

Step 2: Coding, the First Step

Instructor: Eunil Park (eunilpark@skku.edu)



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.x?
- Did you run python command?



Run Python!

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

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

```
>>> 365      < 값 (value)
```

```
365
```

```
>>> 3*4+2    < 표현식 (expression)
```

```
14
```

```
>>>
```

02. Arithmetic Expression (수식)

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  
1  
>>> 2**5  
32  
>>> |
```

Binary Operator (이항연산자)

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

+	더하기	//	몫 구하기
-	빼기	%	나머지구하기
*	곱하기	**	지수 승
/	나누기		

Unary Operator (단항연산자)

- Prefix (전위) 표기법 사용
- Minus sign (-, 부호바꾸기)
 - Representative unary operator (단항연산자)
 - Operator (연산자) – Operand (피연산자)
Ex) -3

02. Arithmetic Expression (수식)

부동소수점 오차

```
>>>  
>>>  
>>>  
>>> 0.1  
0.1  
>>> 0.1*0.1  
0.010000000000000002 < Why?  
>>> |
```

- 컴퓨터
= 0과 1로 이루어진 계산기!
- 컴퓨터로 곱셈을 하려면?

0.1

컴퓨터 입력!

이진수로 변환

!?!?

1. 0.1을 이진수로 정확하게 표현할 수 없음
2. 정확히 표현 불가능 > 근사치로 표현!
3. 곱셈 수행
4. 오차 발생!

예시)

1. 십진수 0.125 입력!
2. $1/10 + 2/100 + 5/1000 = 125/1000$
 $= 1/8 = 0/2 + 0/4 + 1/8$
3. 이진수 0.001로 표현!

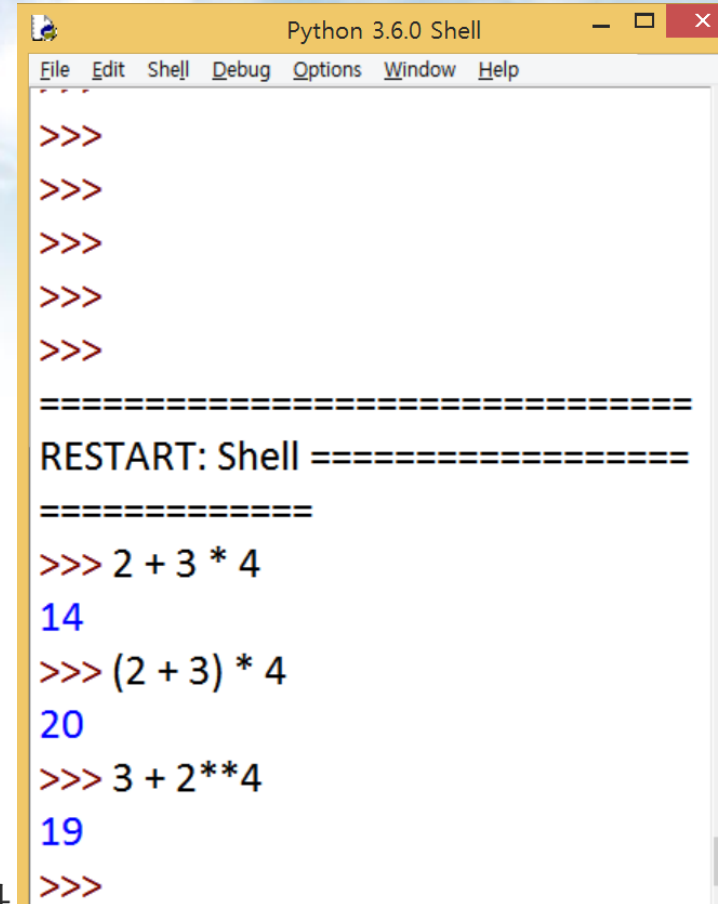
02. Arithmetic Expression (수식)

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
20
>>> 3 + 2**4
19
>>>
```

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
```

```
3
```

```
>>> float(3)
```

```
3.0
```

```
>>> |
```

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
SyntaxError: EOL while scanning string literal
>>> Computer Science
SyntaxError: invalid syntax
>>>
```

String Concatenation

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

```
>>>
```

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

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

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

```
'ComputerScience'
```

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

```
'Computer Science'
```

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

```
'ComputerScience'
```

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

```
'Computer Science'
```

```
>>>
```

< “ ” 빈칸 하나로 구성된 문자열

< 일렬로 나열해도 문자열이 붙음

String Concatenation

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

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

>>> ""
''

>>> 'Computer' + "" + 'Science'
'ComputerScience'
>>> |
```

Type Setting

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

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

```
=====
```

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

```
Traceback (most recent call last):
```

```
File "<pyshell#0>", line 1, in <module>
```

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

```
TypeError: must be str, not int
```

```
>>> "World Cup" + str(2017)
```

< str(2017) → “2017”

