



## **PYTHON SESSION**

By Edu  
@ 이영준 고병욱  
Date \_ 2018.03.24

# CONTENTS

**01 PYTHON 기초**

**02 클래스 / 모듈**

**03 라이브러리 활용**

# 01 Python 기초

## 세션 목표

1. 데이터 사이언스 세션 코드를 이해하고 스스로 짤 수 있는 수준 도달
2. 구글링을 통해 스스로 파이썬 공부를 원활히 할 수 있는 수준 도달
3. 스스로 코드 짜보는 연습 많이 해보기

# O1 Python 기초

다양한 프로그래밍 언어들

C, C++, Java, Python, Fortran 등

# 01 Python 기초

## 파이썬의 특징

1. 객체 지향 언어
2. 인터프리터 언어
3. 들여쓰기를 사용하여 가독성이 좋음
4. 내장함수가 풍부함
5. 데이터 분석 모듈이 잘 갖춰져 있음

# O1 Python 기초

## 명령 프롬프트 / 에디터

```

명령 프롬프트
Microsoft Windows [Version 10.0.16299.251]
(c) 2017 Microsoft Corporation. All rights reserved.
C:\Users\WLYJ>
  
```

```

1  import glob, re, math, random
2  from collections import Counter, defaultdict
3
4  def tokenize(message):
5      message = message.lower()
6      all_words = re.findall("[a-z0-9]+", message)
7      return set(all_words)
8
9  def count_words(training_set):
10     counts = defaultdict(lambda : [0,0])
11     for message, is_spam in training_set:
12         for word in tokenize(message):
13             counts[word][0 if is_spam else 1] += 1
14     return counts
15
16 def word_probabilities(counts, total_spams, total_non_spams, k=0.5):
17     return [(w,
18             (spam + k) / (total_spams + 2*k),
19             (non_spam + k) / (total_non_spams + 2*k))
20            for w, (spam, non_spam) in counts.items()]
21
  
```

**명령프롬프트: CLI 기반의 명령어를 수행할 수 있게 해주는 셸**

**\* CLI: 텍스트 터미널을 통해 사용자와 컴퓨터가 상호작용 하는 방식**

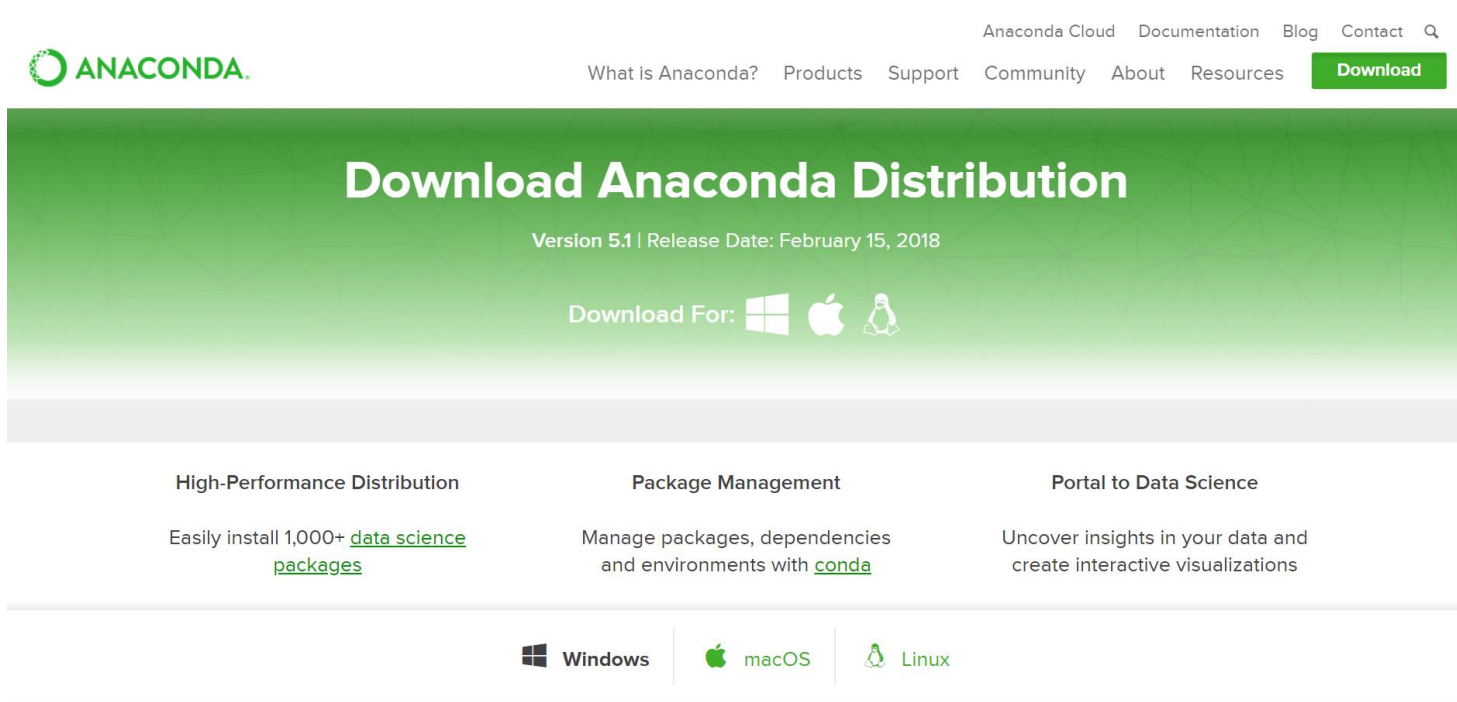
**에디터: 간편한 코드 작성을 도와주는 도구 (ex. Pycharm, Sublime Text )등**

