

인천광역시교육청교육과학정보원



기초부터 배우는

파이썬 프로그래밍

이광식, 최용운

파이썬 프로그래밍 목차

1. 파이썬 기초

1.1. 파이썬 설치	1
1.2. 입력 / 출력	5
1.3. 연산자	12

2. 제어문

2.1. 조건문	15
2.2. 반복문	26
2.3. break	31
2.4. continue	33

3. 자료형

3.1. 리스트(List)	44
3.2. 튜플(Tuple)	57
3.3. 딕셔너리(Dictionary)	59

4. 함수

4.1. 함수	75
4.2. 사용자 정의 함수	80
4.3. 재귀함수	82

5. 파일

5.1. 파일 쓰기	87
5.2. 파일 읽기	88
5.3. 디렉터리, 파일 관리	90

6. 인공지능

6.1. Google colab 소개	105
6.2. 텐서플로를 이용한 숫자인식	108
6.3. 텐서플로를 활용한 이미지 분류하기	118
6.4. 텐서플로를 활용한 마스크 착용 여부 확인하기	124
6.5. 머신러닝을 이용한 예측	129
6.6. 머신러닝을 이용한 회귀	139
6.7. 딥러닝을 이용한 자연어처리	144

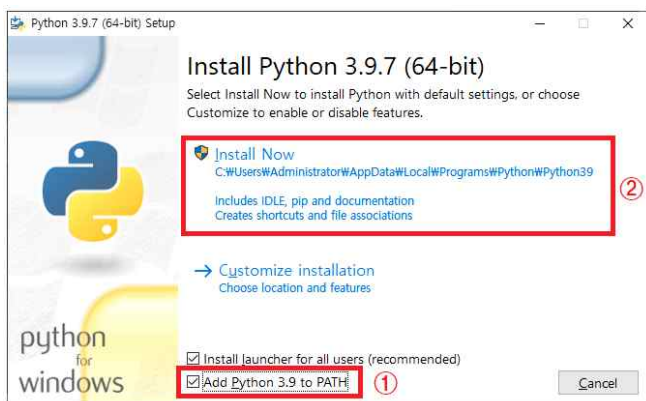
1. 파이썬 기초

1.1. 파이썬 설치

파이썬은 Windows, Linux, macOS 등 다양한 운영체제를 지원한다.

· <https://www.python.org>

위 사이트에 접속하여 최신 버전의 파이썬을 다운받자. 다만 최신 버전은 외부 라이브러리가 지원되지 않는 경우가 있기 때문에 기존에 파이썬3 사용자라면 그대로 사용하는 것을 추천한다. 파이썬은 버전별로 설치가 가능하기 때문에 버전별 관리만 할 수 있다면 추가 설치해도 문제없다. 본 교재는 3.9.7 버전을 기준으로 진행한다.



☐ Add Python 3.9 to PATH

파이썬 파일을 어느 곳에서 실행할 수 있도록 이 옵션을 선택하자.

이 옵션을 선택하면 외부 라이브러리 설치시에도 쉽게 사용할 수 있다.

만약 미선택한 상태에서 설치를 진행했더라도 Windows의 환경변수에서 PATH 설정을 통해 변경 가능하다.



파이썬을 설치하면 윈도우 버튼을 클릭하여 프로그램 메뉴를 확인해보자.

기존 사용하던 3.8 버전과 새로 설치한 3.9 버전을 따로 사용할 수 있는 것을 확인할 수 있다.

새로 설치한 Python 3.9(64-bit)를 실행해보자.

◦ Pyhon 3.9(64-bit)

```
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>>
```

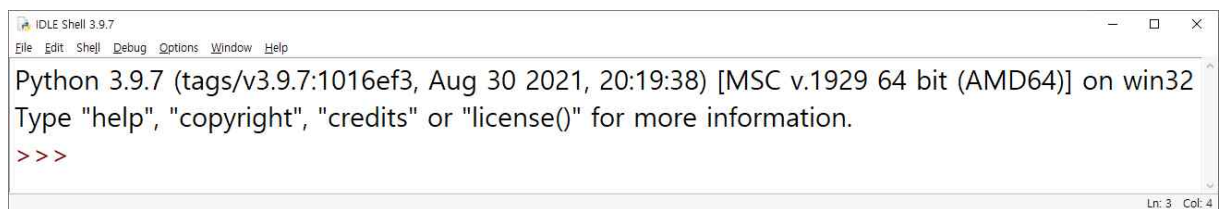
이와 같은 프롬프트(prompt) 창이 나타나며 명령에 따른 결과를 바로 확인할 수 있다. 컴퓨터와 대화하는 것 같다고 하여 대화형 인터프리터 언어라고도 한다.

```
>>> 1+2
```

```
3
```

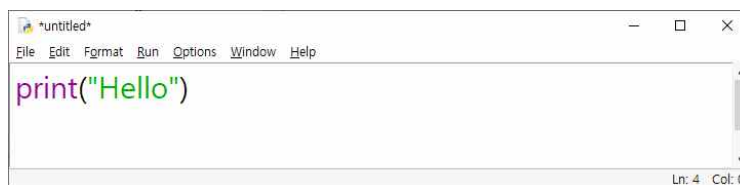
◦ IDLE (Pyhon 3.9 64-bit)

윈도우 버튼을 클릭한 후 [시작→Python 3.9→IDLE]를 클릭하자. 파이썬 셸과 동일한 내용의 창이 나타난다.

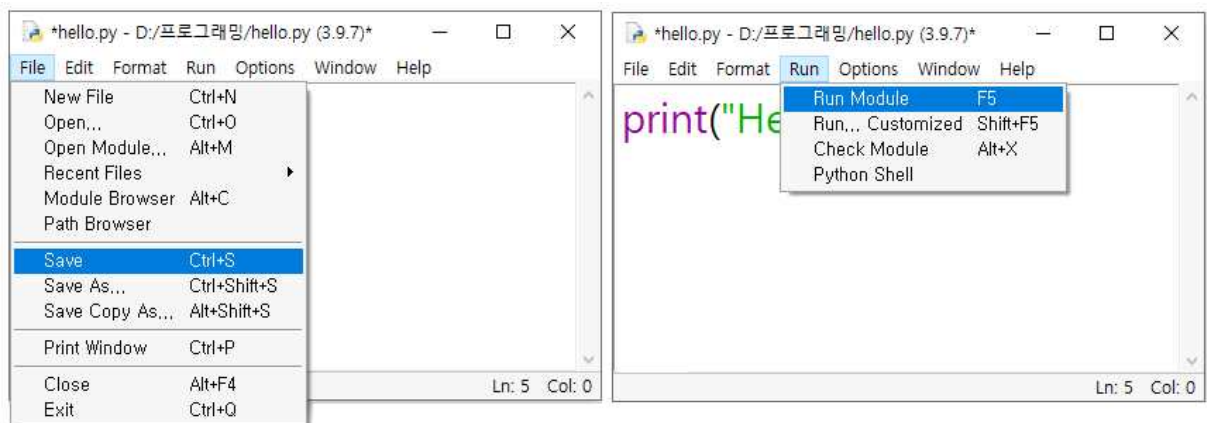


IDLE Shell Window는 에디터에서 실행한 결과가 나타난다.

IDLE Editor는 IDLE Shell Window에서 File 메뉴에서 New File을 선택한다.



위 그림과 같이 간단한 출력문을 작성한 후 저장하자. 확장자는 py이며, 실행 단축키는 F5이다. 파이썬 프로그램 실행 전에 저장을 해야 한다. 저장을 하지 않고 실행한다면 저장 확인 메시지 창이 나타난다. 자주 사용하는 기능이기 때문에 단축키 사용에 익숙해지도록 하자. (저장: Ctrl+S, 실행: F5)



IDLE editor는 파이썬을 설치할 때 함께 설치되는 기본 에디터이다. 파이썬의 기초 내용을 학습할 때 적합하며, 외부 라이브러리 사용이나 복잡한 프로그램을 작성할 경우 전문 editor 사용을 권장합니다.

• visual studio code

마이크로소프트에서 개발한 프리웨어 소프트웨어이다. 파이썬 뿐만 아니라 C/C++, C#, java, html, css, 웹프로그래밍 언어 등 다양한 언어를 지원하는 IDE(Integrated Development Environment)이다. Windows 뿐만 아니라 macOS, Linux를 지원하며, 개인 용과 상업용 모두 무료로 사용할 수 있다.

• PyCharm

Jet Brains에서 제작한 파이썬 전용 IDE이다. HTML, CSS, Javascript, CoffeeScript, SQL, Node.js 언어도 지원한다. 개인 사용자는 무료 버전을 사용할 수 있다. 학교에서 설치할 경우에는 공인 교육 기관(고등학교)용 무료 라이선스를 신청해서 사용할 것을 추천한다.

무료 라이선스 프로그램

교육 기관용 라이선스 오픈 소스 사용자 그룹 행사 파트너십 개발자 인정

학생 및 교사용 **학교 및 대학교용** 교육과정 및 부트캠프

교육 기관을 위한 강의실용 라이선스

모든 JetBrains IDE와 팀 도구를 학교에서 프로그래밍을 교육할 때 무료로 이용하세요.

무료 강의실용 라이선스

✔

공인 교육 기관(고등학교, 전문대학교, 4년제 대학교, 비영리 교육 단체)을 대상으로 발급됩니다.

✔

해당 교육 단체의 교직원, 교수, IT 지원 담당자 및 기타 공식 대표자가 관리합니다. 라이선스 관리자는 해당 [JetBrains 계정](#)에서 라이선스를 관리합니다.

✔

1년간 유효하며 라이선스가 만료되기 30일 전에 갱신할 수 있습니다.

✔

학생을 가르치는 데만 사용해야 합니다.

✘

개인 학생 또는 교사가 개인적으로 사용하도록 제공되지 않습니다(개인적 사용의 경우 [개인 교육용 구독](#) 신청).

✘

영리적인 목적으로 사용하면 안 됩니다.

✘

해당 단체의 제품 또는 서비스를 내부에서 개발하는 데 사용하면 안 됩니다.

✘

제3자와 공유하면 안 됩니다.

<JetBrains IDE 무료 라이선스>

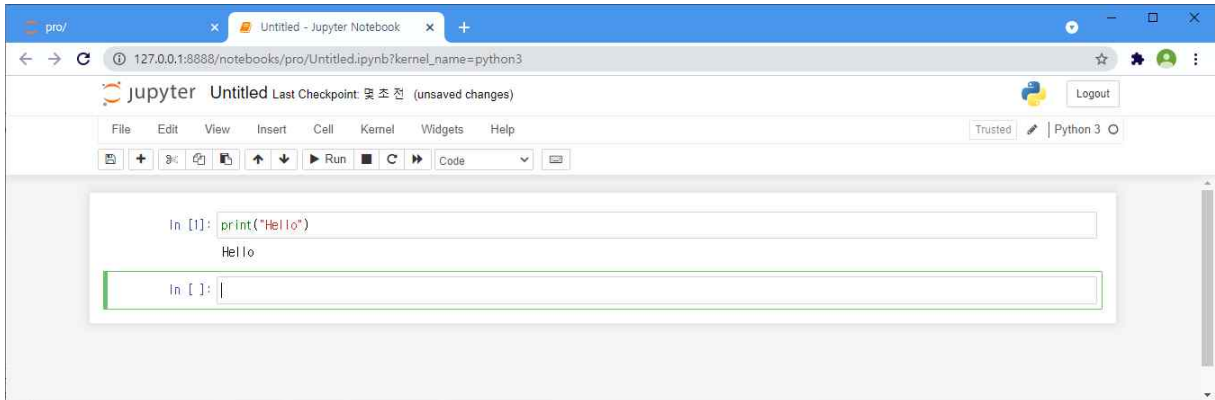
• Sublime Text

오픈소스가 아니기 때문에 라이선스 구입을 요청한다. 무료로 계속 사용할 수 있지만 구입 요청 창이 비정기적으로 나타난다.

Windows, Linux, macOS에서 사용할 수 있으며, 다양한 언어를 지원한다. Visual studio code나 PyCharm의 설치 파일이 각각 80MB, 360MB인 것에 비해 8MB 정도의 가벼운 소프트웨어이다. 메모장과 비슷하게 생겼으며, 실행 속도가 빠르다. 유사한 소프트웨어로는 notepad++, Editplus, atom 등이 있다.

• Anaconda

데이터 과학, 기계 학습 응용 프로그램, 빅데이터 처리, 예측, 분석 등에 특화된 패키지를 통합하여 오픈 소스이다. Anaconda에서 배포하는 Individual Edition은 개인 사용자만 무료로 사용할 수 있다. 많은 양의 패키지를 통합 설치하기 때문에 설치 공간도 2.9GB가 필요하다. 주피터 노트북(Jupyter notebook)에서 파이썬을 실행할 수 있다. (주피터 노트북: 웹 브라우저에서 코드를 작성하고 실행시킬 수 있는 환경)

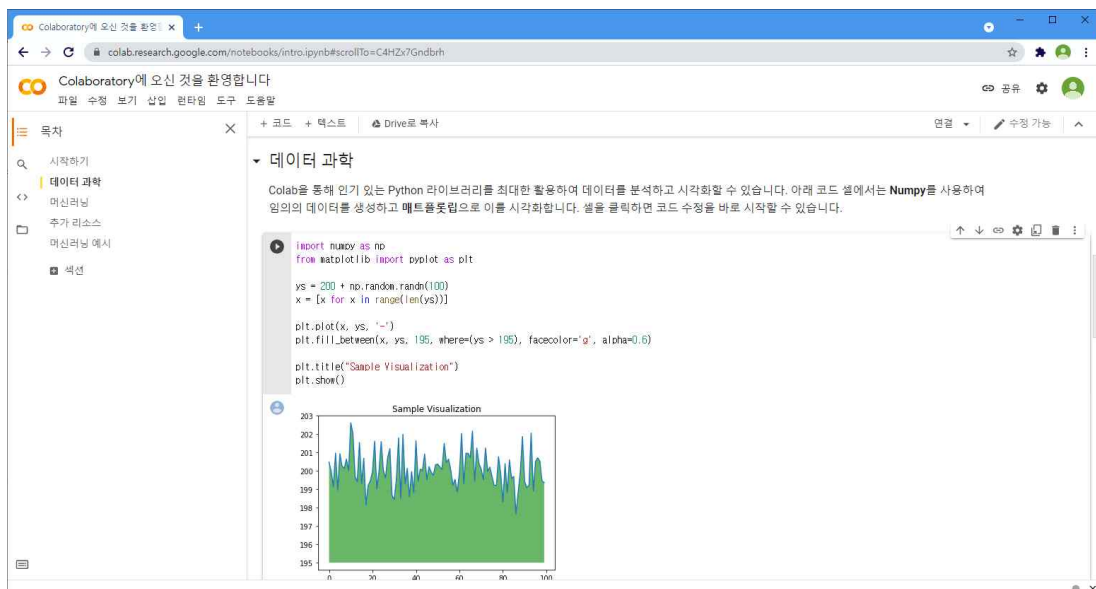


<Jupyter notebook>

• Google Colaboratory

Google에서 사용하던 jupyter notebook을 교육 및 연구용으로 커스텀마이징한 클라우드 서비스이다. 구글 드라이브에 파일을 저장할 수 있으며, 구글의 GPU 자원을 활용할 있기 때문에 인공지능 프로그램 작성에 적합하다.

python, tensorflow, keras, pandas를 포함한 여러 라이브러리를 설치되어 쉽게 사용할 수 있으며, 구글의 GPU를 사용하기 때문에 개인 컴퓨터를 사용하는 것보다 훨씬 빠른 시간에 프로그램을 실행시킬 수 있다.



<Google colaboratory>

1.2. 입력 / 출력

1.2.1. 변수

변수란 프로그래밍 언어에서 값을 저장하기 위한 공간이다. 변수의 값은 언제든지 변할 수 있는 수이다. 변수는 숫자 뿐만 아니라 문자, 리스트 등 다양한 자료형을 저장할 수 있다.

```
email = "test@abc.com"
point = 23
point = point + 10
```

email 변수에는 문자열을 저장했다. point 변수에는 숫자 23을 저장한 후에 10을 더했다. 오른쪽에서 먼저 계산한 후에 왼쪽 변수에 저장을 한다.

point 변수에 23을 저장한 후에 point 변수에 10을 더한 값을 다시 point 변수에 저장한다. 수학식에서는 성립될 수 없는 식이지만, 프로그래밍 언어에서 자주 사용한다.

```
a=3
b="python"
print(a, type(a), id(a))
print(b, type(b), id(b))
```

type(변수)은 변수에 저장된 자료형을 반환한다. id(변수)는 변수가 가리키는 메모리의 주소를 반환한다.

<실행 결과>

```
3 <class 'int'> 2964798400880
python <class 'str'> 2964810632304
```

변수 a는 int(정수형), b는 str(문자형) 이라는 것을 확인할 수 있다. 변수의 메모리 주소는 실행하는 컴퓨터마다 다르게 나타난다.

참고

변수 이름 만드는 규칙

1. 숫자, 알파벳, 한글, 밑줄(_)만 사용한다.
공백, 특수문자, 연산자 등은 사용할 수 없다.
2. 숫자로만 된 변수는 만들 수 없으며, 첫 글자로도 사용할 수 없다.
3. 프로그래밍 언어에서 사용하는 예약어는 변수 이름으로 사용할 수 없다.

※ 변수 이름을 정할 때는 변수 이름만 보더라도 어떤 데이터를 담을지 알 수 있는지 알 수 있도록 이름을 정하는 것이 좋다.

1.2.2. 입력

표준 입력(키보드를 통한 입력)의 경우 `input()` 함수를 사용하여 입력을 받는다.

```
score = input()
print(score)
```

실행하면 `input()` 함수를 실행하는 부분에서 커서가 깜빡인다. 내용을 입력하면, 입력한 내용이 출력된다. 단, 입력한 자료는 모두 문자열로 저장된다. 단순히 커서만 깜빡거리기 때문에 `input()` 함수안에 문자열을 입력해서 사용할 수 있다.

```
score = input("점수: ")
print(score + 10)
```

<실행 결과>

점수:

점수 15를 입력하고 엔터를 치면 에러가 발생한다.

점수: 15

예외가 발생했습니다. `TypeError`

can only concatenate str (not "int") to str

File "D:\프로그래밍\test1.py", line 2, in <module>: print(score + 10)

문자와 숫자를 연결할 수 없다는 내용이다. ‘+’가 더하기 연산자의 역할도 있지만 문자열 연결을 의미하기도 한다. `input()` 함수를 통해 입력한 내용은 사람이 보기에 숫자 일 뿐 컴퓨터는 문자 ‘15’를 저장한다.

이 경우에는 문자열 자료를 정수형으로 변환해야 더하기 연산이 실행된다. 입력할 때부터 변환하는 방법과 입력한 변수를 형변환 시키는 방법이 있다.

```
score1 = int(input("점수: "))
print(score1 + 10)
score2 = input("점수: ")
print(int(score2) + 10)
```

참고

파이썬 형 변환(자세한 내용은 자료형을 참고)

- | | |
|---------------------------------------|--|
| 1. <code>int(x)</code> : x를 정수로 변환 | 2. <code>float(x)</code> : x를 실수형으로 변환 |
| 3. <code>str(x)</code> : x를 문자열로 변환 | 4. <code>chr(x)</code> : x를 문자로 변환 |
| 5. <code>bool(x)</code> : x를 bool로 변환 | |

파이썬이 제공하는 도움말을 살펴보자. IDLE Shell에서 `help(input)`을 입력해보자.

```
>>> help(input)
```

Help on built-in function input in module builtins:

```
input(prompt=None, /)
```

Read a string from standard input. The trailing newline is stripped.

The prompt string, if given, is printed to standard output without a trailing newline before reading input.

If the user hits EOF (*nix: Ctrl-D, Windows: Ctrl-Z+Return), raise EOFError. On *nix systems, readline is used if available.

표준 입력(standad input)으로 문자열을 읽으며, 개행문자(newline)을 제거한다는 내용이다. 입력할 때 엔터키를 쳐야 입력이 완료된다. 이 때 입력한 엔터 값을 제거한다는 내용으로 당연히 받아들이겠지만, 추후 파일 입력에서는 줄이 바뀔 때마다 개행문자를 제거해야 하는 경우도 발생한다.

1.2.3. 출력

`help()` 함수를 이용하여 `print`에 대해 알아보자.

Help on built-in function print in module builtins:

```
print(...)
```

```
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

Prints the values to a stream, or to `sys.stdout` by default.

Optional keyword arguments:

`file`: a file-like object (stream); defaults to the current `sys.stdout`.

`sep`: string inserted between values, default a space.

`end`: string appended after the last value, default a newline.

`flush`: whether to forcibly flush the stream.

`sep=' '` : `print()` 함수에서 출력하는 값들을 구분자에 의해 띄어쓰기를 할 때 사용

`end='\n'` : 문장을 출력하고 마지막에 어떤 내용을 출력할 때 사용

`file=sys.stdout` : 특별한 설정이 없다면 화면에 출력. 파일 출력시에도 사용

`flush=False` : I/O버퍼 사용 유무 설정. 기본 `False`로 설정

다음 출력 명령을 통해 print() 함수의 기능을 알아보자.

◦ 큰 따옴표(“”), 작은 따옴표(‘ ’)

파이썬은 print()에서 큰 따옴표와 작은 따옴표를 모두 허용한다. 큰 따옴표로 시작했다면 큰 따옴표로 닫아야 한다.

```
print(1+2)           #1+2의 연산 결과가 출력 3
print("1+2")         # 문자 "1+2"가 출력 1+2
print('Hello')       #작은 따옴표 사용 Hello
print("Hello")       #큰 따옴표 사용 Hello
print("I can't swim") #큰 따옴표 안에 작은 따옴표 사용 I can't swim
print('"파이썬"언어') #작은 따옴표 안에 큰 따옴표 사용 "파이썬" 언어
```

큰 따옴표 안에 큰 따옴표를 사용하고 싶다면 이스케이프 문자(\)를 사용한다. 글꼴에 따라서 '₩'이나 '\ '로 출력된다.

코드	설명
\'	문자열에서 작은 따옴표(')를 그대로 표현
\"	문자열에서 큰 따옴표(")를 그대로 표현
\\	문자열에서 큰 따옴표(")를 그대로 표현
\\n	문자열에서 줄을 바꿀 때 사용
\\t	문자열 사이에 Tab 간격을 줄 때 사용
\\r	캐리지 리턴(현재 커서를 가장 앞으로 이동)
\\f	폼 피드(현재 커서를 다음 줄로 이동)

참고

주석

프로그램 실행에 영향을 주지 않도록 앞글자에 기호를 붙여 사용한다.

프로그래밍 언어마다 차이가 있으며, 파이썬은 #을 사용한다. 주석은 소스 코드를 쉽게 이해하도록 하는 것이 목적이며, 협업을 할 경우 주석을 사용하면 효율적이다.

한 줄 주석은 #을 사용한다.

여러 줄 주석은 작은 따옴표 3개로 시작해서 작은 따옴표 3개로 닫는다. 작은 따옴표 3개로 둘러싸인 부분은 주석문이 되어 실행에 영향을 미치지 않는다.

◦ 연결 연산

문자를 연결할 때는 '+', ',', ' ' 기호를 사용하거나 생략할 수 있다. 단, 심표(',')를 사용할 경우 문자열끼리 한 칸의 공백이 발생한다.

```
print("Hello"+"Python") #HelloPython
print("Hello", "Python") #Hello Python
print("Hello" "Python") #HelloPython
```

문자와 숫자를 출력할 때는 다음 사항에 유의하자. 심표(',')를 사용하면 공백 한 칸을 두고 출력한다. 곱하기('*')를 사용할 경우 해당 문자가 n번 출력된다.

```
print("Hello", 3) #Hello 3
print("Hello"3) #SyntaxError: invalid syntax
print("Hello"+3) #can only concatenate str (not "int") to str
print("Hello"*3) #HelloHelloHello
```

◦ 포매팅(formatting)

파이썬에서는 복잡한 문자열 출력을 위해 포매팅(formatting)을 지원한다. %를 사용하는 C언어 스타일도 지원하며, fomrat 메서드(method)를 사용하는 방식과 f 문자열을 사용하는 방식이 있다.

· % 기호를 사용하는 포매팅

"문자열" %값

문자열에는 숫자, 문자열을 대입할 수 있고 %d(정수), %f(실수), %s(문자열)을 사용할 수 있다.

```
name = "홍길동"
age = 17
score=91.35
print("이름: %s, 나이:%d, 점수:%f" %(name, age, score))
```

% 기호를 사용하는 포매팅에서는 출력하는 문자열의 길이를 설정할 수 있다.

```
print("[%10s]"%name) # 전체 10칸 중에 문자열 앞에 공백을 채운다.
print("[% -10s]"%name) # 전체 10칸 중에 문자열 뒤에 공백을 채운다.
```

<실행 결과>

```
[      홍길동]
[홍길동      ]
```

%d(정수)도 동일하게 사용할 수 있다. %f의 경우 소수점 아래 자리와 전체 자리 모두를 설정할 수 있다.

<pre> score=91.35 print("[%.1f]"%score) print("[%.3f]"%score) print("[%8.1f]"%score) print("[%8.3f]"%score) print("[%8.1f]"%score) print("[%8.3f]"%score) </pre>	<p><실행 결과></p> <pre> [91.3] [91.350] [91.3] [91.350] [91.3] [91.350] </pre>
--	--

%f의 경우에도 음수를 붙이는 경우에는 실수를 모두 작성하고 뒷부분에 공백을 둔다. score가 91.35일 때 %.3f는 소수점 자릿수를 3자리로 지정하며, 해당하는 값이 없을 때는 0으로 채운다.

%8.3f는 소수점을 포함한 모든 내용이 8자리에 들어간다는 것을 확인해보자.

1	2	3	4	5	6	7	8
		9	1	.	3	5	0

· format 메서드를 사용하는 방법

% 기호를 사용하는 대신 형식을 지정할 곳에 {}기호를 사용한다. % 기호처럼 자료형을 선언할 필요가 없다.

```

print("이름:{}, 나이:{}, 점수:{}".format(name, age, score))
print("이름:{1}, 나이:{2}, 점수:{0}".format(score, name, age))

```

{ }안에 순서를 생략할 경우 format()에 작성한 차례대로 출력된다. 가장 앞에 있는 값은 {0}부터 시작한다.

%10s처럼 출력할 글자수를 지정하기 위해\ 다음과 같이 설정할 수 있다.

형식 지정	내용
{:>10}	전체 10칸을 차지하며 공백을 앞에 붙임
{:<10}	전체 10칸을 차지하며 공백을 뒤에 붙임
{:^10}	전체 10칸을 차지하며 문자열을 중앙에 출력
{:.3f}	소수점 아래 3자리까지 표시
{:,}	천단위 쉼표를 출력

```

print("[{:>10}"].format(name))      # [      홍길동]
print("[{:<10}"].format(name))      # [홍길동      ]
print("[{: ^10}"].format(name))      # [   홍길동   ]
print("[{: .5f}"].format(score))     # [91.35000]
print("[{: 8.3f}"].format(score))    # [ 91.350]
print("[{:<8.3f}"].format(score))    # [91.350  ]
print("[{:>8.3f}"].format(score))    # [ 91.350]
print("[{: ,}"].format(1230000))     # [1,230,000]

```

> 기호 앞에 문자열을 쓰면 공백을 해당 문자열로 채워 준다.

```

print("[{: * > 10}"].format(name))
print("[{: # < 10}"].format(name))
print("[{: ? ^ 10}"].format(name))

```

<실행결과>

```

[*****홍길동]
[홍길동#####]
[???홍길동????]

```

format 메서드는 다음과 같이 분리한 후에도 사용 가능하다.

```

temp = "이름:{}, 나이:{}, 점수:{}"
temp = temp.format(name, age, score)
print(temp)

```

· f문자열을 사용하는 방법

format 메서드를 사용하는 것보다 내용을 줄여쓰기 위해 f 글자를 붙인 문자열이다. 파이썬 3.6버전부터 지원한다.

