

# PYTHON TUTORING #3

School of Computing, KAIST & 대덕고등학교 빛나리

# INTRO

- 파이썬 오류 해결하기, 구글링하기
- 저번 시간 REVIEW
- 연산자, 조건문
- 반복문
- 예제

## TIP) 에러가 났을때 해결해보아요!

문제 해결 능력은 큰 힘이 됩니다.

1. 에러를 읽기
2. 읽어도 모르겠으면 구글에 검색하기
3. 상단 검색 결과 중 stackoverflow가 있으면 들어가서  
읽어보기

영어를 모르겠으면 구글 번역 기능 사용!

# 1. 에러를 읽기

=> IndentationError: unexpected indent

알면? 고치면 됨!

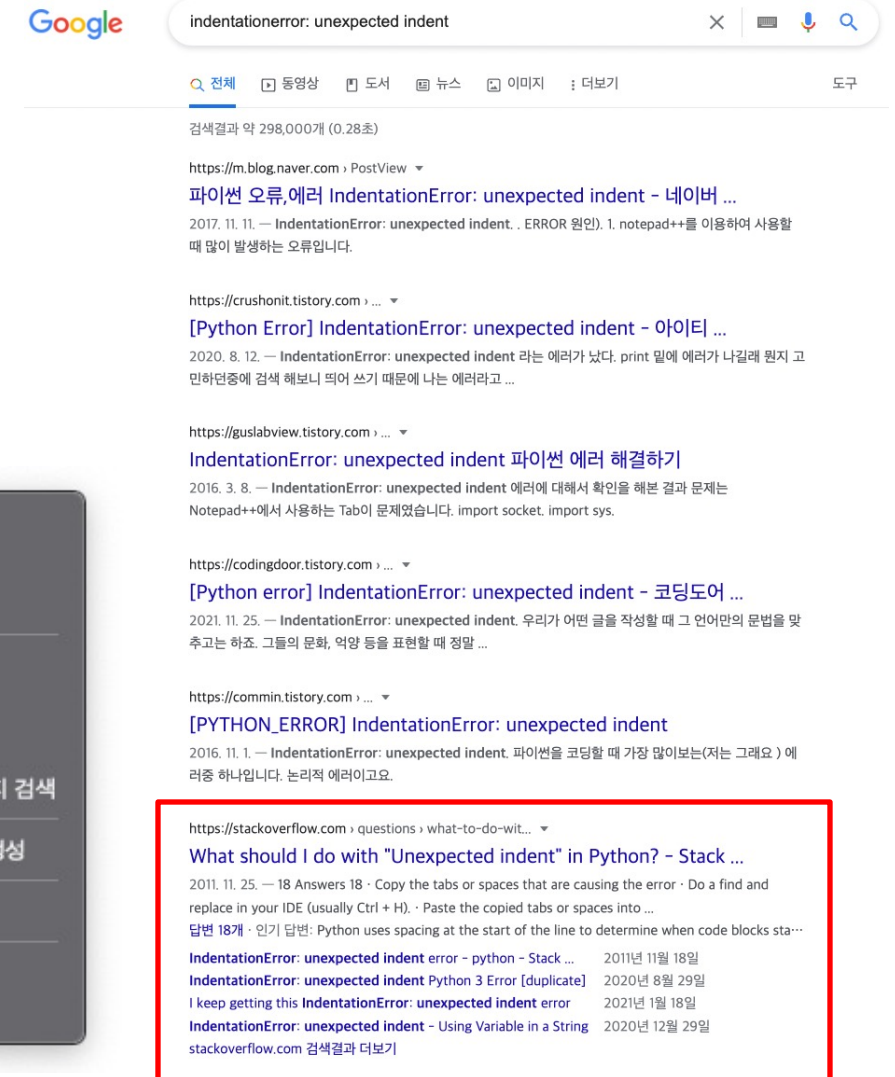
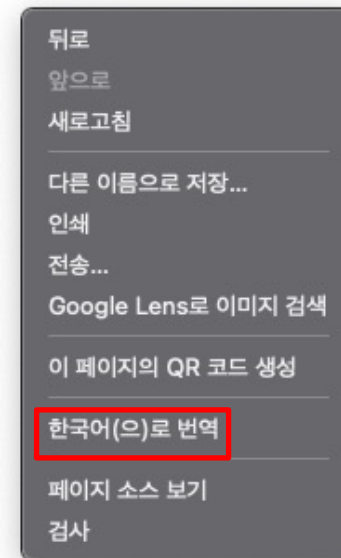
모르면? 음.. Indent가 뭐지? 구글 검색

```
>>> aa = 1
>>> aa
    File "<stdin>", line 1
      aa
IndentationError: unexpected indent
```

