codingOn

C# 기본 문법

C# 및 .Net 버전

```
9   using System.Windows.Forms;
10
11  int num = 111;
12
13  namesp
14  {
15      pu
16      {
17
18
19
20
21
22          InitializeComponent();
23      }
```

(local variable) int num

CS8370: Feature 'top-level statements' is not available in C# 7.3. Please use language version 9.0 or greater.

CS0219: The variable 'num' is assigned but its value is never used

IDE0059: Unnecessary assignment of a value to 'num'

Show potential fixes (Alt+Enter or Ctrl+.)

C# 7.3 에서는 전역변수 사용이 불가능

C#의 역사 (공식 문서)

https://learn.microsoft.com/ko-kr/dotnet/csharp/whats-new/csharp-version-history

C# 및 .Net 버전

기본값

컴파일러는 다음 규칙에 따라 기본값을 결정합니다.

Target	버전	C# 언어 버전 기본값
.NET	7.x	C# 11
.NET	6.x	C# 10
.NET	5.x	C# 9.0
.NET Core	3.x	C# 8.0
.NET Core	2.x	C# 7.3
.NET Standard	2.1	C# 8.0
.NET Standard	2.0	C# 7.3
.NET Standard	1.x	C# 7.3
.NET Framework	모두	C# 7.3

C# 버전은 닷넷 버전에 따라 달라짐

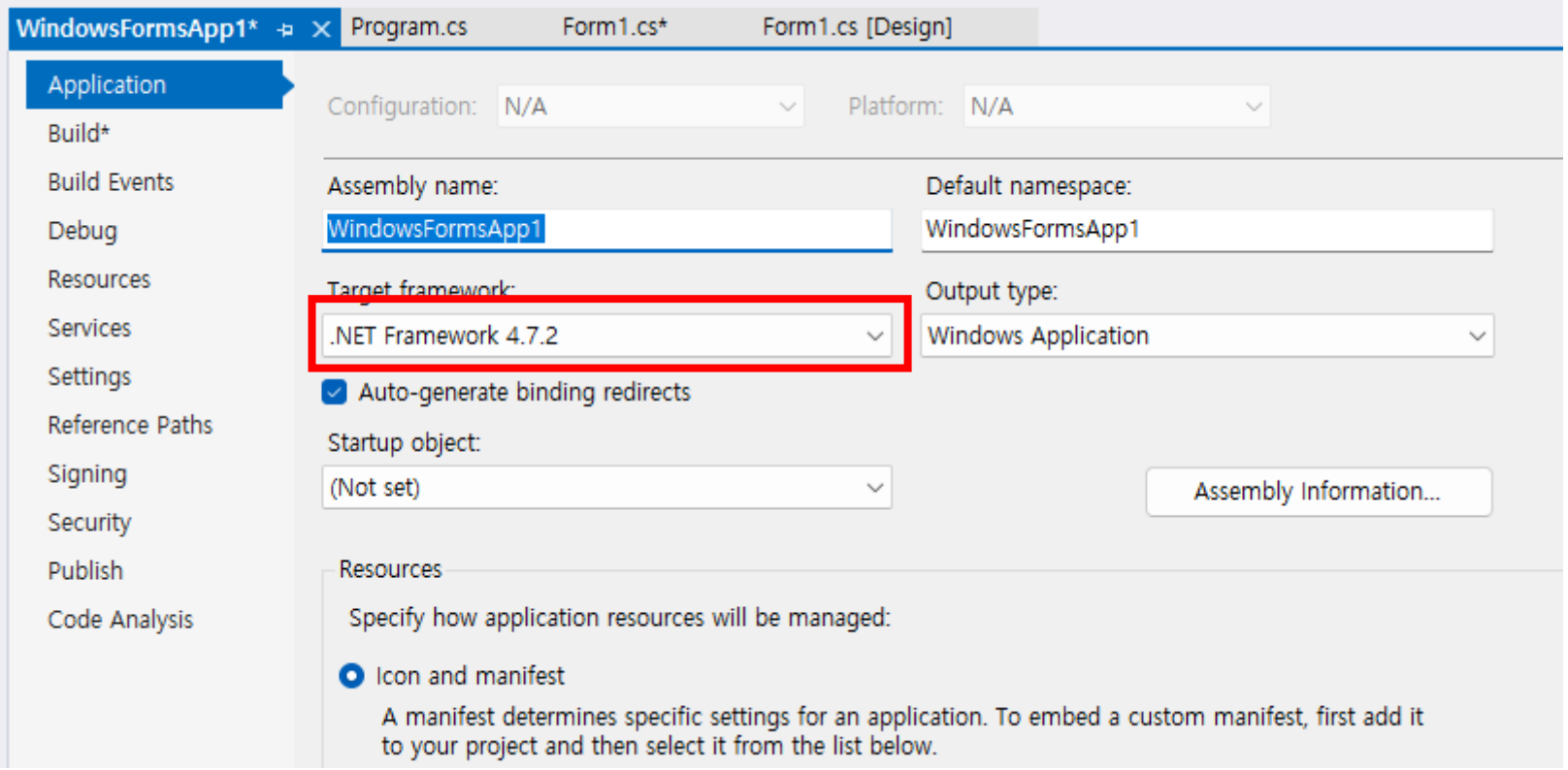
.Net Core -> 오픈소스, 멀티 플랫폼
.Net Framework -> 윈도우용