```

print(f"이름:{name}, 나이:{age}, 점수:{score}")

```

공백의 크기나 특정 문자열을 채우는 방법은 다음과 같다.

```

print(f"[{name:>10}], [{age:>10}]")
print(f"[{name:<10}], [{age:<10}]")
print(f"[{name:^10}], [{age:^10}]")

```

<실행 결과>

```

[      홍길동], [      17]
[홍길동      ], [17      ]
[   홍길동   ], [   17   ]

```

포매팅으로 url 작성하기

url 접속을 통한 데이터를 전달하는 IoT 장비가 있다.

```
https://api.thingspeak.com/update?api_key=YTRUYF6Y045M0B37&field1=5&field2=10
```

이와 같이 url을 통해 접속하면 두 개의 값을 전송할 수 있다.

api_key는 개인별 API 키 중 하나이다.

field1과 field2는 전송하려는 값이다.

포매팅을 사용하여 url을 작성하면 다음과 같다.

```
site = "https://api.thingspeak.com/update?api_key="
apikey = "YTRUYF6Y045M0B37"
v1 = 5
v2 = 10
url = f"{site}{apikey}&field1={v1}&field2={v2}"
print(url)
```

apikey는 개인마다 변경될 수 있으며 전송하려는 v1, v2 값은 계속 변할 때, 포매팅을 사용하면 가독성을 높이면서 편하게 작성할 수 있다.

사용하는 언어의 습관과 취향에 따라 다음과 같이 작성할 수 있다.

```
url2 = "%s%s&field1=%d&field2=%d"%(site,apikey,v1,v2)
url3 = "{}{}&field1={}&field2={}".format(site,apikey,v1,v2)
```

1.3. 연산자

연산이란 프로그램에서 데이터를 처리하여 결과를 산출하는 것이다. 연산자란 연산을 수행하는 기호를 말한다. 대부분의 언어가 비슷한 연산자를 사용하고 있다. 파이썬에서는 다음과 같은 연산자를 지원하고 있다.

1.3.1. 연산자의 종류

◦ 산술 연산자

산술 연산은 수학적 계산을 위한 연산자들이다. 더하기(+), 빼기(-), 곱하기(*), 나누기(/)를 사용한다. 나머지(%), 제곱(**), 몫(//) 연산자는 실습을 통해 특징을 살펴보자.

파이썬에서 나누기 연산의 특징은 정수를 나눌 때 나머지가 없이 나누어 떨어지더라도 실수로 표현된다.

간단한 연산은 IDLE Shell Window에서 실행하자.

>>> 2+3 5 >>> 2+3*4-5 9 >>> 4/2 2.0	>>> 7%3 1 >>> 7//3 2 >>> 2**3 8
--	--

7%3 : 7을 3으로 나눈 나머지

7//3 : 7을 3으로 나눈 몫

2**3 : $2^3(2*2*2)$

$\sqrt{4}$ 는 어떻게 표현할까?

```
>>> 4**0.5  
2.0
```

◦ 할당 연산자

‘=’ 연산 기호는 같다(equal)의 의미가 아니라 오른쪽 값을 왼쪽 변수에 할당하는 연산자이다.

a=3

값 3을 변수 a에 할당한다. a와 3이 같다는 뜻이 아니라 할당하는 것을 알아두자. 산술 연산자와 함께 사용하여 코딩을 간결하게 표현하는 +=, -=, *=, /=, %=, //= 등의 연산이 있다.

a=a+1 과 a+=1 은 동일한 표현 방법이다.

>>> a=5 >>> a=a+1 >>> a 6 >>> a+=2 >>> a 8	>>> a+=2 >>> a 8 >>> a-=4 >>> a 4
--	--

◦ 비교 연산자

관계 연산, 조건 연산이라고도 부르기도 하며, 연산의 결과는 참(True)과 거짓(False)으로만 나타난다.

비교연산자	의미	사용 예	설명
>	크다	A > B	A가 B보다 크다.
>=	크거나 같다	A >= B	A가 B보다 크거나 같다.
<	작다	A < B	A가 B보다 작다.
<=	작거나 같다	A <= B	A가 B보다 작거나 같지 않다.
=	같다	A == B	A와 B가 같다. *등호(=)가 두 개이다.
!=	다르다, 같지 않다	A != B	A와 B가 같지 않다.

◦ 논리 연산자

논리 연산은 두 개 이상의 조건에서 사용된다. and, or, not 연산이 있다.

a=5, b=3일 때 논리 연산의 결과를 살펴보자.

표현	설명	사용 예
and	조건1 and 조건2 조건이 모두 참(True)일 때 참(True)을 반환	>>>a>2 and b>2 True
or	조건1 or 조건2 조건 중 하나만 참(True)이면 참(True)을 반환	>>>a>3 or b>5 True
not	not 비교연산 연산의 결과를 반대로 변경한다.	>>>not a>b False

◦ 비트 연산자

주로 하드웨어 관련 프로그래밍에 활용하는 연산이다. 비트 연산의 And(&), Or(!)은 비교 연산자의 and, or과 비슷하게 동작한다. 차이점이라면 전체 값이 아니라 비트 단위에 적용된다.

Exclusive 연산은 피연산자의 논리가 서로 다를 때 참(True)이 된다.

a=13, b=10일 때 비트 연산의 결과를 살펴보자. (단, 비트 단위로 연산되며, 이진수 표현 방법을 알아야 한다. 13 = 1101₂ 10=1010₂ 이다.)

표현	설명
a&b	1101 1000 값이 서로 같을 때만 1을 반환한다. 1010 결과는 8을 출력
a b	1101 1111 값이 하나라도 참(True)이면 1을 반환한다. 1010 결과는 15를 출력한다.
a^b	1101 0111 값이 서로 다르면 1을 반환한다. 1010 결과는 7을 출력한다.

2. 제어문

2.1. 조건문

2.1.1. 단일 if

파이썬에서 사용하는 조건문 if는 다음과 같이 들여쓰기를 사용한다.

```
if 조건식:
    조건이 참일 때 실행할 명령1
    조건이 참일 때 실행할 명령2
```

조건식 뒤에 콜론(:)을 붙이며, 조건이 참일 때 실행할 명령은 반드시 들여쓰기를 해야 한다. 들여쓰기를 n칸 했다면, 조건이 참일 때 실행할 명령 모두 n칸 들여쓰기를 해야 한다. 파이썬에서는 들여쓰기 4칸을 권장한다.

점수를 입력했을 때 80점 이상이면 "pass"를 출력하는 프로그램을 작성해보자.

```
score = int(input())
if score >= 80:
    print("pass")
```

점수를 입력하면 int() 함수를 통해 정수형으로 변환하여 score에 저장한다.
score 값이 80 이상이면 "pass"를 출력한다.

입력한 값이 70점이면 어떤 결과가 나타날까?

조건문은 80점 이상일 때만 작동하기 때문에 아무 변환도 나타나지 않는다.

80점 미만이라면 "fail"을 출력하려면 어떻게 해야 할까?

단일 if문을 사용하면 두 개를 연속해서 사용해야 한다. 가독성이 떨어지며, 조건식의 수정과 조건이 참과 거짓일 때 실행할 명령을 수정하기 힘들다.

```
score = int(input())
if score >= 80:
    print("pass")
if score < 80:
    print("fail")
```

2.1.2. if ~ else

조건식의 참과 거짓에 따라 조건문을 작성할 때에는 if문 두 개를 사용하여 표현할 수도 있지만 if ~ else를 추천한다.

```
if 조건식:
    조건이 참일 때 실행할 명령
else:
    조건이 거짓일 때 실행할 명령
```

들여쓰기는 위, 아래 동일하게 4칸을 적용해야 한다.

```
score=int(input())
if score>=80:
    print("pass")
else:
    print("fail")
```

80점 이상이면 "A", 60점 이상이면 "B", 60점 미만이면 "C"를 출력하는 프로그램을 작성하는 프로그램을 작성해보자.

단일 if문을 사용한 아래 소스는 어떤 문제가 있을까?

```
score=int(input())
if score>=80:    # 조건1
    print("A")
if score>=60:    # 조건2
    print("B")
if score<60:     # 조건3
    print("C")
```

88점을 입력할 경우 score>=80과 score>=60을 만족하기 때문에 AB를 출력한다.

if ~ else를 사용하면 다음과 같이 작성할 수 있다.

```
score=int(input())
if score>=80:    # 조건1이 참이면
    print("A")
else:            # 조건1이 거짓이면
    if score>=60: # 조건2가 참이면
        print("B")
    else:        # 조건2가 거짓이면
        print("C")
```

80점 이상이면 "A"를 출력하고, 그렇지 않은 경우 if문을 다시 연산한다.

이번에는 입력한 점수에 따라 5단계 평가를 출력하는 프로그램을 작성하려고 한다.

90점 이상	80점 이상	70점 이상	60점 이상	60점 미만
A	B	C	D	F

if ~ else를 사용하면 다음과 같다.

```
score=int(input())
if score>=90:           # 90점 이상이면
    print("A")          # A를 출력하고 if문은 종료
else:                  # 그렇지 않으면(90점 이상이 아니면)
    if score>=80:       # 80점 이상이면
        print("B")     # B를 출력하고 종료
    else:               # 80점 이상이 아니라면
        if score>=70:  # 70점 이상이면
            print("C")  # C를 출력하고 종료
        else:          # 70점 이상이 아니라면
            if score>=60: # 60점 이상이면
                print("D") # D를 출력하고 종료
            else:       # 60점 이상이 아니라면
                print("F") # F를 출력하고 종료한다.
```

2.1.3. if ~ elif ~ else

조건식이 여러 개인 경우 중첩 if를 사용할 수 있지만 코딩의 간결성이 떨어진다. 조건이 여러 개인 경우 if ~ elif ~ else를 제공한다.

```
if 조건식1:
    조건식1이 참일 때 실행할 명령
elif 조건식2:
    조건식2이 참일 때 실행할 명령
elif 조건식n:
    조건식n이 참일 때 실행할 명령
else: #else는 생략할 수 있음. else는 elif 뒤에 위치해야 함.
    조건이 거짓일 때 실행할 명령
```

```

score=int(input())
if score>=90:
    print("A")
elif score>=80:
    print("B")
elif score>=70:
    print("C")
elif score>=60:
    print("D")
else:
    print("F")

```

퀴즈

Quiz 1. 체중(Kg)과 키(m)를 이용한 BMI 계산식을 이용하여 신체지수에 의한 비만 분류를 출력하는 프로그램을 제작하시오.

$$\text{BMI} = \text{체중(Kg)} / \text{키(m)}^2$$

입력: 체중(Kg)을 먼저 입력한 후 키(m)를 입력한다.

출력: BMI 계산식에 따른 판단 결과를 출력한다.

BMI	20미만	20이상 25미만	25이상 30미만	30이상 40미만	40이상
판단	저체중	정상	과체중	비만	고도비만

입력1	출력1	입력2	출력2
60		85	
1.65	정상	1.75	과체중

Quiz 2. 로그인 처리 프로그램을 제작하시오.

입력: 아이디를 입력한 후 비밀번호를 입력한다.

출력: 아이디는 "admin", 비밀번호는 "test"를 입력했을 경우 "Hello"를 출력하고, 틀릴 경우 "fail"을 출력

입력1	출력1	입력2	출력2
amdin		admin	
koko	fail	test	Hello

Quiz 3. 로그인 처리 프로그램을 개선하려고 한다.

입력: 아이디를 입력한 후 비밀번호를 입력한다.

출력: 아이디와 비밀번호를 모두 입력한 후에 아이디는 "admin"을 입력하지 않으면 "wrong id"를 출력한다. 비밀번호는 "test"를 입력하면 "Hello"를 출력하고 그렇지 않은 경우 "wrong pw"를 출력하시오.

입력1	출력1	입력2	출력2
test	wrong id	admin	wrong pw
koko		kim	

Quiz 4. 연도를 입력해서 윤년이면 1을, 아니면 0을 출력하는 프로그램을 작성하시오.

윤년은 다음 조건을 모두 만족해야 한다.

- 연도가 4로 나누어 떨어지면 윤년으로 한다.
- 하지만 100으로 나누어 떨어지는 해는 윤년이 아니다.
- 단, 400으로 나누어 떨어지면 윤년이다.

입력1	출력1	입력2	출력2
2000	1	2021	0

Quiz 5. 나이를 입력해서 17, 18, 19세이면 "출입허용", 그렇지 않은 경우 "출입금지"를 출력하는 프로그램을 작성하시오.

입력1	출력1	입력2	출력2
12	출입허용	18	출입금지

Quiz 6. 평면 상의 좌표를 입력해서 어느 사분면에 속하는지 출력하는 프로그램을 작성하시오. 단, x와 y는 0이 아닌 정수임.

입력1	출력1	입력2	출력2
2	1	-3	2
5		5	

Quiz 7. 삼각형 세 변의 길이를 입력하여 직각삼각형이면 1, 아니면 0을 출력하는 프로그램을 프로그램을 작성하시오. 단, 세 변의 길이는 변의 길이에 상관없이 입력한다.

입력1	출력1	입력2	출력2
3	1	5	0
5		5	
4		5	

해설

Quiz 1. BMI 계산

```
weight =float(input("체중(kg): ")) # 체중은 실수로 입력받는다.
height =float(input("키(m): "))    # 키는 실수로 입력받는다. 165cm : 1.65
bmi =weight /height**2            # 제곱: 연산자 **를 사용하거나 height*height
if bmi>40:
    print("고도비만")
elif bmi>=30:
    print("비만")
elif bmi>=25:
    print("과체중")
elif bmi>=20:
    print("정상")
else:
    print("저체중")
```


Quiz 2. 로그인 처리

아이디와 비밀번호를 모두 입력받은 후에 조건문을 진행한다. 논리 연산자 and를 사용하여 아이디와 비밀번호가 모두 제시하는 단어와 일치할 때 "Hello"를 출력하고 그렇지 않은 경우 "Fail"을 출력한다.

```
id=input("ID : ")
pw=input("PW : ")
if id=="admin" and pw=="test":
    print("Hello")
else:
    print("fail")
```

if ~ else에서 처리할 문장이 하나라면 다음과 같이 코딩할 수 있다.

```
print("Hello"if id=="admin"and pw=="test"else "fail")
```

변수에 결과를 저장한 후 사용할 수도 있다.

```
result ="Hello"if id=="admin"and pw=="test"else "fail"
print(result)
```

Quiz 3. 로그인 처리 개선

논리연산자 and를 사용하여 아이디와 비밀번호 일치여부를 판단하는 인증시스템의 문제점은 무엇일까? ID와 비밀번호 모두 정확하게 입력해야 "Hello"를 출력한다. "Fail"을 출력하는 경우에 ID가 틀렸는지 비밀번호가 틀렸는지 알 수 없다. 중첩 if를 사용하여 로그인 시스템을 개선해보자.

```
id=input("ID : ")
pw=input("PW : ")
if id=="admin":    # id가 admin이라면
    if pw=="test":    # pw가 test라면
        print("Hello")
    else:            # id는 일치하지만 비밀번호가 틀린 경우
        print("Wrong PW")
else:                # id가 admin이 아닌 경우
    print("Wrong ID")
```

Quiz 4. 윤년 판별

윤년 계산을 위해서는 나머지 연산자 %를 사용할 수 있어야 한다. 나머지 연산자는 홀짝을 판별하거나 배수를 구할 때에도 사용한다.

$n \% 2 = 0$ n 을 2로 나눈 나머지가 0이면 --- 이 조건을 만족하면 짝수를 의미
 $n \% 3 = 0$ n 을 3으로 나눈 나머지가 0이면 --- 3의 배수는 모두 만족

```
year =int(input())
if year%4==0:      # year%4==0   4로 나눈 나머지가 0
    if year%100==0: # 100으로 나누어 떨어지면 평년
        if year%400==0: # 100으로 나누어 떨어지지만 400으로 나누어 떨어지면 윤년
            print(1)
        else:
            print(0)
    else:          # 100으로 나누어 떨어지지 않으면 윤년
        print(1)
else:              # 4로 나누어 떨어지지 않으면
    print(0)
```

논리 연산자(and, or)를 사용하면 다음과 같이 코딩을 줄일 수 있다.

```
year =int(input())
if year%4==0 and year%100!=0 or year%400==0:
    print(1)
else:
    print(0)
```

다음 코딩 방법이 익숙해진다면 더욱 간결한 코딩을 작성할 수 있다.

```
year =int(input())
result =1 if year%4==0 and year%100!=0 or year%400==0 else 0
print(result)
```

Quiz 5. 특정 나이만 출입 허용

조건식은 입력한 나이의 범위가 17이상 19이하일 때로 지정해야 한다. 파이썬은 논리 연산자를 사용할 수 있지만 일반 수학식에서 사용하는 연산식도 지원한다.

논리연산자	연산식
<code>age>=17 and age<=19</code>	<code>17<= age <=19</code>

단, 다른 프로그래밍 언어에서는 지원하지 않는 경우가 많다.

```
age =int(input())
print("출입허용"if 17<=age<=19 else "출입금지")
```

Quiz 6. 좌표 입력 후 사분면 판단

x, y 좌표를 입력(단, x, y는 0을 입력하지 않음)

2사분면	1사분면	if문에 사용할 조건식을 만드는 법은 다양하다. 논리 연산자를 사용하는 사람도 있고, 중첩 if문을 사용하는 사람도 있을 것이다.
3사분면	4사분면	

◦ 논리 연산자를 사용하는 경우

```
if x>0 and y>0: r=1
elif x>0 and y<0: r=4
elif x<0 and y>0: r=2
else: r=3
```

◦ 중첩 if문을 사용하는 경우

```
if x>0: r =1 if y>0 else 4
else: r =2 if y>0 else 3
```

가독성이 떨어지지만 더욱 간결하게 코딩하면 다음과 같다.

```
x=int(input())
y=int(input())
r =(1 if y>0 else 4) if x>0 else (2 if y>0 else 3)
print(r)
```

Quiz 7. swap, 피타고라스의 정리

삼각형 세 변의 길이를 입력한 후 변의 길이가 작은 것부터 큰 순으로 나열할 수 있어야 한다. 변수의 값을 서로 교체하는 방법을 swap이라고 한다.

```
a=5
b=3
a=b # b의 값이 a에 저장된다. 변수 a에는 3이 저장된다.
b=a # a의 값이 b에 저장된다. 위에서 a가 3이므로 b도 3이다.
print(a,b) # 3 3
```

위와 같이 두 변수의 값을 서로 교환하기 위해서 대부분의 프로그래밍 언어들은 새로운 변수가 하나 더 필요했다. 파이썬도 같은 방법으로 코딩할 수 있으나 새로운 방법을 지원한다.

변수 a와 b의 값을 서로 교환하는 방법은 다음과 같다.

기존 프로그래밍	파이썬 지원
temp = a a=b b=temp	a, b = b, a

```
a,b=5,3 # 변수를 선언할 때도 줄여서 코딩 가능 a=5, b=3
a,b =b,a # a와 b의 값을 서로 바꾼다.
print(a,b) # 3 5
```

삼각형 세 변의 길이를 변수 a, b, c에 저장하자. 변수 c에 가장 큰 값이 있으면 피타고라스의 정리를 이용하여 문제를 해결할 수 있다.

```
a=int(input())
b=int(input())
c=int(input())

if a>b: a,b =b,a # a가 b보다 크면 두 변수의 값을 바꾼다.
if b>c: b,c =c,b # b가 c보다 크면 두 변수의 값을 바꾼다.

print(a,b,c)
```

이 방법은 오름차순으로 정렬하는 방법이 아니라 c에 가장 큰 값이 저장되도록 하는 방법이다.

참고

변수 3개를 한 줄에 입력하려면

파이썬 자료형 중에 리스트(list)를 사용해서 한 줄에 입력할 수 있다.

```
temp=input().split()  
print(temp)
```

<실행 결과>

```
23 12 45  
['23', '12', '45']
```

input() 함수는 입력한 문자를 모두 문자열로 저장하기 때문에 정수를 입력했어도 위와 같이 문자 형태로 저장한다.

반복문이나 함수를 통해 숫자형으로 변환할 수 있다. 자세한 방법은 다음 단원을 참고하자. map() 함수를 이용해서 리스트의 내용을 정수로 변환하여 변수에 할당할 수 있다.

```
a,b,c=map(int, input().split())
```

input().split()을 통해 생성된 리스트(list)를 정수형으로 변환하는 int() 함수를 적용하여 변수 a,b,c에 할당한다. 입력한 문자열이 3개가 아니라면 다음과 같은 에러가 발생한다.

- not enough values to unpack (expected 3, got 2) 2개만 입력한 경우
- too many values to unpack (expected 3) 3개를 초과해서 입력한 경우

2.2. 반복문

조건문(if)이 프로그램의 흐름을 변경할 수 있었다면, 반복문은 프로그램의 흐름을 반복한다. 파이썬에서 제공하는 반복문은 while문과 for문이다.

while	for
조건에 의한 반복 조건이 참인 동안 반복 실행한다.	계수에 의한 반복 지정한 숫자 범위, 혹은 자료 범위내에서 반복 실행한다.

2.2.1. while

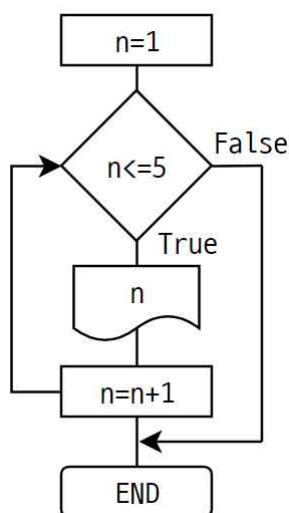
While은 "~하는 동안에"라는 뜻을 가진 단어이다. 조건이 참인 동안 반복 실행된다.

while 조건:

반복 실행할 명령 *#들여쓰기(4칸)*

1부터 5까지 출력하는 예제를 통해 while문의 사용법을 살펴보자.

n=1		실행 결과
while n<=5:	<i># 조건(n이 5이하이면)이 만족하면</i>	1
print(n)	<i>#n을 출력한다.</i>	2
n=n+1	<i>#n값을 1 증가시킨다.</i>	3
		4
		5



if문과 같이 조건식 뒤에 콜론(:)이 있으며, 조건식이 만족하는 동안 반복 실행한다. 반복할 코드는 들여쓰기를 사용한다.

n=n+1이 없다면 n 값이 계속 1이고, 조건(n<=5)을 항상 만족하기 때문에 1만 출력된다.

5부터 1까지 감소하면서 출력하는 방법을 살펴보자.

n=5		실행 결과
while n>0:	# 조건(n이 0이상이면)이 만족하면	5
print(n)	#n을 출력한다.	4
n=n-1	#n값을 1 감소시킨다.	3
		2
		1

조건을 다음과 같이 변경해도 같은 결과가 나타난다.

while n>0:	▷	while n:
------------	---	----------

n 값이 0이 되면 while 0:이 되므로 조건이 거짓이 되고, 반복문은 더 이상 실행되지 않는다.

참고

무한루프

비밀번호를 올바르게 입력할 때까지 계속 비밀번호 입력을 요구하는 경우가 있다. 의도적으로 무한루프를 발생시킬 때 파이썬에서는 다음과 같이 사용한다.

```
while True:
    반복실행할 명령
```

비밀번호 "test"를 입력할 때까지 계속 비밀번호를 입력해야 하는 프로그램을 제작해보자. 비밀번호를 틀리게 입력하면 "비밀번호 오류!"라고 출력한 후 비밀번호를 다시 입력해야 한다. 단, break 명령은 사용하지 않는다.

```
login=True      # 반복문 조건에 사용할 변수(login)을 True로 지정
while login:    # login이 True이기 때문에 무한 반복
    pw =input("pw:")    # 비밀번호를 입력 받는다.
    if pw=="test":      # 비밀번호가 test이면
        login=False    # login을 False로 설정. 다음 반복문 조건이 실행되지 않음
        print("접속 성공")
    else:
        print("비밀번호 오류!")
```

2.2.2. for

while문은 조건을 이용하여 반복 실행할 여부를 결정했다면, for문은 순회 가능한 객체의 처음부터 마지막까지 반복하는 명령문이다. for문의 형식은 다음과 같다.

```
for 변수 in 순회 가능한 객체:  
    반복 실행할 명령
```

순회 가능한 객체에는 다음 단원에서 다룰 리스트, 튜플이 있으며, 문자열이나 range()도 가능하다. 1부터 5까지 출력하는 반복문을 for문으로 작성하면 다음과 같다.

```
for a in [1,2,3,4,5]:  
    print(a,end='')
```

실행 결과
1 2 3 4 5

순회 가능한 객체로 문자열을 사용할 수 있다.

```
for a in "python":  
    print(a,end='')
```

실행 결과
1 2 3 4 5

문자열에서는 한글 처리도 별다른 처리없이 실행된다.

```
for a in "프로그래밍":  
    print(a,end='')
```

실행 결과
프 로 그 래 밍

◦ range(n)

0부터 100까지 print 명령으로 모두 작성하는 것은 너무나 비효율적이다. 반복할 수가 정해졌을 때는 range를 사용한다.

```
for a in range(100):
```

range(100)을 사용하면 0부터 99까지 100개의 숫자를 생성하여, 변수 a에 저장하여 100번을 반복한다. range(n)일 경우 0부터 n-1까지 수만큼 반복되므로 0부터 100까지 반복하기 위해서는 다음과 같이 range 범위를 101로 정해야 한다.

```
for a in range(101):
```

◦ range(시작값, 종료값)

프로그램을 작성할 때, 0부터 시작하지 않고 특정 숫자부터 반복해야 할 경우가 있다. range에서 시작값과 종료값을 정할 수 있다.

```
for a in range(x, y):
```

주의할 점은 range의 범위는 x부터 y-1까지 반복한다. range에서 종료값보다 1이 작은 수까지 반복된다는 것을 기억하자.

```
for a in range(1,5):  
    print(a,end='')
```

실행 결과
1 2 3 4

1부터 10까지 홀수만 출력하자.

```
for a in range(1,11):  
    if a%2==1:    #a를 2로 나눈 나머지가 1이면  
        print(a,end='')
```

실행 결과
1 3 5 7 9

◦ range(시작값, 종료값, 증감값)

위 반복문은 10번을 실행하면서 조건문을 통해 홀수만 출력했다. if문을 사용하지 않고 range에서 증감값을 이용하여 홀수만 출력한다면 반복문의 실행 횟수를 줄일 수 있다.

```
for a in range(1,10,2):  
    print(a,end='')
```

실행 결과
1 3 5 7 9

for a range(1,10,2):

range(1,10)을 사용할 경우 1부터 9까지 1씩 증가하는 것이다. 1씩 증가하면 증감값을 생략할 수 있지만 다른 증감값이면 세 번째 지정 값을 사용할 수 있다. range(1,10,2)는 1부터 9까지 2씩 증가하기 때문에 1, 3, 5, 7, 9 총 5회 반복한다.

range() 함수에 증감값만 추가시킨 간단한 예제로 보이지만, 반복문의 실행 횟수를 반으로 줄인 효율적인 방법이다. 컴퓨터의 빠른 실행속도로는 아주 미미한 차이지만, 실행속도를 중요시하는 프로그래밍에서는 반복문 구조도 효율적으로 구성해야 한다.

증감값을 사용하여 역순으로 출력하는 것도 가능하다.

```
for a in range(5,0,-1):  
    print(a,end='')
```

실행 결과
5 4 3 2 1

reversed() 함수를 이용한 출력 방법도 확인해보자. 반복문에 사용하는 변수를 어떻게 사용할지 여부에 따라 각 상황에 맞는 방법을 선택할 수 있다.