## 2. 모르겠으면 구글에 검색하기

Stackoverflow: 개발자들의 지식 in  
참고: tistory, naver 등의 블로그는 비추천

영어를 잘 모르겠으면 마우스 우클릭 후  
한국어(으)로 번역 선택하면  
어느정도 읽을 수 있어요



# 3. stackoverflow 읽어보고 문제 해결하기

(권장) 영어가 되시면 영어 버전으로

Python uses spacing at the start of the line to determine when code blocks start and end. Errors you can get are:


**Unexpected indent.** This line of code has more spaces at the start than the one before, but the one before is not the start of a subblock (e.g., the *if*, *while*, and *for* statements). All lines of code in a block must start with exactly the same string of whitespace. For instance:

```
>>> def a():  
...     print "foo"  
...     print "bar"  
IndentationError: unexpected indent
```

This one is especially common when running Python interactively: make sure you don't put any extra spaces before your commands. (Very annoying when copy-and-pasting example code!)

```
>>>   print "hello"  
IndentationError: unexpected indent
```

```
>>> aa = 1  
>>> aa  
File "<stdin>", line 1  
aa  
IndentationError: unexpected indent
```



```
>>> aa = 1  
>>> aa  
1
```

# 3. stackoverflow 읽어보고 문제 해결하기

한국어로 번역 -> 한국어 버전  
그리고 답변을 읽으면서 지식도 쌓을 수 있어요.  
(복사 붙여넣기 할 때는 주의해야겠다! 등등)

Python은 줄 시작 부분에 공백을 사용하여 코드 블록이 시작하고 끝나는 시점을 결정합니다. 얻을 수 있는 오류는 다음과 같습니다.


예기치 않은 들여쓰기. 이 코드 줄은 시작 부분에 이전 줄보다 더 많은 공백이 있지만 이전 줄은 하위 블록의 시작 부분이 아닙니다(예: *if*, *while* 및 *for* 문). 블록의 모든 코드 줄은 정확히 동일한 공백 문자열로 시작해야 합니다  
예를 들어:

```
>>> def a():  
...     print "foo"  
...     print "bar"  
IndentationError: unexpected indent
```

이것은 Python을 대화식으로 실행할 때 특히 일반적입니다. 명령 앞에 추가 공백을 두지 않도록 하십시오. (예제 코드를 복사하여 붙여넣을 때 매우 성가시다!)

```
>>> print "hello"  
IndentationError: unexpected indent
```

```
>>> aa = 1  
>>> aa  
File "<stdin>", line 1  
    aa  
IndentationError: unexpected indent
```



```
>>> aa = 1  
>>> aa  
1
```

# 모르는 개념이 생겼을때도 검색

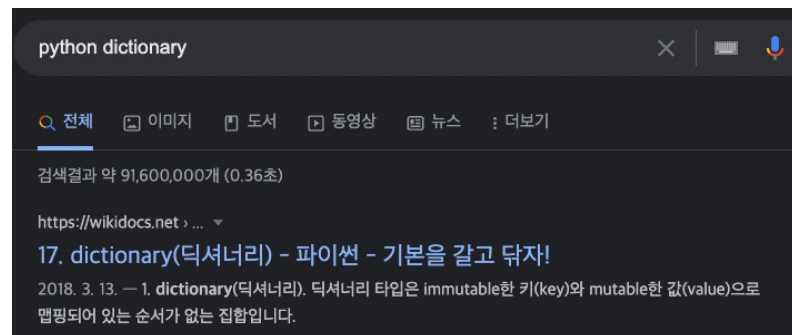
딕셔너리를 모를때 -> Python Dictionary

Docs.python.org

stackoverflow

Wikidocs.net

등등



- 순서가 없기 때문에 인덱스로는 접근할수 없고, 키로 접근 할 수 있습니다.

```
>>> d = {'abc' : 1, 'def' : 2}
>>> d[0]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 0
>>> d['abc']
1
```

<https://docs.python.org> > c-api > dict

## 딕셔너리 객체 — Python 3.10.4 문서

p가 dict 객체이지만, dict 형의 서브 형의 인스턴스는 아니면 참을  
PyObject \* PyDict\_New ()¶: Return value: New reference ...