.Net 5.x 부터 Core와 Framework 통합

https://learn.microsoft.com/ko-kr/dotnet/csharp/language-reference/configure-language-version

C# 및 .Net 버전

.NET Framework 버전 확인하기
솔루션 탐색기 > 프로젝트 우클릭 > 속성



Form1.cs 살펴보기

네임스페이스를 가져오기, 파이썬의 import와 비슷

```
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp1
{
    3 references
    public partial class Form1 : Form
    {
        1 reference
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

회색: 현재 코드에서 사용하지 않음

검은색: 현재 코드에서 사용 중

내 프로그램의 네임스페이스

코드 영역을 { } 로 구분

Form 클래스를 상속 받는 Form1 클래스
- 접근 제어: public
- partial 클래스

Form1 클래스와 같은 이름의 메소드
➔ Form1 클래스의 생성자

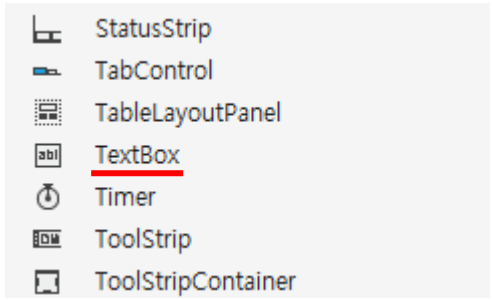
코드를 실행하려면 반드시 ;(세미콜론) 을 붙임

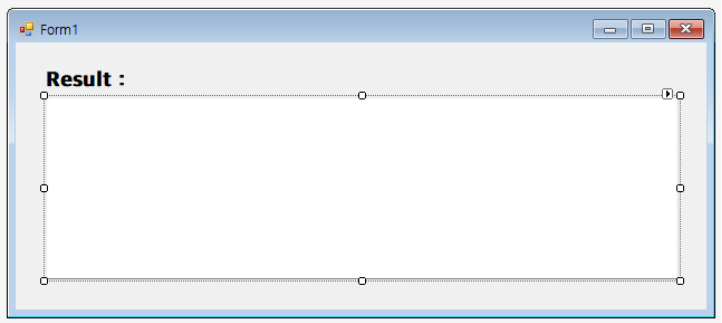
윈폼 초기화를 위한 함수

실습을 위한 준비

- 지난 강의에서 다루었던 내용과 같이 TextBox를 이용해 결과를 확인

ToolBox





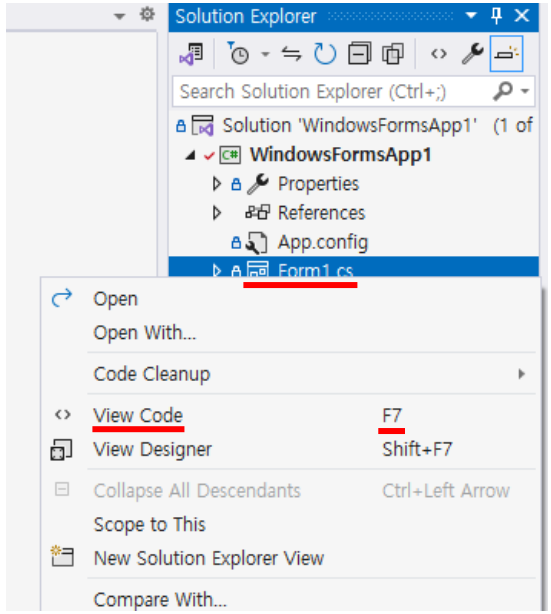
Properties

ImeMode	NoControl
MaxLength	32767
Multiline	True
PasswordChar	
ReadOnly	False
ShortcutsEnabled	True

Design	
(Name)	textBox_print
GenerateMember	True
Locked	False
Modifiers	Private

실습을 위한 준비

- 소스코드는 public Form1() 중괄호 안에 작성



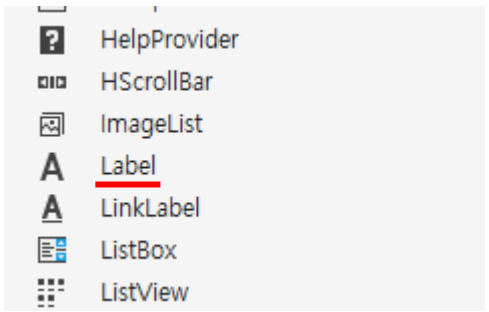
Form1.cs 에서 우클릭

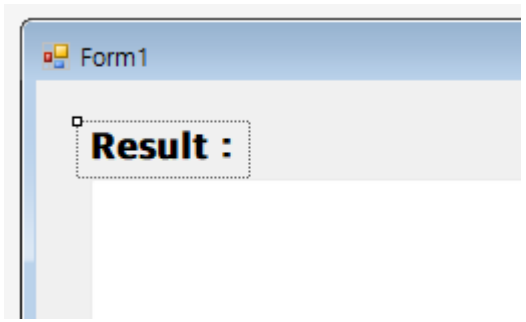
```
namespace WindowsFormsApp1
{
    3 references
    public partial class Form1 : Form
    {
        1 reference
        public Form1()
        {
            InitializeComponent();