```
for a in reversed(range(1, 6)):  
    print(a, end='')
```

```
for b in range(5):    # b의 값은 5회 반복(0,1,2,3,4)  
    print(5-b, end='') # b의 값을 그대로 출력하지 않고 5-b로 역순 출력이 가능
```

입력한 정수에 해당하는 구구단 중 5단을 출력해보자.
for와 while을 사용한 예제로 두 반복문의 사용법을 익혀보자.

while 사용

```
a=5
b=1
while b<=9:
    print(f"{a}*{b}={a*b}")
    b+=1
```

<실행 결과>

```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
```

for 사용

```
a =5
for b in range(1,10):
    print(f"{a}*{b}={a*b}")
```

◦ 중첩 반복문

조건문 안에 조건문을 사용했던 중첩 if와 마찬가지로 반복문 안에 다른 반복문을 사용하는 방법이다. 구구단 전체를 출력하는 프로그램을 분석해보자.

```
a=2
b=1
while a<=9:
    while b<=9:
        print(f"{a}*{b}={a*b}")
        b+=1
    a+=1
```

2단부터 9단까지 출력하려고 했지만 2단만 출력된다. 2단만 출력되는 이유를 무엇일까?
변수 a는 2부터 시작한다. a<=9를 만족하기 때문에 두 번째 while b<=9를 실행하여 2단을 출력한다. 이후에 a는 값을 1 증가하여 3이 된다. 그렇지만, b의 값은 10이 되었기 때문에 두 번째 while문이 실행되지 않는다. a에 해당하는 구구단을 출력하 후 a 값을 1 증가하면서 b의 값은 다시 1부터 설정해야 한다.

```
a=2
b=1
while a<=9:
    while b<=9:
        print(f"{a}*{b}={a*b}")
        b+=1
    a+=1
    b=1
```

while을 사용할 경우 변수의 초기값과 증감값을 정확히 설정해야 한다. for를 사용할 경우에는 range()에서 범위 설정을 통해 훨씬 간결한 코딩으로 작성할 수 있다.

```
for a in range(2,10):
    for b in range(1,10):
        print(f"{a}*{b}={a*b}")
```

2.3. break

break는 반복문을 벗어나기 위해 사용한다. 반복문을 실행하는 도중 특정 조건을 만족할 경우 반복문을 끝낼 수 있다. 1부터 1000까지 출력하면서 변수 a가 5일 때 반복문을 끝내는 예제를 while과 for를 이용해보자.

```
while
a=1
while a<=1000:
    print(a)
    if a==5:
        break
    a+=1
```

```
for
for a in range(1,1000):
    print(a)
    if a==5:
        break
```

while을 사용할 경우 a를 언제 증감시키는지에 따라 a의 초기값을 변경하거나 break의 위치가 달라져야 한다.. print앞에 위치할 경우 다음과 같이 변경해야 한다.

초기값 변경

```
a=0
while a<=1000:
    a+=1
    print(a)
    if a==5:
        break
```

if 조건식 변경

```
a=1
while a<=1000:
    if a>5:
        break
    print(a)
    a+=1
```

-1을 입력하기 전까지 계속 정수를 입력받으면 해당 수의 제곱값을 출력하는 프로그램을 작성해보자. 특정 조건이 만족하기 전까지 반복할 경우에는 for 보다는 while을 사용할 것을 추천한다.

```
while True:
    n=int(input())
    if n==-1 : break    # break은 반복문이 종료되기 때문에 if~else를 미사용
    print(n**2)
```

◦ 중첩 반복문에서 break

구구단 전체를 출력하면서 구구단의 결과가 20 이상일 경우에는 출력을 종료하는 프로그램을 작성하려고 한다.

```
for a in range(2,10):
    for b in range(1,10):
        if a*b>=20:    # a*b값이 20이상이면 반복문 종료
            break
        print(f"{a}*{b}={a*b}")
```

2단을 출력한 후 3단은 3*6=18까지 출력한 후 종료해야겠지만, 실행 결과는 각 단에서 20미만의 값이 모두 출력된다. break 명령은 두 번째 반복문만 탈출하기 때문에 나타나는 현상이다.

3단에서 3*7의 값이 20이상이기 때문에 break 된다. 하지만 a값이 1 증가하고, 두 번째 반복문은 다시 b가 1부터 시작한다.

이중 반복문을 탈출하기 위해 조건을 설정할 수 있는 변수를 따로 설정한다.

```
ck=False    #첫 번째 반복문을 탈출하기 위한 변수
for a in range(2,10):
    for b in range(1,10):
        if a*b>=20:
            ck=True    # ck 값을 변경
            break      # 두 번째 반복문 탈출
        print(f"{a}*{b}={a*b}")
    if ck:    # ck가 True로 변경됐다면
        break # 첫 번째 반복문 탈출
```

while을 사용할 경우에는 변수 a값의 변경을 통해 이중 반복문을 탈출할 수 있다.

```
a=2
while a<=9:
    b=1
    while b<=9:
        if a*b>=20:
            a=10 # a<=9 조건이 거짓이 될 수 있는 값으로 변경(10이상의 값)
            break
        print(f"{a}*{b}={a*b}")
        b+=1
    a+=1
```

2.4. continue

반복문을 유지하면서 코드 실행만 건너뛴다. 1부터 10까지 3의 배수를 제외한 숫자만 출력하는 예제를 보자.

```
for a in range(1,11):  
    if a%3==0:  
        continue  
    print(a, end='')
```

실행 결과
1 2 4 5 7 8 10

반복문 내에 `a%3==0` 조건이 만족하면 `continue` 뒤에 반복 실행할 명령은 실행되지 않고 건너뛴다.

a	조건(<code>a%3==0</code>)	실행 결과
1	False	1을 출력
2	False	2를 출력
3	True	반복문 건너뛴
4	False	4를 출력

`continue`를 사용하지 않더라도 `if`문의 조건식을 이용하여 구현할 수 있다.

```
for a in range(1,11):  
    if a%3!=0:  
        print(a, end='')
```

반복문 내에 반복 실행할 명령이 많을 경우 `continue` 사용을 추천한다.

응용문제

곱하기 연산 문제를 제시하는 퀴즈 프로그램을 제작하시오. 난수(임의의 수)를 사용하여 곱하기 문제를 출제하며, 난수 사용법은 다음과 같다.

```
import random
a=random.random()    # 0과 1사이의 난수(실수)를 생성   $0 < x < 1$ 
b=random.randint(10,20)    # 10부터 20까지 난수(정수)를 생성   $10 \leq x \leq 20$ 
print(a, b)
```

<실행 결과>

0.2978531803454205 11

※ 실행할 때마다 난수를 출력한다.

random은 파이썬에서 제공하는 라이브러리 중에 난수를 발생하는 모듈이다. 라이브러리와 모듈은 함수 단원에서 자세히 다룰 것이며, 지금은 난수를 발생하는 기능만 사용해보자.

• 요구사항

1. 한 자리 수 곱하기 10문제를 출제한다.
2. 각 문제의 답을 입력할 때마다 정답 여부와 누적 정답 수와 오답 수를 출력한다.

• 요구사항 분석 및 코딩 설계

- ① 변수 a, b에 1부터 9까지 난수를 생성한다.

```
import random
a=random.randint(1,9)
b=random.randint(1,9)
```

1부터 9까지 난수가 생성된 것을 확인하려면 print(a, b)를 입력한 후 실행하면 결과를 확인할 수 있다.

- ② 곱하기 문제를 출제하기 위한 문제를 문자열 서식을 이용해서 출력한다.

예) $3 \times 5 =$

f string 포매팅을 사용하여 퀴즈 문장을 생성한다.

```
a=random.randint(1,9)
b=random.randint(1,9)
print(f"{a}*{b}=")
```

③ 반복문을 사용하여 10문제를 생성하여 출력한다.

문제를 입력받으려면 문제를 출력한 후에 줄바꿈을 하면 다음 줄에 입력해야한다.
줄 바꿈을 막기 위해 `print(f"{a}*{b}=", end='')`를 사용할 수 있고, 퀴즈 문장 전체를 변수에 지정하여 `input()`함수에 사용할 수 있다.
문제 번호까지 지정하기 위해 다음과 같이 변경해보자.

```
for n in range(1,11):
    a=random.randint(1,9)
    b=random.randint(1,9)
    quiz =f"문제{n}. {a}*{b}="
    print(quiz)
```

④ 입력한 답을 저장할 변수를 지정한다.

③ 과정에서 `print(quiz)`를 사용하여 10문제를 출력했다면, `input()` 함수를 사용하여 10문제를 출제한 후 입력을 받는다.

```
quiz =f"문제{n}. {a}*{b}="
answer =int(input(quiz))
```

10문제를 모두 입력해야 프로그램이 종료된다. 테스트를 위해서 반복문의 실행 횟수를 줄이거나 중간에 탈출(CTRL+C)할 수 있다.

⑤ 답을 입력받은 후 정답여부와 누적 정답수와 오답수를 출력한다.

④ 과정에서 답을 입력한 후에 조건문을 사용한다.

```
import random
good, wrong =0, 0 #정답, 오답
for n in range(1,11):
    a=random.randint(1,9)
    b=random.randint(1,9)
    quiz =f"문제{n}. {a}*{b}="
    answer =int(input(quiz))
    if answer ==a*b:
        print("정답입니다.")
        good+=1
    else:
        print("오답입니다.")
        wrong+=1
    print(f"정답({good}), 오답({wrong})\n")
```

<실행 결과>

문제1. 9*3=27
정답입니다.
정답(1), 오답(0)

문제2. 4*9=36
정답입니다.
정답(2), 오답(0)

....중간 생략....

문제10. 7*7=20
오답입니다.
정답(6), 오답(4)

퀴즈

Quiz 1. 입력한 정수의 약수를 한 줄로 출력하시오.

입력: 2이상 1000000 미만의 정수를 입력한다.

출력: 입력한 정수의 약수를 한 칸 간격으로 출력한다.

입력

6

출력

1 2 3 6

Quiz 2. 입력한 정수의 약수의 개수를 출력하시오.

입력: 2이상 1000000 미만의 정수를 입력한다.

입력

6

출력

4

Quiz 3. 입력한 정수가 소수이면 1, 아니면 0을 출력하시오.

입력: 2이상 1000000 미만의 정수를 입력한다.

입력1

11

출력1

1

입력2

25

출력2

0

Quiz 4. 2부터 20까지 각 수의 약수를 한 줄로 출력하시오.

출력

2: 1 2

3: 1 3

생략

19: 1 19

20: 1 2 4 5 10 20

Quiz 5. 2부터 100까지 소수를 한 칸 단위로 출력하시오.

출력

2 3 5 7(생략)... 83 89 97

Quiz 6. n개의 정수를 입력한 후 최댓값과 최솟값을 출력하시오.

입력: 2이상 100 미만의 정수를 입력한다.

출력: 입력받은 정수 중에 최솟값, 최댓값을 출력

<u>입력1</u>	<u>출력1</u>	<u>입력2</u>	<u>출력2</u>
5	3	4	26 200
10		100	
21		50	
42		200	
3		26	
29			

Quiz 7. $1+2+3+4+\dots+n > 1000$ 을 만족하는 n의 최솟값을 구하는 프로그램을 작성하시오.

Quiz 8. 입력한 정수를 소인수 분해 하시오.

입력: 2 이상 100000000이하의 정수를 입력한다.

출력: 소인수 분해 결과는 각 숫자를 한 칸 단위로 출력한다.

<u>입력</u>	<u>출력</u>
24	2 2 2 3

Quiz 1. 약수 출력

정수를 입력받아서 프로그램을 작성하면 실행 결과를 확인할 때마다 정수를 입력해야 하는 불편함이 있다. 변수에 값을 미리 지정한 후 프로그램 결과가 완성되면 입력받는 형식으로 변경하는 것이 효율적이다.

```
n=6 #6으로 지정한 후 프로그램 작성한 후에 입력받는 형식으로 변경
for a in range(1, n+1):
    print(n, a, n%a)
    #n, 반복(2부터 n까지), n을 a로 나눈 나머지를 출력
```

<실행 결과>

```
6 1 0
6 2 0
6 3 0
6 4 2
6 5 1
6 6 0
```

정수 n(6)을 a(1,2,3,4,5,6)로 나눈 나머지가 0일 때 a가 n의 약수이다.

조건문을 사용하여 n을 a로 나눈 나머지가 0일 때만 a를 출력한다.

```
n=int(input())
for a in range(1, n+1):
    if n%a==0:
        print(a, end='')
```

이 프로그램에서 n의 약수를 출력하기 위해서는 반복문을 n번 실행해야 한다. 반복문을 실행하는 횟수를 줄일 수 있을까?

100의 약수를 출력하면 1 2 4 5 10 20 25 50 100이다.

n의 약수에서 n을 제외한 가장 큰 약수는 n/2이다. 반복문을 n번 실행하지 않고 절반까지만 실행해도 구할 수 있다.

반복문을 for a in range(1, n/2+1)으로 변경한다면 다음과 같은 에러가 발생한다.

'float' object cannot be interpreted as an integer

n/2로 설정하면 실수로 변환되기 때문에 발생하는 에러이다. n/2에 int() 함수를 사용하여 정수형으로 변환하면 에러를 방지할 수 있다.

```
n=int(input())
for a in range(1, int(n/2)+1):
    if n%a==0:
        print(a, end='')
print(n)
```

Quiz 2. 약수의 개수 출력

약수를 출력하지 않고 약수의 개수를 세기 위한 변수가 필요하다.

```
n=6 #제대로 실행된다면 int(input())으로 변경
cnt=0 #약수의 개수를 저장하기 위한 변수
for a in range(1, n+1):
    if n%a==0:
        cnt+=1 #cnt 변수의 값을 1씩 증가
print(cnt)
```

약수의 개수를 구할 때에도 반복문의 횟수를 줄일 수 있는 방법을 생각해보자. 1은 항상 나누어 떨어지며, $n/2$ 까지만 실행하면 된다는 것을 적용한다.

이 때 1과 n 은 무조건 나누어 떨어지기 때문에 `range(1, n+1)`을 `range(2, int(n/2)+1)`로 변경하고, `cnt=0` 대신 `cnt=2`로 설정한다.

```
n=int(input())
cnt=2 #1과 자기 자신의 수는 미리 포함
for a in range(2, int(n/2)+1):
    if n%a==0:
        cnt+=1 #cnt 변수의 값을 1씩 증가
print(cnt)
```

Quiz 3. 소수 판별

약수의 개수를 출력(Quiz 2)했다면 이를 이용해서 소수 여부를 판별할 수 있다.

소수란 1과 자기 자신으로만 나누어 떨어지는 수이다. 약수의 개수가 2이면 소수라고 할 수 있다.

```
n=int(input())
cnt=0
for a in range(1, n+1):
    if n%a==0:
        cnt+=1 #cnt 변수의 값을 1씩 증가
if cnt==2: #cnt가 2이면... 약수가 2개라면..
    print(1)
else:
    print(0)
```

반복 실행 횟수를 줄이기 위한 방법을 생각해보자.

- `for a in range(2, int(n/2)+1)`로 변경하여 반복문의 실행 횟수를 줄인다.
- `print(1 if cnt==2 else 0)`을 사용하여 조건문 코딩을 줄인다.

이 방법보다 훨씬 더 효율적으로 소수 여부를 판별할 수 있다. 10000이 소수인지 아닌지 물어본다면 바로 소수가 아니라고 대답할 수 있다. 2로 나누어 떨어진다는 것을 알 수 있기 때문이다.

1과 자기 자신이 아닌 다른 수로 나누어 떨어졌다면, 더 이상 소수 여부를 확인하기 위해 반복문을 실행할 필요가 없다. 반복문을 탈출할 수 있는 break를 사용해보자.

```
n=int(input())
cnt=1 # 1이면 소수.. 모든 수를 소수라고 가정한다.
for a in range(2, int(n/2)+1):
    if n%a==0: #1과 자기 자신의 수를 제외한 수로 나누어 떨어지면
        cnt=0 #0으로 변경. 소수가 아님(합성수)
        break #반복문 탈출
print(cnt)
```

Quiz 4. 2부터 20까지 각 수의 약수를 한 줄로 출력

방법 1

```
for a in range(2, 21):
    print(a,":", end='')
    for b in range(1, a+1):
        if a%b==0:
            print(b, end='')
    print()
```

방법 2

```
for a in range(2, 21):
    print(a,": 1", end='')
    for b in range(2, int(a/2)+1):
        if a%b==0:
            print(b, end='')
    print(a)
```

방법2는 Quiz 1번에서 3번까지 실행 횟수를 적용한 방법을 적용한 것이다.

두 방법 모두 다음과 같은 결과가 나타난다.

<실행 결과>

```
2 : 1 2
3 : 1 3
...생략...
19 : 1 19
20 : 1 2 4 5 10 20
```

Quiz 5. 2부터 100까지 소수를 출력

Quiz 4를 변경하여 약수를 출력하지 않고 약수의 개수를 구하여, 약수가 2개일 때만 해당 정수를 출력하면 소수를 출력할 수 있다.

```
for a in range(2, 101):
    cnt=True    #cnt의 값을 0,1 대신 boolean 값으로 설정할 수 있다.
    for b in range(2, a):    #int(a/2)+1로 변경하면 더 효율적
        if a%b==0:    #1과 자기 자신을 제외한 수로 나누어 떨어지면
            cnt=False    # cnt 값을 False로 변경
            break
    if cnt:
        print(a, end='')
```

<실행 결과>

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

Quiz 6. 정수 n개의 최댓값과 최솟값을 출력

```
n=int(input())
vmax =0    #최댓값으로 사용할 값은 입력한 정수의 가장 작은 범위의 값을 지정한다.
for a in range(n):
    v=int(input())
    if v>vmax:    #입력받은 값이 현재의 최댓값(vmax)보다 크면
        vmax=v    #최댓값(vmax)을 입력받은 값(v)으로 변경
print(vmax)
```

최댓값을 구하는 방법을 구현했다면, 최솟값과 함께 적용할 수 있다.

```
n=int(input())
vmax =0
vmin =1000000000
for a in range(n):
    v=int(input())
    if v>vmax: vmax=v
    if v<vmin: vmin=v
print(vmin, vmax)
```

Quiz 7. $1+2+3+4+\dots+n > 1000$ 을 만족하는 n 의 최솟값 출력

반복문을 사용하여 $1+2+3+4+\dots+100$ 을 구해보자.

for <code>s=0</code> <code>for a in range(1,101):</code> <code>s+=a</code> <code>print(s)</code>	while <code>s, n=0, 1</code> <code>while n<=100:</code> <code>s+=n</code> <code>n+=1</code> <code>print(s)</code>
---	--

반복 실행할 횟수를 사용할 경우에는 for문이 간결하다. 이 문제와 같이 조건에 의한 반복은 while을 추천한다. for를 사용할 경우에는 $1+2+3+\dots+n > 1000$ 을 만족하는 n 의 값이 1000 이하의 값일 것이다. 반복할 횟수를 1000으로 설정한 후에 break를 이용해 반복문을 중단할 수 있다.

for <code>s=0</code> <code>for a in range(1,1000):</code> <code>s+=a</code> <code>if s>1000:</code> <code>print(a)</code> <code>break</code>	while <code>s, n=0, 0</code> <code>while s<=1000:</code> <code>n+=1</code> <code>s+=n</code> <code>print(n)</code>
--	---

Quiz 8. 소인수 분해

소인수 분해하는 과정을 살펴보자.

20: $2 \times 2 \times 5$	65: 5×13	50: $2 \times 5 \times 5$
$\begin{array}{r l} 2 & 20 \\ 2 & 10 \\ 5 & 5 \\ \hline & 1 \end{array}$	$\begin{array}{r l} 5 & 65 \\ 13 & 13 \\ \hline & 1 \end{array}$	$\begin{array}{r l} 2 & 50 \\ 5 & 25 \\ 5 & 5 \\ \hline & 1 \end{array}$

소인수 분해 과정을 분석해보자.

1. 정수 n 을 2로 나눈다.
2. 나누어 떨어지면 n 의 값은 2로 나눈 몫으로 설정한다. 나누어 떨어지지 않으면 나누려는 값을 1씩 증가한다.
3. n 이 1이 될 때까지 반복한다.

```

n=int(input())
d=2 #나누려는 값을 2부터 설정
while n!=1: #n이 1이 아닐때까지 반복. (n이 1이면 반복 종료)
    if n%d==0: #n이 d로 나누어 떨어지면
        print(d, end=' ') #d를 출력하고
        n//=d #n은 d로 나눈 몫으로 변경한다.
    else: #n이 d로 나누어 떨어지지 않으면
        d+=1 # d의 값을 1씩 증가한다.

```

for를 이용하면 다음과 같이 작성할 수 있다.

```

n=int(input())
d=2
for a in range(n):
    if n==1:
        break
    elif n%d==0:
        print(d, end=' ')
        n//=d
    else:
        d+=1

```

3. 자료형

3.1. 리스트(List)

25명 학생들의 시험 점수를 계산하는 프로그램을 제작한다고 해보자. 개인별 점수를 저장할 변수를 `score1`, `score2`, ... , `score25`까지 25개의 변수가 필요하다. 학생이 100명, 1000명 증가한다면 인원에 맞는 변수의 선언은 비효율적이다.

C언어나 Java와 같은 언어에서는 다음과 같이 배열을 사용한다.

언어	사용방법	설명
C, C++	<code>int score[25];</code>	자료형 변수명[크기]
JAVA	<code>int[] score = new int[25];</code>	자료형[] 변수 = new 자료형[크기]

C, C++, JAVA는 배열을 선언한 후 같은 자료형을 저장하여 사용하는 방식이다. 파이썬에서는 리스트(list)를 사용한다.

```
score = [90, 92, .... , 88, 74]    #25명 학생의 점수를 저장한다.
```

리스트는 자료형을 선언하지 않고 리스트 이름을 선언한 후에 대괄호([]) 사이에 값을 넣어놓고 사용하거나 `list()`를 사용한다. 다른 프로그래밍 언어와 다르게 리스트에는 여러 자료형을 저장할 수 있다.

```
a=[90,92,84,88,74]
b=list((90,92,84,88,74)) # (90,92,84,88,74)을 list()함수로 리스트로 만든다.
c=["파이썬",900,30.5,"자바",850,78.75] #문자열, 숫자(정수, 실수)를 함께 저장
d=list(("파이썬",900,30.5,"자바",850,78.75))
e=list("프로그래밍") #문자를 하나씩 분리하여 리스트로 만든다.
print(a,b,c,d,e, sep='\n') #print할 내용마다 엔터로 분리한다
```

<실행 결과>

```
[90, 92, 84, 88, 74]
[90, 92, 84, 88, 74]
['파이썬', 900, 30.5, '자바', 850, 78.75]
['파이썬', 900, 30.5, '자바', 850, 78.75]
['프', '로', '그', '래', '밍']
```


3.1.1. 리스트 항목 다루기(인덱스, 슬라이싱)

리스트에 값을 사용하기 위해서 인덱스를 이해해야 한다. C, C++, JAVA와 마찬가지로 처음 값의 인덱스는 0이다.

```
a=[90,91,92,93,94,95,96]
print(a[0])
```

<실행 결과>

90

값	90	91	92	93	94	95	96
인덱스	0	1	2	3	4	5	6
	-7	-6	-5	-4	-3	-2	-1

리스트에 사용하는 인덱스는 음수를 지정할 수 있다. 음수로 지정한 경우 마지막 요소부터 접근한다. a[-1]인 경우 마지막 값인 96이다.

리스트 a의 슬라이싱 결과는 다음과 같다.

슬라이싱	결과	설명
a[0:3]	[90, 91, 92]	a[0]부터 a[2]까지 값
a[3:5]	[93, 94]	a[3]부터 a[4]까지 값
a[0:]	[90, 91, 92, 93, 94, 95, 96]	a[0]부터 마지막까지
a[3:]	[93, 94, 95, 96]	a[3]부터 마지막까지
a[:]	[90, 91, 92, 93, 94, 95, 96]	리스트 a 전체
a[1:5:2]	[91, 93]	a[1]부터 a[4]까지 2씩 건너뛴
a[:2]	[90, 92, 94, 96]	처음부터 마지막까지 2씩 건너뛴
a[1::2]	[91, 93, 95]	a[1]부터 마지막까지 2씩 건너뛴
a[-4:-2]	[93, 94]	a[-4]부터 a[-1]까지
a[-4:]	[93, 94, 95, 96]	a[-4]부터 마지막까지
a[100]	IndexError	IndexError 발생
a[5:100]	[95, 96]	a[5]부터 마지막까지(인덱스가 범위를 초과한 경우 마지막까지)

슬라이싱은 문자열에도 동일하게 동작한다.

```
ct = '프로그래밍'
print(ct[1],ct[-1])
```

<실행 결과>

로 밍

3.1.2. 리스트 항목 수정

리스트 prog에는 프로그래밍 언어 4개를 저장하였다. 첫 번째 C를 C++로 수정하려면 prog에 해당 인덱스를 표시하고 내용을 입력하여 수정한다.

```
prog=["C","PYTHON","JAVA","GO"]
prog[0]="C++"
print(prog)
```

<실행 결과>

```
['C++', 'PYTHON', 'JAVA', 'GO']
```

그러나 인덱스를 벗어나 prog[4]="RUBY"를 실행할 경우 에러(IndexError)가 발생된다.
IndexError: list assignment index out of range

리스트 내 항목 값을 수정하는 것은 다른 언어의 배열 값을 수정하는 방법과 같지만 다음 예제를 통해 리스트의 특징을 알아보자.

리스트 복사 1

```
prog=["C","PYTHON","JAVA"]
myprog=prog
print("prog:",prog)
print("myprog:",myprog)
prog[0]="C++"
print("prog:",prog)
print("myprog:",myprog)
```

<실행 결과>

```
prog: ['C', 'PYTHON', 'JAVA']
myprog: ['C', 'PYTHON', 'JAVA']
prog: ['C++', 'PYTHON', 'JAVA']
myprog: ['C++', 'PYTHON', 'JAVA']
```

리스트 복사 2

```
prog=["C","PYTHON","JAVA"]
myprog=prog
print("prog:",prog)
print("myprog:",myprog)
prog[0]="C++"
print("prog:",prog)
print("myprog:",myprog)
```

<실행 결과>

```
prog: ['C', 'PYTHON', 'JAVA']
myprog: ['C', 'PYTHON', 'JAVA']
prog: ['C++', 'PYTHON', 'JAVA']
myprog: ['C', 'PYTHON', 'JAVA']
```

myprog=prog라고 하면 myprog가 prog를 복제했다고 생각했을 것이다. 하지만 prog[0] 항목 값을 수정하고 prog와 myprog를 출력하면 내용이 모두 변경된 것을 확인할 수 있다.

메모리상에 있는 리스트를 여러 변수가 함께 사용하기 때문에 이와 같은 현상이 발생한다. 파이썬에는 얇은 복사라고 한다. 리스트 값 전체를 다른 리스트에 복사하려면 myprog = prog[:]를 사용하여 prog 리스트의 모든 항목 값을 myprog에 저장하는 것이다. 실행 결과를 확인하면 리스트 prog와 myprog의 항목 값이 동일하지 않다.

3.1.3. 리스트 연산

리스트 자료형도 몇 개의 연산을 지원한다. 함수를 사용하지 않고 연산자를 통한 예제를 통해 리스트 연산을 알아보자.

```
prog=["C","PYTHON","JAVA","GO"]
webp=["PHP","JSP","ASP.NET"]
mypro=prog+webp
prog[0]="D"
print("prog:",prog)
print("mypro:",mypro)
```

<실행 결과>

```
['C', 'PYTHON', 'JAVA', 'GO', 'PHP', 'JSP', 'ASP.NET']
['D', 'PYTHON', 'JAVA', 'GO']
```

리스트 prog와 webp를 잇는 연산을 mypro에 저장한다. prgo+webp 연산 이후 prog[0] 항목 값을 수정했지만 리스트 mypro에는 영향이 없는 것을 확인할 수 있다.

리스트 myprog는 prog의 값과 "html"을 잇는 연산을 하려고 한다.