# 지난시간 REVIEW

- 변수(variable)
- 자료형(List, Tuple, Dictionary, String, Bool 등)

## String + 변수 사용 예시

String을 변수에 저장  
두 변수와 또 다른 String을 더하고,  
print 를 통해 결과를 출력함

```
>>> var1 = "Hello"  
>>> var2 = "LightNari"  
>>> print(var1 + " " + var2)  
Hello LightNari
```

## List 사용 예시(심화)

List로 이뤄진 List를 만들고

people 변수를 만들어 첫번째 인덱스를 제외한 리스트를 다시 만듦

People[0]: Junsoo의 정보, People[1]: LightNari의 정보

People[0][2]: Junsoo의 정보 중 3번째 정보(나이)

```
>>> information = [["Name", "Birthday", "Age"], ["Junsoo", "0625", 24], ["LightNari", "1111", 18]]
>>> people = information[1:len(information)]
>>> people
[['Junsoo', '0625', 24], ['LightNari', '1111', 18]]
>>> print("Total Age:", people[0][2] + people[1][2])
Total Age: 42
```

# Python 연산자

연산자: 피연산자의 값을 조작할 수 있는 구조  
산술, 비교, 논리, 비트 연산자 등

산술 연산자 예시:

$1 + 2 = 3$

피연산자인 1, 2를 조작해 3을 만들기

Operator	Example
+ 덧하기	$a + b = 30$
- 빼기	$a - b = -10$
* 곱하기	$a * b = 200$
/ 나누기	$b / a = 2$
** 제곱	$a = 3; b = 4; \text{print}(a * b)$
// 정수 몫	$9 // 2 = 4$ and $9.0 // 2.0 = 4.5$
% 나머지	$b \% a = 0$

# Python 연산자

비교, 논리 연산자는 주로 조건문(특히 if문)에서 사용해요  
True, False(Not True)를 반환하는데, 이 값으로 조건문 실행 여부를 결정

## 비교 연산자

비교연산자	설명
<code>x &lt; y</code>	x가 y보다 작다
<code>x &gt; y</code>	x가 y보다 크다
<code>x == y</code>	x와 y가 같다
<code>x != y</code>	x와 y가 같지 않다
<code>x &gt;= y</code>	x가 y보다 크거나 같다
<code>x &lt;= y</code>	x가 y보다 작거나 같다

## 논리 연산자

연산자	설명
<code>x or y</code>	x와 y 둘중에 하나만 참이면 참이다
<code>x and y</code>	x와 y 모두 참이어야 참이다
<code>not x</code>	x가 거짓이면 참이다

# 조건문(Conditional Statement)

현재 조건을 판단하여

특정 조건을 만족할 때만 실행되는 코드

ex) score가 59 / 60 / 61 일 때 각각의 결과는?

예시

```
grade = 0
score = int(input())

if score >= 60:
    print("pass")
    grade = grade + 1
else:
    print("fail")
print(grade)
```

# 조건문

if 다음에는 어떤 조건 하에서 실행할지

조건문을 이용하여 표현해줘야 한다

연산자	설명
x or y	x와 y 둘중에 하나만 참이면 참이다
x and y	x와 y 모두 참이어야 참이다
not x	x가 거짓이면 참이다

비교연산자	설명
x < y	x가 y보다 작다
x > y	x가 y보다 크다
x == y	x와 y가 같다
x != y	x와 y가 같지 않다
x >= y	x가 y보다 크거나 같다
x <= y	x가 y보다 작거나 같다

## 예시

```
if 조건문:
    수행할 문장1
    수행할 문장2

if x > y or x < y:
    print("not same")
else:
    print("same!")
```

# 조건문

if + 조건문 다음에는 반드시 콜론 :  
수행할 문장들은 반드시 **들여쓰기(indent)**

else: 뒤에는  
조건문을 만족하지 않을 경우의 코드

만약 셋 이상의 분기문을 만들고 싶다면?