            // 이곳에 소스코드를 작성

            textBox_print.Text = "어떤 결과가 나올까?";
        }
    }
}
```

실습을 위한 준비

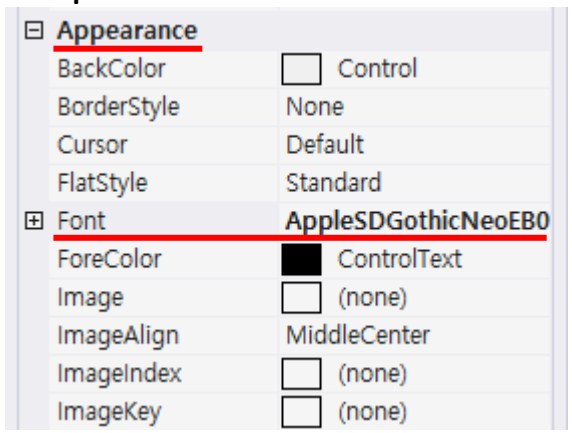
- ToolBox의 Label 컨트롤을 사용하여 텍스트를 Form에 넣을 수 있음

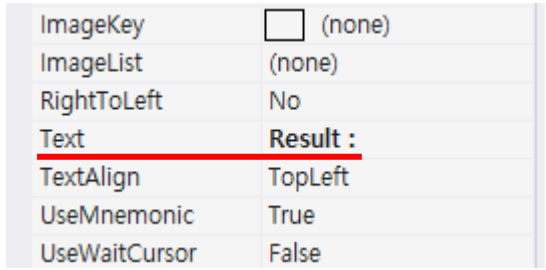
ToolBox





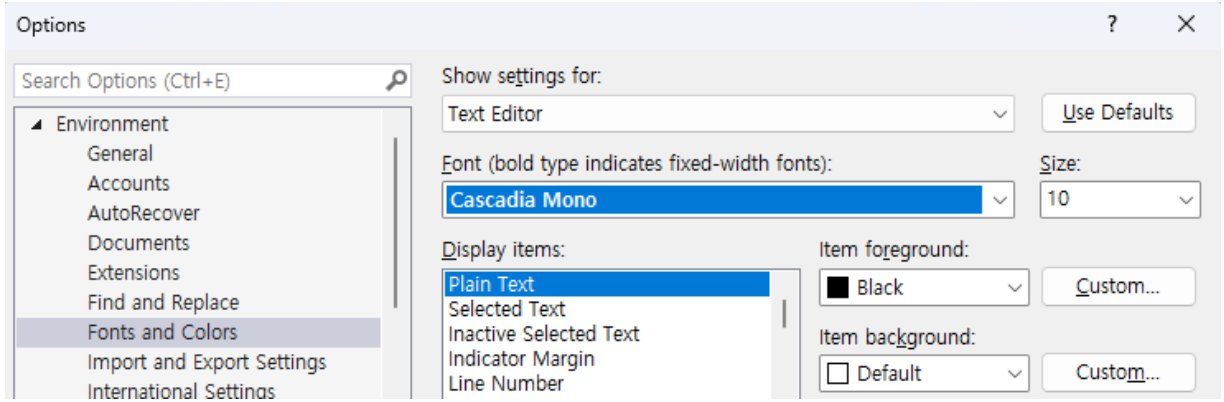
Properties

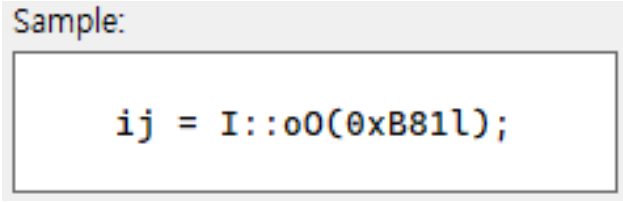




실습을 위한 준비