```
prog=["C","PYTHON","JAVA"]
mypro=prog+"HTML"
print(mypro)
```

<실행 결과>

```
TypeError: can only concatenate list (not "str") to list
```

하지만 리스트는 리스트끼리만 잇는 연산이 가능하다. int(정수)나 str(문자) 자료형은 리스트와 연결할 수 없기 때문에 c=prog+["html"]로 변경해보자.

```
prog=["C","PYTHON","JAVA"]
mypro=prog+["HTML"]
print(mypro)
```

<실행 결과>

```
['C', 'PYTHON', 'JAVA', 'HTML']
```

리스트 항목 값을 반복하기 위해 곱 연산을 사용한다.

```
prog=["C","PYTHON","JAVA"]
data=[1,2,3]
print(prog*2)
print(data*3)
```

<실행 결과>

```
['C', 'PYTHON', 'JAVA', 'GO', 'C', 'PYTHON', 'JAVA', 'GO']  
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

만약에 리스트끼리 곱한다면 어떻게 될까? `print(prog*data)`를 실행할 경우 다음과 같은 에러(TypeError)가 발생한다.

`TypeError: can't multiply sequence by non-int of type 'list'`

3.1.4. 리스트 메서드(Method)

1) 위치 : `index`

리스트에서 특정 값의 위치를 반환한다. 특정 값이 여러 개일 경우 가장 낮은 인덱스를 반환하며, 특정 값이 리스트내 존재하지 않을 경우 에러(ValueError)가 발생한다.

```
prog=["WEB", "OA", "Multi", "OA"]  
print(prog.index("OA"))
```

<실행 결과>

```
1
```

리스트 a에 없는 값의 `index`를 구할 경우 다음과 같은 에러가 발생한다.

```
prog=["WEB", "OA", "Multi", "OA"]  
print(prog.index("python"))
```

<실행 결과>

`ValueError: 'python' is not in list`

다음과 같이 대소문자를 변경한 경우는 어떻게 될까?

```
a=["WEB", "OA", "Multi", "OA"]  
print(a.index("oa"))
```

<실행 결과>

`ValueError: 'oa' is not in list`

`index()`는 대소문자를 구별한다는 것을 확인할 수 있다.

2) 추가 : append, extend, insert

◦ append()

리스트의 마지막에 값을 추가할 수 있다. 리스트는 자료형의 제한이 없기 때문에 모든 자료형을 추가할 수 있다.

```
a=[1,2,3,4]
b=["WEB","OA","Multi"]
a.append(5)           #[1, 2, 3, 4, 5]
a.append("python")    #[1, 2, 3, 4, 5, python]
a.append([6,7])       #[1, 2, 3, 4, 5, python, [6,7]]
a.append(b)           #[1, 2, 3, 4, 5, 'python', [6, 7], ['WEB', 'OA', 'Multi']]
```

◦ extend()

리스트의 마지막에 값을 추가할 수 있다. 하지만 append()와 달리 두 리스트를 하나로 합친다.

```
a=[1,2,3,4]
b=["WEB","OA","Multi"]
a.extend(b)
print(a)
```

<실행 결과>

```
[1, 2, 3, 4, 'WEB', 'OA', 'Multi']
```

참고

extend()

```
a=[1,2,3,4]
b=["WEB","OA","Multi"]
c=("a","b","c") #튜플(tuple)
```

a.extend(5)	extend()는 리스트끼리 사용하는 함수이다. 왼쪽과 같이 실행할 경우 에러(TypeError)가 발생한다. TypeError: 'int' object is not iterable ※5를 추가하려면 a.extend([5])로 사용해야 한다.
a.extend(b)	[1, 2, 3, 4, 'WEB', 'OA', 'Multi'] append와 달리 리스트 항목의 값만 추가한다. a.append(b)는 다음과 같이 리스트 내에 리스트를 삽입한다. [1, 2, 3, 4, ['WEB', 'OA', 'Multi']]
a.extend(c)	[1, 2, 3, 4, 'a', 'b', 'c'] 리스트 자료형이 아닌 튜플 자료형도 추가 가능하다.

3) insert(위치, 값)

a.insert(2, 5) : 리스트에서 2번째 위치에 5를 추가한다. 리스트 인덱스는 0부터 시작되므로 a[2]에 5를 삽입한다.

```
a=[1,2,3,4]
a.insert(2,5)
print(a)
```

<실행 결과>

[1, 2, 5, 3, 4]

리스트 a에서 "OA" 다음에 "Excel"을 추가해보자.

```
a=["WEB", "OA", "Multi", "RUBY"]
a.insert(_____)
```

방법.

- ① 리스트 a에서 "OA"의 index 값을 구한다.
- ② insert()를 이용하여 해당 위치에 "Excel"을 추가한다.

```
a=["WEB", "OA", "Multi", "RUBY"]
a.insert(a.index("OA")+1, "Excel")
print(a)
```

리스트 a에서 "OA"의 인덱스는 1이다. "Excel"의 위치를 "OA" 다음에 위치시키기 위해서 "OA" 인덱스에 1을 더해준다.

리스트 a의 항목 순서에 상관없이 "Excel"을 "OA" 다음에 위치시킬 수 있다. 하지만 리스트 내에 "OA" 항목 값이 존재하지 않는 경우 에러가 발생한다.

4) 삭제 : del, pop, remove

파이썬에서 리스트 내 항목을 삭제하기 위해 del, pop(), remove()를 제공한다.

◦ del

리스트이름[인덱스] : 리스트내 해당하는 인덱스 값을 삭제한다.

del은 리스트의 method가 아니라는 것은 명령어 사용법을 통해 확인할 수 있다.

```
a=["WEB", "OA", "Multi", "RUBY"]
del a[1]
print(a)
```

<실행 결과>

['WEB', 'Multi', 'RUBY']

del은 인덱스 값이 리스트 항목의 범위를 초과한 경우 에러(IndexError)가 발생한다. 인덱스를 작성하지 않을 경우 리스트 전체를 삭제하기 때문에 주의해야 한다.

```
a=["WEB","OA","Multi","RUBY"]
del a
print(a)
```

<실행 결과>

```
[NameError: name 'a' is not defined]
```

◦ pop()

pop()	리스트내 마지막 값을 삭제한다.
pop(인덱스)	리스트에 해당하는 인덱스 값을 삭제한다.

```
a=["WEB","OA","Multi","RUBY"]
a.pop() #리스트 항목 마지막 값 삭제
print(a)
b=a.pop(1) #b에 a[1]을 저장하고, a[1]은 삭제한다.
print(b, a)
```

<실행 결과>

```
['WEB', 'OA', 'Multi']
OA ['WEB', 'Multi']
```

pop()은 스택(stack), 큐(Queue)를 구현할 경우 간결한 코딩을 가능하게 한다.
처음 값을 삭제할 경우 : a.pop(0)

◦ remove()

리스트 내 항목 값으로 삭제할 경우 remove()를 사용한다. 단, 리스트에 중복된 값이 있는 경우 처음에 등장하는 값을 삭제한다.

```
a=["WEB","OA","Multi","RUBY","OA"]
a.remove("OA")
print(a)
```

<실행 결과>

```
['WEB', 'Multi', 'RUBY', 'OA']
```

remove()를 사용할 경우 리스트내 해당 값이 존재하지 않을 경우 에러(ValueError)가 발생한다.

```
ValueError: list.remove(x): x not in list
```

5) 개수 : count

count는 단어 뜻으로 인해 항목의 전체 개수를 세는 함수로 착각하는 경우가 많다. 파이썬에서 count()는 리스트내 특정 값의 개수를 세는 함수이다.

```
a=["WEB","OA","Multi","OA"]
print(a.count("OA"))
```

<실행 결과>

2

count()를 사용하여 리스트 내에 없는 항목을 삭제할 경우 에러 발생을 방지해보자.

```
a=["WEB","OA","Multi","RUBY"]
if a.count("python"):
    a.remove("python")
    print(a)
else:
    print("삭제할 값이 없습니다.")
```

리스트 내에 'python'이 없기 때문에 a.count("python") 결과 값은 0이다. a.count>0이라고 작성해도 되지만, if 0은 항상 거짓(False)이기 때문에 if a.count("python")으로 더욱 간결하게 사용할 수 있다.

참고

in 키워드

리스트 내 항목 중에 찾는 값의 존재 여부를 확인할 때 in 키워드를 사용한다

```
a=["WEB","OA","Multi","RUBY"]
print("OA" in a)      #a 리스트 내에 "OA"가 존재하는가?   True
print("python" in a)  #a 리스트 내에 "python"가 존재하는가?   False
```

in 키워드를 사용해 삭제 에러를 방지해보자.

```
a=["WEB","OA","Multi","RUBY"]
if "python" in a:
    a.remove("python")
    print(a)
else:
    print("삭제할 값이 없습니다.")
```


6) 정렬 : sort, reverse

리스트의 순서를 변경하려면 sort()를 사용한다.

```
a=[7,9,5,1,4,8,3,2,6]
a.sort()
print(a)
```

<실행 결과>

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
a=[7,9,5,1,4,8,3,2,6]
print(a.sort())
```

<실행 결과>

```
None
```

sort()는 리스트 자체를 변경시키기 때문에 다시 되돌릴 수 없다. sort()는 리턴(Return)이 없기 때문에 print(a.sort())를 실행시킬 경우 None을 리턴한다.

sort()는 기본적으로 오름차순 정렬이다. 리스트 항목 값을 내림차순 정렬을 하기 위해 reverse()를 사용한다.

```
a=[7,9,5,1,4,8,3,2,6]
a.reverse()
print(a)
```

<실행 결과>

```
[6, 2, 3, 8, 4, 1, 5, 9, 7]
```

reverse()는 리스트 항목의 위치를 역순으로 변경시킨다. sort()를 적용한 후에 reverse()를 적용하면 내림차순으로 정렬시킬 수 있다.

```
a=[7,9,5,1,4,8,3,2,6]
a.sort()
a.reverse()
print(a)
```

<실행 결과>

```
[9, 8, 7, 6, 5, 4, 3, 2, 1]
```

간결한 코딩을 원한다면 다음과 같이 작성할 수 있다.

```
a=[7,9,5,1,4,8,3,2,6]
a.sort(reverse=True)
print(a)
```

7) 정렬 : sorted

리스트의 값을 보존하고 정렬하기를 원한다면 sorted()를 사용해야 한다. sorted()는 리스트의 사본을 사용한다.

```
a=[7,9,5,1,4,8,3,2,6]
b=sorted(a)
print(a)
print(b)
```

<실행 결과>

```
[7, 9, 5, 1, 4, 8, 3, 2, 6]
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

sorted()를 사용한 내림차순 정렬은 다음과 같이 사용한다.

```
b=sorted(a,reverse=True)
```

8) 길이 : len

리스트 요소의 개수를 구할 수 있다. 리스트 항목 중 특정 값의 개수를 세는 count와 혼동하지 않도록 주의하자.

```
prog=["C","PYTHON","JAVA","GO"]  
print(len(prog))
```

<실행 결과>

4

len()을 통해 배열의 마지막 값의 위치를 알 수 있다. 파이썬에서는 음수 인덱스를 통해 마지막 값을 쉽게 찾을 수 있다.

```
prog=["C","JAVA","PYTHON"]  
s=len(prog)  
print(prog[s-1])  
print(prog[-1])
```

<실행 결과>

PYTHON

PYTHON

3.1.5. 리스트에서 반복문 사용

리스트의 값을 하나씩 출력하려고 한다. len()을 사용하여 리스트 내에 항목의 개수를 구하고, 해당 숫자만큼 반복하여 내용을 출력한다.

```
data=["cpu","ram","vga","ssd","dvd"]  
for a in range(len(data)): #len(data)를 사용하여 data의 개수만큼 반복  
    print(data[a])
```

for문은 다음과 같은 방법도 제공한다. 프로그래밍 방법에 따라 두 방법 중 하나를 선택해보자.

```
data=["cpu","ram","vga","ssd","dvd"]  
for a in data:  
    print(a)
```

join() 함수를 이용한 방법은 반복문을 사용하지 않고 1줄로도 구현이 가능하다.

```
data=["cpu","ram","vga","ssd","dvd"]  
print("\n".join(data))
```

data가 숫자일 경우에 join()을 사용할 경우 에러가 발생한다.

```
data=[1,2,3,4,5]  
print("-".join(data))
```

<실행 결과>

sequence item 0: expected str instance, int found

리스트의 숫자형 자료를 map()과 str()을 이용해 문자형으로 변환하여 해결할 수 있다.

```
data=[1,2,3,4,5]
print("-".join(list(map(str,data))))
```

map(함수, 리스트) : 원본 리스트를 변경하지 않고 새 리스트를 생성한다.

```
data=[1,2,3,4,5]
print(map(str,data)) # 리스트 data의 항목 하나씩 str() 함수를 적용
a1,a2,a3,a4,a5=map(str,data) # map() 함수로 변수 5개에 문자열을 저장한다.
print(a1,a2,a3,a4,a5,type(a1))
```

<실행 결과>

```
<map object at 0x000002308E363730>
1 2 3 4 5 <class 'str'>
```

list(map(함수, 리스트)) : 리스트의 항목 값들에 함수를 적용하여 새로운 리스트로 생성한다.

```
data=[1,2,3,4,5]
print(list(map(str,data)))
```

<실행 결과>

```
['1', '2', '3', '4', '5']
```

리스트 data에 난수 10개를 저장해보자. 반복문과 append()를 이용하여 난수를 10개 추가할 수 있다.

```
import random
data=[]
for a in range(10):
    data.append(random.randint(10,99))
```

리스트 내에서 반복문을 사용하는 방법으로 같은 기능을 구현할 수 있다.

```
import random
data1=[random.randint(10,99) for a in range(10)] # [식 for 변수 in 리스트]
data2=list(random.randint(10,99) for a in range(10)) # list(식 for 변수 in 리스트)
```

리스트 data의 값의 제곱값을 data2에 저장해보자.

```
data=[1,2,3,4,5]
data2=[a*a for a in data] # [1, 4, 9, 16, 25]
print(data2)
```

3.1.6. 이차원 리스트

리스트 안에 리스트나 튜플이 여러 개 있을 때 다음과 같이 리스트로 묶을 수 있다.

```
a=[[1,2,3],["a","b"],(10,20,30)]
```

리스트 a 안에는 3개의 요소가 있다. 다음 출력문을 통해 이차원 리스트의 항목 값을 확인해보자.

```
a=[[1,2,3],["a","b"],(10,20,30)]
print(a[0], a[0][1])
print(a[1][1])
print(a[2][1])
```

<실행 결과>
[1, 2, 3] 2
b
20

a[0]은 [1,2,3], a[1]은 ["a","b"], a[2]는 (10,20,30)이다.
print(a[0][1])은 [1,2,3] 중에 2를 출력한다.

다음 표의 데이터를 이차원 리스트로 정의해보자.

학번	이름	이메일	포인트
10101	김창준	robert@abc.com	34
10305	심재호	jhsim@abc.com	23
10515	김은희	ehkim@xyz.com	15
10823	박지현	jhpark@xyz.com	25

```
data=[[10101,"김창준","robert@abc.com",34],
       [10305,"심재호","jhsim@abc.com",23],
       [10515,"김은희","ehkim@xyz.com",15],
       [10823,"박지현","jhpark@xyz.com",25]]
```

이름과 이메일만 출력해보자.

◦ len()을 사용하는 경우

```
for a in range(len(data)):
    print(data[a][1], data[a][2])
```

◦ 리스트의 요소만 가져올 경우

```
for a in data:
    print(a[1], a[2])
```

3.2. 튜플(Tuple)

3.2.1. 튜플 선언

튜플은 소괄호(())를 사용하며 리스트와 유사한 기능을 가진다. 단, 튜플은 항목을 수정하지 못한다. 따라서 insert, del, remove, pop, sort, reverse와 같이 항목을 추가, 삭제, 위치를 변경하는 메소드를 사용할 수 없다.

```
a=(5,3,1,4,2)
b=tuple((5,3,1,4,2))
c=("python",500,3.14)
d=(1,3,2),(10,30,20),"java","C++")
print(type(a),type(b),type(c),type(d))
```

<실행 결과>

```
<class 'tuple'> <class 'tuple'> <class 'tuple'> <class 'tuple'>
```

튜플 선언시 다음과 같은 방법은 선언할 경우 튜플이 되지 않는다.

```
a=[5])
b=(5)
c=("python")
print(type(a),type(b),type(c))
```

```
<class 'list'> <class 'int'> <class 'str'>
```

리스트는 항목 값 하나만 사용해도 리스트로 선언되지만, 튜플은 항목이 하나일 경우 위와 같이 사용하면 해당 값의 자료형인 정수형(int)과 문자형(str)으로 선언된다. 항목이 하나일 경우 튜플로 선언하기 위한 방법은 다음과 같다.

```
a=(5,)
b=(5,)
c=("python",)
print(type(a),type(b),type(c))
```

리스트와 튜플은 서로 변환할 수 있다.

```
data1=[1,2,3,4,5]
data2=(1,2,3,4,5)
v1=tuple(data1)
v2=list(data2)
print(v1, v2)
```

<실행 결과>

```
(1, 2, 3, 4, 5) [1, 2, 3, 4, 5]
```

튜플의 항목이 다음과 같을 때 각 명령을 실행시 발생하는 에러는 다음과 같다.

a=(5,3,1,4,2)

명령	실행 결과
a[0]=4	TypeError: 'tuple' object does not support item assignment
a.remove(1)	AttributeError: 'tuple' object has no attribute 'remove'
a.pop()	AttributeError: 'tuple' object has no attribute 'pop'
del a[1]	TypeError: 'tuple' object doesn't support item deletion
a.append(6)	AttributeError: 'tuple' object has no attribute 'append'
a.extend((10,30))	AttributeError: 'tuple' object has no attribute 'extend'
a.sort()	AttributeError: 'tuple' object has no attribute 'sort'

튜플에서 사용할 수 있는 메서드는 count와 index가 있으며, 리스트에서 사용했던 내장 함수를 모두 동일하게 사용할 수 있다.

```
a=("python", "C++", "JAVA", "GO")
b=(4,2,5,1,3)
```

명령	실행 결과
print(a[1:2])	('C++',) <i>#항목이 한 개이기 때문에 콤마(,)를 표시한다.</i>
print(a[1:3])	('C++', 'JAVA')
print(b[-3:-1])	(5, 1)
print(a.index("JAVA"))	2 <i>#"JAVA"의 인덱스 값을 출력</i>
print(a.count("C++"))	1 <i>#튜플에 "C++"의 개수</i>
print(len(a))	4 <i>#튜플 항목의 개수</i>
c=sorted(b)	[1, 2, 3, 4, 5]

3.3. 딕셔너리(Dictionary)

3.3.1. 딕셔너리 선언 및 구조

딕셔너리의 기본 모습은 키(Key)와 값(Value)이 중괄호({}) 사이에 위치한 구조이다.

```
{key1: value1, key2:value2, key3:value3,.....}
```

key 값은 중복되지 않고 변하지 않는 값을 사용한다. value는 하나의 값뿐만 아니라 리스트 형태도 가능하다.

딕셔너리를 선언하는 방법은 다음과 같다.

```
d1={}
d2={"name":"홍길동", "email":"test1@abc.com", "phone":"010-1234-5678"}
d3=dict(name="김철수", email="test2@abc.com", phone="010-8765-4321")
print(type(d1), type(d2), type(d3))
```

<실행 결과>

```
<class 'dict'> <class 'dict'> <class 'dict'>
```

3.3.2. 딕셔너리 추가 / 수정 / 삭제

딕셔너리에 키(Key)와 값(value)을 추가하는 방법은 다음과 같다.

```
fruits={"a":"apple", "b":"banana"}
fruits["c"]="carot"
print(fruits)
```

<실행 결과>

```
{'a': 'apple', 'b': 'banana', 'c': 'carot'}
```

기존 키 값이 존재하면 값을 변경할 수 있다.

```
fruits={'a':'apple', 'b':'banana'}
fruits['b']='blueberry'
print(fruits)
```

<실행 결과>

```
{'a': 'apple', 'b': 'blueberry'}
```

딕셔너리의 값은 리스트 형태도 가능하다.

```
fruits={"a":"apple","b":"banana"}
fruits["c"]=["coconut","cherry"]
print(fruits)
```

<실행 결과>

```
{'a': 'apple', 'b': 'banana', 'c': ['coconut', 'cherry']}
```

딕셔너리의 키와 값을 삭제하려면 del을 사용한다.

```
fruits={"a":"apple","b":"banana","c":["coconut","cherry"]}
del fruits["c"]
print(fruits)
```

<실행 결과>

```
{'a': 'apple', 'b': 'banana'}
```

단, 딕셔너리에 존재하지 않는 키를 삭제할 때는 에러(KeyError: 'd')가 발생한다.

```
del fruits["d"] # 키 값에 d가 없으면 에러(KeyError: 'd') 발생
```

딕셔너리의 모든 값을 삭제할 때는 clear() 함수를 사용한다.

```
fruits={"a":"apple","b":"banana","c":["coconut","cherry"]}
fruits.clear()
```

<실행 결과>

```
{}
```

딕셔너리의 키(key)와 값(value)을 삭제한 것이다. 딕셔너리 전체를 삭제할 때에는 del을 사용한다.

```
del fruits
```

3.3.3. 딕셔너리 검색

딕셔너리는 index를 사용하지 않는다. 딕셔너리 키 값을 검색할 때는 in을 사용한다.

```
fruits={"a":"apple","b":"banana","c":["coconut","cherry"]}
print("b" in fruits, "banana" in fruits)
```

<실행 결과>

```
True False
```


값(Value)으로 검색할 때는 values()나 items() 함수를 사용한다.

```
fruits = {"a": "apple", "b": "banana", "c": "cherry"}
print(fruits.keys(), type(fruits.keys()))
print(fruits.values(), type(fruits.values()))
print(fruits.items(), type(fruits.items()))
```

<실행 결과>

```
dict_keys(['a', 'b', 'c']) <class 'dict_keys'>
dict_values(['apple', 'banana', 'cherry']) <class 'dict_values'>
dict_items([('a', 'apple'), ('b', 'banana'), ('c', 'cherry')]) <class 'dict_items'>
```

딕셔너리의 값중에 "banana"를 검색해보자.

```
fruits = {"a": "apple", "b": "banana", "c": "cherry"}
print("banana" in fruits.values()) #방법1. in 키워드를 사용하는 방법
for k, v in fruits.items(): #방법2. 반복문을 사용할 때는 items()를 사용
    if v=="banana":
        print(k, v)
```

<실행 결과>

```
True
b banana
```

3.3.4. 딕셔너리 합치기

두 개의 딕셔너리를 합칠 때는 update() 함수를 사용한다. 중복된 값이 있으면, 가장 마지막에 있는 값으로 변경된다.

```
f1 = {'a': 'apple', 'b': 'banana'}
f2 = {'b': 'blueberry', 'c': 'coconut'}
f1.update(f2)
print(f1)
```

<실행 결과>

```
{'a': 'apple', 'b': 'blueberry', 'c': 'coconut'}
```

3.3.5. 딕셔너리 정렬

◦ 키(keys)를 기준으로 정렬

```
data={"d":8,"c":5,"a":9,"b":2}
d1=sorted(data)
d2=sorted(data.keys())# 키(key)를 오름차순으로 정렬
d3=sorted(data.keys(), reverse=True) #키(key)를 내림차순으로 정렬
print(d1, d2, d3)
```

<실행 결과>

```
['a', 'b', 'c', 'd'] ['a', 'b', 'c', 'd'] ['d', 'c', 'b', 'a']
```

내림차순으로 정렬하기 위해서는 reverse 인자의 값을 True로 설정한다.

키를 기준으로 내림차순하여 정렬한 후에 키와 값을 모두 출력하면 다음과 같다.

```
data={"d":8,"c":5,"a":9,"b":2}
d3=sorted(data.keys(), reverse=True)
for k in d3:
    print(k, data[k])
```

◦ 값(values)을 기준으로 정렬

값을 기준으로 정렬하려면 data.keys() 대신에 data.values()를 적용한다.

```
data={"d":8,"c":5,"a":9,"b":2}
d2=sorted(data.values())
d3=sorted(data.values(), reverse=True)
print(d2, d3)
```

<실행 결과>

```
[2, 5, 8, 9] [9, 8, 5, 2]
```

◦ 키(keys)와 값(values)의 쌍(items())을 기준으로 정렬

```
data={"d":8,"c":5,"a":9,"b":2}
d1=sorted(data.items())
d2=sorted(data.items(), reverse=True)
print(d1, d2, sep='\n')
```

<실행 결과>

```
[('a', 9), ('b', 2), ('c', 5), ('d', 8)]
[('d', 8), ('c', 5), ('b', 2), ('a', 9)]
```

◦ 람다식을 사용한 정렬

```
data={"d":8,"c":5,"a":9,"b":2}
#키를 기준으로 정렬
k1=sorted(data.items(), key=lambda x:x[0]) #오름차순 정렬
k2=sorted(data.items(), key=lambda x:x[0],reverse=True) #내림차순 정렬

#값을 기준으로 정렬
v1=sorted(data.items(), key=lambda x:x[1]) #오름차순 정렬
v2=sorted(data.items(), key=lambda x:x[1],reverse=True) #내림차순 정렬
print(k1, k2, v1, v2, sep='\n')
```

<실행 결과>

```
[('a', 9), ('b', 2), ('c', 5), ('d', 8)]
[('d', 8), ('c', 5), ('b', 2), ('a', 9)]
[('b', 2), ('c', 5), ('d', 8), ('a', 9)]
[('a', 9), ('d', 8), ('c', 5), ('b', 2)]
```

값(value)이 가장 큰 키(key)를 출력하는 방법은 다음과 같다.

```
data={"d":8,"c":5,"a":9,"b":2}
sdata=sorted(data.items(),key=lambda x:x[1])#오름차순 정렬
print(sdata[-1], sdata[-1][0], sdata[-1][1])
```

<실행 결과>

```
('a', 9) a 9
```

딕셔너리 data를 값(values)을 기준으로 오름차순 정렬한 후에 마지막 항목을 가져온다. sdata[-1]은 ('a', 9)이다. 키는 sdata[-1][0], 값은 sdata[-1][1]로 표현할 수 있다.

값을 기준으로 내림차순하여 모든 항목의 키와 값을 출력하는 방법을 보고 딕셔너리 정렬 방법을 살펴보자.

```
data={"d":8,"c":5,"a":9,"b":2}
sdata=sorted(data.items(),key=lambda x:x[1],reverse=True) #값을 기준으로 내림차순
for k, v in sdata:
    print(k, v)
```

<실행 결과>

```
a 9
d 8
c 5
b 2
```

퀴즈

Quiz 1. 입력한 정수를 소인수 분해 하시오.

입력: 2 이상 10000000이하의 정수를 입력한다.

출력: 소인수 분해 결과는 각 숫자의 곱으로 출력한다.

입력

24

출력

2*2*2*3

Quiz 2. 입력한 시간의 형식을 변경하시오.

입력: 시간을 [시:분:초] 형식으로 입력한다.

$0 \leq \text{시} < 24, 0 \leq \text{분} < 60, 0 \leq \text{초} < 60$

출력: 24시간 형식의 시간을 [오전 0시 0분 0초]로 출력한다.

입력1

8:23:12

출력1

오전 8시 23분 12초

입력2

16:40:5

출력2

오후 4시 40분 5초

Quiz3. 입력한 점수에 따라 5단계 평가를 출력하는 프로그램을 작성하시오.

단, 조건문을 사용하지 않는다.

입력: 0이상 100이하의 점수를 정수로 입력한다.

출력: 점수별 등급표를 참고하여 출력하시오.

90점 이상	80점 이상	70점 이상	60점 이상	60점 미만
A	B	C	D	F

입력1

85

출력1

B

입력2

77

출력2

C

Quiz 4. N명의 심사위원의 점수의 합계를 출력하시오. 단, 가장 높은 점수와 낮은 점수를 제외한 점수는 제외하시오. 가장 높은 점수가 중복될 경우 값 하나만 삭제
 입력: 심사위원의 수만큼 점수를 한 칸 단위로 입력한다.
 출력: 최댓값과 최솟값을 제외한 점수의 합계를 출력한다.

