Theory of Multi-Paradigm Programming Languages

Lecture 3-1

Chapter 3 ML Programming Language

Overview

Prof. Moonkun Lee, Ph.D.

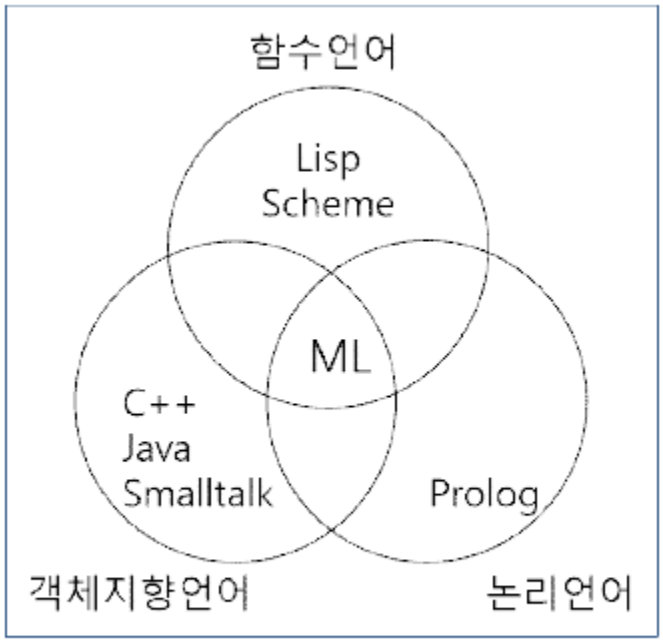
Division of Computer Science and Engineering
Jeonbuk National University, Republic of Korea

Contents

3.1 ML언어의 특징

Main Characteristics

- A functional language
- Side-effect freedom
- High-order function
- Polymorphism
- Abstract data types
- Recursion
- Rule-based programming
- Strong type
- Pattern
- Generic data/function/module



<그림 3-1> ML Meta-Domain

3.2 BASIC

[L#]	화면
1	(UNIX command level)
2	**> sml**
3	*Standard ML of New Jersey, Version 0.93, February 15, 1993*
4	*val it = () : unit*
5	**5;**
6	*val it = 5 : int*
7	**"abc";**
8	*val it = "abc" : string*
9	**^d**
10	**>** (Return to UNIX command level)

<프로그램 3-1> Interpreter 방식

다음 주 실습 방법 강의

PC에서의 IDE(Integrated Development Environment) 및
Cloud 서버의 Unix/Linux 상에서의
ML 프로그램 작성 및 실행 방법
Video 특강

```
3  Standard ML of New Jersey, Version 0.93, February 15, 1993
4  val it = () : unit
5  use "foo.sml"
6  (opening foo.sml)
7  5;
8  val it = 5 : int
9  "abc";
10 val it = "abc" : string
11 > (Return to UNIX command level)
```

<프로그램 3-3> Compilation 결과

[L#]	화면
1	**5;**
2	*val it = 5 : int*
3	**~5;**
4	*val it = ~5 : int*
5	**0x1234;**
6	*val it = 4660 : int*
7	**1 + 2;**
8	*val it = 3 : int*
9	**5 - 2;**
10	*val it = 3 : int*
11	**5 * 5;**
12	*val it = 25 : int*
13	**5 div 5;**
14	*val it = 1 : int*
15	**5 mod 5;**
16	*val it = 0 : int*
17	**5 = 5;**
18	*val it = true : bool*
19	**5 <> 5;**
20	*val it = false : bool*
21	**5 < 2;**
22	*val it = false : bool*
23	**5 <= 5;**
24	*val it = true : bool*
25	**5 > 2;**
26	*val it = true : bool*
27	**5 >= 5;**
28	*val it = true : bool*