## 예시

```
if 조건문:  
    수행할 문장1  
    수행할 문장2  
  
if x > y or x < y:  
    print("not same")  
else:  
    print("same!")
```



# 조건문

셋 이상의 분기문을 만들고 싶다면  
elif를 이용할 수 있다.

## 예시

```
if 조건문1:  
    수행할 문장1  
    수행할 문장2  
elif 조건문2:  
    수행할 문장3  
else:  
    수행할 문장4
```

## 예제

input() 함수를 이용하여 성적을 입력 받아서

성적에 따라 "A", "B", "F" 중 하나를 출력하는 프로그램을 작성하시오.

A : 90 이상 100 이하  
B : 80 이상 89 이하  
F : 0 이상 79 이하

```
>>> grade = int(input())  
87  
>>> print(grade)  
87
```

## 예제

input() 함수를 이용하여 성적을 입력 받아서  
성적에 따라 "A", "B", "F" 중 하나를 출력하는 프로그램을 작성하시오.

A : 90 이상 100 이하  
B : 80 이상 89 이하  
F : 0 이상 79 이하

```
>>> grade = int(input())  
87  
>>> print(grade)  
87
```

```
>>> if 90 <= grade and grade <= 100:  
...     print("A")
```

## 예제

input() 함수를 이용하여 성적을 입력 받아서

성적에 따라 "A", "B", "F" 중 하나를 출력하는 프로그램을 작성하시오.

A : 90 이상 100 이하  
B : 80 이상 89 이하  
F : 0 이상 79 이하

```
>>> grade = int(input())  
87  
>>> print(grade)  
87
```

```
>>> if 90 <= grade and grade <= 100:  
...     print("A")  
... elif 80 <= grade <= 89:  
...     print("B")
```

## 예제

input() 함수를 이용하여 성적을 입력 받아서

성적에 따라 "A", "B", "F" 중 하나를 출력하는 프로그램을 작성하시오.

A : 90 이상 100 이하  
B : 80 이상 89 이하  
F : 0 이상 79 이하

```
>>> grade = int(input())  
87  
>>> print(grade)  
87
```

```
if 90 <= grade and grade <= 100:  
    print("A")  
elif 80 <= grade <= 89:  
    print("B")  
else:  
    print("F")
```

# 반복문(Loop Control)

숫자 하나를 입력받아 그 숫자를 3번 출력하기

3000번 출력하기?

같은 코드를 여러 번 반복시키고 싶을 때  
반복문을 사용한다

예시

```
print("hello")  
print("hello")  
print("hello")  
print("hello")
```

```
for i in range(4):  
    print("hello")
```

## 반복문 - for

n번 반복하고 싶다면

for 변수이름 in range(n):

매 코드 실행 시마다  
l의 값이 0에서 9까지 변화

if와 마찬가지로 콜론 :, 들여쓰기 중요!

예시

```
for i in range(10):  
    print(i)
```

```
for i in range(10):  
    for j in range(10):  
        print(i, j)
```

## 반복문 - while

while은 조건문과 함께 사용

조건문이 참일 경우 반복, 거짓이면 멈춤

반복횟수가 특별히 정해지지 않을 때 주로 사용

마찬가지로 콜론 :, **들여쓰기** 중요!

예시

while 조건문:  
반복할 문장1

i = 0

```
while i < 10:  
    print(i)  
    i = i + 1
```



# 반복문 Statement

pass, continue, break

**pass**는 실행할 코드가 없다는 뜻,  
다음 코드를 계속해서 진행한다.

**continue**는 즉시 다음 순번의 loop를 실행한다.

**break**는 반복문을 멈추고 loop 밖으로 나간다

예시

```
for i in range(10):  
    if i % 2 == 1:  
        continue  
    print(i)
```

```
for i in range(10):  
    if i % 2 == 1:  
        break  
    print(i)
```

