

# Python Syntax

ESM2017-41

객체지향프로그래밍 이론 및 실습

SKKU 시스템경영공학과

조영일

# Python Syntax

- Syntax : 프로그래밍 언어의 문법 또는 구조에 관한 정의

# Python Design Philosophy

“There should be one—and preferably only one—obvious way to do it”  
from "The Zen of Python"

- 사람이 읽기 좋은 형태로 설계
- 이해하기 어려운 각종 기호들 대신 영어 단어를 사용하는 문법
- 세미콜론 등을 사용하지 않고 들여쓰기를 통해 정돈된 코드 레이아웃 사용

# Python Syntax – 명령문 종결

- 컴퓨터 프로그램은 명령문(Control Statement)이 모여서 하나의 전체적인 프로그램을 구성
- C를 비롯한 대부분의 언어의 경우 하나의 명령문 단위를 종결하기 위해 세미콜론(;)을 사용
- Python의 경우 별도의 세미콜론 없이 개행하는 것으로 명령문의 종결을 표시

===Java===

```
public class Main{  
    public static void main(String[] args)  
    {  
        System.out.println("Hello World");  
    }  
}
```

===Python===

```
print( "Hello World")
```

# Python Syntax – Indentation

- Control Flow Block : 함수, 제어문 등의 내용을 구성하는 코드 뭉치

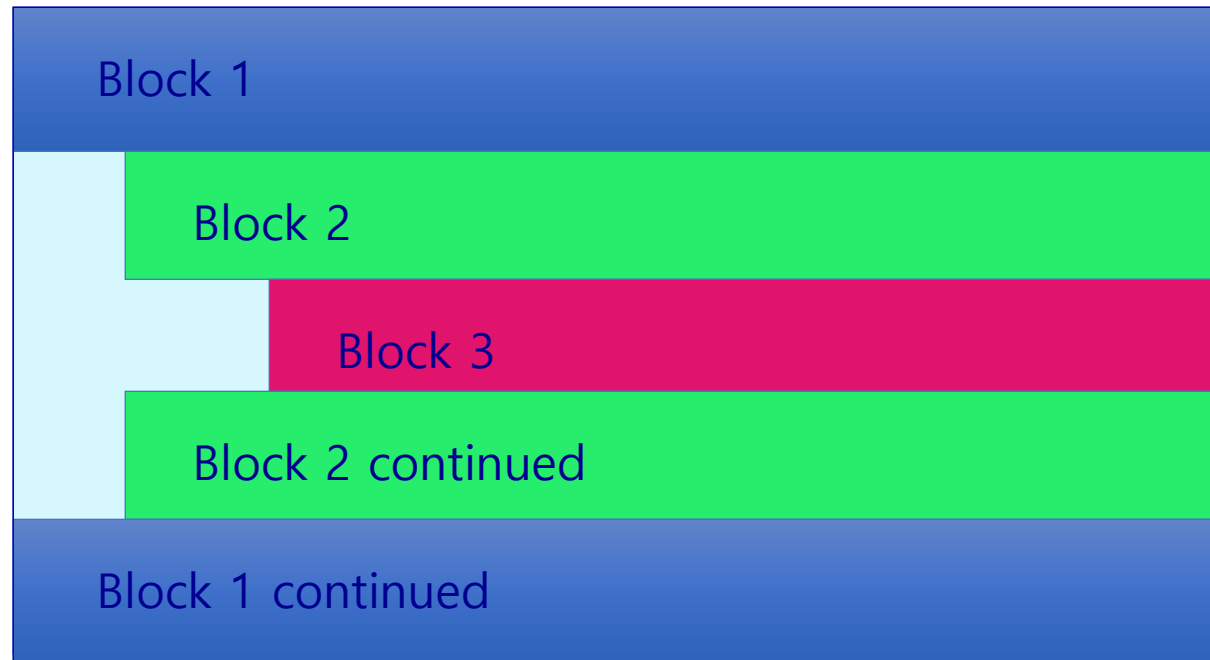
# Python Syntax – Indentation

- Python 은 들여쓰기를 통해 Control Flow Block 을 구분함
- 타 언어에서 들여쓰기는 단지 코드의 미관을 위해서 사용하는 경우가 많지만, Python에선 문법적인 강제 사항임
- 들여쓰기는 \t(탭문자열)이 아닌 띄어쓰기 4개로 구성하는 것이 표준임

```
void foo(int x)
{
    if (x == 0) {
        bar();
        baz();
    } else {
        qux(x);
        foo(x - 1);
    }
}
```

```
def foo(x):
    if x == 0:
        bar()
        baz()
    else:
        qux(x)
        foo(x - 1)
```