<프로그램 3-3-1> integer 관련 연산자 예시 및 결과

[L#]	화면
1	**5.1;**
2	*val it = 5.1 : real*
3	**~5.2;**
4	*val it = ~5.2 : real*
5	**3.14E3;**
6	*val it = 3140.0 : real*
7	**3E~3;**
8	*val it = 0.003 : real*
9	**1.2 + 2.2;**
10	*val it = 3.4 : real*
11	**5.2 - 2.1;**
12	*val it = 3.1 : real*
13	**5.4 * 5.2;**
14	*val it = 28.08 : real*
15	**5.5 / 5.2;**
16	*val it = 1.05769230769 : real*
17	**5.1 = 5.1;**
18	*val it = true : bool*
19	**5.1 <> 5.1;**
20	*val it = false : bool*
21	**5.1 < 2.1;**
22	*val it = false : bool*
23	**5.1 <= 5.1;**
24	*val it = true : bool*
25	**5.1 > 2.1;**
26	*val it = true : bool*
27	**5.1 >= 5.1;**
28	*val it = true : bool*

<프로그램 3-3-2> real 관련 연산자 예시 및 결과

[L#]	화면
1	**true;**
2	*val it = true : bool*
3	**false;**
4	*val it = false : bool*
5	**not true;**
6	*val it = false : bool*
7	**not false;**
8	*val it = true : bool*
9	**(5 < 1) andalso (5 <= 5);**
10	*val it = false : bool*
11	**(5 < 1) orelse (5 <= 5);**
12	*val it = true : bool*
13	**1=1;**
14	*val it = true : bool*
15	**"string" = "string";**
16	*val it = true : bool*
17	**1<>1;**
18	*val it = false : bool*
19	**"string" <> "String";**
20	*val it = true : bool*

<프로그램 3-3-3> Boolean 관련 연산자 예시 및 결과

[L#]	화면
1	**#"A";**
2	*val it = #"A" : char*
3	**#"A" = #"a";**
4	*val it = false : bool*
5	**#"A" <> #"B";**
6	*val it = true : bool*
7	**#"A" < #"B";**
8	*val it = true : bool*
9	**#"A" <= #"B";**
10	*val it = true : bool*
11	**#"A" > #"B";**
12	*val it = false : bool*
13	**#"A" >= #"B";**
14	*val it = false : bool*

<프로그램 3-3-4> Char 관련 연산자 예시 및 결과

[L#]	화면
1	**"this is a string";**
2	*val it = "this is a string" : string*
3	**"abc" ^ "defg";**
4	*val it = "abcdefg" : string*
5	**"string" = "string";**
6	*val it = true : bool*
7	**"string" <> "String";**
8	*val it = true : bool*
9	**"string" < "string";**
10	*val it = false : bool*
11	**"string" <= "string";**
12	*val it = true : bool*
13	**"string" > "string";**
14	*val it = false : bool*
15	**"string" >= "string";**
16	*val it = true : bool*

<프로그램 3-3-5> String 관련 연산자 예시 및 결과

	integer	real	boolean	char	string
values	positive negetive hexidecimal	positive negetive scientific	true false	character ASCII code	string
연산자	+ - * div mod ~ = <> < <= > >=	+ - * / ~ = <> < <= > >=	not andalso orelse = <>	= <> < <= > >=	^ = <> < <= > >=

<표 3-1> 데이터 유형의 값과 연산

```
1   val x = 1;
2   val x = 1 : int
3   val y = 1.1;
4   val y = 1.1 : real
5   val z = true;
6   val z = true : bool
7   val s = "abc";
8   val s = "abc" : string
9   val t = #"d";
10  val t = #"d" : char
```

<프로그램 3-4> Variables

```
1   1+2;
2   val it = 3 : integer
3   1.0+2.0;
4   val it = 3.0 : real
5   20 div 6;
6   val it = 3 : int
7   20 mod 6;
8   val it = 2 : int
9   ~2*(~3);195)
10  val it = 6 : int
```