# 01 Python 기초

## 아나콘다

<https://www.anaconda.com/download/>

Windows는 pip로 패키지를 설치할 때 문제가 많아  
주요 패키지가 포함된 Anaconda를 많이 사용함



The screenshot shows the Anaconda website's download page. At the top, there's a navigation bar with links to Anaconda Cloud, Documentation, Blog, Contact, and a search icon. Below this is the Anaconda logo and a 'Download' button. The main heading is 'Download Anaconda Distribution' with a subtext 'Version 5.1 | Release Date: February 15, 2018'. Underneath, it says 'Download For:' followed by icons for Windows, macOS, and Linux. The page is divided into three columns: 'High-Performance Distribution' (mentioning 1,000+ data science packages), 'Package Management' (mentioning managing packages with conda), and 'Portal to Data Science' (mentioning interactive visualizations). At the bottom, there are icons and labels for Windows, macOS, and Linux.




ANACONDA

Anaconda Cloud Documentation Blog Contact




What is Anaconda? Products Support Community About Resources [Download](#)

## Download Anaconda Distribution

Version 5.1 | Release Date: February 15, 2018

Download For:   

<b>High-Performance Distribution</b>	<b>Package Management</b>	<b>Portal to Data Science</b>
Easily install 1,000+ <a href="#">data science packages</a>	Manage packages, dependencies and environments with <a href="#">conda</a>	Uncover insights in your data and create interactive visualizations