<u>입력1</u>	<u>출력1</u>	<u>입력2</u>	<u>출력2</u>
5 7 4 9 1	16	9 9 9 8 8	26

Quiz 5. 연속해서 정답을 맞출 경우 배점이 달라지는 시험이 있다.
 입력: n개의 채점결과를 'O', 'X'로 연속해서 입력한다.
 출력: 'O'가 1개만 있을 경우 1점, 연속해서 있을 때는 연속한 수가 점수이다.
 예) OXX : 1점(1+0+0), OOX: 3점(1+2+0), OXOXO: 3점(1+0+1+0+1)
 XOOOX: 6점(0+1+2+3+0), OOXOO: 6점(1+2+0+1+2)
 (출처: ICPC > Regionals > Asia Pacific > Korea > Asia Regional - Seoul 2005 A번)

<u>입력</u>	<u>출력</u>
OXOOXOOXOOX	10점

Quiz 6. 입력한 문장에서 사용하지 않은 알파벳 글자를 출력하시오.
 입력: 한 문장을 입력한다.
 출력: a부터 z까지 사용하지 않은 글자를 출력한다.
 단, 소문자만 입력한다.

<u>입력</u>	<u>출력</u>
python is a programming language.	bcdfjkqvwxyz

Quiz 7. 정수를 입력하면 각 자릿수의 숫자의 합을 출력하시오.

<u>입력1</u>	<u>출력1</u>	<u>입력2</u>	<u>출력2</u>
123	6	250000	7

Quiz 8. 입력한 n 개의 단어를 출현 빈도수가 높은 단어 3개를 출력하시오.

입력: 단어를 한 칸 단위로 n 개 입력한다.

출력: 가장 많이 등장한 단어부터 단어와 출현 빈도수를 3개 출력한다.

입력

py java C py html java C py java py

출력

py: 4

java: 3

C: 2

Quiz 9. 정수 n 을 입력하면 2부터 n 까지 소수의 개수를 출력하시오.

n 은 2이상 1000000이하로 입력하며, 에라토스테네스의 체를 이용하시오.

입력1

출력1

입력2

출력2

20

8

50

15

참고

에라토스테네스의 체

2부터 n 까지의 정수를 차례로 쓴다. (2, 3, 4, ..., n)

1. 2 이외의 2의 배수를 지운다. 2는 소수가 된다.(짝수 중 유일한 소수)

2 다음 수 3을 제외한 3의 배수를 지운다

3. 위 과정대로 진행한다.(n 까지 진행)

1~3 과정을 진행한 후 남은 수가 소수가 된다.

1단계. 2를 제외한 2의 배수 모두 삭제

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

2단계. 3을 제외한 3의 배수를 삭제

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

이 과정을 반복하면 소수만 남게 된다. 이 방법이 에라토스테네스의 체이다.

Quiz 1. 소인수 분해

2단원에 소인수 분해 과정에서는 곱하기 연산자를 생략했다. 곱하기 연산자를 붙일 경우 분해 과정을 분석해보자.

1. 출력결과를 저장할 변수를 공백으로 선언한다. (문자형이나 리스트를 사용)
2. 정수 n 을 2로 나눈다.
3. 나누어 떨어지면 n 의 값은 2로 나눈 몫으로 설정한다. 출력 결과를 저장할 변수에 나누어 떨어진 수와 곱하기 연산자를 연결한다. 나누어 떨어지지 않으면 나누려는 값을 1씩 증가한다.
4. n 이 1이 될 때까지 반복한다.
5. n 이 1이 되면 출력 결과를 저장할 변수에서 필요한 부분만 추출한다.

```
n=int(input())
s='' #출력 결과를 저장할 변수
d=2 #나누려는 값을 2부터 설정
while n!=1: #n이 1이 아닐때까지 반복. (n이 1이면 반복 종료)
    if n%d==0: #n이 d로 나누어 떨어지면
        s+=f'{d}*'
        n//=d #n은 d로 나눈 몫으로 변경한다.
    else: #n이 d로 나누어 떨어지지 않으면
        d+=1 # d의 값을 1씩 증가한다.
print(s[:-1]) # s의 마지막은 '*'이기 때문에 마지막 글자는 제외하고 출력
```

리스트를 사용한다면 출력 결과를 저장할 변수를 리스트로 선언한다.

문자 연결 부분은 `s+=f'{d}*`은 `append()`로 변경하고, 반복문이 끝나면 `join()`을 이용해 곱하기 연산자로 연결하여 출력한다. `join`을 사용하려면 리스트에는 문자열로 저장한다.

```
n=int(input())
s=[] #출력 결과는 리스트로 선언
d=2
while n!=1:
    if n%d==0:
        s.append(str(d)) # 정수를 문자로 변환하여 리스트에 추가
        n//=d
    else:
        d+=1
print('*'.join(s))
```

Quiz 2. 시간 형식 변경

시간 형식(시:분:초)으로 입력하더라도 문자열로 인식한다. ':'를 기준으로 분리하고 시, 분, 초를 정수로 분리한다.

◦ split()으로 시, 분, 초를 분리

```
time=input().split(':')
```

':'를 기준으로 문자를 분리하여 리스트에 저장한다.

12:34:21을 입력할 경우 ['12', '34', '21']로 저장된다.

◦ map()을 사용하여 변수 3개에 저장

```
h, m, s = map(int, input().split(':'))
```

input().split(':')의 결과는 리스트에 저장되고, 리스트의 값들을 int()함수를 사용해 정수로 변환한다. 시, 분, 초는 각각 h, m, s에 저장된다.

◦ list()를 사용하여 시, 분, 초를 리스트에 저장

변수 3개를 사용하지 않고 리스트에 저장하려면 list()함수를 사용한다.

```
time=list(map(int, input().split(':')))
```

① 입력한 문자열을 ':'으로 분리하여 리스트에 저장

② 리스트의 값들을 int()함수를 사용해 정수로 변환

③ map() 함수로 정수로 분리된 값을 리스트로 변환

12:04:21을 입력할 경우 [12, 4, 21]로 저장된다.

◦ 오전, 오후를 판단하는 기준

조건문을 사용해 12시 이상인 경우 '오후', 미만인 경우는(그렇지 않은 경우) '오전'

```
ap = '오후' if h>=12 else '오전'
```

◦ 24시간 출력을 12시간으로 변경

가장 많이 하는 실수가 다음과 같이 작성하는 것이다.

```
if h>=12: h-=12
```

13:30:40인 경우 오후 1시이지만, 12:30:40인 경우 오후 0시가 된다.

오전 오후의 기준은 12시 이상이지만, 시간 표현은 12 초과로 해야 한다.

```
h, m, s =map(int, input().split(':'))
ap='오후'if h>=12 else '오전'
if h>13: h-=12
print(f'{ap}{h}시 {m}분 {s}초')
```


Quiz3. if문을 사용하지 않고 성적 평가(5단계)

평가 기준을 10점 단위로 나눠 보자.

100	90	80	70	60	50	40	30	20	10	0
A	A	B	C	D	F	F	F	F	F	F

10점 단위별로 평가 결과를 문자나 리스트로 저장한다.

0점부터 10점대, 20점대 순서로 저장하고 점수를 10으로 나눈 몫을 구한다. 그 순서에 해당하는 값을 출력하면 조건문(if)을 사용하지 않고 해결할 수 있다.

```
t='FFFFFFDCBAA'
score=int(input())
print(t[score//10])
```

Quiz 4. N명의 심사위원의 점수의 합계 (최댓값, 최솟값 제외)

list, map, int, input, split을 사용해 점수를 한 줄에 입력한 후 공백으로 분리하여 리스트에 저장한다.

```
score =list(map(int, input().split()))
```

리스트에서 최댓값, 최솟값을 삭제한 후 합계를 출력하려면 max, min, remove, sum을 사용한다.

```
score =list(map(int, input().split()))
print(score)
score.remove(max(score))
score.remove(min(score))
print(sum(score))
```

참고

remove()로 중복 값 모두 삭제하기

최댓값이나 최솟값이 중복될 경우라도 remove() 함수는 앞에 있는 값만 삭제한다. 모두 삭제하려면 리스트에서 삭제하려는 값만큼 반복하면서 삭제를 해야 한다.

```
data=[1,2,1,2,1,2,3]
vmin=min(data)
for i in range(data.count(vmin)):
    data.remove(vmin)
print(data)
```

Quiz 5. 채점 프로그램

입력한 문자는 문자열 그대로 사용하거나 리스트로 변환해서 사용할 수 있다.

문제 해결과정을 분석해보자.

1. 전체 점수(총점)를 계산할 변수 *s*를 0으로 선언한다.
2. 각 문제의 배점을 계산할 변수 *k*를 0로 선언한다.
3. 첫 글자가 '0'이면 *k*를 1더한 후 *s*에 *k*를 더한다. 'X'이면 *k*를 0로 변경한다.
4. 입력한 글자 수만큼 반복한다.

```
data = input()
s=0 #총점
k=0 #문항별 배점
for a in range(len(data)):
    if data[a]=='0':
        k+=1 #배점을 1 더한다. 연속 정답인 경우 1씩 증가한다.
        s+=k #총점을 k만큼 더한다.
    else:
        k=0 #배점을 0으로 변경. 오답일 경우 연속 증가값을 0으로 변경
print(s) #총점을 출력
```

Quiz 6. 미사용 알파벳 글자 출력

알파벳 a부터 z까지 리스트에 저장한다.

```
alpha = ['a','b','c',..., 'y','z']
```

위와 같이 한 글자씩 지정하는 방법도 있지만 문자열에 `list()` 함수를 사용하면 한 글자씩 리스트 항목 값으로 변경할 수 있다. 알파벳을 순서대로 작성할 필요는 없다.

```
alpha=list('abcdefghijklmnopqrstuvwxyz')
```

※ 문자열 함수를 이용하면 알파벳 문자 26글자를 사용하지 않고 입력할 수 있다.(4단원 참고)

방법1. 입력한 문장의 한 글자씩 리스트에 해당하는 글자가 있으면 삭제한다.

리스트에 남아있는 글자가 미사용 문자이다.

```
alpha=list('abcdefghijklmnopqrstuvwxyz')
data=input()
for t in range(len(data)): #입력한 글자 길이만큼 반복한다.
    if data[t]in alpha: #t번째 글자가 리스트 alpha에 있으면
        alpha.remove(data[t]) #리스트 alpha에서 t번째 글자를 삭제
print(''.join(alpha)) #리스트 alpha의 값을 공백을 주지 않고 연속해서 출력
```

방법2. 입력한 문자를 한 글자씩 다른 리스트에 저장한다. 두 리스트간 비교하여 사용하지 않은 문자를 출력한다.

```
alpha=list('abcdefghijklmnopqrstuvwxyz')
data=list(input()) # 입력한 문자를 한 글자씩 리스트로 변환
for t in data: #리스트 data에서 한 글자씩 가져온다.
    if t in alpha: #글자(t)가 리스트 alpha에 있으면
        alpha.remove(t) # 리스트 alpha에서 글자(t)를 삭제
print(''.join(alpha)) #리스트 alpha의 값을 공백을 주지 않고 연속해서 출력
```

※이 문제를 해결하기 위해 리스트, 딕셔너리, 아스키 코드 등 다양한 방법으로 접근할 수 있다.

Quiz 7. 자릿수의 합

입력한 정수의 자릿수의 합을 구하는 과정을 분석해보자.

방법1. 정수로 계산할 경우

1. 자릿수의 합을 저장할 변수 s 를 0으로 선언한다.
2. 입력한 정수 n 을 10으로 나눈 나머지는 s 에 더한다.
3. n 은 n 을 10으로 나눈 몫으로 변경한다.
4. n 이 0이 될 때까지 2~3 과정을 반복한다.

```
n=int(input()) #입력받은 문자를 정수로 변환한다.
s=0 #자릿수의 합을 계산하기 위한 변수를 0으로 선언한다.
while n!=0: #n이 0이 아닐 때까지 반복(n이 0이면 반복문 종료)
    s+=n%10 #자릿수의 합은 10으로 나눈 나머지를 더한다.
    n//=10 #입력한 정수는 10으로 나눈 몫이다.
print(s)
```

방법2. 문자로 계산할 경우

1. 입력할 때 정수로 변환하지 않고 한 글자씩 리스트로 변환한다.
2. 리스트의 한 글자씩 정수로 변환한다.
3. 리스트의 총합을 출력한다.

◦ len()을 사용

```
data=list(input())
for a in range(len(data)):
    data[a]=int(data[a])
print(sum(data))
```

◦ map()을 사용

```
data=list(map(int, list(input())))
print(sum(data))
```

Quiz 8. 출현 빈도수가 높은 단어 3개 출력

입력한 문장은 split() 함수를 사용해 공백 단위로 분리하여 리스트 data에 저장한다.

방법1. 리스트를 사용할 경우

1. 단어와 빈도를 저장할 리스트 word, cnt를 선언한다.
2. 리스트 data의 값들을 반복문을 사용해서 첫 번째 값부터 읽는다.
3. 리스트 data 첫 번째 값이 리스트 word에 있으면
리스트 word에 해당 단어의 위치를 찾는다. 리스트 cnt의 위치값의 항목을 1 증가
4. 리스트 data 첫 번째 값이 리스트 word에 없으면
리스트 word에 해당 단어를 추가하고, 리스트 cnt는 1을 추가한다. (첫 번째 등장)
5. 다른 리스트에 리스트 cnt를 정렬한 값을 저장한다.
6. 오름차순일 경우 0, 1, 2번째, 내림차순일 경우 -1, -2, -3번째 값을 출력한다.

```
data=input().split()
data=input().split()
word=[] #단어를 등록
cnt=[] #빈도 수를 등록
for t in data:
    if t in word: #입력한 단어가 word에 있으면
        p=data.index(t) #해당 단어의 위치를 p에 저장
        cnt[p]+=1 #빈도 수를 1씩 증가
    else: #입력한 단어가 word에 없으면
        word.append(t) #해당 단어를 추가
        cnt.append(1) #빈도 수를 1로 설정
print(word)
print(cnt)
```

<실행 결과>

```
py py c html py java c java jpg java gif html
['py', 'c', 'html', 'java', 'jpg', 'gif']
[3, 2, 2, 3, 1, 1]
```

리스트 cnt를 내림차순으로 정렬하여 다른 리스트(scnt)에 저장한다.

리스트 scnt의 첫 번째 값부터 세 번째 값이 리스트 cnt에서 몇 번째 순서인지 계산한다.

해당 순서를 리스트 word에서 찾아 출력한다.

단, 중복값이 있을 경우 다른 항목을 출력하기 위해서는 출력한 단어를 삭제해야 한다.

```
scnt =sorted(cnt, reverse=True)
for a in range(3):
    x=cnt.index(scnt[a])
    print(f'{word[x]}: {cnt[x]}')
    del word[x]
    del cnt[x]
```

방법2. 딕셔너리를 사용할 경우

1. 딕셔너리 word를 선언한다. 키: 단어, 값: 빈도
2. 리스트 data의 값들을 반복문을 사용해서 첫 번째 값부터 읽는다.
3. 리스트 data 첫 번째 값이 딕셔너리 word에 있으면
딕셔너리 word[단어]의 값에 1을 더한다.
4. 리스트 data 첫 번째 값이 딕셔너리 word에 없으면
딕셔너리 word에 word[단어]를 1로 설정한다.
5. 딕셔너리를 값을 기준으로 정렬한다.
6. 딕셔너리 items()로 키와 값을 3개 출력한다.

```
data=input().split()
word ={} #딕셔너리 선언
for t in data:
    if t in word.keys():
        word[t]+=1
    else:
        word[t]=1
print(word)
```

<실행 결과>

```
py py c html py java c java jpg java gif html
{'py': 3, 'c': 2, 'html': 2, 'java': 3, 'jpg': 1, 'gif': 1}
```

딕셔너리 word를 내림차순 정렬하여 키와 값을 3개만 출력한다.

```
sort_word =sorted(word.items(), key=lambda x:x[1],reverse=True)
for a in range(3):
    print(sort_word[a][0],':',sort_word[a][1])
```

Quiz 9. 소수의 개수 출력(에라토스테네스의 체)

리스트의 순서가 0부터 시작하기 때문에 0부터 100까지 리스트에 저장한다.

소수는 0과 1은 제외하기 때문에 리스트의 0, 1번째 순서의 값은 0으로 설정한다.

파이썬은 리스트 내에서 반복문을 사용할 수 있다.

```
data =[0,0]+[a for a in range(2,101)]
```

2부터 시작해서 2를 제외한 2의 배수는 모두 0으로 변경한다. 3을 제외한 3의 배수를 모두 0으로 변경한다. n의 배수는 값을 기준으로 설정할 수 있고, 위치 값을 기준으로 설정할 수 있다. 위 과정을 반복하면 리스트 data에서 0이 아닌 값은 소수이다.

```

data=[0,0]+[a for a in range(2,101)]
cnt=0 #소수의 개수를 저장하기 위한 변수
for a in range(2, 101): #2부터 100까지
    for b in range(a*2, 101, a): #2의 배수(4부터 2씩 증가)
        data[b]=0 #해당 위치의 값을 0으로 변경
for a in data:
    if a>0: cnt+=1 #0이 아닌 값이면 cnt를 1씩 증가
print(cnt)

```

위 프로그램을 개선할 수 있는 방법을 연구해보자. 중첩 반복문에서 실행 횟수를 줄일 수 있는 방법은 다음과 같다.

- 2를 제외한 짝수는 소수가 아니다.
- n번째 값이 0이면 n번째 값의 배수는 모두 0이다. 반복문을 실행할 필요가 없다.
예) 2를 제외한 2의 배수를 0으로 변경, 3을 제외한 3의 배수를 0으로 변경
4는 2의 배수를 0으로 변경할 때 이미 0으로 변경했기 때문에 반복문 미실행
- 반복문은 배수만큼 변경하기 때문에 전체 크기가 100이라면 50까지 반복하면 된다.

```

data=[0,0]+[a for a in range(2,101)]
cnt=0
for a in range(2, 51): #아래 반복문에 배수를 진행하기 때문에 50까지만 반복
    if data[a]==0: continue #해당 순서의 값이 0이면 continue를 사용
    for b in range(a*2, 101, a): #a를 제외한 a의 배수를 0으로 설정
        data[b]=0
for a in data:
    if a>0: cnt+=1 #0이 아닌 값이면 cnt를 1씩 증가
print(cnt)

```

소수에 해당하는 숫자만 다른 리스트에 추가하는 방법도 있다.

```

data=[0,0]+[a for a in range(2,101)]
result=[]
for a in range(2, 101):
    if data[a]==0: continue
    result.append(data[a])
    for b in range(a*2, 101, a):
        data[b]=0
print(len(result))

```

4. 함수

4.1. 함수

함수란 어떤 일을 수행하는 코드를 정의해 놓은 것이다. 화면에 출력하기 위한 `print()`도 함수이다. 파이썬에서 미리 정의해 놓은 내장 함수와 사용자가 직접 제작할 수 있는 방법도 제공한다.

4.1.1. 문자열 함수

함수	설명
<code>len()</code>	입력값 <code>x</code> 의 길이(항목의 전체 개수)를 반환한다.
<code>count()</code>	입력값 <code>x</code> 에서 특정 문자의 개수를 출력한다.
<code>index()</code>	문자열 중에 해당 문자가 등장하는 위치를 반환한다. 찾는 값이 없을 경우 오류가 발생한다. (ValueError: substring not found)
<code>find()</code>	문자열 중에 해당 문자가 등장하는 위치를 반환한다. 해당 문자열이 존재하지 않을 경우 <code>-1</code> 을 반환한다. <code>find()</code> 는 문자열만 사용 가능하다. 리스트, 튜플, 딕셔너리 자료형에서는 <code>AttributeError</code> 가 발생한다.
<code>replace()</code>	기존 문자를 새로운 문자로 치환한다.
<code>upper()</code>	문자열을 모두 대문자로 변환한다.
<code>lower()</code>	문자열을 모두 소문자로 변환한다.
<code>title()</code>	단어의 첫 글자만 대문자로 변환하고 다른 글자는 소문자로 변환한다.
<code>lstrip()</code>	문자열의 왼쪽 공백을 제거한다.
<code>rstrip()</code>	문자열의 오른쪽 공백을 제거한다.
<code>split()</code>	문자열을 나누어 준다. <code>split()</code> 함수에 인자 값이 없으면 공백을 기준으로 문자열을 나누어 준다. 특정 문자를 입력하면, 해당 문자로 나누어 준다.

```
engt="python programming"
kot="파이썬 프로그래밍"
pro=["0A","Prog","Multi","Prog"]
print("len:",len(engt),len(kot),len(pro))          # len: 18 9 4
print("count:", engt.count("p"), pro.count("Prog")) # count: 2 2
print("index:", engt.index("h"), pro.index("0A"))   # index: 3 0
print("find:", engt.find("y"), engt.find("z"))      # find: 1 -1
```

◦ 대·소문자 변환

```
prog="java c++ PythOn html5"
print(prog.upper())
print(prog.lower())
print(prog.title())
print(prog.replace("c++","G0"))
print(prog)
```

<실행 결과>

```
JAVA C++ PYTHON HTML5
java c++ python html5
Java C++ Python Html5
java c++ PythOn html5
java G0 PythOn html5
java c++ PythOn html5
```

변수 prog의 내용을 upper(), lower(), title(), replace()를 적용해도 위와 같은 방법은 변수 prog에 영향을 주지 않는다. prog를 변경하는 방법은 다음과 같다.

```
prog="java c++ PythOn html5"
prog = prog.title()
prog = prog.replace("c++","C++")
print(prog)
```

<실행 결과>

```
Java C++ Python Html5
```

◦ 문자 연결

```
text="123abc가나다"
prog=["java","c++","python","html5"]
print("-".join(text))
print(", ".join(prog))
```

<실행 결과>

```
1-2-3-a-b-c-가-나-다
java, c++, python, html5
```

◦ 문자 분리

```
t1='java py html'
t2='java/py/html'
print(t1.split())    # 공백으로 분리
print(t2.split('/')) # '/'를 기준으로 분리
```

<실행 결과>

```
['java', 'py', 'html']
['java', 'py', 'html']
```

◦ 공백 제거

```
t=" korea 123 "
print(len(t), t)
print(len(t.lstrip()), t.lstrip()) #왼쪽 공백 제거
print(len(t.rstrip()), t.rstrip()) #오른쪽 공백 제거
print(len(t.strip()), t.strip())   #양쪽 공백 제거
```

<실행 결과>

```
14 korea 123
12 korea 123
12 korea 123
10 korea 123
```


참고

아스키(ASCII) 코드

미국정보교환표준부호(American Standard Code for Information Interchange)

미국 ANSI에서 1963년에 정보교환용 7비트의 이진수 조합으로 만든 부호이다.
각 문자를 7비트로 표현하기 때문에 27(128개)의 문자를 표현할 수 있다.

파이썬에서는 ord(), chr()을 통해 아스키 코드를 다룰 수 있다.

ord(문자): 문자의 아스키 코드를 반환한다. print(ord('A')) #65

chr(숫자): 숫자에 맞는 아스키 코드를 반환한다. print(chr(65)) #A

```
for a in range(65, 91):  
    print(chr(a),end='')
```

<실행 결과>

ABCDEFGHIJKLMNOPQRSTUVWXYZ

4.1.2. 숫자 함수

◦ 파이썬에 내장된 함수 중 수학과 관련한 함수는 다음과 같다.

print(abs(-3))	#절대값	<실행 결과> 3
print(pow(3,4))	#3의 4제곱	81
print(max(5,2,8,3,4))	#최댓값	8
print(min(5,2,8,3,4))	#최솟값	2
print(round(3.147))	#반올림	3
print(round(3.547))	#반올림	4
print(round(3.147,2))	#소수 셋째자리에서 반올림	3.15

◦ 진법 관련

파이썬에서는 10진수를 2, 8, 16진수로 변환하기 위해 bin, oct, hex를 사용한다.
진법 변환시 결과 값은 모두 문자열로 저장된다.

x=35	<실행 결과>
b, o, h=bin(x), oct(x), hex(x)	0b100011 0o43 0x23
print(b, o, h)	<class 'str'>
print(type(b), type(o), type(h), sep='\n')	<class 'str'>
	<class 'str'>

n진수를 10진수로 변환하기 위해 int() 함수를 사용한다.

int(string, base) string: n진수에 해당하는 문자, base: 10

#각 진수에 해당하는 기호 생략 가능

```
print(int('0b11', 2), int('11',2))
print(int('0o11', 8), int('11', 8))
print(int('0x11', 16), int('11',16))
```

<실행 결과>

```
3 3
9 9
17 17
```

◦ math 라이브러리

```
import math
print(math.pi)
print(math.e)
print(math.trunc(1.78))
print(math.factorial(5))
print(math.degrees(math.pi))
print(math.radians(180))
print(math.sin(math.pi/2))
print(math.cos(math.pi/2))
print(math.log10(100))
print(math.log2(8))
print(math.pow(2,4))
```

<실행 결과>

```
3.141592653589793
2.718281828459045
1
120
180.0
3.141592653589793
1.0
6.123233995736766e-17
2.0
3.0
16.0
```

◦ random

```
import random as r
print(r.random())           #0부터 1사이의 난수(실수)
print(r.randint(10,20))     #10이상 20이하의 난수
print(r.randrange(10,20))   #10이상 20미만의 난수
```

<실행 결과>

```
0.9964217871507911
18
14
```

4.1.3. 날짜/시간 함수

◦ time

1970년 1월 1일 0시 0분 이후 경과한 시간을 초단위로 반환한다. 시간대는 표준시를 사용한다.

```
import time
print(time.time())
print(time.localtime(time.time()))
```

<실행 결과>

1620142357.4180183

```
time.struct_time(tm_year=2021, tm_mon=5, tm_mday=5, tm_hour=0, tm_min=32,
tm_sec=37, tm_wday=2, tm_yday=125, tm_isdst=0)
```

struct_time 구조체에서 각 요소별로 출력할 수 있지만 strftime를 사용하면 형식에 따라 출력할 수 있다.

```
import time
t=time.localtime(time.time())
print(f'{t.tm_year}-{t.tm_mon}-{t.tm_mday}')
print(time.strftime('%Y-%m-%d',t))
```

<실행 결과>

2021-10-23

2021-10-23

(오늘 날짜가 출력됨)

코드	설명	예
%a	요일 줄임말	Sun, Mon, ... Sat
%A	요일	Sunday, Monday,
%w	요일을 숫자로 표시, 월요일~일요일, 0~6	0, 1, ..., 6
%d	일	01, 02, ..., 31
%b	월 줄임말	Jan, Feb, ..., Dec
%B	월	January, February, ...,
%m	숫자 월	01, 02, ..., 12
%y	두 자릿수 연도	01, 02, ...,
%Y	네 자릿수 연도	2021
%H	시간(24시간)	00, 01, ..., 23
%I	시간(12시간)	01, 02, ..., 12
%p	AM, PM	AM, PM
%M	분	00, 01, ..., 59
%S	초	00, 01, ..., 59
%Z	시간대	대한민국 표준시
%j	1월 1일부터 경과한 일수	001, 002, ...,
%U	1년중 주차, 월요일이 한 주의 시작으로	00, 01, ...,
%W	1년중 주차, 월요일이 한 주의 시작으로	00, 01, ...,
%c	날짜, 요일, 시간을 출력, 현재 시간대 기준	Wed May 5 00:38:33 2021
%x	날짜를 출력, 현재 시간대 기준	05/19/18
%X	시간을 출력, 현재 시간대 기준	'11:44:22'

- datetime

```
import datetime
now = datetime.datetime.today()
print(datetime.datetime.today())
print(datetime.datetime.now())
print(now.year, now.month, now.day)
print(now.hour, now.minute, now.second)
print(now.strftime('%Y-%m-%d'))
print(now.strftime('%H-%M-%S'))
```

〈실행 결과〉

```
2021-10-23 10:49:36.057515
2021-10-23 10:49:36.057514
2021 10 23
10 49 36
2021-10-23
10-49-36
```

4.2. 사용자 정의 함수

- 함수 만들기

```
def function(parameter) :
    #함수에 작성할 코드
```

function은 사용자가 지정함. 변수 이름을 만드는 규칙과 동일함.
parameter는 인수이며, 여러 개일 경우 쉼표(,)로 구분한다.