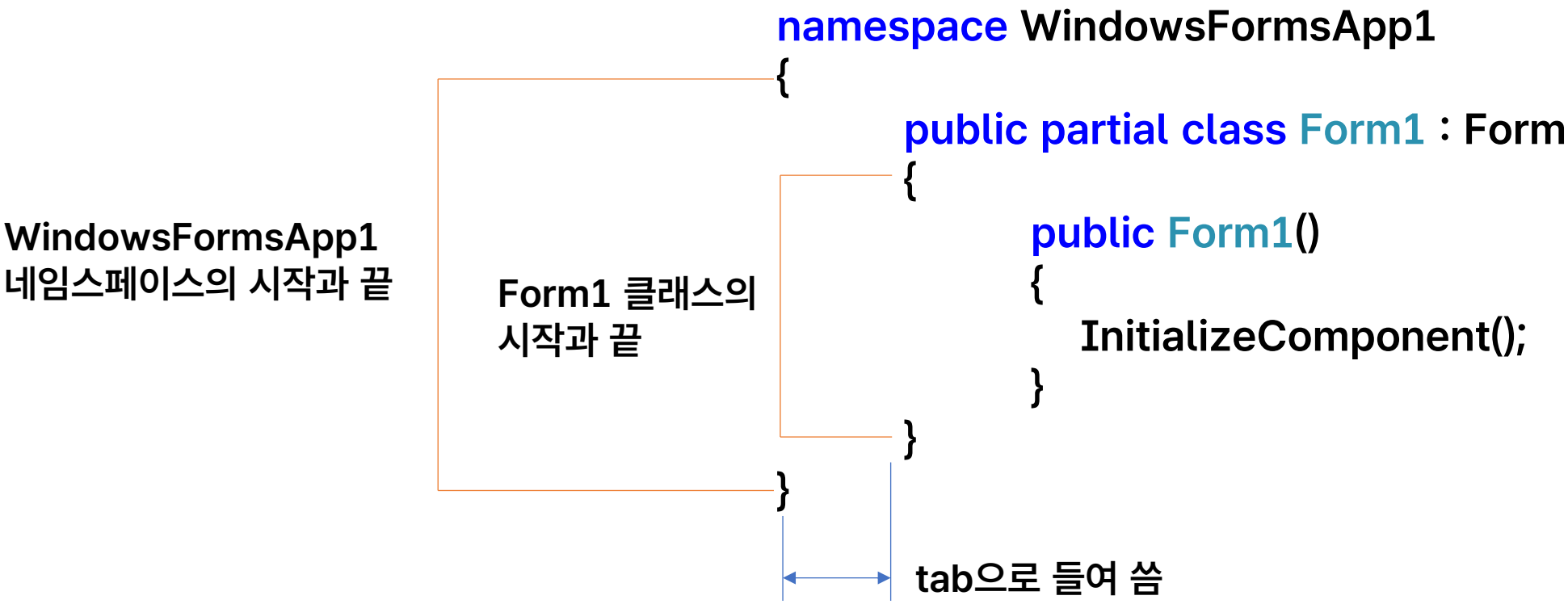
- 폰트는 B80OlI1를 구분하기 좋고 consola, fira, cascadia 등 문자 간격이 일정한 mono 타입 폰트를 사용
  - consola: 윈도우에서 기본 제공
  - fira: https://fonts.google.com/specimen/Fira+Code
  - cascadia: Visual Studio 기본 제공
- Tools > Options > Environment > Fonts and Colors 에서 변경





Scope (코드 영역)

- { } 를 사용하여 해당 클래스, 함수 등의 시작과 끝을 표시



Scope (코드 영역)

- 파이썬과 다르게 줄 간격, 들여쓰기와 상관 없음