 Windows  macOS  Linux

# 01 Python 기초

## 환경변수

프로세스가 컴퓨터에서 동작하는 방식에 영향을 미치는 동적인 값들의 모임

현재 폴더에 실행 파일이 없더라도 환경변수에 존재하는 파일들은 바로 실행이 가능함

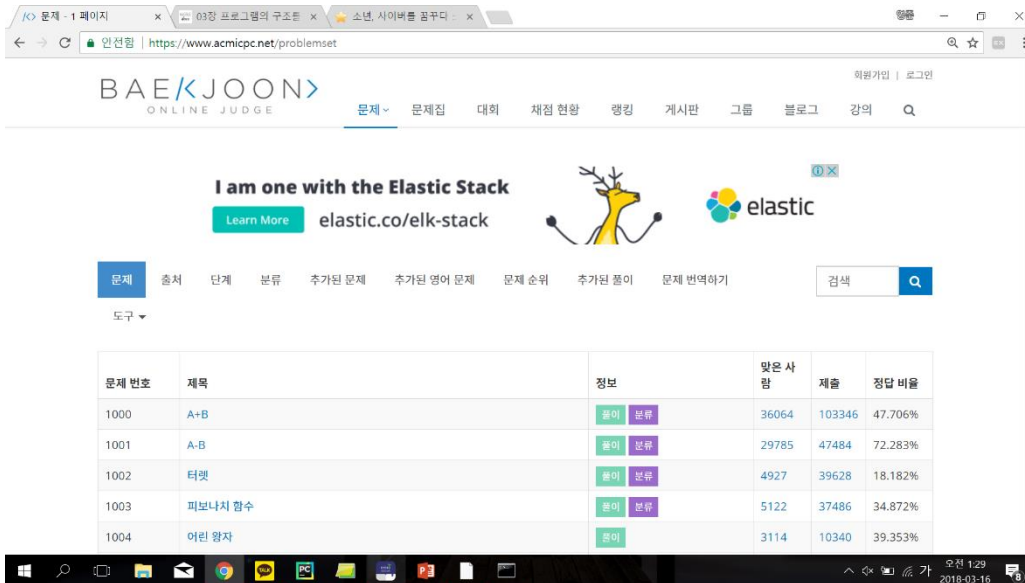


# 01 Python 기초

## 백준 알고리즘

<https://www.acmicpc.net/>

국내에서 가장 유명한 알고리즘 문제 사이트  
다양한 프로그래밍 문제/ 질문/ 풀이 존재



The screenshot shows the Baekjoon Online Judge website. The header includes the site name 'BAEKJOON ONLINE JUDGE' and navigation links like '문제', '문제집', '대회', etc. Below the header is a banner for 'I am one with the Elastic Stack' with a 'Learn More' button and the Elastic logo. A search bar is present. Below the banner is a table of problems.

문제 번호	제목	정보	맞은 사람	제출	정답 비율
1000	A+B	풀이 분류	36064	103346	47.706%
1001	A-B	풀이 분류	29785	47484	72.283%
1002	터렛	풀이 분류	4927	39628	18.182%
1003	피보나치 함수	풀이 분류	5122	37486	34.872%
1004	어린 왕자	풀이	3114	10340	39.353%

# 01 Python 기초

## 변수, 할당

파이썬에서 모든 것들은 객체(object)로 구현되어 있음

변수: 컴퓨터 메모리에 존재하는 값을 참조하기 위한 이름

변수 이름으로 사용할 수 있는 문자

소문자, 대문자, 숫자(숫자로 시작할 수는 없음), 언더스코어

변수 이름으로 사용할 수 없는 문자

예약어 ex.False, True, except, def etc..

할당: 데이터가 담긴 객체에 이름을 붙이는 것. 변수에 값을 할당한다는 표현 사용

Ex) `a = 32` 에서 `a`는 변수이고 32라는 정수를 `a`에 할당 한 것이라고 표현

\* 변수 명은 가독성을 고려해서 설정해야 협업 용이

# 01 Python 기초

## 숫자 자료형

### 숫자 자료형

정수형

실수형

8진수 / 16진수

```
_integer = 123  
_float = 1.23  
_float2 = 1.23e10  
_float3 = 1.23e-10  
_octal = 0o123  
_Hexadecimal = 0x8123f
```

데이터 타입 변환

int(), float()

```
print(float(3)) | 3.0  
print(int(12.3444)) | 12
```

# 01 Python 기초

## 숫자 자료형

### 연산자

사칙연산: +, -, \*, /

// 연산자: 정수 나누기

% 연산자: 나머지

\*\* 연산자: 지수

기타: math 모듈

### 사칙연산

```
>>> 1 + 3
4
>>> 1 - 3
-2
>>> 2 * 3
6
>>> 3 / 5
0.6
```

### 기타연산

```
>>> 12 // 5
2
>>> 12 % 5
2
>>> 12 ** 5
248832
```

# 01 Python 기초

## 문자열 자료형

### 문자열 자료형

```
"Python Session"
'Python Session'
"""Python Session"""
'''Python Session'''
```

### 문자열에 ', ' 포함시키는 경우

```
"Python' Session"
'Python" Session'
```

### 여러 줄인 문자열

```
'''
Python Session
I like Python
'''

"""Python Session\nI like Python"""
```

### 데이터 타입 변환: str()

```
>>> type(str(123))
<class 'str'>
>>> str(123)
'123'
```

# 01 Python 기초

## 문자열 자료형

### 인덱싱 / 슬라이싱 (데이터 부분 선택)

```
>>> a = "I like Python"
>>> a[0]
'I'
>>> a[5]
'e'
>>> a[0:5]
'I lik'
>>> a[1:6]
' like'
>>> a[-1]
'n'
```

### 문자열 연산

```
>>> "Py" + "thon"
'Python'
>>> "---" * 20
'-----'
```

1번 라인

-----

2번 라인

-----

3번 라인

- \* 0부터 세기 때문에 항상 주의!
- \* 마지막 인덱스는 포함하지 않음
- \* 인덱스의 부호가 음수면 마지막부터 카운트

# 01 Python 기초

## 문자열 자료형

### 문자열 포매팅

```
>>> a = "딱다구리"
>>> print("{0}는 너무 시끄러워요".format(a))
딱다구리는 너무 시끄러워요
>>> b = "개구리"
>>> print("{0}, {1}는 너무 시끄러워요".format(a, b))
딱다구리, 개구리는 너무 시끄러워요
```