- 함수 호출하기

```
def func1():
    #함수에 작성할 코드
func1()
```

함수는 호출하기 전에 정의되어 있어야 한다.

- 함수: 인수와 반환값이 없는 경우

```
def hi():
    print('Hi!')
hi()
```

〈실행 결과〉

Hi!

- 함수: 인수가 있고 반환값이 없는 경우

```
def hi(n):
    print('Hi!'*n)
hi(3)
```

〈실행 결과〉

Hi!Hi!Hi!

- 함수: 인수는 없고 반환값이 있는 경우

```
def hi():
    return 'Hi!'
print(hi())
```

<실행 결과>
Hi!

- 함수: 인수와 반환값이 있는 경우

```
def plus(a,b):
    return a+b
print(plus(3,2))
```

<실행 결과>
5

함수 호출시 사용자 정의 함수에서 인수의 순서에 맞게 지정한다. 순서를 변경해서 출력하려면 다음과 같이 사용할 수 있다.

```
print(calc(y=2,x=4))
```

print(plus(3))과 같이 함수의 인수의 개수가 맞지 않는 경우 에러가 발생한다.
TypeError: plus() missing 1 required positional argument: 'b'

함수에서 인수 값을 미리 지정한 경우 에러가 발생하지 않는다.

```
def plus(a,b=2):
    return a+b
print(plus(3))
```

<실행 결과>
5

- 함수: 반환값이 여러 개인 경우

```
def calc(a,b):
    return a+b, a-b, a*b
print(calc(3,2))
```

<실행 결과>
(5, 1, 6)

반환값이 여러 개인 경우 튜플(tuple) 형태로 반환된다.

- 함수: 가변 인수(인수의 개수를 정하지 않고 사용)

```
def data_sum(*n):
    s=0
    for i in n:
        s+=i
    return s
print(data_sum(1,2,3,4,5,6,7,8,9,10))
```

<실행 결과>
45

4.3. 재귀함수

함수에서 자기 자신을 호출(recursive call)하는 방식이다.

```
def test(x):  
    print(x)  
    test(x-1)  
print(test(10))
```

위와 같은 방식은 무한 루프에 빠지게 된다. 다음 예제를 비교하면서 재귀함수를 이해해 보자.

```
def test(x):  
    if x==0: return  
    print(x, end=' ')  
    test(x-1)  
print(test(5))
```

<실행 결과>

5 4 3 2 1

```
def test(x):  
    if x==0: return  
    test(x-1)  
    print(x, end=' ')  
print(test(5))
```

<실행 결과>

1 2 3 4 5

◦ 재귀함수로 factorial 구현하기

```
def factorial(n):  
    if n==1:      # n이 1일 때  
        return 1  # 1을 반환하고 재귀호출을 끝냄  
    return n*factorial(n-1)  
print(factorial(5))
```

<실행 결과>

120

factorial(5)의 진행과정을 살펴보자.

5 * factorial(4)

4 * factorial(3)

3 * factorial(2)

2 * factorial(1)

1을 return

return 5 * 4 * 3 * 2 * 1

return 4 * 3 * 2 * 1

return 3 * 2 * 1

return 2 * 1

if n==1: return 1 1을 반환

파이썬 math 라이브러리에는 factorial을 지원한다.

```
import math  
print(math.factorial(5))
```

퀴즈

Quiz 1. 피보나치 수열

피보나치 수는 0, 1로 시작한다. 세 번째 수는 앞의 두 숫자의 합이다.

식으로 작성하면 $F_n = F_{n-1} + F_{n-2}$ 이다. (n 은 2이상이다.)

입력: 정수 n 을 입력한다. ($n \geq 2$)

출력: n 번째 피보나치 수를 출력한다.

입력

10

출력

55

Quiz 2. 최대공약수와 최소공배수

재귀함수를 사용하여 입력받은 두 수의 최대공약수와 최소공배수를 출력하시오.

입력: 정수 두 개를 한 칸 단위로 입력한다. (두 수는 2이상이다.)

출력: 최대공약수와 최소공배수를 출력한다.

입력

12 18

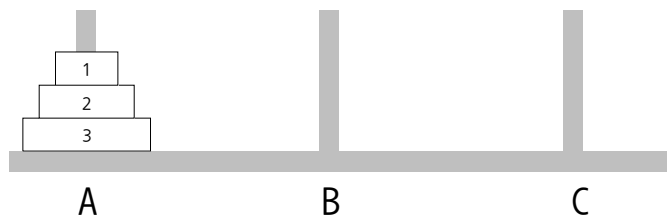
출력

6 36

Quiz 3. 하노이 타워

3개의 장대가 있고 첫 번째 장대에는 크기가 서로 다른 3개의 원판이 있다. A장대에 있던 원판을 다음 규칙에 따라 C로 이동시키려고 한다.

- ▶ 한 번에 한 개의 원판만 다른 장대로 이동시킬 수 있다.
- ▶ 위에 올려 놓은 원판은 아래 원판보다 작아야 한다.



원판의 개수를 입력하면 원판의 이동 회수를 첫 줄에 출력하고, 다음 줄부터는 이동 과정을 출력하는 프로그램을 작성하시오.

입력

3

출력

1번 원판: A → C
 2번 원판: A → B
 1번 원판: C → B
 3번 원판: A → C
 1번 원판: B → A
 2번 원판: B → C
 1번 원판: A → C

해설

Quiz 1. 피보나치 수열

$F_n = F_{n-1} + F_{n-2}$ n번째 피보나치 수를 출력하는 방법

◦ 반복문 사용

```
a,b=0,1
n=int(input())
for i in range(2, n+1):
    a,b=b,a+b
print(b)
```

n	0	1	2	3	4	5	6	7	8	9	10
a	0	0	1	1	2	3	5	8	13	21	34
b		1	1	2	3	5	8	13	21	34	55

◦ 재귀함수 사용

```
def fibo(n):
    if n==0: return 0
    elif n<2: return 1
    else:
        return fibo(n-1)+fibo(n-2)
n=int(input())
print(fibo(n))
```

그러나 위 프로그램은 n값이 커질수록 실행 속도가 느려지는 단점이 있다. 재귀함수를 사용할 때는 다음과 같이 반복문의 실행 횟수를 줄일 수 있다.

```
def fibo(n):
    global v
    if n>=len(v):
        v.append(fibo(n-1)+fibo(n-2))
    return v[n]
v=[0,1]
n=int(input())
print(fibo(n))
```


Quiz 2. 최대공약수와 최소공배수

◦ 반복문을 사용

1. 입력받은 두 수 중 더 작은 값을 d에 저장한다.
2. 두 수가 모두 d로 나누어 떨어지면 d가 최대 공약수이다.
3. 나누어 떨어지지 않으면 d의 값을 1 감소시킨다.
4. 2~3과정을 반복한다. d의 값이 1이면 모든 수는 1로 나누어 떨어지기 때문에 d가 1이 될 때까지 반복한다.

```
def gcd(a,b):
    d=min(a,b)
    while True:
        if a%d==0 and b%d==0:
            return d
        d-=1
a,b =map(int, input().split())
print(gcd(a,b))
```

◦ 재귀함수를 사용 (유클리드 알고리즘)

1. 두 수 a, b를 입력받음
2. 함수 호출시(b, a%b)의 형태로 변형(a%b는 b보다 크지 않음)
3. 나머지 연산의 결과가 0이 될 때까지 반복한다.
4. 어떤 수와 0의 최대공약수는 자기 자신이다.

```
def gcd(a,b):
    if b==0: return a
    return gcd(b, a%b)
a,b=map(int, input().split())
print(gcd(a,b))
```

Quiz 3. 하노이 타워

n개의 원반이 있다고 할 때

과정 1. A(처음)에서 (n-1)번째 원반을 B(경유지)로 옮긴다

과정 2. A(처음)에서 n번째 원반을 C(목적지)로 옮긴다

과정 3. B(경유지)에서 (n-1)번째 원반을 C(목적지)로 옮긴다

이 과정을 반복하면 하노이 타워 이동을 끝낼 수 있다.

```
def hanoi(n,a,b,c):
    if n==1:
        print(f'{n}번 원반: {a}-> {c}')
        return
    hanoi(n-1, a, c, b)
    print(f'{n}번 원반: {a}-> {c}')
    hanoi(n-1, b, a, c)
n=int(input())
hanoi(n,'A','B','C')
```

5. 파일

5.1. 파일 쓰기

파일을 읽거나 쓰기 위해서는 파이썬 내장 함수 `open`을 사용한다. `open()`을 사용한 경우 마지막에 `close()`를 사용해야한다.

```
파일 객체 = open(파일이름, 파일열기 모드)
파일 처리 명령
파일 객체.close()
```

```
f=open('test.txt','w')
```

위와 같이 지정할 경우, 파이썬 소스 파일이 있는 경로에 `test.txt` 파일이 생성된다.

특정 위치를 지정하고 싶다면, 다음과 같이 경로를 지정한다.

```
f=open('c:\\temp\\test.txt','w')
```

```
f=open(r'c:\temp\test.txt','w')
```

`with as`를 사용하면 `close()`를 사용하지 않아도 된다.

```
with open('test.txt','r') as f:
    print(f.read()) #들여쓰기를 해야 한다.
```

파일 열기 모드는 읽기(`r`), 쓰기(`w`), 추가(`a`)가 있다.

- `r` : 읽기 모드(파일을 읽기만 할 때 사용)
- `w` : 쓰기 모드(파일에 내용을 쓸 때 사용. 기존 내용은 삭제)
- `a` : 추가 모드(기존 파일 마지막에 내용을 추가할 때 사용)



`test.txt` 파일에는 다음과 같은 내용이 있다.

텍스트 파일의 인코딩 형식은 UTF-8이다.

파일 처리 프로그래밍을 할 경우 인코딩 형식에 따라 에러가 발생할 수 있다.

- `w` 모드로 파일을 쓸 경우: 기존 내용은 삭제되고 새로운 내용으로 채워짐.

```
f=open('pro.txt','w')
f.write('python\n')
f.write('java\n')
f.close()
```

```
f=open('pro.txt','w')
print('python', file=f)
print('java', file=f)
f.close()
```

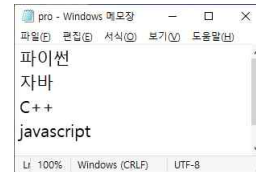
```
with open('pro.txt','w') as f:
    f.write('python\n')
    print('java',file=f)
```

`write` 명령은 다음 줄로 변경을 하지 않기 때문에 줄바꿈(`\n`)을 했다.

`with as`는 `close()`를 사용하지 않지만 들여쓰기를 해야한다.

- a 모드로 파일을 쓸 경우: 기존 내용 뒤에 새로운 내용이 추가됨

```
with open('pro.txt', 'a') as f:
    f.write('C++\n')
    f.write('javascript\n')
```



5.2. 파일 읽기

파일 읽기 모드는 read, readline, readlines를 지원한다.

- 읽기 모드 read(): 파일 전체를 읽는다.

```
with open('pro.txt', 'r') as f:
    t=f.read()
    print(t)
```

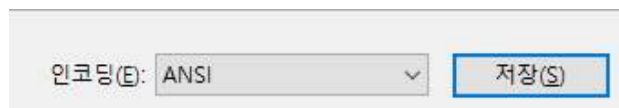
pro.txt 파일의 내용을 화면에 출력한다.

실행시 다음과 같은 에러가 발생할 수 있다.

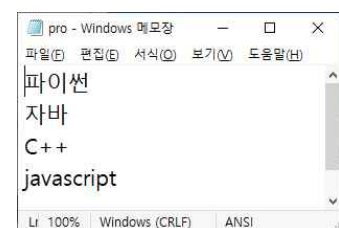
'cp949' codec can't decode byte 0xed in position 0: illegal multibyte sequence

Windows 운영체제의 메모장에서는 메모장 인코딩 형식이 UTF-8이다.

메모장에서 파일 저장시 인코딩 형식을 ANSI로 변경한다.



메모장 오른쪽 하단 부분에 ANSI를 확인할 수 있다.



파일 인코딩 형식을 변경할 수 없는 경우에 open()에 encoding 옵션을 설정한다.

```
with open('pro.txt', 'r', encoding='utf8') as f:
```

- 읽기 모드 readline(): 파일에서 한 줄을 읽는다.

```
with open('pro.txt', 'r') as f:
    t=f.readline()
    print(t)
```

<실행 결과>
파이썬

readline() 함수는 한 줄씩 읽는다.

전체 내용을 출력하고 싶다면 반복문을 사용해야 한다.

```
with open('pro.txt','r') as f:
    while True:
        t=f.readline()
        if not t: break
        print(t, end='')
```

<실행 결과>

```
python
java
C++
html
```

한 줄씩 읽다가 내용이 없다면 break를 사용하여 반복문을 탈출한다.

pro.txt 파일에도 한 줄마다 엔터 값이 있고, print() 함수도 내용을 출력한 후에 줄 바꿈기를 하기 때문에 실행 결과는 위와 같이 줄바꿈('\n')이 두 번씩 발생한다.

이를 방지하기 위해서는 print(t)를 print(t, end='')로 변경해야 한다.

◦ 읽기 모드 readlines(): 파일 전체 내용을 리스트 형태로 저장한다.

```
with open('pro.txt','r') as f:
    t=f.readlines()
    print(t)
```

<실행 결과>

```
['파이썬\n', '자바\n', 'C++\n', 'javascript\n']
```

readlines()는 파일 내용 전체를 리스트 형태로 저장한다. readlines() 함수로 읽은 경우 줄바꿈('\n')을 공백으로 대체하기 위해 replace() 함수를 사용한다.

파일 전체를 출력하는 명령은 다음과 같다. readlines() 함수를 통해 저장한 리스트 값의 줄바꿈 기호를 replace() 함수를 사용하여 줄바꿈을 공백으로 대체한다.

```
with open('pro.txt','r') as f:
    t=f.readlines()
    for k in t:
        print(k.replace('\n',''))
```

<실행 결과>

```
파이썬
자바
C++
javascript
```

5.3. 디렉터리, 파일 관리

파이썬 os, shutil 모듈에서 제공하는 함수로 폴더(디렉토리)와 파일을 관리할 수 있다.

◦ 폴더 정보

함수	설명
os.getcwd()	현재 작업 중인 디렉토리를 정보를 반환한다.
os.listdir(path)	디렉터리 내 파일과 하위 폴더를 리스트 형태로 반환한다. path 생략시 현재 디렉터리내 파일과 하위 폴더를 반환한다.
os.scandir(path)	파이썬 3.5버전부터 등장. 디렉터리내 모든 항목을 리스트가 아닌 iterator로 반환한다. 폴더와 파일 목록 뿐만 아니라 파일 크기, 수정된 날짜 등과 같은 속성 정보도 가져온다.
os.walk()	디렉터리 내 파일과 하위 폴더 정보를 모두 찾아서 반환한다.
os.chdir(path)	현재 디렉터리를 path로 변경

```
import os
print(os.getcwd())
os.chdir(r'c:\temp')
print(os.getcwd())
```

<실행 결과>
D:\프로그래밍
C:\temp

os.getcwd()는 파이썬 파일이 저장된 현재 경로를 출력한다.

os.chdir(경로명)을 실행하여 현재 경로를 C 드라이브의 temp 폴더로 변경하였다.

다시 os.getcwd()를 통해 변경된 경로를 확인할 수 있다.

```
f1=os.listdir()
f2=os.listdir('test')
f3=os.listdir(r'C:\temp')
print(f1, f2, f3, sep='\n')
```

<실행 결과>
['f51.py', 't43.py', 'test',]
['데이터.xlsx', '보고서.hwp', '사진']
['AUtempR', 'ffff', 'NE02', 'ODT']

listdir()은 파이썬 파일이 저장된 경로에서 폴더와 파일을 리스트 형태로 반환한다.

listdir('test')은 파이썬 파일이 저장된 경로에서 test 폴더 내의 폴더와 파일을 리스트 형태로 반환한다.

listdir(r'C:\temp')은 특정 경로의 폴더와 파일을 리스트 형태로 반환한다.

unix, linux의 경로 표시는 그대로 사용 가능하다. ("C:/Users/User/Desktop/")

```
file_list = os.scandir()
print(file_list)
for filename in file_list:
    print(filename, filename.name)
```

<실행 결과>
 <nt.ScandirIterator object at 0x0000000003535c70>
 <DirEntry 'f51.py'> f51.py
 <DirEntry 't43.py'> t43.py
 <DirEntry 'test'> test

os.scandir()은 폴더와 파일의 목록을 list 형태가 아닌 iterator로 반환한다.

<nt.ScandirIterator object at 0x0000000003535c70>

ScandirIterator는 파이썬 파일이 저장된 곳의 모든 항목을 포인팅 하기 때문에 반복문을 사용해 폴더와 파일 이름을 가져올 수 있다.

```
t = os.walk(r'D:\프로그래밍')
print(t)
for path, folders, files in t:
    print(path)
    print(folders)
    print(files)
```

<실행 결과>
 <generator object walk at 0x0000000003b914a0>
 D:\프로그래밍
 ['test']
 ['f51.py', 'test1.py']
 D:\프로그래밍\test
 []
 ['데이터.xlsx', '보고서.hwp']

os.walk()는 경로 내 모든 하위 폴더와 파일을 반환해준다. list 형태가 아닌 generator로 반환하며, 반복문으로 항목을 가져올 수 있다.

- path D:\프로그래밍 처음 방문한 곳의 경로
- folders ['test'] 방문한 경로에서 폴더 목록
- files ['f51.py', 'test1.py'] 방문한 경로에서 파일 목록

반복문을 통해 각각의 항목을 클릭한 후에 다시 하위 폴더를 방문하여 path, folders, files를 출력한다.

◦ 폴더 생성, 삭제, 이름 변경

함수	설명
os.mkdir(폴더명)	디렉토리를 생성. 기존에 폴더가 있으면 에러가 발생한다.
os.rmdir(폴더명)	디렉토리를 삭제. 디렉토리내 파일이 있으면 에러가 발생한다.
os.unlink(파일명) os.remove(파일명)	폴더(파일)를 삭제한다.
os.rename(src,dest)	폴더(파일)의 이름을 src에서 dest로 변경한다.

```
os.mkdir('test')    #현재 폴더에서 test 폴더를 만든다.
os.mkdir(r'test\tt') #test 폴더 내에서 tt 폴더를 만든다.
#unix식 경로 표현
os.mkdir(r'korea\incheon') #현재 폴더에서 korea 폴더 내에 incheon 폴더를 만든다.
os.mkdir('c:/test/프로그래밍')# 폴더를 만들 전체 경로 표시
```

os.mkdir(path)

path는 폴더명, 경로명으로 작성할 수 있으며, r스트링과 unix식 경로 표현을 지원한다. os.mkdir('test')는 현재 폴더에서 test 폴더를 만든다. 만일 이 프로그램을 두 번째 실행한다면 같은 폴더명이 존재하기 때문에 에러가 발생한다.

test 폴더가 만들어진 후에 os.mkdir(r'test\tt')는 test 폴더내에 tt 폴더를 만든다. 그러나 os.mkdir(r'korea\incheon')는 현재 폴더에서 korea 폴더가 없을 경우에는 에러가 발생한다.

```
os.mkdir('c:/test/프로그래밍') #unix식 경로 표현
```

C 드라이브에서 test 폴더가 존재하면 test 폴더에 프로그래밍 폴더를 만든다. 그러나 test 폴더가 존재하지 않는다면 에러가 발생한다.

C:\프로그래밍\파이썬\파일\os

C 드라이브 내에 프로그래밍 폴더가 없다면 폴더명을 하나씩 만들어야 한다. 이럴 경우 os.makedirs(path)를 사용하면 경로의 모든 하위 폴더를 생성한다.

```
os.makedirs('c:/프로그래밍/파이썬/파일/os1')    #unix식 경로 표현
os.makedirs('c:\\프로그래밍\\파이썬\\파일\\os2')  # || 사용
os.makedirs(r'c:\프로그래밍\파이썬\파일\os3')    # r스트링
```

os.rmdir(path)은 빈 폴더를 삭제하고, 빈 폴더가 아니면 에러가 발생한다.

```
os.rmdir(path) #unix식 경로 표현
```

파일을 삭제할 때는 remove()와 unlink()를 사용한다.

```
os.remove('test.hwp')
os.unlink('korea.hwp')
```

삭제하려는 파일이 없을 때는 에러가 발생한다.

FileNotFoundError (note: full exception trace is shown but execution is paused at: <module>)

◦ `os.path`

함수	설명
<code>os.path.exists()</code>	폴더(파일)의 존재 여부를 반환한다.
<code>os.path.isdir()</code>	폴더의 존재 여부를 반환한다.
<code>os.path.path.isfile()</code>	파일의 존재 여부를 반환한다.
<code>os.path.getsize()</code>	파일의 크기를 Byte 단위로 반환한다.
<code>os.path.split()</code> <code>os.path.splitext()</code>	경로와 파일을 분리한다.
<code>os.path.join()</code>	파일명과 경로를 합친다.
<code>os.path.dirname()</code>	파일을 제외한 경로만 반환한다.
<code>os.path.basename()</code>	파일 이름만 반환한다.

파일을 삭제할 때 파일의 존재 여부를 확인한 후에 삭제를 한다.

```
if os.path.isfile('test.hwp'):
    os.remove('test.hwp')
else:
    print('파일이 없습니다.')
```

```
if os.path.exists('test.hwp'):
    os.remove('test.hwp')
else:
    print('파일이 없습니다.')
```

폴더를 생성하거나 삭제할 때에도 `os.path.isdir()`이나 `os.path.exists()`를 사용하면 에러를 예방할 수 있다.

```
if os.path.isdir('test'):
    print('폴더가 존재함')
else:
    os.mkdir('test')
```

```
if not os.path.isdir('test'):
    os.mkdir('test')
```

현재 폴더에서 파이썬 파일의 목록과 파일의 크기를 출력해보자.

```
import os
for flist in os.listdir():
    if os.path.isfile(flist):
        print(flist, os.path.getsize(flist))
```

`os.scandir()`을 사용하면 다음과 같이 작성한다.

```
for flist in os.scandir():
    if os.path.isfile(flist):
        print(flist.name, os.path.getsize(flist))
```

os.path.dirname()과 os.path.basename()은 경로와 파일 이름만 출력하는 함수이다.

```
import os
mypath = r'D:\프로그래밍\hello.py'
print(os.path.dirname(mypath))    #D:\프로그래밍
print(os.path.basename(mypath))  #hello.py
```

현재 폴더내 파이썬 파일만 출력하려고 한다. 한글 파일(hwp)의 목록만 출력하는 방법을 생각해보자.

- 파일 이름에서 마지막 3글자를 추출

파일 이름에서 슬라이싱을 통해 마지막 3글자를 추출한 글자가 'hwp'이면 출력한다.

```
for flist in os.scandir():
    if os.path.isfile(flist) and flist.name[-3:] == 'hwp':
        print(flist.name)
```

- split()을 사용하여 점(.)을 기준으로 파일 이름과 확장자를 분리

split()을 사용하여 점(.)을 기준으로 확장자를 분리할 경우 파일 이름에 점(.)을 한 개만 사용해야 한다.

```
fname = 'hello.py'
file1 = fname.split('.')
print(file1[0], file1[1])
```

파일 이름에 점(.)을 여러 개 사용하는 경우는 리스트의 마지막 값이 확장자이다.

예) python-3.9.7-amd64.exe

```
fname = 'python-3.9.7-amd64.exe'
file1 = fname.split('.')
print(''.join(file1[:-1]), file1[-1])
```

파일 이름은 처음부터 마지막 전까지 점(.)을 기준으로 분리됐기 때문에 join() 함수를 사용하면 파일 이름을 출력할 수 있다.

- 파일 이름에서 파일 확장자를 분리

os.path.splitext()는 파일 이름과 확장자를 분리한다. 단, 확장자 이름만 추출되지 않고 점(.)을 포함한다.

```
for flist in os.scandir():
    if os.path.isfile(flist):
        fn, fe = os.path.splitext(flist)
        if fe == '.hwp':
            print(flist.name)
```

참고

os.path 모듈

os.path에서는 파일 처리

미국 ANSI에서 1963년에 정보교환용 7비트의 이진수 조합으로 만든 부호이다.
각 문자를 7비트로 표현하기 때문에 27(128개)의 문자를 표현할 수 있다.

파이썬에서는 ord(), chr()을 통해 아스키 코드를 다룰 수 있다.

ord(문자): 문자의 아스키 코드를 반환한다. print(ord('A')) #65

chr(숫자): 숫자에 맞는 아스키 코드를 반환한다. print(chr(65)) #A

```
for a in range(65, 91):  
    print(chr(a),end='')
```

<실행 결과>

ABCDEFGHIJKLMNOPQRSTUVWXYZ

◦ 시스템 명령어

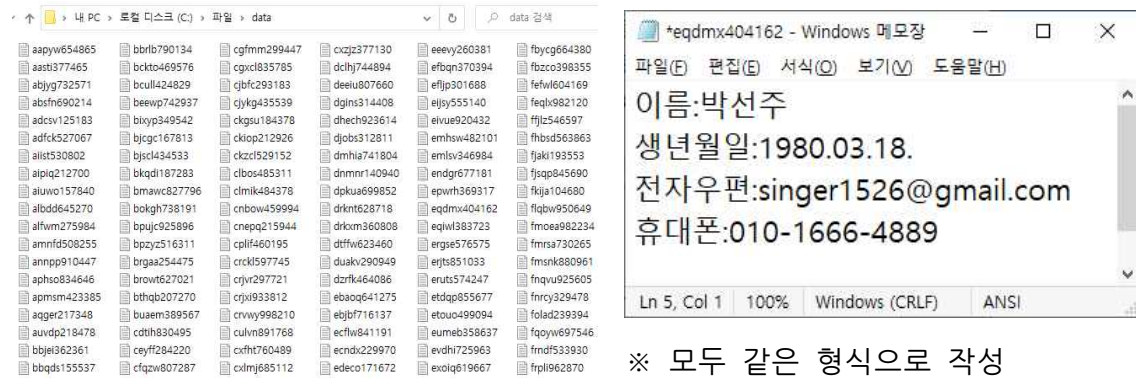
os.system(cmd명령어): cmd명령어를 실행(Windows 운영체제)

https://yganalyst.github.io/data_handling/memo_1/

특정 파일을 드라이브에서 검색하여 해당 파일이 존재하는 경로를 출력하려고 한다.
test.png 파일을 D 드라이브 특정 폴더에 숨겨놓았다.

[Quiz 1~3]

C:\파일\data에는 아래 그림과 같이 이름, 생년월일, 전자우편, 휴대폰 번호가 저장된 텍스트 파일이 500여개 있다.



※ 모두 같은 형식으로 작성

Quiz 1. 현재 파일 이름은 아무 의미없는 문자와 숫자의 조합이다. 파일 이름을 텍스트 파일에 있는 이름으로 변경하시오.(단, 동명이인은 없음)

eqdmx404162.txt ▶ 박선주.txt

Quiz 2. 여러 파일들을 하나의 파일(data1.txt)로 취합하시오.

