

충북대학교 SW 중심 대학 산업단

# Python 기반의 오픈소스 프로그래밍

박상수 강사

2020년 8월 20일



# 강의 목차

- **Python** 시작하기
- 기본 데이터 타입
- 제어: 조건문, 반복문

# 리스트 (List)

## ■ 리스트: 여러 요소들을 갖는 집합 (컬렉션)

- 요소: 숫자, 문자열

- 예) 가족 (어머니, 아버지, 본인, 동생)

```
>>> family = ['mother', 'father', 'gentleman', 'sexy lady']
```

- 리스트의 요소는 숫자, 문자, 빈 리스트 가능

```
a = []      # 빈 리스트  
a = ["AB", 10, False]
```

- 인덱싱: 리스트의 특정 한 요소만을 선택하기 위한 방법

- 첫번째 요소는 리스트 [0], 두번째 요소는 리스트 [1] 처럼 표현

```
a = ["AB", 10, False]  
x = a[1]           # a의 두번째 요소 읽기  
a[1] = "Test"      # a의 두번째 요소 변경  
y = a[-1]          # False  
print(a[1])
```

- 인덱스에 -1, -2 같은 음수 사용 가능 (-1: 현재 리스트의 마지막 요소, -2는 뒤에서 두번째 요소)

# 리스트 (List)

## ■ 리스트: 여러 요소들을 갖는 집합 (컬렉션)

- 요소: 숫자, 문자열

- 예) 가족 (어머니, 아버지, 본인, 동생)

```
>>> family = ['mother', 'father', 'gentleman', 'sexy lady']
```

- 리스트의 요소는 숫자, 문자, 빈 리스트 가능

```
a = []      # 빈 리스트  
a = ["AB", 10, False]
```

- 인덱싱: 리스트의 특정 한 요소만을 선택하기 위한 방법

- 첫번째 요소는 리스트 [0], 두번째 요소는 리스트 [1] 처럼 표현

```
a = ["AB", 10, False]  
x = a[1]           # a의 두번째 요소 읽기  
a[1] = "Test"      # a의 두번째 요소 변경  
y = a[-1]          # False  
print(a[1])
```

- 인덱스에 -1, -2 같은 음수 사용 가능 (-1: 현재 리스트의 마지막 요소, -2는 뒤에서 두번째 요소)

# Python 시작하기

## ■ Python 실행

- 윈도우에서 실행
  - [시작]-[프로그램]-[Python3.x]에서 "Python 3.x 콘솔프로그램" 또는 "IDLE (Python 3.x)" 프로그램 실행
- Mac/리눅스
  - "python" 입력 후 실행
  - Python 프로그램의 종류는 `exit()`

```
esoc@esoc-desktop:~/1.Workspace_1/constant/python$ python
Python 3.6.10 (default, Dec 19 2019, 23:04:32)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
esoc@esoc-desktop:~/1.Workspace_1/constant/python$
```

리눅스에서의 python 실행의 예

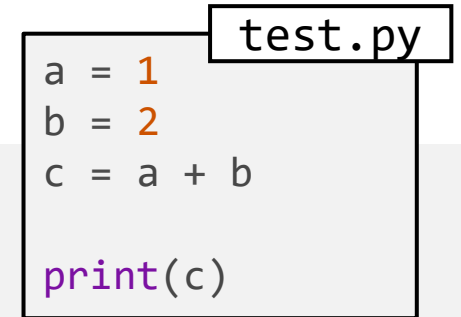
# Python 간단한 프로그램

## ■ Python 프로그램의 프로그래밍 과정

- 대화형 인터프리터를 사용하지 않고, 에디터를 사용하여 python 프로그램을 작성하고 실행
  - Notepad와 같은 간단한 에디터를 사용하여 다음의 코드를 작성하고 test.py로 저장
  - Python 프로그램은 확장자로 py 사용
  - Test.py를 실행하기 위해서는 python (python.exe) 뒤에 해당 python 파일을 명시

```
esoc@esoc-desktop:~/1.Workspace_1/constant/python$ python
Python 3.6.10 (default, Dec 19 2019, 23:04:32)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a=1
>>> b=2
>>> c=a+b
>>> print(c)
3
>>> exit()
```

```
esoc@esoc-desktop:~/1.Workspace_1/constant/python$ python test.py
3
```



```
test.py
a = 1
b = 2
c = a + b

print(c)
```