\* {0}, {1} 순서로 대입

\* {0}, {1} 외에도 다양한 용법 존재

\* 원래는 %d와 같은 형태 사용

### 문자열 관련 주요 메서드

replace()

split()

```
>>> a = "I like Python"
>>> a.replace("like", "love")
'I love Python'
>>> a.split()
['I', 'like', 'Python']
>>> a.split('e')
['I lik', ' Python']
```

\* 메서드: 클래스 안에서 사용하는 함수

# 01 Python 기초

## ■ 연습문제

- \* a = "I love python" 을 "I like c++"로 바꾸어 보세요
- \* "hello"를 30번 출력해주세요
- \* "ABCDEFGF"를 "ABCZEFG"로 바꾸어주세요
- \* 1255 / 54 의 몫과 나머지를 구해주세요



# 01 Python 기초

## 리스트 자료형

리스트 자료형: 대괄호로 묶어서 표현되며 각 원소는 콤마로 구분

```
_python = [1,2,3,4,5,"PYPY", ["ABC", (123, 111)]]
```

### 리스트 자료형 특징

1. 데이터를 순차적으로 파악할 수 있음
2. 내용의 순서를 바꿀 수 있음
3. 문자열, 튜플과 달리 변경 가능함
4. 동일한 값 반복 가능

# 01 Python 기초

## 리스트 자료형

리스트 자료형 인덱싱, 슬라이싱

```
>>> _python = [1,2,3,4,5,"PYPY", ["ABC", (123, 111)]]
>>> _python[0]
1
>>> _python[5]
'PYPY'
```

\* 문자열과의 차이점: 개별 원소를 인덱싱을 통해 수정할 수 있음  
(문자열은 불가)

```
>>> example = [1,2,3,4,5]
>>> example[0] = 5
>>> example
[5, 2, 3, 4, 5]
```

연습문제

\_python에서 "ABC"를 출력해주세요

# 01 Python 기초

## 리스트 자료형

### 리스트 자료형 연산

#### 덧셈

```
>>> a = [1,2,3]
>>> b = [4,5,6]
>>> a+b
[1, 2, 3, 4, 5, 6]
```

#### 곱셈

```
>>> a = ["abc", "bcf"]
>>> a * 3
['abc', 'bcf', 'abc', 'bcf', 'abc', 'bcf']
```

# O1 Python 기초

## 리스트 자료형

### 리스트 수정/ 삭제

```
>>> a = [1,2,3,4,5]
>>> a[0] = 10
>>> a
[10, 2, 3, 4, 5]
```

```
>>> del a[0]
>>> a
[2, 3, 4, 5]
```

### 리스트 관련 주요 메서드

**append()**

**extend()**

**len()**

```
>>> a = [1,2,3,4,5]
>>> a.append([6,7])
>>> a
[1, 2, 3, 4, 5, [6, 7]]
```

```
>>> a = [1,2,3,4,5]
>>> a.extend([6,7])
>>> a
[1, 2, 3, 4, 5, 6, 7]
```

```
>>> a
[1, 2, 3, 4, 5, 6, 7]
>>> len(a)
7
```

# 01 Python 기초

## 튜플 자료형

튜플 자료형: 소괄호 묶여서 표현되며 각 원소는 콤마로 구분

```
>>> _tuple = (1,2,3)
>>> _tuple = (1,)
>>> _tuple = 1,2,3
```

### 튜플과 리스트의 차이점

1. 튜플도 인덱싱은 가능하나 값의 추가, 수정, 삭제가 불가능함
2. 튜플은 더 작은 공간을 사용함
3. 튜플은 딕셔너리의 키로 사용할 수 있음
4. 함수의 인자들은 튜플로 전달됨

# 01 Python 기초

## 딕셔너리 자료형

딕셔너리 자료형: {key1 : value1, key2:value2, key3:value3 }

```
>>> _dictionary = {"a":1, "b":2, "c":3}
```

딕셔너리 key, value 할당 / 삭제

```
>>> _dictionary = {"a":1, "b":2, "c":3}
>>> _dictionary["d"] = 4
>>> _dictionary
{'a': 1, 'b': 2, 'c': 3, 'd': 4}
>>> del _dictionary["a"]
>>> _dictionary
{'b': 2, 'c': 3, 'd': 4}
```