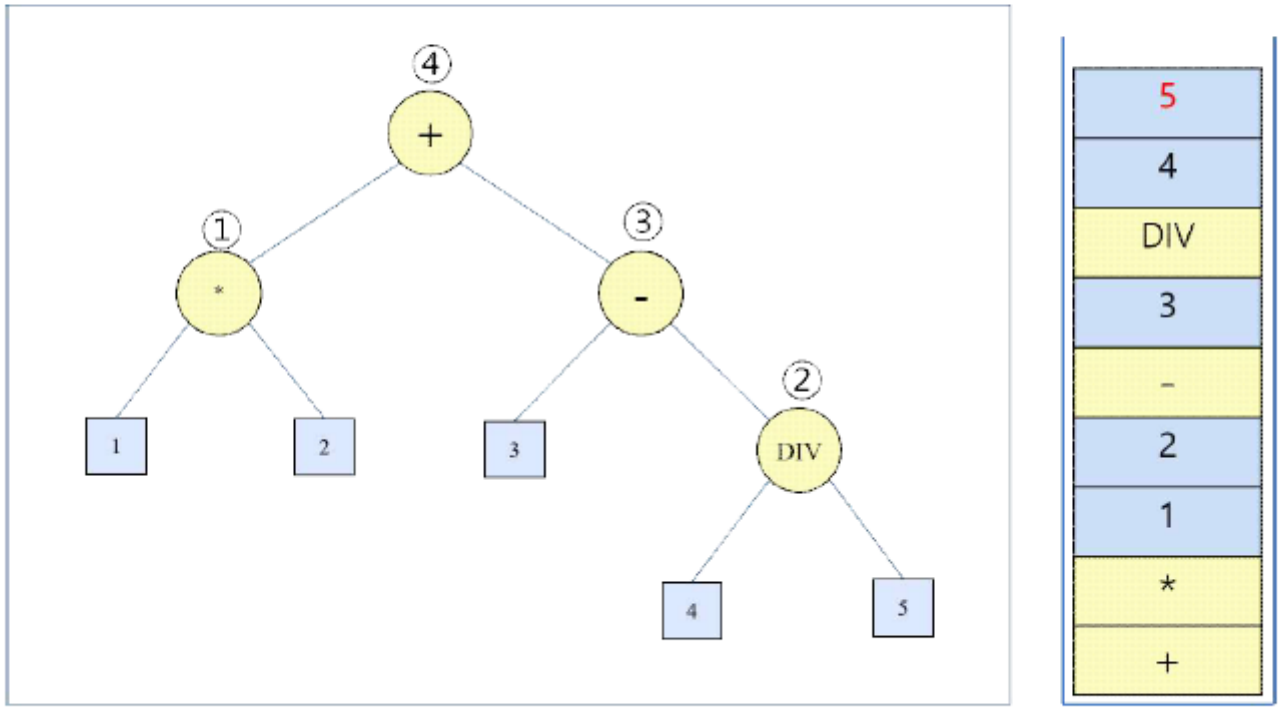
<프로그램 3-5> Arithmetic Operations

```
1  val x = 1;
2  val x = 1 : int
3  val y = 2;
4  val y = 2 : int
5  val z = x+y;
6  val z = 3 : int
```