파일 한 개당 4줄을 사용. data1.txt 파일은 파일 형식 그대로 전체를 취합

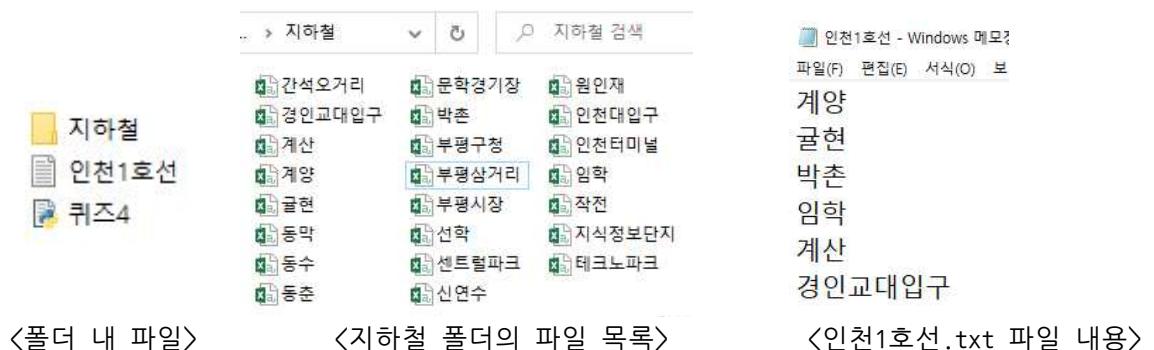
Quiz 3. 여러 파일들을 하나의 파일(data2.txt)로 취합하시오.

단, 줄 단위로 이름 생년월일 전자우편 휴대폰번호를 탭 간격으로 출력하시오.

Quiz 4. 인천1호선.txt 파일에는 전철역 이름이 한 줄씩 있다.

지하철 폴더에는 각 역에서 제출한 파일(역이름.csv)이 있다.

아직 제출하지 않은 지하철 역명을 미제출.txt 파일로 출력하시오.



<폴더 내 파일>

<지하철 폴더의 파일 목록>

<인천1호선.txt 파일 내용>

500 여개의 파일을 다루는 프로그램도 반복문을 사용한다. 모든 파일을 모두 사용하지 않고 3개 정도만 사용하는 것을 추천한다. 에러가 발생할 경우 많은 파일을 다시 초기화 해야 한다. 반복문을 사용하기 때문에 파일을 2개 이상 사용해서 에러가 발생하지 않는다면, 파일 모두를 대상으로 실행시킨다.

Quiz 1. 파일 이름 변경하기

아이디어

1. 파일 목록을 불러온다.
2. 파일을 한 개씩 열어서 첫 줄을 읽는다.
3. 첫 줄의 내용(이름:홍길동) 중 콜론(:)을 기준으로 내용을 분리한다.
4. 읽은 파일을 닫고, 파일 이름을 첫 줄에 읽은 이름으로 변경한다.

```
import os
file_list=os.listdir(r'C:\파일\data')
for fname in file_list: #파일 목록을 하나씩 반복한다.
    f=open(fname, 'r') # 첫 번째 파일을 읽기 모드로 연다.
    print(f.readline()) # 파일의 첫 번째 줄을 출력한다.
    f.close() # 파일 닫기
```

그러나 의도했던 것과는 다르게 에러가 발생한다.

Traceback (most recent call last):

File "C:\파일\파일문제.py", line 4, in <module>

f=open(fname, 'r')

FileNotFoundError: [Errno 2] No such file or directory: 'aapyw654865.txt '

파이썬 파일이 저장된 곳과 파일을 읽어할 경로가 다르기 때문에 에러가 발생한다.

os.chdir(r'C:\파일\data')을 사용해서 현재 경로를 변경할 수 있다.

그러나 os.scandir()을 사용할 경우 경로 변경없이 사용할 수 있다. with as로 파일을 open()할 경우 close()를 사용하지 않아도 된다.

```
import os
file_list=os.scandir(r'C:\파일\data')
for fname in file_list:
    with open(fname, 'r')as f:
        print(f.readline())
```

이름:최락

이름:황금심

이름:민경훈

f.readline()으로 파일의 첫 줄을 읽어서 출력했다. print 명령이 한 줄을 띄어쓰기 하며, 파일의 내용에서도 줄바꿈을 가져왔기 때문에 두 줄씩 띄어쓰기가 된다.

파일에서 읽은 줄바꿈(Enter)을 없애는 방법

```
data=f.readline().replace('\n','')
```

'이름:'을 삭제하는 방법은 다음과 같다.

◦ **replace()**를 사용

```
data=data.replace('이름:', '')
```

파일의 첫 줄만 읽어서 '이름:'을 삭제할 때는 `replace()`를 사용할 수 있다. 하지만 파일의 모든 줄에서 ':'을 기준으로 앞 내용을 삭제할 때는 ':'을 기준으로 분리하는 `split()` 사용을 추천한다.

◦ **split()**을 사용

```
data=f.readline().replace('\n','') # 이름:홍길동   줄변경을 삭제
data=data.split(':') # data[0]: 이름, data[1]: 홍길동
print(data[-1])      # 홍길동   이름만 추출
```

os.rename(src, des)를 사용할 경우

```
with open(fname, 'r') as f:
    data=f.readline().replace('\n','')
    data=data.split(':')
    re_file=f'{data[-1]}.txt'
os.rename(fname, re_file)
```

위 소스는 다음과 같은 에러가 발생한다.

(note: full exception trace is shown but execution is paused at: <module>)

```
os.rename(fname, re_file)
```

변경할 파일 이름도 파일 경로를 설정해야 한다.

```
import os
file_path = r'C:\파일\data'
file_list =os.scandir(file_path)
for fname in file_list:
    with open(fname, 'r') as f:
        data=f.readline().replace('\n','')
        data=data.split(':')
        re_file =os.path.join(file_path, f'{data[-1]}.txt')
    os.rename(fname, re_file)
```

Quiz 2. 파일 취합(파일 내용 그대로 취합)

아이디어

1. data1.txt 파일을 쓰기 모드로 연다.
2. 파일 목록을 불러온다.
3. 파일을 한 개씩 열어서 전체 내용을 읽어서 data1.txt에 기록한다.
4. 반복문으로 열었던 파일은 닫는다.
5. data1.txt 파일을 닫는다.

방법 1.

```
import os
file_path=r'C:\파일\data'
f=open('data1.txt', 'w')
os.chdir(file_path)
flist=os.listdir()
for fname in flist:
    read_file =open(fname, 'r')
    f.write(read_file.read())
    read_file.close()
f.close()
```

방법 2.

```
import os
flist=os.scandir(r'C:\파일\data')
f=open('data1.txt', 'w')
for fname in flist:
    read_file=open(fname, 'r')
    f.write(read_file.read())
    read_file.close()
f.close()
```

방법 3.

```
import os
flist=os.scandir(r'C:\파일\data')
with open('data1.txt', 'w') as f:
    for fname in flist:
        with open(fname, 'r') as read_file:
            f.write(read_file.read())
```

os.scandir()과 with as를 사용하여 코딩의 길이를 줄일 수 있다.

Quiz 3. 파일 취합(줄 단위 취합)

아이디어

1. data2.txt 파일을 쓰기 모드로 연다.
2. 파일 목록을 불러온다.
3. 파일을 한 개씩 열어서 전체 내용을 리스트 단위로 연다.
4. 내용 중 콜론(:)을 기준으로 내용을 분리한다.

기존 : ['이름:홍길동\n', '생년월일:2000.10.06.\n', '전자우편:singer1065@daum.net\n', '휴대폰:010-2733-9468\n']

변경 : ['홍길동', '2000.10.06', 'singer1065@daum.net', '010-2733-9468']

5. 변경한 내용을 data1.txt 파일에 기록한다.
6. data2.txt 파일을 닫는다.

```
import os
flist=os.scandir(r'C:\파일\data')
with open('data1.txt', 'w') as f:
    for fname in flist:
        with open(fname, 'r') as read_file:
            data=read_file.readlines() #내용을 분리
            print(data)
```

<실행 결과>

['이름:권혜경\n', '생년월일:1973.04.04.\n', '전자우편:singer1067@gmail.com\n', '휴대폰:010-6543-3908\n']

...반복 생략....

['이름:명국환\n', '생년월일:1984.06.03.\n', '전자우편:singer1328@naver.com\n', '휴대폰:010-5376-9392\n']

• 파일에 제목과 줄바꿈을 삭제하는 방법

```
data=read_file.readlines() #내용을 분리
for a in range(len(data)):
    data[a]=data[a].replace('\n','')
    cdata =data[a].split(':')
    data[a]=cdata[-1]
print(data)
```

```
data[a]=data[a].replace('\n','')
```

['이름:권혜경', '생년월일:1973.04.04.', '전자우편:singer1067@gmail.com', '휴대폰:010-6543-3908']


```
cdata =data[a].split(':')
data[a]=cdata[-1]
```

실제 데이터만 추출한 결과가 리스트로 나타난다.

```
['권혜경', '1973.04.04.', 'singer1067@gmail.com', '010-6543-3908']
```

리스트의 내용을 Tab으로 연결하기 위해 join()을 사용한다.

```
print('\t'.join(data))
```

<실행 결과>

```
권혜경  1973.04.04.      singer1067@gmail.com    010-6543-3908
```

원하는 결과를 얻었다면 data2.txt 파일에 기록한다.

```
import os
flist=os.scandir(r'C:\파일\data')
with open('data2.txt', 'w') as f:
    for fname in flist:
        with open(fname, 'r') as read_file:
            data=read_file.readlines() #내용을 분리
            for a in range(len(data)):
                data[a]=data[a].replace('\n','')
                cdata =data[a].split(':')
                data[a]=cdata[-1]
            print('\t'.join(data), file=f)
```

Quiz 4. 파일 목록에 없는 내용 추출

아이디어

1. 인천1호선.txt 파일의 내용을 리스트 형태로 저장한다. (list1)
2. 지하철 폴더의 파일 목록을 리스트 형태로 저장한다. (list2)
3. list1과 list2의 내용이 중복된게 있다면 list1에서 해당 내용을 삭제한다.
4. list1에 남은 내용은 지하철 폴더에 없는 파일이다.
5. list1에 남은 내용을 파일(미제출.txt)로 저장한다.

```
import os
subway=os.listdir('지하철')
print(subway)
```

<실행 결과>

['간석오거리.csv', '경인교대입구.csv', '생략', '지식정보단지.csv', '테크노파크.csv']

```
with open('인천1호선.txt','r') as f:
    data=f.readlines()
print(data)
```

<실행 결과>

['계양\n', '굴현\n', '박촌\n', '생략', '센트럴파크\n', '국제업무지구']

지하철 폴더의 목록에서는 .csv를 삭제하고 인천1호선.txt 파일 내용의 목록에서는 줄바꿈을 삭제한 후 비교해야 한다.

```
import os
subway=os.listdir('지하철')
for i in range(len(subway)):
    subway[i]=subway[i].replace('.csv','')
incheon=[]
with open('인천1호선.txt','r')as f:
    data=f.readlines()
    for i in range(len(data)):
        incheon.append(data[i].replace('\n',''))
print(incheon)
print(subway)
```

두 개의 리스트 모두 특정 글자를 삭제해야 한다. 삭제하려는 내용만 다를 뿐 반복문의 형식은 동일하기 때문에 사용자 정의 함수를 제작해보자.

```
import os
def Only_Data(mylist, del_txt):
    test=[]
    for t in mylist:
        test.append(t.replace(del_txt, ''))
    return test
subway=Only_Data(os.listdir('지하철'), '.csv')
with open('인천1호선.txt', 'r') as f:
    incheon=Only_Data(f.readlines(), '\n')

print(subway)
print(incheon)
```

사용자 정의 함수 Only_data()는 리스트 처리 방법에 따라 다음과 같이 변경할 수 있다.

```
def Only_Data2(mylist, del_txt):
    for a in range(len(mylist)):
        mylist[a]=mylist[a].replace(del_txt, '')
    return mylist
```

두 리스트를 출력하면 '.csv'와 줄바꿈 기호('\n')가 사라진 것을 확인할 수 있다.

인천1호선.txt 파일의 내용은 전체 역을 포함하고 있다.

지하철 폴더의 내용이 인천1호선.txt에 있다면 해당 내용을 삭제해보자.

지하철 폴더의 모든 내용을 삭제하면 제출하지 않은 지하철 역의 이름만 남게 된다.

```
for station in subway:
    if station in incheon:
        incheon.remove(station)

with open('미제출.txt', 'w') as f2:
    f2.write('\n'.join(incheon))
```

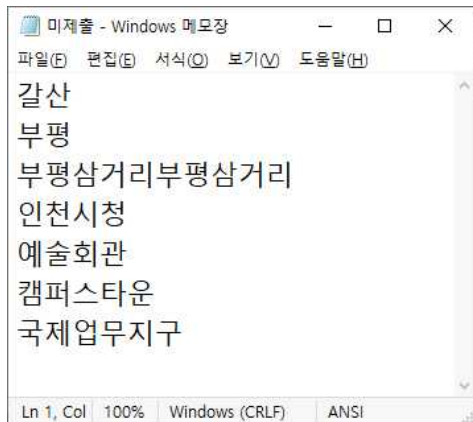
```

import os
def Only_Data2(mylist,del_txt):
    for a in range(len(mylist)):
        mylist[a]=mylist[a].replace(del_txt, '')
    return mylist

subway=Only_Data2(os.listdir('지하철'), '.csv')
with open('인천1호선.txt', 'r')as f:
    incheon=Only_Data2(f.readlines(), '\n')
for station in subway:
    if station in incheon:
        incheon.remove(station)
with open('미제출.txt','w')as f2:
    f2.write('\n'.join(incheon))

```

<실행 결과>



6. 인공지능

6.1. Google colab 소개

◦시작하기

Colaboratory는 설치할 필요가 없으며, 클라우드에서 실행되는 무료 Jupyter 노트북 환경이다. Colaboratory를 사용하면 브라우저를 통해 무료로 코드를 작성 및 실행하고, 분석을 저장 및 공유하며, 강력한 컴퓨팅 리소스를 이용할 수 있다.

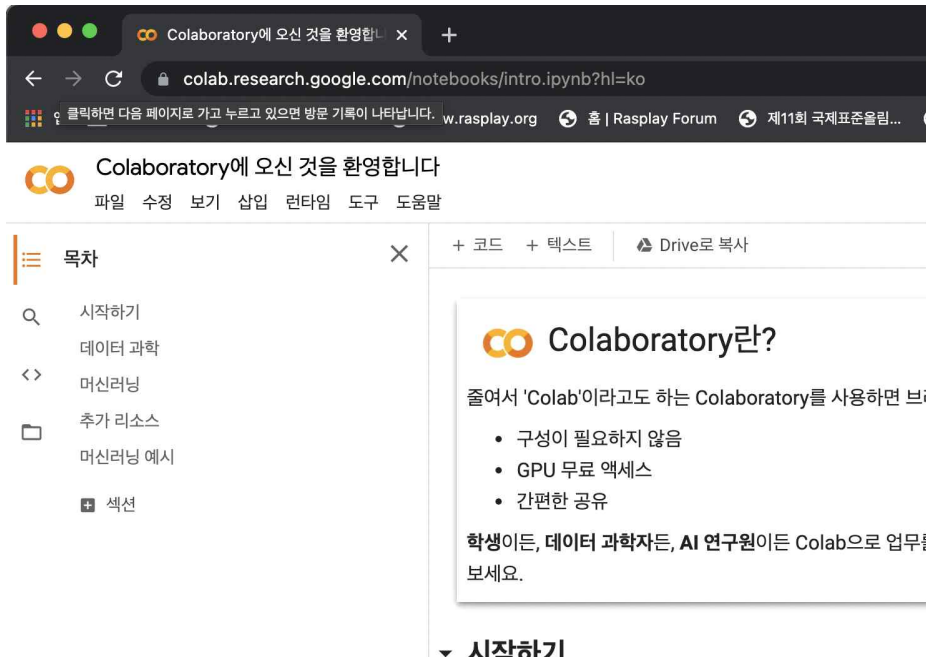
참고 영상 : <https://youtu.be/inN8seMm7UI>

◦Tensorflow 개념

텐서플로우는 데이터 흐름 그래프를 이용하여 계산을 하는 오픈소스 라이브러리이다. 구글 브레인 팀에서 개발하였고, 현재 머신러닝 분야에서 활발하게 사용되고 있다. 텐서는 여러 차원의 배열의 형태를 한 숫자값의 모음으로 구성하였다. 텐서의 rank는 차원의 수이다. 텐서플로우를 사용하면 차원 수가 높은 텐서를 조작할 수 있다. 텐서 플로우 연산은 텐서를 만들고 삭제하고 조작한다. 일반적인 텐서플로우 프로그램에서 코드는 연산이다.

◦Google Colaboratory 시작하기

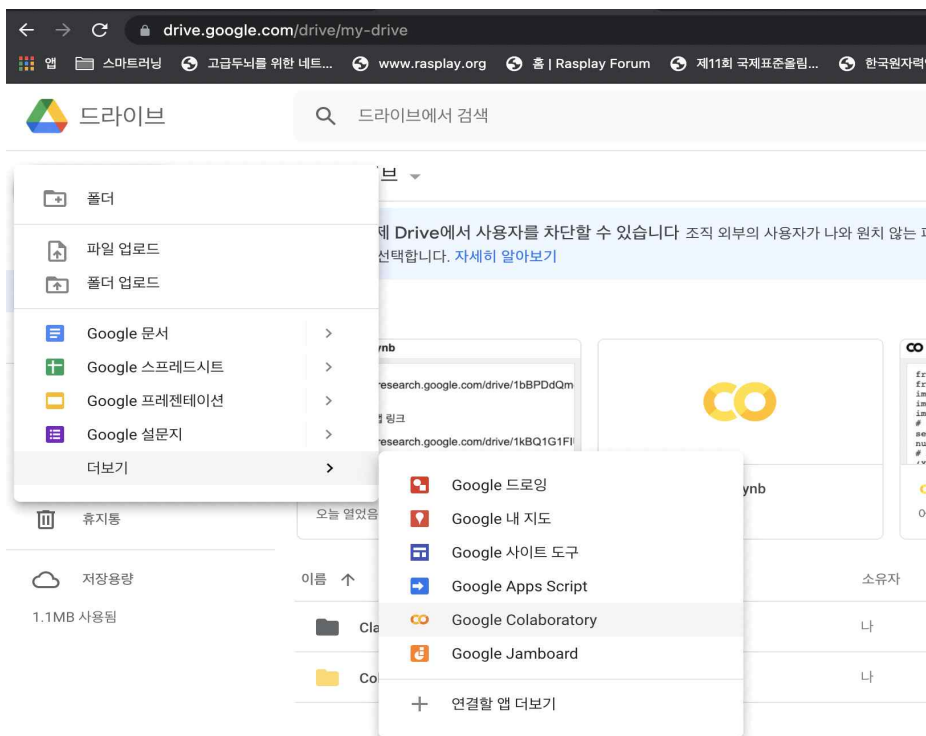
구글 계정을 만들거나 구글 계정이 있으면 <http://colab.research.google.com> 에 접속하여 다음과 같이 로그인한다.



출처 : <http://colab.research.google.com>

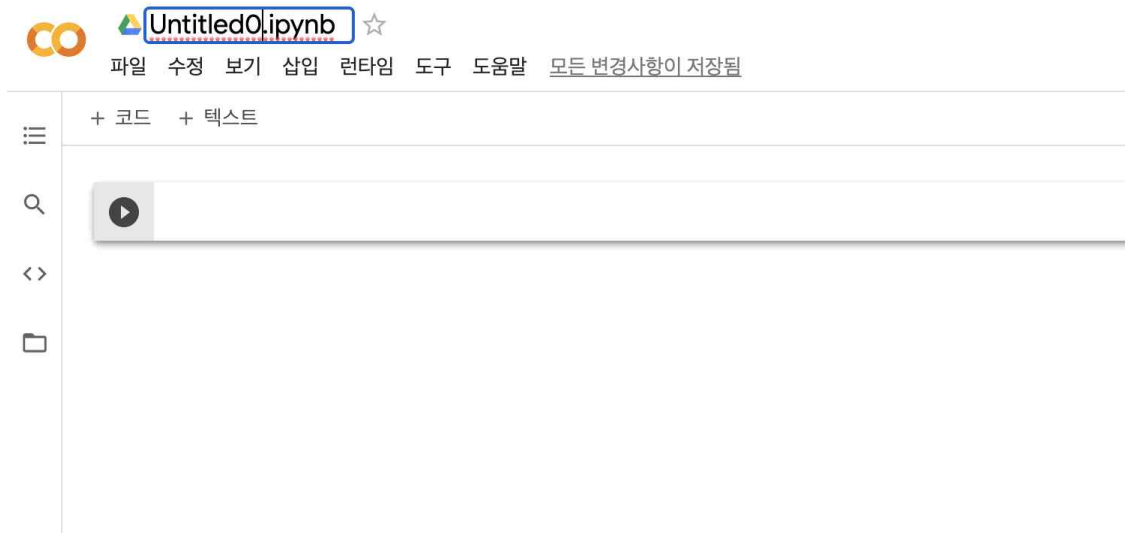
°Google Colaboratory 설치하기

구글 계정에 접속한 후 구글 드라이브에 접속하면 다음과 같은 화면이 보이게 된다. 더보기 버튼을 클릭하여 Google Colaboratory에 연결하면 링크로 접속하는 화면과 동일한 화면이 나타난다.



◦Google Colaboratory 시작하기

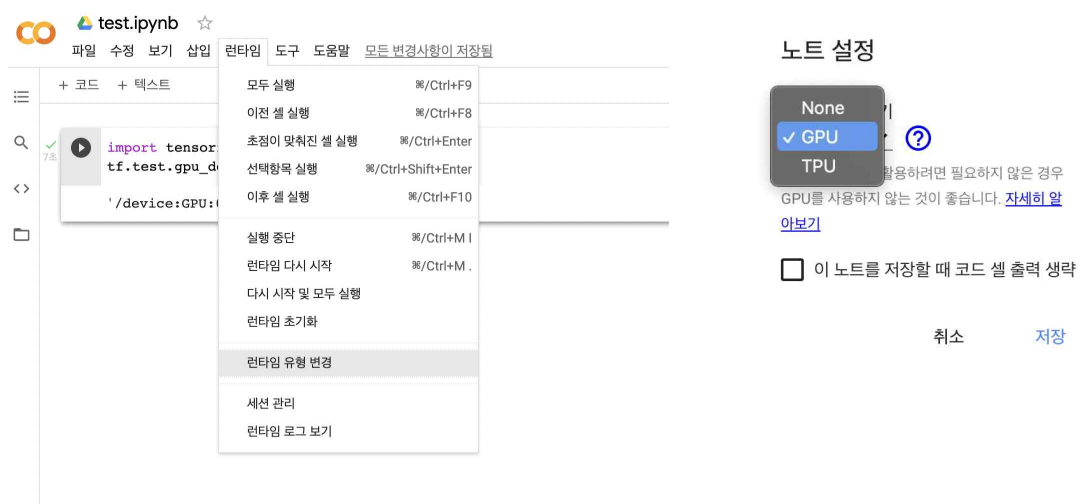
구글 계정에 접속한 후 구글 드라이브에 접속하면 다음과 같은 화면이 보이게 된다. 더보기를 클릭하여 Google Colaboratory에 연결하면 링크로 접속하는 화면과 동일한 화면이 나타난다.




출처 : <http://colab.research.google.com>

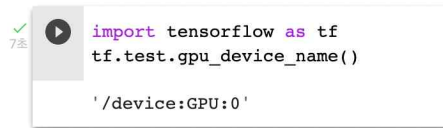
◦GPU 설정

GPU가속기 컴퓨팅은 그래픽 처리 장치(GPU)와 CPU를 함께 이용하여 과학, 수학, 공학, 데이터 분석의 처리 속도를 향상시키는데 있다. GPU향상은 프로그램의 향상을 의미하며 현대 여러 산업 분야에서 다양한 플랫폼의 가속을 의미한다.



런타임 - 런타임유형변경을 클릭 후 노트설정에서 GPU를 선택할 수 있다. 이후 코드입력 창

을 통해서 다음과 같이 입력을 한 후  또는 ctrl + Enter키를 클릭하면 '/device:GPU:0' 출력되면 정상적으로 설정이 마무리 된다.



```
import tensorflow as tf
tf.test.gpu_device_name()

'/device:GPU:0'
```

출처 : <http://colab.research.google.com>

6.2. 텐서플로를 이용한 숫자인식

◦ MNIST를 활용한 손글씨 인식

1. 도구준비

```
1 import tensorflow as tf
2 import numpy as np
3
4 import matplotlib.pyplot as plt
5 %matplotlib inline
6
7 import warnings
8 warnings.filterwarnings('ignore')
```

일반적인 머신러닝은 구글에서 제공하는 텐서플로를 가장 많이 이용하면 가장 쉽게 접근이 가능하다. 1번줄은 tensorflow를 코랩에서 불러오는 명령어이며 앞으로 계속 사용할 예정이기 때문에 tf로 줄여서 계속하여 호출한다는 의미이다. 또한 2번줄의 numpy 역시 호출하여 사용하되 계속하여 사용할 예정이기 때문에 np로 줄여서 사용한다는 의미이다.

4번줄은 이미지를 화면상에서 출력하기 위한 명령어이다.

7번, 8번줄은 warnings(경고메시지)이 프로그램 실행시에 발생하면 그것을 무시하는 구문이다. 주석처리를 통해서 7번과 8번줄의 코딩은 하지 않아도 결과는 똑같이 나온다.

2. 데이터 읽어오기

```
9  mnist=tf.keras.datasets.mnist
10 (x_train, y_train), (x_test, y_test)=mnist.load_data()
11 print(f'학습데이터 {len(x_train)}개 평가 데이터 {len(x_test)}개')
```

<실행결과>

학습데이터 60000개 평가데이터 10000개

9번줄은 케라스의 데이터셋을 이용하여 mnist데이터 셋을 호출하는 구문이다. 케라스는 딥러닝을 여행에 비유한다면 텐서플로는 목적지까지 빠르게 이동시켜주는 비행기에 해당되고, 케라스는 비행기의 이륙 및 정확한 지점까지 도착을 책임지는 파일럿에 해당된다고 한다. mnist데이터는 케라스에서 무료로 제공하고 있으며 데이터의 완성도도 아주 높다. 10번줄은 9번줄에서 호출한 mnist데이터가 압축이 된 형태로 다운로드되며 결과화면에 주소와 함께 표시된다. 11번 줄에서 학습 데이터와 평가 데이터가 레이블로 각각 나누어서 저장된다. 실행하면 학습데이터 60000개와 평가 데이터 10000개로 총 70000개의 데이터가 들어 있음을 확인할 수 있다.

3. 데이터 살펴보기

```
12 print(f'학습데이터 {x_train.shape} 레이블 {y_train.shape} y {y_train}')
```

```
13 print(f'평가데이터 {x_test.shape} 레이블 {y_test.shape} y {y_test}')
```

<실행결과>

```
학습데이터 (60000, 28, 28) 레이블 (60000,) y [5 0 4 ... 5 6 8]
평가데이터 (10000, 28, 28) 레이블 (10000,) y [7 2 1 ... 4 5 6]
```

실행결과를 확인해 보면 학습데이터가 60000개가 들어 있다고 보여주고 있으며 5, 0, 4로 시작되고 있다. 평가 데이터는 10000개가 들어 있으며 정답 레이블이 7, 2, 1로 보여 주고 있다.