# 01 Python 기초

## 딕셔너리 자료형

### 딕셔너리 사용법

```
>>> _dictionary  
{'b': 2, 'c': 3, 'd': 4}  
>>> _dictionary['b']  
2
```

### 딕셔너리 사용시 주의사항

중복되는 key 값 사용시 하나를 제외한 나머지 것들이 무시됨

# 01 Python 기초

## 딕셔너리 자료형

### 딕셔너리 관련 메서드

#### key 존재 여부 리턴

```
>>> "c" in _dictionary
True
>>> "f" in _dictionary
False
```

#### key 들의 리스트 리턴

```
>>> _dictionary.keys()
dict_keys(['b', 'c', 'd'])
```



# 01 Python 기초

## 집합(set) 자료형

### 집합 자료형

어떤 것이 존재하는지 여부를 판단할 때 쓰는 자료형  
중복을 허용하지 않음  
순서가 없음

```
>>> example_set = set([1,2,3])  
>>> example_set  
{1, 2, 3}
```

### 교집합, 합집합, 차집합 구하기

```
>>> set_1 = set([1,2,3])  
>>> set_2 = set([1,5,6])  
>>> set_1 & set_2  
{1}  
>>> set_1 | set_2  
{1, 2, 3, 5, 6}  
>>> set_1 - set_2  
{2, 3}
```

# 01 Python 기초

## 불(참/거짓) 자료형

### 자료형의 참, 거짓

```
>>> bool("python")
True
>>> bool("")
False
>>> bool([1,2,3])
True
>>> bool([])
False
>>> bool(1)
True
>>> bool(0)
False
>>> bool(None)
False
```

```
>>> True
True
>>> False
False
```

# 01 Python 기초

## 주석

인터프리터에 의해 무시되는 텍스트의 한 부분

```
95 # 스팸일 확률을 오름차순으로 정렬
96 classified.sort(key=lambda row: row[2])
97
98 # 스팸이 아닌 메시지 중에서 스팸일 확률이 가장 높은 메시지
99 spammiest_hams = list(filter(lambda row: not row[1], classified))[-5:]
100
101 # 스팸 중에서 스팸일 확률이 가장 낮은 메시지
102 hammiest_spams = list(filter(lambda row: row[1], classified))[:5]
```

코드 설명 등 각종 코멘트를 할 때 사용함

# 01 Python 기초

## 라인 유지하기

코드의 가독성을 위해 사용

```
>>> 1 + 2 + 3 + 4  
... 4 + 5 + 6  
21
```

\* 코드의 가독성은 매우 중요한 문제!

# 01 Python 기초

## 입출력

### 입력: input() 함수 사용

```
>>> User_Variable = input()
>? 사용자 고유값
>>> print(User_Variable)
사용자 고유값
```

### 안내문구 입력 가능

```
>>> input("입력해주세요")
입력해주세요
>? 입력하는중..|
```

# 01 Python 기초

## 입출력

출력: print() 함수 사용

```
>>> print("you like python")
you like python
```

```
>>> print("you", "like", "python")
you like python
```

,로 띄어쓰기 가능

```
>>> print("you" "like" "python")
youlikepython
>>> print("you"+"like"+"python")
youlikepython
```

여러 문자열을 연달아 쓰거나 덧셈  
을 사용해 같이 출력할 수 있음

# 01 Python 기초

## 연습문제

백준알고리즘 1000 번

<https://www.acmicpc.net/problem/1000>

hint) map 함수, split()메서드

`map(function, iterable)`

첫 번째 인자로 함수를, 두 번째 인자로 반복 가능한 자료형을 받아  
자료형의 각 원소에 함수를 적용시킨 결과를 반환

`split()`

문자열을 분할하여 리스트로 만듦.  
공백으로 뉘둘 시 스페이스를 기준으로 나눔

# 01 Python 기초

## 제어문

### if 문 (조건문)

#### 기본 구조

```
>>> signal = True
>>> if signal == True:
...     print("signal is True")
... else:
...     print("signal is not True")
...
signal is True
```