```
namespace WindowsFormsApp1
{
    public partial class Form1 : Form { public Form1()
        { InitializeComponent(); }}}
```

표기법

dash-case (kebab-case)

snake_case

camelCase

PascalCase

tomakeyoufeelmyloveknowmetoowell

dash-case(kebab-case)

to-make-you-feel-my-love-know-me-too-well

snake_case

to_make_you_feel_my_love_know_me_too_well

camelCase

toMakeYouFeelMyLoveKnowMeTooWell

PascalCase

ToMakeYouFeelMyLoveKnowMeTooWell

codingOn

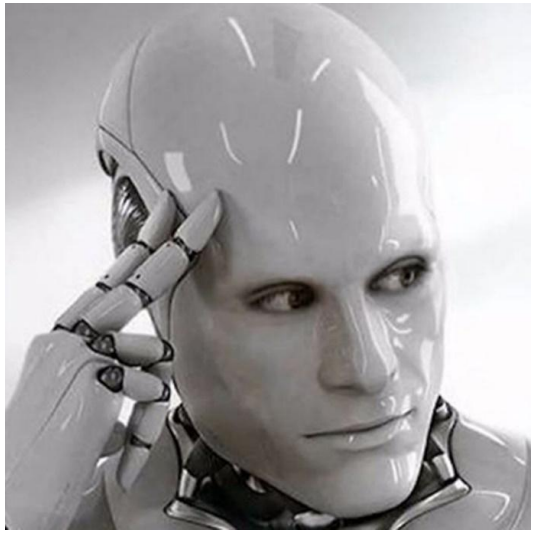
Zero-based numbering

Zero-based Numbering

- 프로그래밍에서는 0 기반으로 번호를 매긴다!
- 특수한 경우를 제외하고는 **0부터 숫자를 시작**함
- 즉, 첫 번째 요소의 번호(index)가 **0**입니다.

codingOn

1 2 3 4 5 6 7 8 9 10 ~



0 1 2 3 4 5 6 7 8 9 ~

인덱스 (Index)

- C#을 포함한 대부분의 프로그래밍 언어에서 데이터의 위치를 가리키는 번호.
- 배열, 리스트, 문자열과 같은 **'순서가 있는 데이터'**에서 각 요소를 구별하기 위해 부여된 번호.
- 특정 위치의 데이터에 빠르게 접근하거나 수정하기 위해 사용.

```
string[] fruits = { "apple", "banana", "cherry" };

// 인덱스:   0          1          2
Console.WriteLine(fruits[0]);  // apple
Console.WriteLine(fruits[1]);  // banana
Console.WriteLine(fruits[2]);  // cherry
```

codingOn

주석

codingOn

한 줄 주석 (//)

- // 이후의 문장은 모두 주석으로 간주.
- 주로 짧은 설명이나 TODO를 기록할 때 사용.

여러 줄 주석 (/* */)

- 블록 단위로 주석을 달고 싶을 때 사용
- 중첩은 안됨! (안에 다른 /* */ 또 쓰면 오류