간단한 python 프로그램의 실행 예

# Python 코딩의 기초

## ■ 코딩블록 들여쓰기 (Indentation)

- Python은 코딩블록을 표시하기 위해 들여쓰기를 사용
  - C/C++, Java에서는 Bracket ({...})을 사용
  - 코딩블록을 시작하는 문장들의 끝에는 콜론 (:)을 사용, 내부의 코딩블록은 들여쓰기

```
x=1  
  
if x>0:           콜론  
    a=1  
    b=2           들여쓰기  
    c=a+b  
else:  
    a=-1  
    b=-2  
    c=a-b  
  
print(c)
```

코딩블록 들여쓰기의 예

# Python 표준 라이브러리 및 주석

## ■ Python은 많은 표준 라이브러리 제공

- 라이브러리 (Library): 프로그램 요소, 특정한 기능을 모듈화 한 대상 (함수)
  - 표준 라이브러리를 사용하기 위해서는 "import" 문을 사용
  - 예) math 라이브러리에 있는 sqrt() 함수를 사용하기 위해서는

## ■ 주석 (Comment): 프로그래밍에 있어 메모를 하는 목적

- Python에서 주석을 표기하기 위해 파운드 (#) 기호를 사용
- 주석은 라인의 처음에 올 수도 있고, 라인의 문장이 끝난 부분에 올 수도 있음

```
Python 3.6.10 (default, Dec 19 2019, 23:04:32)
[GCC 5.4.0 20160609] on linux ~
>>> import math
>>> n=math.sqrt(9.0)
>>> print(n)
3.0
```

표준 라이브러리 예

```
# 주석 1
# 주석 2
run(1)

# 주석 3
run(2) # 주석 4
```

주석 사용의 예



# 강의 목차

- Python 시작하기
- 기본 데이터 타입
- 제어: 조건문, 반복문

# Python 기본 데이터 타입

## ■ Python에서 사용되는 기본 데이터 타입 (Scalar Data Type)

- Integer type: 소수점을 갖지 않는 정수를 갖는 데이터 타입
- Boolean: True나 False만 갖는 타입
- None: 아무 데이터를 갖지 않는 것을 표현하는 타입
  - 데이터 변환 (캐스트 연산): 데이터를 특정 타입으로 변환하는 방법
  - 예) `int(3.5)`를 `print`를 사용하여 출력하면 어떤 값으로 표현될까 ?

타입	설명	표현 예
int	정수형 데이터	100, 0xFF (16진수), 0o56 (8진수)
float	소수점을 포함한 실수	b=10.25
bool	참과 거짓을 표현하는 불	c=True
None	Null과 같은 표현	d=None

Python 데이터 타입

```
int(3.5)      # 3
2e3           # 2000.0
float(1.6)    # 1.6
float("inf")  # 무한대
float("-inf") # -무한대
bool(0)       # False. 숫자에서 0만 False임,
bool(-1)      # True
bool("False") # True
a = None      # a는 None
a is None     # a가 None 이므로 True
```

주석 사용의 예

# Python 기본 데이터 타입 실습

## ■ 다음의 코드의 실행 결과는 어떤 값으로 출력이 되는지 확인하기

- 변수 a,b를 출력할 때 소수점 자리는 유지가 될까 ?
- **Print() 함수**
  - 변수에 값을 저장했을 때, 출력하고 검증 (디버깅)하는 용도로 사용하는 함수
  - Print(변수\_이름)을 사용하여 변수의 값을 출력
  - Print('문자열', 변수\_이름)을 사용하여 사용자가 이해 할수 있는 형태로 출력

```
a=int(3.7)
b=int(3.2)
c=float(1.7)
d=float(-1.7)
e=float("-inf") # ""가 없는 경우는 ?
f=bool(1)
```

```
print('result a is', a)
print('result b is', b)
print('result c is', c)
print('result d is', d)
print('result e is', e)
print('result f is', f)
```

