

CES1017 Programming Fundamentals

# Week 2: Coding, the First Step

Instructor: Eunil Park (pa1324@hanyang.ac.kr)



**HANYANG UNIVERSITY**



## Last Class

1. What is “Interpreter”?
2. Create your own python environment.
3. Run python command!

## Today's Schedule

1. Check-up
  2. Arithmetic Expression
  3. String
  4. Boolean Expression
  5. Output
  6. Editor
  7. Variable and Assignment
  8. Input
  9. Comments
  10. Bug
  11. Exercise
- Expression

## Goal and Course Info

- What is “Python Interpreter”?  
What can we do using “Python Interpreter”?
- Did you try to install python 3.6?
- Did you run python command?



## Run Python!

- Command-line terminal -> “python” or “python3”

: Shell =====

>>> 365      < 값 (value)

>>> 3\*4+2      < 표현식 (expression)

>>>

## 02. Arithmetic Expression (수식)



HANYANG  
UNIVERSITY

### Numeral (수)

```
>>> 0
0
>>> 3
3
>>> 3655
3655
>>> +16
16
>>> -23
-23
>>>
```

[integer]

정수

```
>>> 2.5
2.5
>>> 0.0025e3
2.5
>>> 250e-2
2.5
>>> -3.14
-3.14
>>> |
```

[floating-point number]

부동소수점수

## 02. Arithmetic Expression (수식)



```
RESTART: Shell =====
```

```
=====
```

```
>>> -3
```

```
-3
```

```
>>> 3+5
```

```
8
```

```
>>> 6/3
```

```
2.0
```

```
>>> 7/3
```

```
2.3333333333333333
```

```
>>> 7%3
```

```
>>> 2**5
```

```
>>> |
```

### Binary Operator (이항연산자)

- Infix (중위) 표기법 사용
  - Operand – Operator – Operand  
(피연산자) (연산자) (피연산자)  
Ex)  $3 + 5$ ,  $24 * 365$
- Representative binary operators

+	더하기	//	
-	빼기	%	
*	곱하기	**	
/	나누기		

### Unary Operator (단항연산자)

- Prefix (전위) 표기법 사용

## 02. Arithmetic Expression (수식)



HANYANG  
UNIVERSITY

### 부동소수점 오차

```
>>>
```

```
>>>
```

```
>>>
```

```
>>> 0.1
```

```
0.1
```

```
>>> 0.1*0.1
```



## 02. Arithmetic Expression (수식)



### Precedence (우선순위)

- 연산자 우선순위: 수학과 거의 유사함

가장높음	**	
높음	-	부호바꾸기
낮음	*	
	/	
	//	
	%	
가장낮음	+	
	-	
		빼기

- 우선순위를 바꾸고 싶다면!?
  - 결합순서
    - 좌결합 (left associative) : 왼쪽부터 연산
    - 우결합 (right associative) : 오른쪽부터 연산
- Ex) 2-3-4, 좌결합결과 -5, 우결합결과 3

```
Python 3.6.0 Shell
File Edit Shell Debug Options Window Help
>>>
>>>
>>>
>>>
>>>
=====
RESTART: Shell =====
=====
>>> 2 + 3 * 4
14
>>> (2 + 3) * 4
14
>>> 3 + 2**4
17
>>>
```

## 02. Arithmetic Expression (수식)



### Type Setting (타입 변환)

- 수의 타입 변환 (형태 변환)
- 타입을 변환해주는 함수 활용 (int, float 등)
- 타입을 확인하고 싶을 때!?: type() 활용

RESTART: Shell =====

=====

>>> int(3.84)      < 소수점 이하 버림

>>> float(3)

>>> |

## String Expression

- 문자를 일렬로 나열해 놓은 것
- 문자열 표현식(string expression): 큰따옴표(" ") 혹은 작은따옴표(' ') 문자를 양쪽 끝에 붙여서 표현

```
===== RESTART
: Shell =====
>>> "Computer Science"
'Computer Science'
>>> 'Computer Science'
'Computer Science'
>>> "Computer Science
>>> Computer Science
>>>
```

## String Concatenation

- 문자열 붙이기(String Concatenation): “+” 연산자 이용

```
>>>
```

```
===== RESTART
```

```
: Shell =====
```

```
>>> 'Computer' + "Science"
```

```
'ComputerScience'
```

```
>>> "Computer" + " " + 'Science'
```

< “ ” 빈칸 하나로 구성된 문자열

```
>>> 'Computer' 'Science'
```

< 일렬로 나열해도 문자열이 붙음

```
>>> "Computer" " " 'Science'
```

```
'Computer Science'
```

```
>>>
```

```
>>>
```

## String Concatenation

- 빈 문자열(Empty String): 문자가 하나도 없는 문자열, "" 또는 ""

```
===== RESTART
: Shell =====
>>> ""
''
>>> ""
''
>>> 'Computer' + "" + 'Science'
'ComputerScience'
>>> |
```

# 03. String (문자열)



## Type Setting

- ‘Type’ 변환 함수들: str(), int(), float(), etc.

```
===== RESTART: Shell
```

```
=====
```

```
>>> "World Cup" + 2017
```



```
>>>
```

< str(2017) → “2017”

- str(2017): 2017의 타입을 string으로 바꿔주세요!