```
'World Cup2017'
```

```
>>>
```

- 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")  
Traceback (most recent call last):  
  File "<pyshell#5>", line 1, in <module>  
    int("3+4")  
ValueError: invalid literal for int() with base 10: '3+4'  
>>> float("3.12")  
3.12  
>>> float("0.3e2")  
30.0
```

```
>>> int("3.14")  
Traceback (most recent call last):  
  File "<pyshell#8>", line 1, in <module>  
    int("3.14")  
ValueError: invalid literal for int() with base 10: '3.14'  
>>> int("0.3e2")  
Traceback (most recent call last):  
  File "<pyshell#9>", line 1, in <module>  
    int("0.3e2")  
ValueError: invalid literal for int() with base 10: '0.3e2'  
>>> float("3")  
3.0  
>>> |
```

03. String (문자열)

반복 붙이기

- <문자열> * n은 <문자열>을 n번 연속 붙인다는 의미

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

```
>>> "Pooh"*5
'PoohPoohPoohPoohPooh'
```

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

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

```
>>> |
```


구분문자와 특수문자

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

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

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

```
SyntaxError: invalid syntax
```

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

```
"Eunil's Dog"    < 해결책 1
```

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

```
"Eunil's Dog"    < 해결책 2, 역슬래쉬(backslash, \) 사용
```

```
>>> |
```

04. Boolean Expression (논리식)

논리값, 논리연산자

- 논리값(Boolean)
 - 참(True)과 거짓(False)으로 구성
- 논리연산자(Boolean Operator)
 - 1) 논리곱(and), 논리합(or)
 - 이항연산자(중위 표기법 사용)
 - 2) 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))
False
>>> (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()
Traceback (most recent call last):
  File "<pyshell#49>", line 1, in <module>
    True and loop()
  File "<pyshell#48>", line 1, in loop
    def loop():loop()
  File "<pyshell#48>", line 1, in loop
    def loop():loop()
  File "<pyshell#48>", line 1, in loop
    def loop():loop()
  [Previous line repeated 990 more times]
RecursionError: maximum recursion depth exceeded
>>> False and loop()
False
>>> True or loop()
True
>>> False or loop()
Traceback (most recent call last):
  File "<pyshell#52>", line 1, in <module>
    False or loop()
  File "<pyshell#48>", line 1, in loop
    def loop():loop()
  File "<pyshell#48>", line 1, in loop
    def loop():loop()
  File "<pyshell#48>", line 1, in loop
    def loop():loop()
  [Previous line repeated 990 more times]
RecursionError: maximum recursion depth exceeded
>>>|
```

비교논리식

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

```
>>> 11==11
True
>>> 11>=11
True
>>> 11!=11
False
>>> "SKKU" == "SKKU"
True
>>> "SKKU" <= "AAI"
False
>>> "SKKU" <= "aai"
True
>>> 3 == "three"
False
>>> False == 0
True
>>>
```

Standard Output

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

```
>>> print(3)
3
>>> print(3.64)
3.64
>>> print("Welcome to \nSungkyunkwan University\nAAI Department!")
Welcome to
Sungkyunkwan University
AAI Department!
>>> "Welcome to\nSKKU\nAAI Department!"
'Welcome to\nSKKU\nAAI Department!'
>>>
```

`\t` : 탭
`\n` : 다음줄

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)
```

```
7.5
```

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

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

```
6.0
```

```
>>> pooh
```

```
Traceback (most recent call last):
```

```
File "<pyshell#76>", line 1, in <module>
```

```
pooh
```

```
NameError: name 'pooh' is not defined
```

```
>>>
```

```
< 변수는 지정해야 쓸 수 있음
```

Naming Rules

- 문자(a-z, A-Z), 숫자(0-9), 아래줄(_)의 조합으로만 만들어야 함
예) aai20 (O), aai_17 (O), aai@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)
```

```
10 일을 분으로 따지면 7200 분이다. >>>
```

```
>>> day = 10
```

```
>>> minute = day * 24 * 60
```

```
>>> print(day, c1, minute, c2)
```

```
10 일을 분으로 따지면 14400 분이다.
```

```
>>>
```


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

동전 총액 계산하기

사용자 입력 받기

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

```
print("음수는 입력하지 마세요.")
```

```
coin500 = int(input("500원짜리는 몇 개 입니까?"))
```

```
coin100 = int(input("100원짜리는 몇 개 입니까?"))
```

```
coin50 = int(input("50원짜리는 몇 개 입니까?"))
```

```
coin10 = int(input("10원짜리는 몇 개 입니까?"))
```

계산

```
total = 500 * coin500 + 100 * coin100 + 50 * coin50 + 10 * coin10
```

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

Step 2: Coding, the First Step

Instructor: Eunil Park (eunilpark@skku.edu)