```
// 한 줄 주석

/*
 *
 * 여러 줄 주석
 *
 */
```

변수, 자료형

변수 및 기본 자료형

```
int num = -100;
uint p_num = 321; // 양수만 가능
float f_num = 124.5213f;
double d_num = 321.321;
decimal D_num = 987.654m;
char word = 'A'; // 유니코드 16bit 문자 (한 글자)
string name = "John"; // 유니코드 문자열
```

자료형의 종류

- 실제로 변수의 크기 만큼 메모리(RAM)에서 용량을 차지함

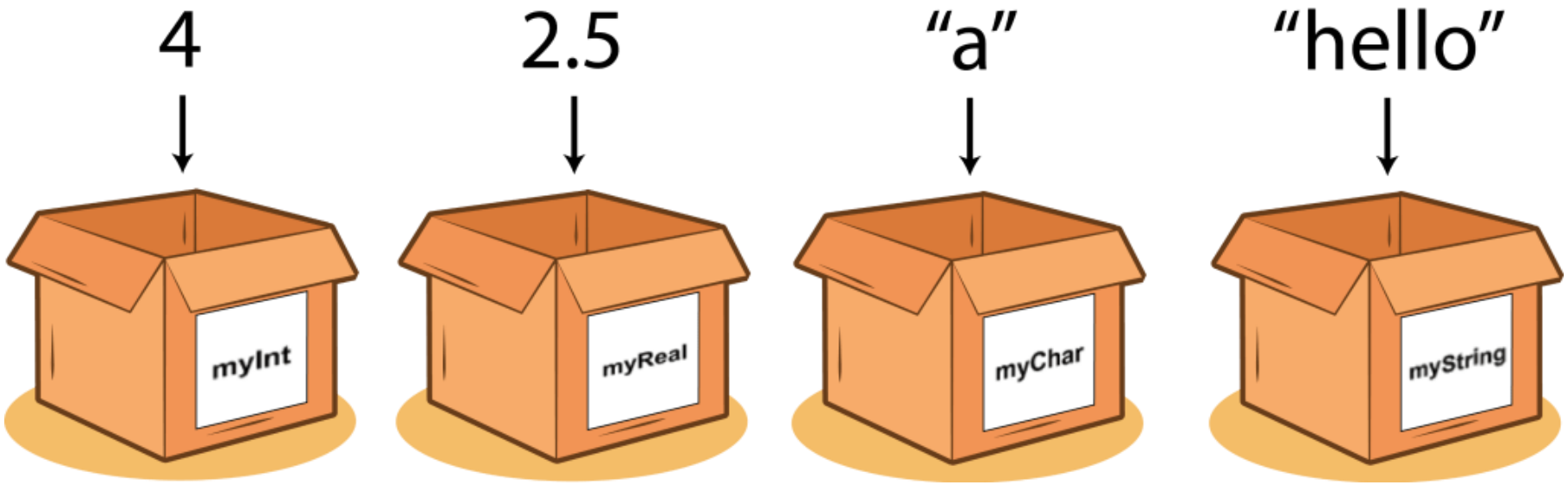
자료형	형식	범위	크기
sbyte	System.SByte	-128 ~ 127	부호 있는 8 bit 정수
byte	System.Byte	0 ~ 255	부호 없는 8 bit 정수
short	System.Int16	-32,768 ~ 32,767	부호 있는 16 bit 정수
ushort	System.UInt16	0 ~ 65,535	부호 없는 16 bit 정수
int	System.Int32	-2,147,483,648 ~ 2,147,483,647	부호 있는 32 bit 정수
uint	System.UInt32	0 ~ 4,294,967,295	부호 없는 32 bit 정수
long	System.Int64	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,808	부호 있는 64 bit 정수
ulong	System.UInt64	0 ~ 18,446,744,073,709,551,615	부호 없는 64 bit 정수

자료형의 종류

float	System.Single	±1.5e-45 ~ ±3.4e38	4 byte
double	System.Double	±5.0e-324 ~ ±1.7e308	8 byte
decimal	System.Decimal	$\pm 1.0 \times 10^{-28}$ $\pm 7.9 \times 10^{28}$	16 byte
char	System.Char	U+0000 ~ U+ffff	유니코드 16 bit 문자
string	System.String		유니코드 문자열
bool	System.Boolean		4 byte

변수

- 변수 = variable, 변하는 값
- 변수는 데이터를 담는 빈 그릇!
- 변수 이름은 식별자 네이밍 규칙이 적용 됨



변수

식별자

- 프로그래밍 언어에서 **이름을 붙일 때** 사용하는 단어
- 주로 변수, 함수 이름 등으로 사용함
- 규칙
  - 첫 문자는 알파벳 문자이거나 밑줄(_)이어야 함
  - 나머지 문자는 문자, 밑줄(_) 또는 숫자(0-9)여야 함
  - **대소문자 구분**
  - 예약어는 사용 불가
- 관례
  - camelCase : 변수명, 매개변수
  - PascalCase : 클래스명, 메서드명, 속성명

hi apple (x) → 공백

hello* (x) → 특수문자

hello_ (o) → 특수문자이기는 하지만, _ 허용

hello23 (o)

1hello (x) → 숫자로 시작 불가능

키워드 (예약어)

- 컴파일러에 특별한 의미가 있는 미리 정의된 예약 식별자
- C# 언어 자체에서 **미리 정해놓은 의미가 있는 키워드**로,
  프로그래머가 **식별자(변수명, 함수명 등)**로 사용할 수 없음.

abstract	event	namespace	static
as	explicit	new	string
base	extern	null	struct
bool	false	object	switch
break	finally	operator	this
byte	fixed	out	throw
case	float	override	true
catch	for	params	try
char	foreach	private	typeof
checked	goto	protected	uint
class	if	public	ulong
const	implicit	readonly	unchecked
continue	in	ref	unsafe
decimal	int	return	ushort
default	interface	sbyte	using
delegate	internal	sealed	virtual
do	is	short	void
double	lock	sizeof	volatile
else	long	stackalloc	while
enum			

Q1. 퀴즈

Q. 다음 중 가장 적절하게 이름이 지어진 것은?

① totalCount
② _score
③ 3rdPlayer
④ user_name

Q2. 퀴즈

Q. 다음 중 **클래스 이름으로 가장 적절한 식별자는?**

① student_info
② StudentInfo
③ studentinfo
④ STUDENTINFO

Q3. 퀴즈

Q. 다음 코드 중 **컴파일 오류가 발생하는 경우**는?

① int @int = 5;
② int total$ = 100;
③ int _value = 10;
④ int count2 = 2;

Q4. 퀴즈

Q. 다음 중 변수 이름으로 가장 권장되는 작명은?

① PrintResult
② Total_Count
③ userScore
④ name list

codingOn

변수명 Tip!