실행결과

```
result a is 3
result b is 3
result c is 1.7
result d is -1.7
result e is -inf
result f is True
```

Python 기본 데이터 타입 실습

# Python 연산자 #1

## ■ 연산자: 데이터를 어떠한 방식으로 계산하는 지를 결정

- 산술연산자: 사칙연산자, 제곱 (\*\*), 나머지 (%), Modulus, 소수점 이하를 버리는 연산자 (//, Floor division)
- 비교연산자: 등호 (==), 같지 않음 (!=), 부등호 (<, >, <=, >=)
- 할당연산자: 변수에 값을 할당하기 위해 사용, 기본적으로 = (Equal sign)을 사용
- 논리연산자: and (모두 참인 경우 참), or (어느 한쪽만 참이면 참), not (참이면 거짓, 거짓이면 참)
- Bitwise 연산자: 비트 단위의 연산을 하는데 사용

```
# 산술연산자
5 % 2    # 1
5 // 2   # 2

# 비교연산자
if a != 1:
    print("1이 아님")

# 할당연산자
a = a * 10
a *= 10    # 위와 동일한 표현
```

```
# 논리연산자
x = True
y = False

if x and y:
    print("Yes")
else:
    print("No")

# Bitwise 연산자
a = 8      # 0000 1000
b = 11     # 0000 1011
c = a & b  # 0000 1000 (8)
d = a ^ b  # 0000 0011 (3)

print(c) print(d)
```

연산자 사용의 예 #1

## Python 연산자 #2

### ■ 연산자: 데이터를 어떠한 방식으로 계산하는 지를 결정

- 멤버십 연산자: 종류 (in, not in), 좌측 operand가 우측 컬렉션에 속해 있는지 아닌지를 체크
- Identity 연산자: 종류 (is, is not), 양쪽 operand가 동일한 object를 가리키는지 아닌지를 체크

```
# 멤버십 연산자
a = [1,2,3,4]
b = 3 in a      # True
print(b)

# Identity 연산자
a = "ABC"
b = a
print(a is b)  # True
```

연산자 사용의 예 #2

# Python 연산자 실습

## ■ 연산자 결과의 타입을 고려한 프로그래밍 필요

- 나눗셈을 하는 경우
  - 몫과 나머지 값은 반드시 정수 값으로 저장되어야 함
  - 정수 값으로 저장하기 위해서는 연산자의 결과가 특정 데이터 타입으로 저장 될 수 있도록 해야 함 !

```
a=6/4 # Quotient (몫)  
b=6%4 # Remainder (나머지)
```

```
print(a)  
if a!=1:  
    print("Wrong")  
else:  
    print("Right")
```

```
a=__(6/4) # Quotient (몫)  
b=6%4 # Remainder (나머지)
```

```
print(a)  
if a!=1:  
    print("Wrong")  
else:  
    print("Right")
```

Python 연산자 실습: 저장되는 데이터의 타입을 고려한 방법

# Python 문자열과 바이트

## ■ Python에서 문자열은 단일인용부호 (')나 이중인용부호 (")를 사용하여 표현

- `s='충북대'`, `s="충북대"`는 변수 `s`에 충북대 라는 문자열을 할당하는 동일한 표현
- 여러 라인에 걸쳐 있는 문자열을 표현하고 싶다면 3개의 인용부호를 사용 (`'''`)
- 문자열 포매팅
  - 일정한 포맷에 맞춰 문자열을 조합하는 방법
  - 템플릿 안에 대입 값이 들어갈 자리를 지정하고, 나중에 그 값을 채워 넣는 방식

```
p = "이름: %s 나이: %d" % ("김유신", 65)
print(p)
# 출력: 이름: 김유신 나이: 65
```

```
p = "X = %0.3f, Y = %10.2f" % (3.141592, 3.141592)
print(p)
# 출력: X = 3.142, Y =          3.14
```

```
esoc@esoc-desktop:~/1.Workspace_1/constant/python$ python print.py
이름: 김유신 나이: 65
X = 3.142, Y =          3.14
```