# Python Syntax – Indentation



# Python Syntax – Indentation

```
# Do.  
if 10 > 9:  
    print("10 is bigger than 9")  
  
# Do NOT. (IndentationError: unexpected indent)  
if 10 > 9:  
print("10 is bigger than 9")  
  
# Do NOT. (Syntax Error)  
if 10 > 9 {  
    print("10 is bigger than 9")  
}
```



# Python Syntax – Variables

- Variables(변수) : 프로그램 내에서 변할 수 있는 값.
- Python은 별도의 변수 선언 없이, 변수에 값을 할당하는 즉시 변수가 생성됨

```
school_name = "SKKU"  
department_name = "Industrial Engineering"  
grade = 3
```

# Python Syntax – Variables

- 변수 이름 규칙
  - 변수 이름은 문자열이나 underscore(\_)로 시작되어야 함
  - 변수 이름은 숫자로 시작 할 수 없음
  - 변수 이름은 알파벳, 숫자로 구성해야 함(Python 3부터는 한글도 가능하지만 권장하지 않음)
  - 변수 이름은 공백이나 특수 문자를 사용할 수 없음
  - 변수 이름은 대소문자를 구분함

# Python Syntax – Reserved Keywords

- Reserved Keywords : Python 문법 상 자체적으로 사용되는 영어단어, 변수명에 사용 불가함

and	assert	in
del	else	raise
from	if	continue
not	pass	finally
while	yield	is
as	break	return
elif	except	def
global	import	for
or	print	lambda
with	class	try
exec		

# Python Syntax – Variables

```
# 변수 할당
x = "School"

# 여러개의 변수를 동시에 할당 할 수 있음
a, b, c = 1, 2, 3

# 여러개의 변수에 동일한 값을 할당 할 수 있음
d = e = f = 4

# Do Not (Syntax Error)
1_school = 3
!_school = 4
if = 4

# 변수의 값을 출력
print(a)
print(b)
print(c)
```

# Python Syntax – Comments

- Comments(주석) : 작성된 코드의 이해를 돕기 위해 사람이 읽을 수 있는 설명을 다는 것

# Python Syntax – Comments

- Single line comment : “#”으로 시작

```
# This is a single line comment.  
a = 1  
  
b = 2 # 코드 끝에도 주석을 달 수 있다.
```

# Python Syntax – Comments

- Multi line comment : “””/””” 으로 시작과 끝 표현

```
"""  
This is a comment  
written in  
more than just one line  
"""  
print("Hello, World!")
```

# Python Syntax – Numbers

- Python 에서 숫자는 3가지 방식으로 표현 가능
  - int : 정수형
  - float : 소숫점형
  - complex : 복소수

```
x = 1      # int  
y = 2.8    # float  
z = 1j     # complex
```



# Python Syntax – Arithmetic Operators

- 산술 연산자 : Numbers 자료형에 대한 숫자 계산

Operator	Description	Example
+	더하기	$a + b = 30$
-	빼기	$a - b = -10$
*	곱하기	$a * b = 200$
/	나누기	$b / a = 2.0$
%	나머지	$b \% a = 0$
**	제곱	$a ** c = 1000$
//	몫	$a // c = 3$

`** a = 10, b = 20, c = 3`

# Python Syntax – Arithmetic Operators

- 산술 연산자 : Numbers 자료형에 대한 숫자 계산

```
a = 1 + 1    # 더하기 / a = 2
a = 1 - 1    # 빼기 / a = 0
a = 3 * 4    # 곱하기 / a = 12
a = 3 / 2    # 나누기 / a = 1.5
a = 5 // 2   # 몫 / a = 2
a = 5 % 2    # 나머지 / a = 1
a = 5 ** 2   # 제곱 / a = 25
```

# Python Syntax – Assignment Operators

- 대입 연산자 : 연산을 수행 하고 왼쪽 변수에 값을 다시 대입하는 연산자

Operator	Description	Example	Equivalent Statement
+=	더해서 대입	a += b	a = a + b
-=	빼서 대입	a -= b	a = a - b
*=	곱해서 대입	a *= b	a = a * b
/=	나눠서 대입	a /= b	a = a / b
%=	나머지 대입	a %= b	a = a % b
**=	제곱 대입	a **= b	a = a ** b
//=	몫 대입	a //= b	a = a // b