# 03. String (문자열)



## Type Setting (예)

```
====  
>>> str(3.14)  
'3.14'  
>>> str(0.2e3)  
'200.0'  
>>> int("123")  
123  
>>> int("3+4")
```

```
>>> int("3.14")
```

## 03. String (문자열)



### 반복 붙이기

- <문자열> \* n은 <문자열>을 n번 연속 붙인다는 의미

```
===== RESTART: Shell
```

```
=====
```

```
>>> "Pooh"*5
```

```
'PoohPoohPoohPoohPooh'
```

```
>>> "Pooh"*0
```

```
>>> "Pooh"*-2
```

```
>>> |
```



# 03. String (문자열)



## 구분문자와 특수문자

- 구분문자(delimiter): 독립된 문자열을 구분해주는 문자, e.g., “”

```
===== RESTART: Shell
```

```
=====
```

```
>>> 'Eunil's Dog'
```

```
>>> |
```

# 04. Boolean Expression (논리식)



## 논리값, 논리연산자

- 논리값(Boolean)
  - 참(True)과 거짓(False)으로 구성

- 논리연산자(Boolean Operator)

- 논리곱(and), 논리합(or)
  - 이항연산자(중위 표기법 사용)
- 논리역(not)
  - 단항연산자(전위 표기법 사용)

- 우선순위

가장 높음	not
높음	and
낮음	or

```
>>> True
True
>>> False
False
>>> True and True
True
>>> True and False
False
>>> False and True
False
>>> False and False
False
>>> True or True
True
>>> False or True
True
>>> False or False
False
>>> not True
False
>>> not False
True
>>> not (False or ((not False) and True))
True
>>> (not False) or ((not False) and True)
True
>>> not False or not False and True
True
>>>
```

# 04. Boolean Expression (논리식)



## 계산 순서

- 이 항연산자(논리곱/논리합)의 경우
    - 왼쪽 피연산자 먼저 계산
  - 단축계산 (short-circuit evaluation)
    - 계산결과가 명확하면  
필요 없는 계산 생략
- Ex) False and ?? > False  
True or ?? > True

```
>>> def loop():loop()
```

```
>>> True and loop()
```

```
>>> False and loop()
```

```
>>> True or loop()
```

```
>>> False or loop()
```

```
>>>|
```

# 04. Boolean Expression (논리식)



## 비교논리식

- 비교연산
  - 두 값을 비교하여 같거나 다름, 혹은 크거나 작음을 판명
- 비교연산자(Comparison Operator)
  - 같다(==), 다르다(!=), 크다(>), 작다(<), 크거나같다(>=), 크거나작다(<=) 등

11 == 11	True != True	"ERICA" < "ERICA"
11 != 11	True < False	3 == "three"
11 <= 11	False < True	3 >= "three"
11 < 11	True <= True	3 < "three"
12 <= 23 <= 34	False < False	True == "True"
12 <= 23 >= 14	"ERICA" == "ERICA"	False == 0
3 == 3.0	"ERICA" == "erica"	True == 1
3 == 3.1	"ERICA" != "erica"	True == 3
True == False	"ERICA" <= "Hanyang"	
False == False	"ERICA" <= "ERICA"	

[연습문제] ASCII code

```
RESTART: Shell =====
=====
>>> 11 == 11
True
>>> 11 >= 11
True
>>> 11 != 11
False
>>> "ERICA" == "ERICA"
True
>>> "ERICA" <= "Hanyang"
False
>>> "ERICA" <= "Hanya"
False
>>> 3 == "three"
False
>>> False == 0
True
>>>
```

# 05. Output (출력)



## Standard Output

- 표준출력(Standard Output): Python 실행창에 결과 출력
  - `print(<표현식>)`
  - `print(<표현식>, ..., <표현식>)`

```
===== RESTART: Shell =====
>>> print(3)
3
>>> print(3.64)
3.64
>>> print ("Welcome to \nHanyang University\nERICA Campus\nCollege of Computing!")
Welcome to
Hanyang University
ERICA Campus
College of Computing!
>>> "Welcome to \nHanyang University\nERICA Campus\nCollege of Computing!"
>>>
```

\t : 탭  
\n : 다음줄

## Editor

- Python Program = a sequence of commands (명령문)
  - 여러 명령문으로 구성된 program 작성 및 실행  
→ Editor에서 program 작성 (.py 파일) > python으로 실행  
예) a.py 작성 -> “python a.py”로 실행
- Editors
  - Vim: (Linux and MacOS has it by default) - <http://www.vim.org>
  - Emacs : <https://www.gnu.org/software/emacs/>
  - Sublime Text : <https://www.sublimetext.com/>
  - Atom : <https://atom.io/>