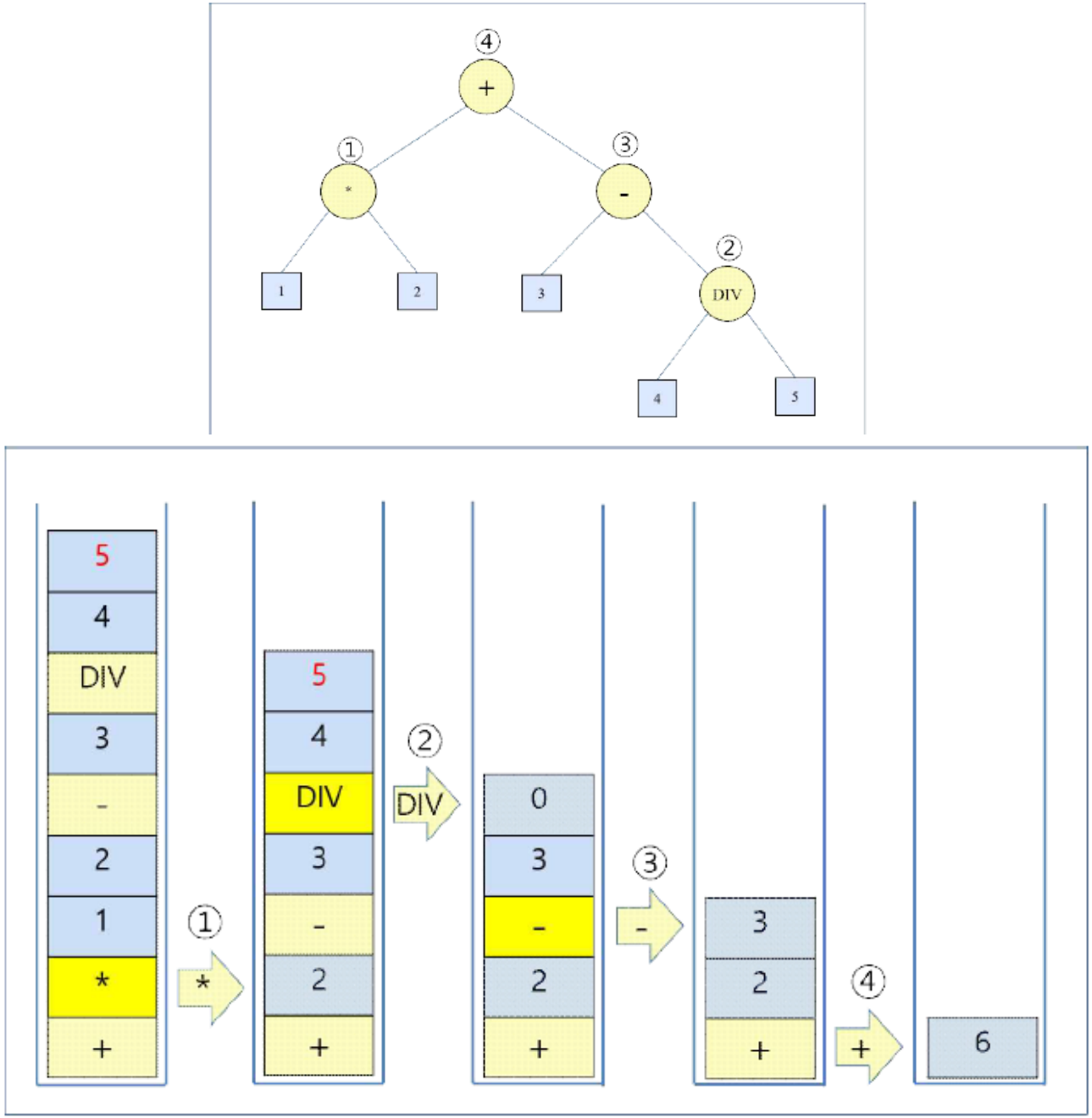
<프로그램 3-6> Integer Addition: Variable

```
1  1*2+3-4div5;
2  val it = 4 : int
```

<프로그램 3-7> Integer Addition: Variable



<그림 3-2> Precedency Tree & Execution Stack



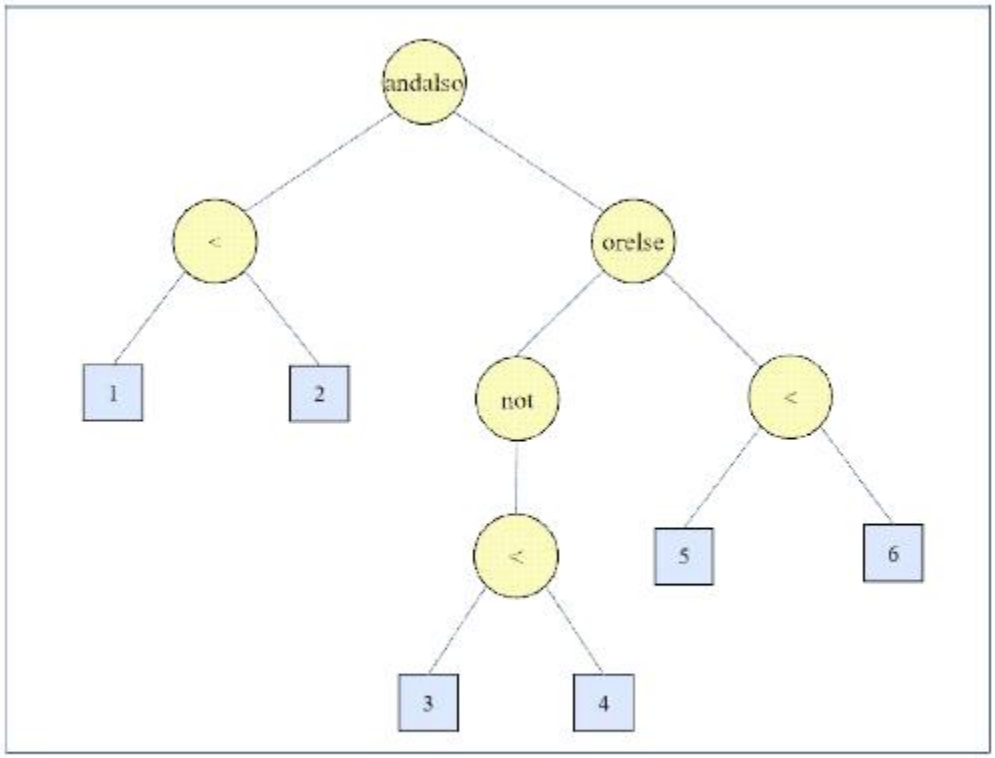
<그림 3-3> Stack Operations

```
1  1<=2;
2  val it = true : bool
3  "abc" < "def";
4  val it = true : bool
```