Python 포매팅의 예

# Python 문자열과 바이트 실습

## ■ 문자열 포매팅

### ■ 예시) "답: %s" % "A" 와 같은 문자열 포매팅 표현

- % 앞 부분은 포맷 템플릿, % 뒤는 실제 대입할 값
- % (포매팅 연산자, Formatting Operator)
- % 앞뒤로 각각 하나의 값 만을 받아들이므로 만약 % 뒤의 값이 복수이면 튜플로 묶어야 함

타입	의미
%s	문자열
%r	문자열
%c	문자열 (character, char)
%d 또는 %i	정수 (int)
%f 또는 %F	부동소수점 (float)
%e 또는 %E	지수형 부동소수점 (exponential)
%o 또는 %O	8진수
%x 또는 %X	16진수
%%	% 퍼센트 리터럴

Python 포매팅 표현 방법

```
p = " 전공: %s 이름: %s " % ("전자전기", "김유신")
print(p)
# 전공: 전자전기 이름: 김유신
```

```
p = "X = %0.3f, Y = %0.2f" % (3.141592, 3.141592)
print(p)
# 출력: X = 3.142, Y = 3.14
```

전공: 전자전기 이름: 김유신  
X = 3.142, Y = 3.14

Python 포매팅 예제



# 강의 목차

- Python 시작하기
- 기본 데이터 타입
- 제어: 조건문, 반복문

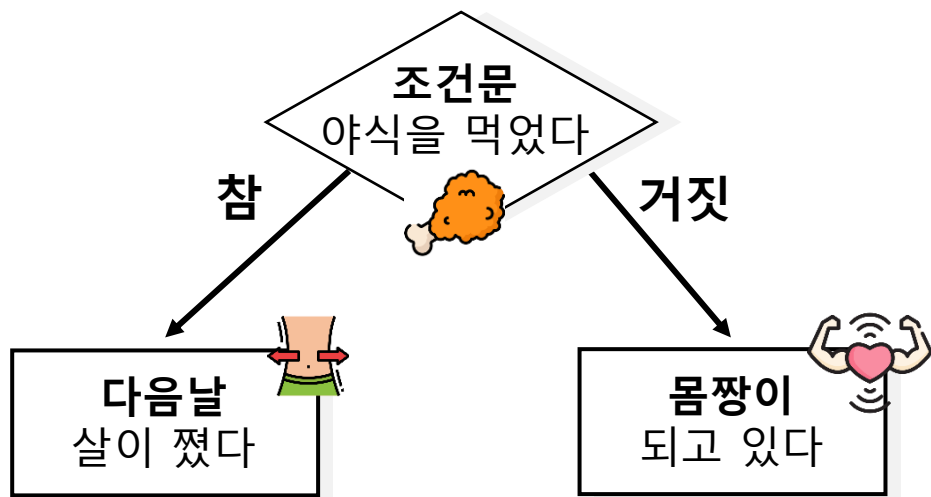
# 강의 목차

- Python 시작하기
- 기본 데이터 타입
- 제어: 조건문, 반복문

# Python 조건문

## ■ 조건문: 프로그래머가 명시한 조건에 따라 달라지는 상황을 수행

- Python에서 조건문을 사용하기 위해서는 **if**문을 사용
  - if** 키워드 다음에 조건을 적고, 조건식 다음에 콜론(:)을 사용해 if 조건식의 끝을 표현
  - if**문 조건식이 참이 아닐때, 다음 **if**문을 체크하기 위해 **elif** (else-if)문 사용 가능
  - if**조건문 안에서 특정 블록/문장을 수행하지 않고 그냥 skip하기 위해 **pass**라는 키워드 사용 가능



조건문의 예: 야식

```
x = 10

if x < 10:
    print("한자리수")
elif x < 100:
    print("두자리수")
else:
    print("세자리 이상")
```

Python 조건문의 예

# Python 조건문 실습

## ■ 시험 점수에 따라 다른 메시지를 출력하는 프로그램

- 점수의 분포에 따라 다른 메시지 출력
  - 1등급 (100점 이하, 90점 초과): "당신의 학점은 A 입니다 "
  - 2등급 (90점 이하, 80점 초과): "당신의 학점은 B 입니다 "
  - 3등급 (80점 이하, 70점 초과): "당신의 학점은 C 입니다 "
  - 4등급 (70점 이하): "재수강 입니다 ^^"

```
score = 100

if (score <= 100) and (score > 90):
    print("당신의 학점은 A입니다")
.
.
.
```