# Python Syntax – Booleans

- 논리 자료형 : 참과 거짓을 표현

```
True
False
not True == False
True and True == True
True or False == True

3 == 5 # => False
10 * 10 == 100 # => True
```

# Python Syntax – Strings

- 문자열

```
school_name = 'SKKU'  
department_name = "Industrial Engineering"  
  
school_name + " " + department_name # => SKKU Industrial Engineering
```

# Python Syntax – String Indexing

```
S = "Hello World"
```

H	e	l	l	o		W	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10

- `S[0] == 'H'`
- `S[1] == 'e'`
- `S[5] == ' '`
- `S[10] == 'd'`

# Python Syntax – String Negative Indexing

```
S = "Hello World"
```

H	e	l	l	o		W	o	r	l	d
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- `S[-1] == 'd'`
- `S[-5] == 'W'`
- `S[-11] == 'H'`

# Python Syntax – String Slicing

```
S = "Hello World"
```

H	e	l	l	o		W	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10

- `S[0:2] == 'He'`
- `S[3:7] == 'lo W'`
- `S[9:11] == 'ld'`
- `S[10] == 'd'`



# Python Syntax – String Slicing

```
S = "Hello World"
```

H	e	l	l	o		W	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10

- `S[:2] == 'He'`
- `S[6:] == 'World'`

# Python Syntax – String Slicing

```
S = "Hello World"
```

H	e	l	l	o		W	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10

- `S[6:-1]` == `'Worl'`
- `S[6:-2]` == `'Wor'`
- `S[-4:-2]` == `'or'`

# Python Syntax – None

- None은 해당 변수에 값이 없음을 의미함
- 값이 할당되지 않은 변수가 필요한 경우 사용

```
a = None  
a is None == True
```

# Python Type Casting

- Python은 변수를 선언할 때 자료형을 지정하지 않는다.
- 할당되는 값에 따라 자료형이 저절로 지정됨

```
a = 1 # Integer  
b = 2 # Integer
```

```
print(a + b) # 3
```

```
a = "1" # String  
b = "2" # String
```

```
print(a + b) # "12"
```

# Python Type Casting

- 서로 연산이 불가능한 자료형의 연산을 시도할 경우 “TypeError”

```
a = 1 # Integer  
b = "2" # String  
  
print(a + b) # TypeError
```

# Python Type Casting

- 형 변환(Type Casting) : 어떤 자료형을 다른 자료형으로 변환 시키는 과정

```
a = 1 # Integer  
b = "2" # String  
  
print(a + int(b)) # 3
```

# Python Type Casting

- 형 변환 클래스
  - `str()` : 객체를 문자형으로 변환
  - `int()` : 객체를 정수형으로 변환
  - `float()` : 객체를 실수형으로 변환
- 정수(integer)를 실수(float)로, 실수(float)를 정수(integer)로 변환할 수 있다.
- 숫자(정수와 실수)를 문자열로 변환할 수 있지만, 문자열은 숫자로 변환 가능한 것만 숫자로 변환할 수 있다.
- 만약 자료형이 서로 변환 가능한 형태가 아니면 오류가 발생한다.