<프로그램 3-9> Comparison Operation

```
1  1<2 andalso 2<1;
2  val it = false : bool
3  1<2 andalso (not (3<4) orelse 5<6);
4  val it = true : bool
```

<프로그램 3-10> Comparison Operation



<그림 3-4> Precedency Tree

단축연산자(short-circuit operator)

a && b

a || b

```
1  if 1<2 then 3 else 4;
2  val it = 3 : int
```

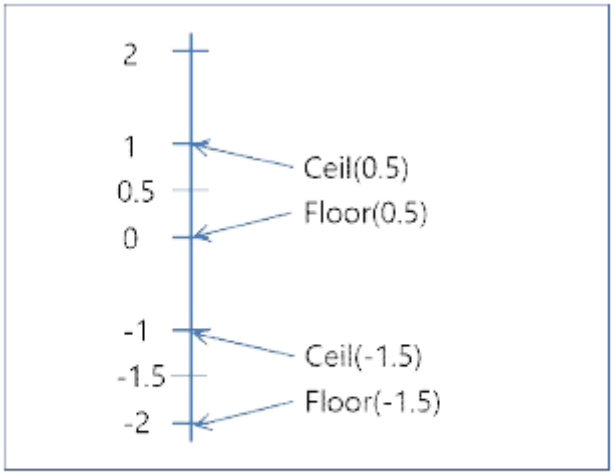
<프로그램 3-11> If-Then-Else Operation

```
1  3 + 4.0;
2  std in:1.1-1.7 Error: operator and operand don't agree
3  (tyconmismatch)
4  operator domain: int * int
5  operand: int * real
6  in expression:
7  + : overloaded (3,4.0)
```