Python 조건문을 활용한 성적분류 프로그램

# Python 반복문

- **반복문: 프로그램 소스 코드내에서 특정한 부분이 반복적으로 수행 될 수 있도록 하는 구문**
  - 1부터 10까지 숫자를 반복해서 출력하는 경우
    - 1, 2, 3, ..., 10
    - `print(N)`, `N`은 1부터 10까지의 숫자
    - 반복문은 반복적으로 수행되는 작업을 쉽고 빠르게 사용하기 위한 방법

```
>>> print(1)
1
>>> print(2)
2
>>> print(3)
3
>>>
```

1부터 10까지 숫자를 출력하는 프로그램

# Python While 반복문

## ■ While문 (~하는 동안)

- While 키워드 다음의 조건식이 참인 경우 계속 While안의 블록을 실행
  - 예) 1, 2, 3, ..., 10 숫자를 출력
  - 다음에 나오는 숫자는 앞의 숫자보다 1만큼 큰 값
  - 계속 앞의 숫자에 1을 더하는 과정을 10이 될 때까지 반복
  - While은 어떤 조건이 만족되는 동안 그 아래 있는 문장은 반복, num이 10이 될 때까지 블록 (3~4 line) 반복

```
>>> num = 1
>>> while num <= 10:
...     print(num)
...     num = num + 1
... 
```

num = 1    num = 2    ...    num = 10  
print(num)    print(num)       print(num)

1부터 10까지 숫자를 출력하는 프로그램

# Python While 반복문 실습

- 정수를 한 개 입력 받아, 1부터 입력 받은 수 까지의 제곱을 구하는 프로그램
  - 5을 입력한 경우 5의 제곱까지 값을 출력

3

1 1  
2 4  
3 9  
4 16  
5 25

```
num = 5  
i=1  
  
while (i*i) <= (____):  
    print(i, " ", i*i)  
    i=(____)
```

제곱을 출력하는 프로그램 (좌: 동작화면, 우: 소스코드)

# Python For 반복문

## ■ For문: 컬렉션으로부터 하나의 요소 (element)를 가져와 루프 내의 블록을 실행

- 순서형 (Sequence) 자료를 이용하여 원하는 명령을 반복할 때 사용
  - **For (요소변수) in (컬렉션) 형식**으로 사용
  - 예) 0부터 10까지의 숫자를 더하는 경우
  - `range(n)`: 0부터 n-1까지의 숫자의 값을 리턴
  - 문자열 요소를 갖는 리스트로부터 각 문자열들을 순차적으로 출력 가능

```
sum = 0

for i in range(11):
    sum += i
print(sum)
```

```
list = ["This", "is", "a", "book"]

for s in list:
    print(s)
```

For문을 사용하는 코드의 예 (좌: 컬렉션, 우: 리스트)



# Python For 반복문 실습

## ■ 100미터 높이에서 고무공을 떨어트리는 경우

- 이 공은 땅에 닿을 때마다 원래 높이의 3/5만큼의 높이에 튀어 오릅니다.
- 공이 열 번 튀는 동안, 그때마다 공의 높이를 계산합니다.
  - `range(a,b)`: a부터 b-1까지의 숫자의 값을 리턴
  - `round(a,b)`: a값을 소수점 b자리 만큼 반올림

```
height = 100 # 공의 높이
bounce = 0   # 공이 튀어 오른 횟수

for bounce in ____:
    height = height * (____)
    print(bounce, round(height, 4))
```

```
esoc@esoc-desktop:~/1.Workspace_1/constant/python$ python bounce.py
1 60.0
2 36.0
3 21.6
4 12.96
5 7.776
6 4.6656
7 2.7994
8 1.6796
9 1.0078
10 0.6047
```

고무공이 튀어 오른 횟수와 높이를 계산하는 프로그램

**감사합니다**

---