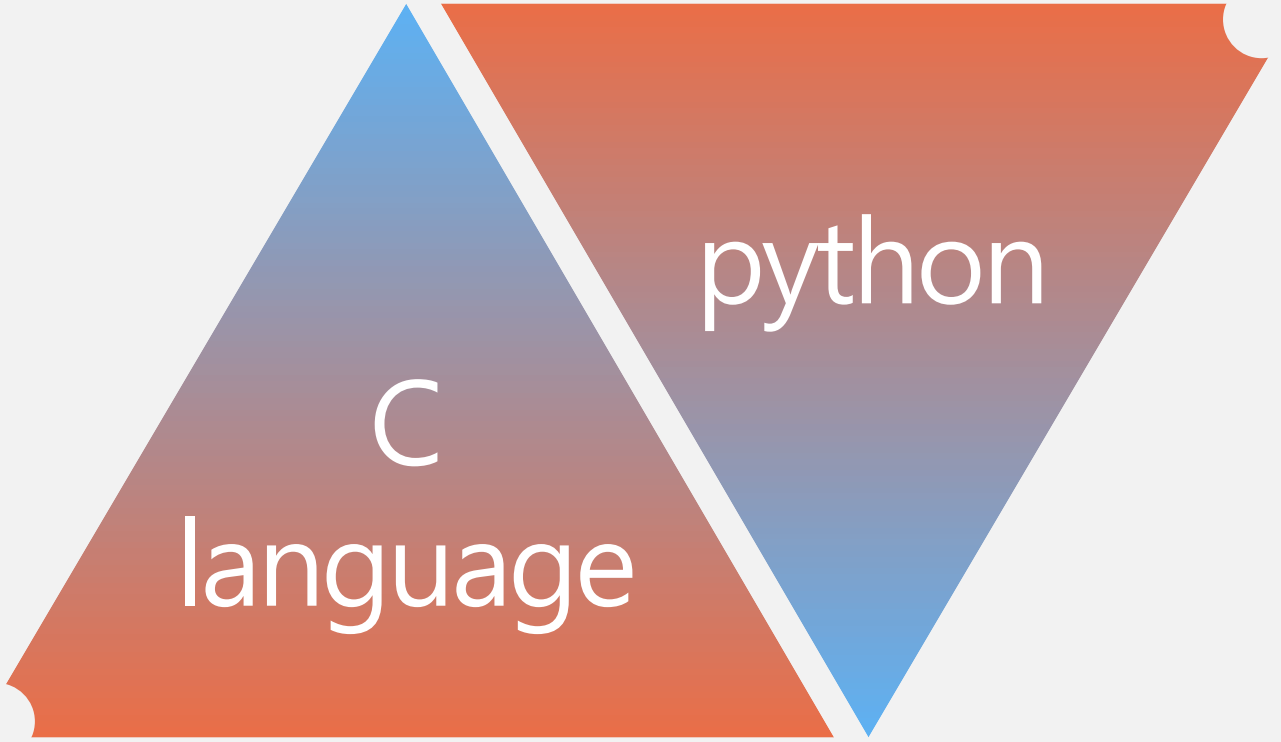
소수 6자리를 넘는 실수 입력 시 %e 사용

단! double형 변수를 사용하는 경우에는 서식 문자 %le를 사용

새로운 변수 ??

1. 배열
2. 포인터 변수
3. 구조체

4. 배열과 포인터



복습