<프로그램 3-12> Type Checking

```
1  (real) 3 + 4.0;
2  val it = 7.0: real
```

<프로그램 3-13> Coercion



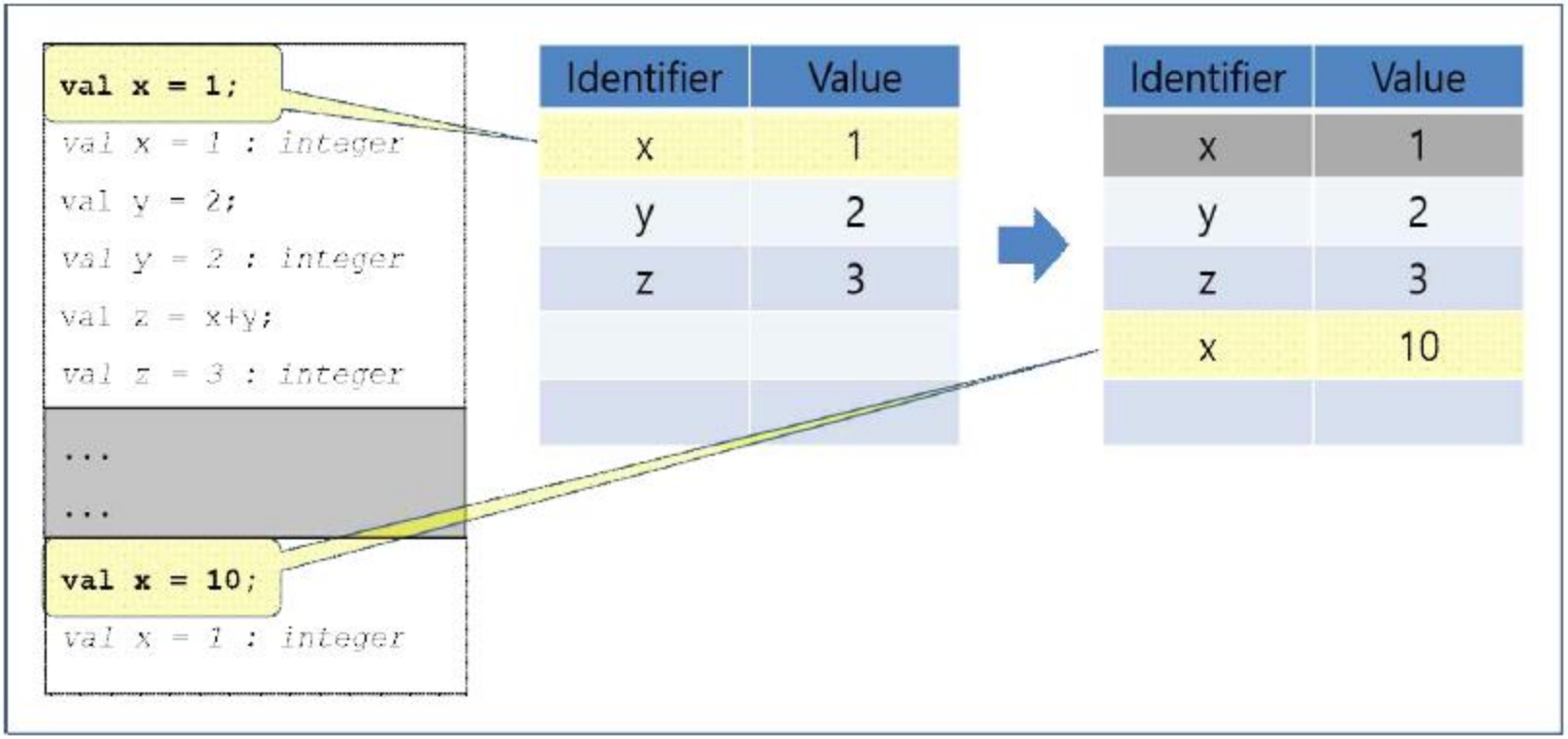
<그림 3-5> Ceil, Floor 함수

```
1  3 + floor(4.0);
2  val it = 7: int
```

<프로그램 3-14> Floor 함수

```
1  ord(#"a");
2  val it = 97 : int
3  chr(97);
4  val it = #"a" : char
5  str(#"a");
6  val it = "a" : string
```

<프로그램 3-15> Type Conversion



<그림 3-6> ML Environment

```
1  (1, 1.0, "one");
2  val it = (1,1.0,"one") : int * real * string
3  (1, (2,3), (4,(5,6)));
4  val it = (1,(2,3),(4,(5,6))) : int * (int*int) * (int * (int*int))
```

<프로그램 3-16> Tuple Example

```
1  #2(1, 1.0, "one");
2  val it = 1.0 : real
```

<프로그램 3-17> Tuple Operation

```
1  ["1", "2", "3"];
2  val it = ["1","2","3"] : string list
3  [(1,2.0), (3,4.0)];
4  val it = [(1,2.0),(3,4.0)] : (int * real) list
```

<프로그램 3-18> List Example

```
1  hd([1,2,3]);
2  val it = 1 : int
3  tl([1,2,3]);
4  val it = [2, 3] : int list
5  tl [1];[218]
6  val it = [] : int list
7  1::[2,3];
8  val it = [1,2,3] : int list
9  [1]@[2,3];
10 val it = [1,2,3] : int list
11 explode("abcd")
12 val it = [#"a",#"b",#"c",#"d"] : char list
13 implode( [#"a", #"b", #"c", #"d" ])
14 val it = "abcd" : string
```

<프로그램 3-19> List Operations

```
1   hd([1,2,3]);
2   val it = 1 : int
3   tl([1,2,3]);
4   val it = [2, 3] : int list
5   tl [1];218)
6   val it = [] : int list
7   1::[2,3];
8   val it = [1,2,3] : int list
9   [1]@[2,3];
10  val it = [1,2,3] : int list
11  explode("abcd")
12  val it = [#"a",#"b",#"c",#"d"] : char list
13  implode( [#"a", #"b", #"c", #"d" ])
14  val it = "abcd" : string
```

<프로그램 3-19> List Operations