# Console I/O(Input/Output)

- The original meaning of Console



# Ancient Computer:

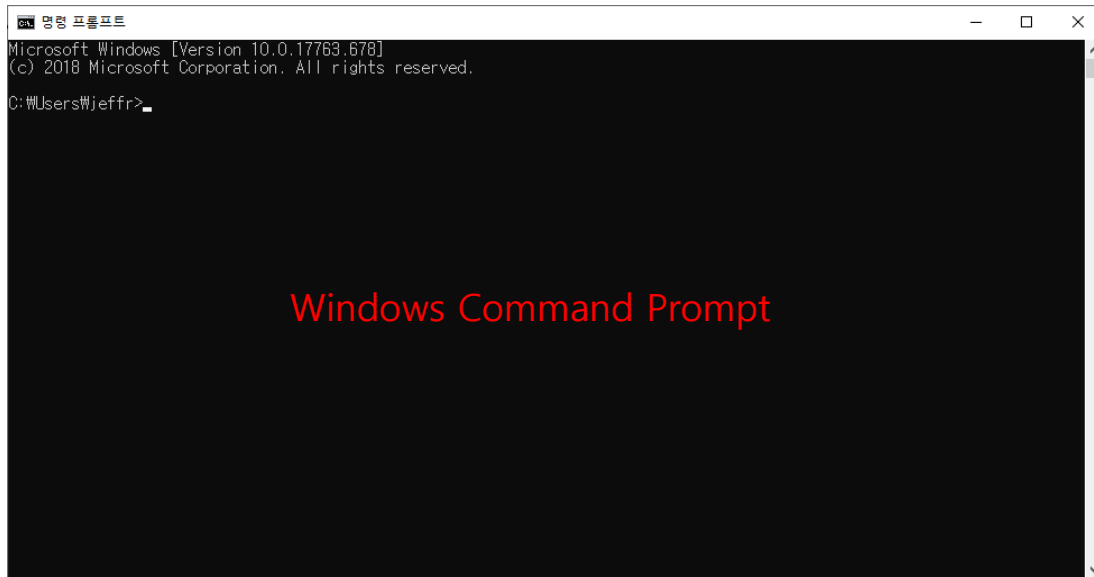
# NO Graphic User Interface

## Only text input / output



# Console I/O(Input/Output)

- Modern console

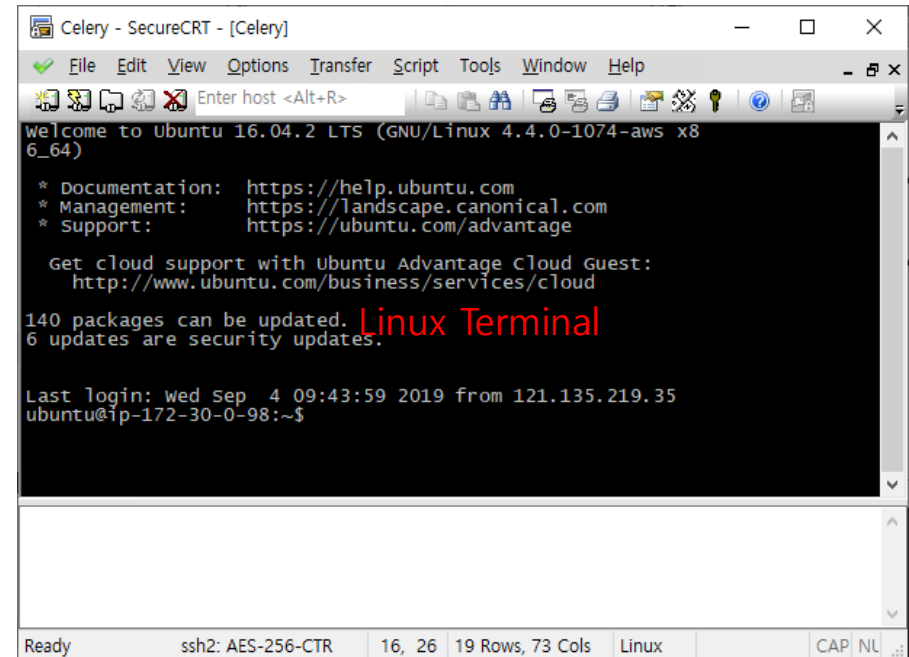


A screenshot of the Windows Command Prompt window. The title bar reads '명령 프롬프트'. The window content shows the standard Windows version and copyright information, followed by the command prompt 'C:\Users\jeffr>'. The text 'Windows Command Prompt' is overlaid in red at the bottom center.

```
Microsoft Windows [Version 10.0.17763.678]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\jeffr>
```

Windows Command Prompt



A screenshot of a Linux terminal window running within SecureCRT. The title bar is 'Celery - SecureCRT - [Celery]'. The terminal shows the Ubuntu 16.04.2 LTS login banner, system information, and login details. The text 'Linux Terminal' is overlaid in red at the bottom center.

```
Celery - SecureCRT - [Celery]
File Edit View Options Transfer Script Tools Window Help
Enter host <Alt+R>
welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-1074-aws x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

140 packages can be updated. 6 updates are security updates.

Last login: wed Sep  4 09:43:59 2019 from 121.135.219.35
ubuntu@ip-172-30-0-98:~$
```

Linux Terminal

# Console I/O(Input/Output)

- Console : (텍스트 기반으로) 사용자가 명령어를 입력하고 그 결과를 확인 하는 환경
- Console Input : 사용자로부터 입력을 받기 - `.input()`

```
>>> name = input("What is your name?")
```

- Console Output : Console에 원하는 값을 출력 - `.print()`

```
>>> name = input("What is your name?")  
What is your name? "Python"  
>>> print(name)  
"Python"
```

# Summary

- Control Flow Block & Indentation
  - Variables
  - Python Types & Basic Operation
  - Type Casting
  - Console I/O
- 
- Python REPL 이용하여 기본적인 Variable 선언, Operator 사용법 익히기