```
// 이름만 보고 용도를 알 수 있게
int score = 95;
int userAge = 22;
decimal totalPrice = 129.99m;

// 너무 짧지도, 너무 길지도 않게
int t = 1;  // ✕ 의미 없음
int totalUserCountInSessionTime = 100; // ✕ 너무 길어서 읽기 힘듦
int userCount = 100;  // O 적절한 길이 + 의미 전달

// 일관된 네이밍 컨벤션 사용
string userName = "Alice";  // camelCase → 지역 변수, 매개 변수
string UserEmail { get; set; } = "alice@example.com"; // PascalCase → 속성, 클래스 등

// 불리언 변수는 질문처럼
bool isActive = true;
bool hasError = false;
bool canRetry = true;
```

변수의 선언, 사용, 초기화

- 선언
  - "자료형"^"변수이름"; 형태로 작성
  - 앞으로 해당 변수를 사용하겠다고 컴파일러에게 알려주는 기능
  - 같은 Scope 안에서 같은 이름의 변수를 사용할 수 없음
- 사용(할당)
  - "변수이름" = "데이터"; 형태로 작성
  - 변수의 선언을 한 뒤에만 사용이 가능
  - 선언된 변수에 특정 데이터를 복사하는 기능
  - = 기호를 중심으로, 좌 = 변수명, 우 = 복사할 데이터 🌟
- 초기화
  - "자료형"^"변수이름" = "데이터";
  - 변수 선언과 동시에 값을 지정

codingOn

변수의 선언 및 사용

```
// 변수의 선언
int numOfCrew;

// 변수의 사용 (값 복사)
numOfCrew = 19;

// 변수의 초기화
string className = "말하기 듣기";

// 변수의 값 덮어쓰기
className = "기술 가정";

// 선언보다 밑에 줄에서 사용가능
lineCount = 10;
int lineCount;
```

```
// 같은 이름 사용 불가
byte buffer;
float buffer;

// 데이터 타입이 다르면 복사 불가
int number = 10;
string word = "안녕";
number = word;

// 변수끼리 값 복사
int var_x = 10;
int var_y = var_x; // x->y로 복사

// 사칙 연산 및 괄호 활용
int var_z = var_x * var_y;
int result = var_z + (var_x + 5);
```

codingOn

변수와 Scope

- 선언된 변수는 같은 Scope (중괄호) 안에서만 사용 가능
- Scope를 벗어난 변수는 메모리에서 반환됨 (더이상 사용 못함)

```
{
    int inside = 100;
}

// inside와 Scope가 달라서 사용 불가
int outside = inside + 50;
```

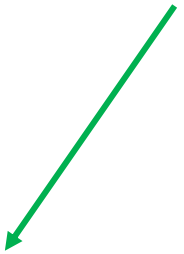
변수 Casting(변환)