## 예제

1이상 1000 이하의 수 중

7로 나누면 나머지가 3

11로 나누면 나머지가 5

13으로 나누면 나머지가 9

가 되는 수를 찾아라!

```
>>> for i in range(1, 1000):
```

# 예제

1이상 1000 이하의 수 중

7로 나누면 나머지가 3

11로 나누면 나머지가 5

13으로 나누면 나머지가 9

가 되는 수를 찾아라!

```
>>> for i in range(1, 1000):  
...     if i % 7 == 3:
```

# 예제

1이상 1000 이하의 수 중

7로 나누면 나머지가 3

11로 나누면 나머지가 5

13으로 나누면 나머지가 9

가 되는 수를 찾아라!

```
>>> for i in range(1, 1000):  
...     if i % 7 == 3 and i % 11 == 5 and i % 13 == 9:
```

## 예제

1이상 1000 이하의 수 중

7로 나누면 나머지가 3

11로 나누면 나머지가 5

13으로 나누면 나머지가 9

가 되는 수를 찾아라!

```
>>> for i in range(1, 1000):  
...     if i % 7 == 3 and i % 11 == 5 and i % 13 == 9:  
...         print(i)  
...  
269
```

## 예제

$n(n > 2)$ 을 입력받고, 첫번째부터  $n$ 번째까지의 피보나치 수를 `>>> n = int(input())`

`append()`를 이용하여 리스트에 저장하고 이를 출력하시오.

첫번째 피보나치 수 = 0, 두번째 = 1, 세번째 = 1

$$f(n) = f(n-1) + f(n-2)$$

Ex)  $n = 11$

>> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]

## 예제

$n(n > 2)$ 을 입력받고, 첫번째부터  $n$ 번째까지의 피보나치 수를 `>>> n = int(input())`

`append()`를 이용하여 리스트에 저장하고 이를 출력하시오.

첫번째 피보나치 수 = 0, 두번째 = 1, 세번째 = 1  
 $f(n) = f(n-1) + f(n-2)$

```
>>> fibonacci_list = [0, 1, 1]
```

Ex)  $n = 11$

>> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]

# 예제

$n(n > 2)$ 을 입력받고, 첫번째부터  $n$ 번째까지의 피보나치 수를 `>>> n = int(input())`

`append()`를 이용하여 리스트에 저장하고 이를 출력하시오.

첫번째 피보나치 수 = 0, 두번째 = 1, 세번째 = 1  
 $f(n) = f(n-1) + f(n-2)$

```
>>> fibonacci_list = [0, 1, 1]
>>> for i in range(3, n):
```

Ex)  $n = 11$

>> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]



## 예제

$n(n > 2)$ 을 입력받고, 첫번째부터  $n$ 번째까지의 피보나치 수를 `>>> n = int(input())`

`append()`를 이용하여 리스트에 저장하고 이를 출력하시오.

첫번째 피보나치 수 = 0, 두번째 = 1, 세번째 = 1

$f(n) = f(n-1) + f(n-2)$

Ex)  $n = 11$

>> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]

```
>>> fibonacci_list = [0, 1, 1]
>>> for i in range(3, n):
...     f_n = fibonacci_list[i-1] + fibonacci_list[i-2]
```

# 예제

$n(n > 2)$ 을 입력받고, 첫번째부터  $n$ 번째까지의 피보나치 수를 `>>> n = int(input())`

`append()`를 이용하여 리스트에 저장하고 이를 출력하시오.

첫번째 피보나치 수 = 0, 두번째 = 1, 세번째 = 1

$f(n) = f(n-1) + f(n-2)$

Ex)  $n = 11$

`>> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]`

```
>>> fibonacci_list = [0, 1, 1]
>>> for i in range(3, n):
...     f_n = fibonacci_list[i-1] + fibonacci_list[i-2]
...     fibonacci_list.append(f_n)
```

## 예제

$n(n > 2)$ 을 입력받고, 첫번째부터  $n$ 번째까지의 피보나치 수를 `>>> n = int(input())`

`append()`를 이용하여 리스트에 저장하고 이를 출력하시오.

첫번째 피보나치 수 = 0, 두번째 = 1, 세번째 = 1

$f(n) = f(n-1) + f(n-2)$

Ex)  $n = 11$

`>> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]`

```
>>> fibonacci_list = [0, 1, 1]
>>> for i in range(3, n):
...     f_n = fibonacci_list[i-1] + fibonacci_list[i-2]
...     fibonacci_list.append(f_n)
...
>>> print(fibonacci_list)
```