## Variable and Assignment

- Variable (변수): 메모리(값 보관 장소)의 “이름”
- Assignment (지정): 표현식을 계산해서 변수에 저장하는 작업

```
=====
```

```
>>> width = 3
```

```
>>> height = width + 2
```

```
>>> print(width, height)
```

```
3 5
```

```
>>> area = width * height / 2.0
```

```
>>> print(area)
```

```
7.5
```

```
>>> height = height - 1 < 변수 “height”
```

```
>>> print(width, height)   새로 4로 지정
```

```
3 4
```

```
>>> print(area)
```

```
>>> pooh
```

## Naming Rules

- 문자(a-z, A-Z), 숫자(0-9), 아래줄(\_)의 조합으로만 만들어야 함  
예) erica17 (O), erica\_17 (O), erica@2 (X)
- 변수 작명에 고려할 사항들
  - 값의 성격을 잘 대변해주는 이름을 고를 것
  - 일관성을 유지할 것
  - 관습을 따를 것 (일반 변수는 소문자로 시작)
  - 너무 길게 만들지 말 것



# 07. Variable and Assignment



## 변수 사용 예

- `print(5, "일을 분으로 따지면", 5*24*60, "분이다.")`
- `print(10, "일을 분으로 따지면", 7*24*60, "분이다.")`



```
: Shell =====  
>>> c1, c2 = "일을 분으로 따지면", "분이다."  
>>> day = 5  
>>> minute = day * 24 * 60  
>>> print(day, c1, minute, c2)  
5 일을 분으로 따지면 7200 분이다.  
>>> day = 10  
>>> print(day, c1, minute, c2)
```

## Standard Input (표준입력)

- 입력함수 input() 사용
- Input(<expression>)으로 사용 가능



: Shell =====

```
>>> input()
```

```
3
```

```
'3'
```

```
>>> x = input("입력?")
```

```
입력?3
```

```
>>> print(x)
```

```
3
```

```
>>> |
```

## Comments (주석)

- Comments (주석) – 실행이 안되는 텍스트
  - - Python에서는 “#”로 표시
- 코드 관리 및 가독성 증진을 위해 주석이용 > 다양한 세부 정보를 기록!
- 프로그램 시작부분
  - 프로그램의 이름, 간단한 설명, 입출력 명세, 작성자, 작성일, 버전, 수정일 등을 명시해두면 좋음 (팀프로젝트 시 활용)
- 코드를 이해하기 쉽게 보충 설명을 붙여두는 것도 권장

예)

```
# title: 동전합산 서비스
# problem: 가지고 있는 동전의 총액 계산
# input: 각 동전의 개수
# output: 총액
# author: 한진영
# date: 2017년 3월 14일
# version: 1.0
```

## Bug

- Bug: 프로그램 안에 있는 오류
- Debugging: 오류를 찾아 수정하는 작업
- Error의 대표적인 종류
  1. 구문오류(syntax error) 또는 문법오류(grammar error)
    - 문법에 맞지 않음
  2. 실행오류(run-time error)
    - 실행 중 비정상적으로 생기는 오류
  3. 타입오류(type error)
    - 연산자와 피연산자들 사이에 타입이 맞지 않아 발생하는 실행오류

예: `apple = "1" + 2`
  4. 값오류(value error)
    - 맞지 않는 값을 사용하는 경우 발생하는 실행오류

예: `banana = int("3.14")`
  5. 나누기0오류(zero division error)
    - 0으로 나누면 발생하는 실행오류

**안전한 코딩 습관이 중요!**  
**(Secure Coding)**

## 프로그램 작성 방법

1. 문제와 입력, 출력의 정의
2. 문제를 푸는 알고리즘 설계
  - 알고리즘(algorithm): 문제를 풀어 해답을 얻는 절차
3. 설계한 알고리즘을 기반으로 프로그램 작성
  - Python 프로그램 작성
4. 실행 검사(test)하면서 프로그램 보수
  - Python 실행기로 프로그램 실행

# 11. Exercise



## 동전 총액 계산하기

# 사용자 입력 받기

```
print("동전 합산 서비스에 오심을 환영합니다.")
```

```
print("음수는 입력하지 마세요.")
```

```
coin500 =
```

```
coin100 =
```

```
coin50 =
```

```
coin10 =
```

# 계산

```
total =
```

# 결과 출력

```
print("\n손님의 동전은 총", total, "원 입니다.")
```

## Summary

1. Check-up
  2. Arithmetic Expression
  3. String
  4. Boolean Expression
  5. Output
  6. Editor
  7. Variable and Assignment
  8. Input
  9. Comments
  10. Bug
  11. Exercise
- Expression

# Thanks

Week 2: Coding, the First Step

Instructor: Eunil Park (pa1324@hanyang.ac.kr)