```
// 데이터를 주고 받을 때 자료형을 반드시 맞춰 줘야 함
int num = -100;
string name = "300";

num = name; // Error
num = int.Parse(name); // name의 값을 숫자로 변환
num = int.Parse("200"); // 이렇게도 가능
```

소괄호 안에 값을 넣을 수 있는 코드를 "함수"라고 함
(자세한 조건은 이후 함수를 배울 때 다룰 예정)

변수 Casting(변환)

```
// 데이터를 주고 받을 때 자료형을 반드시 맞춰 줘야 함
int num = -100;
string name = "300";


name = num; // Error
name = num.ToString(); // num의 값을 문자열로 변환
name = 400.ToString(); // 이렇게도 가능
```



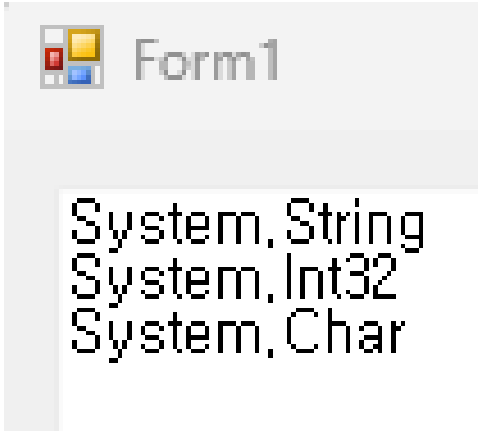
소괄호에 값을 넣지 않아도 작동하는 함수도 있음
(이름 옆에 소괄호가 있으면 함수라고 생각해도 무방)

자동 자료형 지정

- 입력 데이터의 형식이 불분명 할 경우 사용

+= 기호는 덧셈 누적 연산을 수행
값을 덮어쓰지 않고, 기존 값에 추가로 연산을 수행 (-=, *=, /= 도 가능)

```
var name = "John";
var num = 1000;
var one_char = 'A';

textBox_print.Text += name.GetType();
textBox_print.Text += "\r\n";
textBox_print.Text += num.GetType();
textBox_print.Text += "\r\n";
textBox_print.Text += one_char.GetType();
```



* 문자열끼리 덧셈이 가능: "App" + "le" → "Apple"

실습. 변수 및 캐스팅

1. byte, short, int, float, double, decimal 변수를 선언
2. 변수 이름은 자료형과 적합한 것을 선택
   - ex1. byte는 0~255까지만 표현이 가능하므로 초등학교 한 반의 학생 수
   - ex2. double은 매우 많은 소수점 자리 수 표현이 가능하므로 우주 로켓 발사 시 변화하는 압력의 값
   - 변수 이름은 숫자로 시작할 수 없음
3. GetType() 함수 및 ToString() 함수를 이용하여 "데이터 타입"^"변수명":^"데이터" 순서로 표시
4. Push후 레포 주소를 슬랙 댓글에 남기기

Result :

```
System.Byte retroColorRed: 20
System.Int64 distanceToCanada_cm: 20
```

연산자

codingOn

연산자

- 대입 연산자: =
- 비교 연산자: >, >=, <, <=, ==, !=,
- 산술 연산자: +, -, *, /, %(나머지), **(거듭제곱)
- 논리 연산자: !(not), &&(and), ||(or)

기본 연산자

- % 나머지 연산자
  - 홀수 판단 : num % 2 == 1 이면 홀수
  - 짝수 판단 : num % 2 == 0 이면 짝수
- ** 거듭 제곱
  - ** 를 사용
  - 2 ** 3 = 8
  - 3 ** 3 = 27

codingOn

연산자 줄여 쓰기 (대입 연산자)

- num = num + 5 → num += 5
- num = num - 5 → num -= 5
- num = num * 5 → num *= 5
- num = num / 5 → num /= 5

증감 연산자

- int num = 0;
- i++ → i = i + 1
- i-- → i = i - 1

codingOn

비교 연산자

- 비교 연산자
  - a == b : a와 b가 동일하면 참
  - a != b : a와 b가 동일하지 않으면 참
  - a < b : a가 b보다 작으면 (b가 a보다 크면) 참
  - a <= b : a가 b보다 작거나 같으면 참

codingOn

논리 연산자

- || (or) : 여러 개 중 하나라도 true ➞ true
- && (and) : 모든 값이 true ➞ true
- ! (not) : true ➞ false, false ➞ true

codingOn

삼항 연산자

- 형식
- 조건 ? 참일 때 값 : 거짓일 때 값

```
int score = 85;
string result = (score >= 60) ? "합격" : "불합격";
```