if, else로 이루어져 있음

if 문에 속하는 실행문은 들여쓰기를 해야함

조건 뒤에 콜론(:) 이 필요함

들여쓰기: 스페이스 네 번 or tab 한 번

else는 꼭 있을 필요는 없음

if 문을 중첩하여 사용할수도 있음



# 01 Python 기초

## 제어문

### 비교연산자

$x < y \leftrightarrow x \text{가 } y \text{보다 작음}$

$x > y \leftrightarrow x \text{가 } y \text{보다 큼}$

$x \leq y \leftrightarrow x \text{가 } y \text{보다 작거나 같음}$

$x \geq y \leftrightarrow x \text{가 } y \text{보다 크거나 같음}$

$x == y \leftrightarrow x \text{와 } y \text{가 같음}$

$x != y \leftrightarrow x \text{와 } y \text{가 같지 않음}$

# 01 Python 기초

## 제어문

### 논리연산자

$x \text{ and } y \leftrightarrow x \text{와 } y \text{가 모두 참이면 참}$

$x \text{ or } y \leftrightarrow x \text{와 } y \text{ 둘 중 하나만 참이면 참}$

$\text{not } x \leftrightarrow \text{논리 상태를 반전시킴}$

### 멤버 연산자

$x \text{ in 리스트, 튜플, 문자열} \leftrightarrow \text{해당 자료형에 포함되어 있으면 참}$

$x \text{ not in 리스트, 튜플, 문자열} \leftrightarrow \text{해당 자료형에 포함되어 있지 않으면 참}$

# 01 Python 기초

## 제어문

### if 문 (조건문)

### elif 문

```
>>> if count == 1:
...     print("count is 1")
... elif count == 2:
...     print("count is 2")
... else:
...     print("count is not 1 or 2")
...
count is 2
```

if 문이 거짓이고 elif문이 참일 때 실행

# 01 Python 기초

## 제어문

### while 문 (반복문)

while (조건. 불리언 타입):  
    (실행 1)  
    (실행 2)

```
>>> while True:  
...     print("while loop")
```

### 일반적인 while문 사용법

```
>>> while count < 10:  
...     count += 1  
...     print("count: {}".format(count))  
...  
count: 1  
count: 2  
count: 3  
count: 4  
count: 5  
count: 6  
count: 7  
count: 8  
count: 9  
count: 10
```

# 01 Python 기초

## ■ 제어문

### while 문 (반복문)

#### 대입 연산자

$x += y \leftrightarrow x = x + y$

$x -= y \leftrightarrow x = x - y$

$x *= y \leftrightarrow x = x * y$

$x /= y \leftrightarrow x = x / y$

# 01 Python 기초

## 제어문

### while 문 (반복문)

**break 문:** 해당 조건을 만족시키면  
반복문을 빠져나감

```
>>> count = 0
>>> while count < 10:
...     count += 1
...     print("count: {}".format(count))
...     if count == 2:
...         break
...
count: 1
count: 2
```

**continue 문:** 해당 조건을 만족시키  
면 반복문의 처음으로 돌아감

```
>>> count = 0
>>> while count < 10:
...     count += 1
...     if count % 3 != 0:
...         continue
...     print("count: {}".format(count))
...
count: 3
count: 6
count: 9
```

# 01 Python 기초

## 제어문

for 문 (반복문)

for 원소 in 반복가능자료형:  
    (실행 1)  
    (실행 2)  
    ...

```
>>> _list = ["종운", "상연", "건우"]  
>>> for person in _list:  
...     print(person)  
...  
종운  
상연  
건우
```

# 01 Python 기초

## 제어문

for 문 (반복문)

for문, if문 응용

```
>>> for person in _list:
...     if person == "종윤":
...         print("이 사람은 이종윤입니다")
...     else:
...         print("이 사람은 종윤이가 아닙니다")
...
이 사람은 이종윤입니다
이 사람은 종윤이가 아닙니다
이 사람은 종윤이가 아닙니다
```

continue, break문은  
while문에서와 동일

if 문과 마찬가지로 for  
문도 중첩해서 사용할 수  
있음



# 01 Python 기초

## 제어문

for 문 (반복문)

range() 함수: 연속적인 값 생성

range(5) : 0,1,2,3,4

range(2,6) : 2,3,4,5

range(1,6,2) : 1,3,5

```
>>> for i in range(0,5):  
...     print(i)  
...  
0  
1  
2  
3  
4
```

끝 숫자는 포함되지 않음

보통 i를 많이 사용함

# 01 Python 기초

## 1회차 연습문제

1. 백준 알고리즘 1924번

<https://www.acmicpc.net/problem/1924>

2. 백준 알고리즘 2839번

<https://www.acmicpc.net/problem/2839>

3. 백준 알고리즘 4344번

<https://www.acmicpc.net/problem/4344>