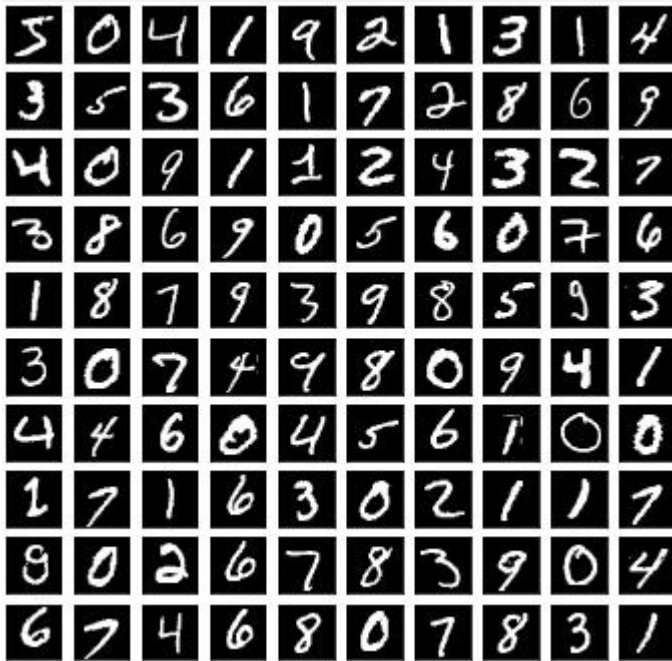
```
14 import matplotlib.pyplot as plt
15
```

```

16 row=10
17 col=10
18 n=row*col
19
20 plt.figure(figsize=(6,6))
21 for i in range(n):
22
23     ax=plt.subplot(row,col,i+1)
24     plt.imshow(x_train[i].reshape(28,28))
25     plt.gray()
26     ax.get_xaxis().set_visible(False)
27     ax.get_yaxis().set_visible(False)
28 plt.show()

```

<실행결과>



학습 이미지가 어떻게 보여지는지 출력하여 보는 구문이다. 가로 X 세로= 10으로 출력하여 보여준다. 20번 줄에서 figsize를 6,6으로 보여주며, 21번줄에서는 for문을 통해서 한글자씩 호출 하되 크기를 글자의 이미지를 28, 28로 줄이고 글자의 색상은 gray색상(흑백)으로 전환하여 보여준다. 14번줄의 matplotlib라이브러리를 호출하는 구문으로 24번줄의 imshow()함수를 이용하여 이미지를 출력한다.

4. 데이터 전처리

```
29 x_train, x_test=x_train/255.0, x_test/255.0
```

29번줄은 데이터 전처리 과정으로 학습데이터와 평가데이터를 255로 나누어준다. 0부터 255까지 값으로 바꾸어서 계산을 쉽게 한다.

5. 모델 정의

```
30 model=tf.keras.models.Sequential([
31     tf.keras.layers.Flatten(input_shape=(28,28)),
32     tf.keras.layers.Dense(128,activation='relu'),
33     tf.keras.layers.Dropout(0.2),
34     tf.keras.layers.Dense(10,activation='softmax')
35 ])
```

30번줄은 모델을 정의하여 생성하는 구문으로 Sequential모델을 생성한다. 31번줄의 input_shape은 12번~13번줄의 출력과 같이 28, 28로 되어 있다. 32번줄의 Dense는 relu함수를 사용하여 128개의 출력값을 가지며, 34번줄의 Dense는 softmax함수를 사용하여 10개로 출력되며 그 값은 0부터 9까지의 값을 가진다.

```
36 model.summary()
```

<실행결과>

```
Model: "sequential_3"
```

Layer (type)	Output Shape	Param #
flatten_6 (Flatten)	(None, 784)	0
dense_6 (Dense)	(None, 128)	100480
dropout_3 (Dropout)	(None, 128)	0
dense_7 (Dense)	(None, 10)	1290

```
Total params: 101,770  
Trainable params: 101,770  
Non-trainable params: 0
```

36번줄은 모델의 어떤 형태인지 확인하는 구문이다.

6. 모델 준비

```
37 model.compile(optimizer='adam',  
38               loss='sparse_categorical_crossentropy',  
39               metrics=['accuracy'])
```

37번줄은 optimizer는 오차를 어떻게 줄여 나갈지 정하는 함수이다. adam을 사용하게 되며, 28번줄의 loss는 한번 신경망이나 머신러닝이 실행할 때마다 오차값을 추적하는 함수이다. 사용된 함수는 sparse_categorical_crossentropy이다.

7. 모델 학습

```
40 model.fit(x_train, y_train, epochs=5)
```

<실행결과>

```
Epoch 1/5
1875/1875 [=====] - 3s 1ms/step - loss: 0.0647 - accuracy: 0.9794
Epoch 2/5
1875/1875 [=====] - 3s 1ms/step - loss: 0.0584 - accuracy: 0.9814
Epoch 3/5
1875/1875 [=====] - 3s 1ms/step - loss: 0.0529 - accuracy: 0.9825
Epoch 4/5
1875/1875 [=====] - 3s 1ms/step - loss: 0.0492 - accuracy: 0.9839
Epoch 5/5
1875/1875 [=====] - 3s 1ms/step - loss: 0.0443 - accuracy: 0.9858
```

40번줄은 모델을 학습하는 구문으로 epochs=5에 따라 5번 학습을 실행한다. 마지막 5번째 실행결과를 보면 98%이상의 정확도가 향상되고 있다. loss로 0.064에서 0.0443으로 줄어들고 있다.

8. 모델 평가

```
41 model.evaluate(x_test, y_test)
```

<실행결과>

```
313/313 [=====] - 0s 872us/step - loss: 0.0684 - accuracy: 0.9800
[0.06840541213750839, 0.9800000190734863]
```

41번줄은 모델을 평가하게 되며 평가데이터를 가지고 평가한다. 이때의 손실값은 0.0684가 나오며 정확도는 98%이다. 학습때보다 손실값과 정확도가 조금 더 적게 나왔다.

9. 결과 예측

```
42 pred=model.predict(x_test)
43 np.set_printoptions(precision=7)
44 print(f'각 클래스에 속할 확률: \n{pred[0]}')
```

<실행결과>

```
각 클래스에 속할 확률:
[8.2565812e-09 3.2683199e-12 5.9243615e-07 8.7962262e-06 3.0448012e-15
 5.8412264e-10 1.9174460e-17 9.9999034e-01 1.4282313e-09 2.3527036e-07]
```

평가 데이터의 첫 번째 데이터를 가지고 어느 클래스에 속하는지 결과를 예측해 보는 구문이다. 순서대로 0, 1, 2, 3, 4, 5, 6, 7, 8, 9에 속할 확률로 7이 9.9×10^{-1} 이 가장 높은 확률을 보이고 있다.

10. 만든 이미지 실행하여 결과 예측하기

```
45 from PIL import Image
46 img_png=Image.open("./7.png").convert('L')
47 img_png.size
```

<실행결과>

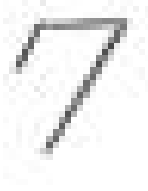
```
(605, 806)
```

45번 ~ 47번 줄은 업로드된 이미지를 가지고 테스트하는 구문이다. 46번 줄의 `.convert('L')`은 이미지 파일(7.png)을 연 뒤 grayscale로 바꾸어 주는 명령어이다. 47번줄은 업로드한 7.png 파일의 사이즈를 결과값으로 리턴해 주고 있다.

```
48 img_mnist=img_png.resize((28,28))
```

```
49  img_mnist
```

<실행결과>



47번줄에서 출력된 이미지 사이즈를 학습하기 위해 28, 28로 바꾸어 주는 구문이다. 배경은 흰색이며 글자는 검은색으로 출력된 것을 볼 수 있다.

```
50  np_im=(255-np.array(img_mnist))
51  np_im.shape
```

<실행결과>

```
(28, 28)
```

50번줄은 255에서 각 픽셀의 값을 빼주면서 반전(흰색-->검정)을 하는 부분이다.

```
52  tt=np.array2string(np_im,separator=",")
53  tt=tt.replace("\n","")
54  tt=tt.replace("[","")
55  tt=tt.replace("]", "")
56  tt=tt.replace(" ", "")
57  tt=tt.replace(" ", "")
58  tt="*, " + tt
59  print(tt)
60
61  data_value=tt.split(",")
62  image_arr = np.asfarray(data_value[1:]).reshape((28,28))
```

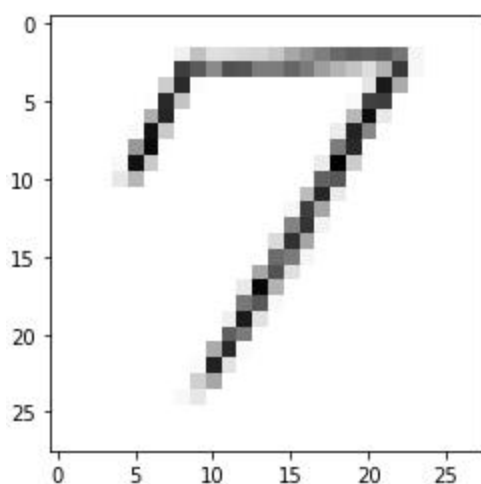
〈실행결과〉

[illegible]

전처리하는 과정으로 실행결과의 중간값을 보면 0이 아닌 다른 숫자들이 출력됨을 확인할 수 있다. 반전한 후의 값이 잘 들어갔는지 확인하는 부분으로 0은 검은색을 나타내고 있다. 따라서 흰색과 검정색으로 반전이 잘 되고 있다.

```
63 plt.imshow(image_arr, cmap='Greys', interpolation='None')
64 tplt.show()
```

〈실행결과〉



전처리 한 후 이미지를 출력하는 모습으로 이미지가 7로 출력되고 있다. 가로 세로 28x28=784개의 픽셀로 구성되어 있다. 반전된 이미지이기 때문에 흰색배경이 0이라면 글자가 들어간 곳은 1~255까지 숫자 중에 하나로 채워져 행렬로 구성된다. 63번줄의 cmap=Greys는 이미지의 옵션을 지정해 흑백으로 처리해 준다.


```

65 image_array=np.asfarray(data_value[1:]).reshape((1,28,28))

66 predictions=model(image_array).numpy()
67 result=np.argmax(tf.nn.softmax(predictions).numpy())

68 print(f'나의 이미지 분류 결과: {result}')

```

<실행결과>

나의 이미지 분류 결과: 7

전처리 한 후 이미지를 출력하는 모습으로 이미지가 7로 출력되고 있다. 65번줄의 reshape() 함수를 이용해 가로 세로 28x28=784개의 픽셀로 구성해 준다. reshape(총 샘플 수, 1차원 속성의 수)형식으로 지정한다. 1차원 속성의 수는 앞에서 본 바와 같이 28x28=784개이다.

6.3. 텐서플로를 활용한 이미지 분류하기

◦ 고양이와 강아지 이미지 분류하기

1. 도구준비

```
1. import cv2
2. import matplotlib.pyplot as plt

3. import tensorflow_datasets as tfds
4. import tensorflow as tf
```

1번줄은 CV2를 코랩에서 불러오는 명령어이며 이미지를 처리하기 위한 라이브러리이다. 2번 줄도 matplotlib를 사용하여 이미지를 처리하기 위한 명령어이며 앞으로 계속해서 사용하기 때문에 plt로 줄여서 사용한다는 의미이다.

3번줄은 텐서플로의 데이터셋을 호출하는 명령어로 앞으로 계속해서 사용하기 위해서 tfds으로 줄여서 사용하게 된다.

4번줄은 딥러닝 모델을 만들기 위해서 텐서플로를 tf로 줄여서 사용하게 된다.

2. 데이터 읽어오기

```
5. data_train, ds_info=tfds.load('cats_vs_dogs',
    split=[tfds.Split.TRAIN],with_info=True)
6. ds_info
```

5번줄은 텐서플로에서 기본적으로 제공되는 데이터를 가져오는 구문이다. tfds 데이터셋이라는 라이브러리를 통해서 cats_vs_dogs 데이터셋을 호출하게 된다. 참고적으로 텐서플로 공식 사이트를 통해서 제공되는 이미지셋은 다음과 같다.

(참고 사이트 : https://www.tensorflow.org/datasets/catalog/cats_vs_dogs)

TensorFlow

설치 학습 API 리소스 커뮤니티 TensorFlow를 사용해야 하는 이유

Datasets

개요 Catalog 가이드 API

Overview

- Audio
- D4ri
- Graphs
- Image
- Image classification
 - beans
 - bigearthnet
 - binary_alpha_digits
 - caltech101
 - caltech_birds2010
 - caltech_birds2011
 - cars196
 - cassava
 - cats_vs_dogs**
 - chexpert (manual)
 - cifar10
 - cifar100
 - cifar10_1
 - cifar10_corrupted
 - citrus_leaves
 - cmaterdb
 - colorectal_histology
 - colorectal_histology_large
 - curated_breast_imaging_dds (manual)
 - cycle_gan
 - deep_weeds
 - diabetic_retinopathy_detection (manual)
 - dm1ab
 - dtd
 - emnist
 - eurosat
 - fashion_mnist
 - food101
 - gehos_conflict_stimuli
 - horses_or_humans
 - imagenet2012 (manual)
 - imagenet2012_corrupted (manual)
 - imagenet2012_multilabel (manual)
 - imagenet2012_real (manual)
 - imagenet2012_subset (manual)
 - imagenet_a
 - imagenet_f
 - imagenet_resized
 - imagenet_v2
 - imagenette
 - imagewang
 - kin8net
 - lfw
 - malaria
 - mnist
 - mnist_corrupted
 - omniglot
 - oxford_flowers102
 - oxford_iiit_pet
 - patch_camelyon
 - pet_finder
 - places365_small
 - plant_leaves

ML 커뮤니티 데이는 11월 9일입니다! TensorFlow, JAX에서 업데이트를 우리와 함께, 더 자세히 알아보다

translated by Google 이 페이지는 Cloud Translation API를 통해 번역되었습니다. Switch to English

TensorFlow > 리소스 > Datasets > Catalog

고양이 대 개

- 설명: 고양이와 개 이미지의 큰 집합입니다. 1738개의 손상된 이미지가 삭제되었습니다.
- 홈페이지: <https://www.microsoft.com/en-us/download/details.aspx?id=54765>
- 소스 코드: `tfds.image_classification.CatsVsDogs`
- 버전:
 - 4.0.0 (기본값): 새로운 분할 API (<https://tensorflow.org/datasets/splits>)
- 다운로드 크기: 786.68 MiB
- 데이터 집합 크기: Unknown size
- 자동 캐시 (문서): 알 수 없음
- 분할:

나뉘다	예
'train'	23,262

- 특징:

```
FeaturesDict({
  'image': Image(shape=(None, None, 3), dtype=tf.uint8),
  'image/filename': Text(shape=(), dtype=tf.string),
  'label': ClassLabel(shape=(), dtype=tf.int64, num_classes=2),
})
```

- 감독 키 (참조 [as_supervised 문서](#)): ('image', 'label')
- 그림 (`tfds.show_examples`):


6번줄은 데이터의 정보를 확인하는 구문이다. 다음과 같은 화면에 데이터 정보가 표시된다.

〈실행결과〉

Downloading and preparing dataset cats_vs_dogs/4.0.0 (download: 786.68 MiB, ge

DI Completed...: 100%  1/1 [00:12<00:00, 12.56s/ url]

DI Size...: 100%  786/786 [00:12<00:00, 58.94 MiB/s]

 23241/0 [00:13<00:00, 1725.68 examples/s]

WARNING:abs1:1738 images were corrupted and were skipped

Shuffling and writing examples to /root/tensorflow_datasets/cats_vs_dogs/4.0.0

100%  23261/23262 [00:02<00:00, 9772.86 examples/s]

Dataset cats_vs_dogs downloaded and prepared to /root/tensorflow_datasets/cats

```
tfd.core.DatasetInfo(
  name='cats_vs_dogs',
  version=4.0.0,
  description='A large set of images of cats and dogs. There are 1738 corrupt
  homepage='https://www.microsoft.com/en-us/download/details.aspx?id=54765',
  features=FeaturesDict({
    'image': Image(shape=(None, None, 3), dtype=tf.uint8),
```

고양이와 강아지의 데이터를 불러 오는 모습을 화면을 통해서 확인할 수 있다. 총 23000개의 데이터를 호출하게 된다.

3. 데이터 살펴보기

```
7. images=[one['image'].numpy() for one in data_train[0].take(30)]  
8. len(images)
```

<실행결과>

30

실행결과를 확인해 보면 이미지를 30개 정도 가지고 왔음을 알 수 있다. 7번줄은 데이터를 받은 변수로부터 30개의 이미지를 가져 오고 각각의 이미지로부터 numpy형식으로 변환해주는 명령어이다. 8번은 가져온 데이터의 양을 반환해주는 구문이다.

```
9. plt.imshow(images[1])  
10. plt.axis('off')
```

<실행결과>

(-0.5, 335.5, 408.5, -0.5)



9번, 10번줄은 불러온 이미지를 확인하기 위한 구문이다. plt.imshow를 통해서 확인할 수 있는데 1번 이미지는 다음과 같은 출력결과를 보이며, 0부터 숫자를 증가시켜주면 이미지가 변화하는 것을 확인할 수 있다.

4. 데이터 전처리

```
11. resnet50_pre=tf.keras.applications.resnet.ResNet50(weights='imagenet',input
    _shape=(224,224,3))
12. resnet50_pre.summary()
```

<실행결과>

conv5_block1_0_bn (BatchNormali	(None, 7, 7, 2048)	8192	conv5_block1_0_conv[0][0]
conv5_block1_3_bn (BatchNormali	(None, 7, 7, 2048)	8192	conv5_block1_3_conv[0][0]
conv5_block1_add (Add)	(None, 7, 7, 2048)	0	conv5_block1_0_bn[0][0] conv5_block1_3_bn[0][0]
conv5_block1_out (Activation)	(None, 7, 7, 2048)	0	conv5_block1_add[0][0]
conv5_block2_1_conv (Conv2D)	(None, 7, 7, 512)	1049088	conv5_block1_out[0][0]
conv5_block2_1_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block2_1_conv[0][0]
conv5_block2_1_relu (Activation	(None, 7, 7, 512)	0	conv5_block2_1_bn[0][0]
conv5_block2_2_conv (Conv2D)	(None, 7, 7, 512)	2359808	conv5_block2_1_relu[0][0]
conv5_block2_2_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block2_2_conv[0][0]
conv5_block2_2_relu (Activation	(None, 7, 7, 512)	0	conv5_block2_2_bn[0][0]
conv5_block2_3_conv (Conv2D)	(None, 7, 7, 2048)	1050624	conv5_block2_2_relu[0][0]
conv5_block2_3_bn (BatchNormali	(None, 7, 7, 2048)	8192	conv5_block2_3_conv[0][0]
conv5_block2_add (Add)	(None, 7, 7, 2048)	0	conv5_block1_out[0][0] conv5_block2_3_bn[0][0]
conv5_block2_out (Activation)	(None, 7, 7, 2048)	0	conv5_block2_add[0][0]
conv5_block3_1_conv (Conv2D)	(None, 7, 7, 512)	1049088	conv5_block2_out[0][0]
conv5_block3_1_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block3_1_conv[0][0]
conv5_block3_1_relu (Activation	(None, 7, 7, 512)	0	conv5_block3_1_bn[0][0]
conv5_block3_2_conv (Conv2D)	(None, 7, 7, 512)	2359808	conv5_block3_1_relu[0][0]
conv5_block3_2_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block3_2_conv[0][0]
conv5_block3_2_relu (Activation	(None, 7, 7, 512)	0	conv5_block3_2_bn[0][0]
conv5_block3_3_conv (Conv2D)	(None, 7, 7, 2048)	1050624	conv5_block3_2_relu[0][0]
conv5_block3_3_bn (BatchNormali	(None, 7, 7, 2048)	8192	conv5_block3_3_conv[0][0]
conv5_block3_add (Add)	(None, 7, 7, 2048)	0	conv5_block2_out[0][0] conv5_block3_3_bn[0][0]
conv5_block3_out (Activation)	(None, 7, 7, 2048)	0	conv5_block3_add[0][0]
avg_pool (GlobalAveragePooling2	(None, 2048)	0	conv5_block3_out[0][0]
predictions (Dense)	(None, 1000)	2049000	avg_pool[0][0]
=====			
Total params: 25,636,712			
Trainable params: 25,583,592			
Non-trainable params: 53,120			

11번줄은 resnet이라는 인공지능 모델을 가지고 오는 구문이다. resnet50_pre라는 변수명을 가지고 텐서플로의 케라스를 호출하게 되며, weight(가중치)라는 파라미터에 imagenet이라는 120만개의 학습된 이미지를 가져오게 된다. input_shape(입력한 이미지의 사이즈)는 224, 224, 3으로 설정한다. 12번줄은 모델의 정보를 summary()함수를 통해서 확인한다.

5. 모델 정의

```
13. from tensorflow.keras.applications.imagenet_utils import decode_predictions
```

13번줄은 모델을 예측하여 결과를 가져오는 모델을 정의한다.

6. 모델 준비

```
14. def pred_img(img):
15.     plt.imshow(img)
16.     plt.axis('off')
17.     plt.show()

18.     img_resized=cv2.resize(img,(224,224))
19.     pred=resnet50_pre.predict(img_resized.reshape([1,224,224,3]))
20.     decode_pred=decode_predictions(pred)

21. for i, instance in enumerate(decode_pred[0]):
22.     print('{}위: {} ({:.2f}%}'.format(i+1, instance[1], instance[2]*100))
```

14번줄은 예측하는 함수를 선언한다. 15번 ~ 17번줄은 이미지를 보여주는 구문이다. 18번줄은 모델에 들어가는 이미지의 크기로 224, 224로 조절하는 구문이다. 19번줄은 예측하는 변수로 pred를 선언하고 좀전에 선언했던 resnet50_pre.predict를 가지고 배치로써 이미지의 크기를 1, 224, 224, 3으로 변환하게 된다. 20번줄은 예측된 결과를 어떤 대상인지 확인하여 바꾸어 주는 구문이다.

21번줄은 결과들을 1위부터 5위까지 보여주기 위한 반복문이다.

7. 모델 예측

```
23. pred_img(images[15])
```

<실행결과>



1위: tabby (45.78%)
2위: tiger_cat (30.16%)
3위: Egyptian_cat (9.22%)
4위: lynx (4.09%)
5위: Siamese_cat (3.26%)

15번 데이터를 통해서 예측되는 순위가 1위부터 5위까지 확률로 보여주는 것을 확인할 수 있다. 고양이의 종류인 tabby가 45.78%로 정확하게 예측되고 있다. 마찬가지로 []안의 숫자를 바꾸어 가면서 다른 모델들로 잘 예측하는지 확인해 볼 이수 있다.

```
24. pred_img(images[1])
```

<실행결과>



1위: Chihuahua (90.09%)
2위: miniature_pinscher (4.65%)
3위: French_bulldog (2.58%)
4위: Boston_bull (0.80%)
5위: boxer (0.45%)

이미지의 1번 데이터를 예측해보면 1위가 chihuahua가 90.09%로 정확하게 예측되고 있다. 마찬가지로 []안의 숫자를 바꾸어 가면서 다른 모델들로 잘 예측하는지 확인해 볼 수 있다.

6.4. 텐서플로를 활용한 마스크 착용 여부 확인하기

◦ 이미지를 활용한 마스크 착용 여부 확인하기

1. 도구준비

```
1. from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
2. from tensorflow.keras.models import load_model

3. import numpy as np
4. import cv2

5. import matplotlib.pyplot as plt
6. import os

7. plt.style.use('dark_background')
```

1번줄은 tensorflow를 코랩에서 불러오는 명령어이며 keras 모델을 사용하게 된다.

3번줄은 numpy를 사용하면서 앞으로 계속 np로 호출하게 된다. 4번줄은 영상 처리를 위한 cv2 라이브러리를 불러오며, 5번줄은 matplotlib.pyplot를 사용하게 되며 plt로 앞으로 호출하게 된다.

2. 모델 불러오기

```
8. facenet = cv2.dnn.readNet('models/deploy.prototxt',
    'models/res10_300x300_ssd_iter_140000.caffemodel')
9. model = load_model('models/mask_detector.model')
```

8번, 9번줄은 두 개의 모델을 불러오는 구문으로 8번줄은 얼굴을 인식하는 모델을 불러오는 구문이다. 모델 파일은 models/deploy.prototxt안에 삽입되어 있으며 opencv의 dnn모델을 호출하여 사용할 것이다. 9번줄은 마스크 모델로 케라스 모델을 가지고 사용하게 된다.

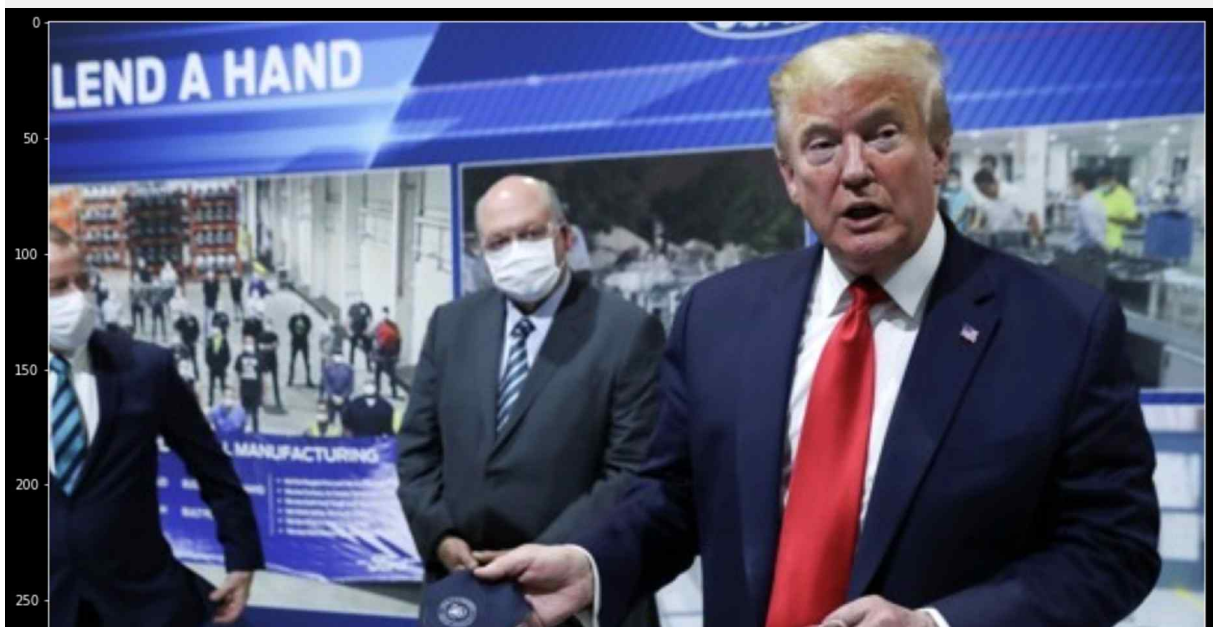
3. 이미지 가져오기

```
10. img = cv2.imread('imgs/02.jpg')
11. h, w = img.shape[:2]
```



```
12. plt.figure(figsize=(16, 10))
13. plt.imshow(img[:, :, ::-1])
```

<실행결과>



사용할 테스트 이미지를 가지고 오는 구문으로 10번줄은 opencv2의 imread를 사용하여 테스트 이미지인 02.jpg를 가지고 온다. 11번줄은 이미지의 형태를 h, w에 각각 저장해 준다. h는 세로(높이)이며 w는 가로(넓이)를 나타낸다. 13번줄은 matplotlib를 통해서 이미지가 잘 저장되어 있는지 확인하는 구문으로 img를 변환하게 되는데 opencv로 읽은 이미지가 BGR로 읽여지기 때문에 RGB로 바꾸어 주는 구문이다.

4. 이미지 중에서 얼굴 부분만 자르기 위한 전처리

```
14. blob = cv2.dnn.blobFromImage(img, scalefactor=1., size=(300, 300),
    mean=(104., 177., 123.))

15. facenet.setInput(blob)
16. dets = facenet.forward()
```

14번줄은 opencv에서 dnn알고리즘을 활용하기 형태로 변환하기 위하여 파라미터값을 scalefactor=1, size는 300, 300 mean값은 104, 177, 123으로 넣어 준다. 15번줄은 모델에 데이터를 삽입하는 부분이다. 16번줄은 결과를 추론하고 결과가 dets에 저장된다.

5. 얼굴만 자르기

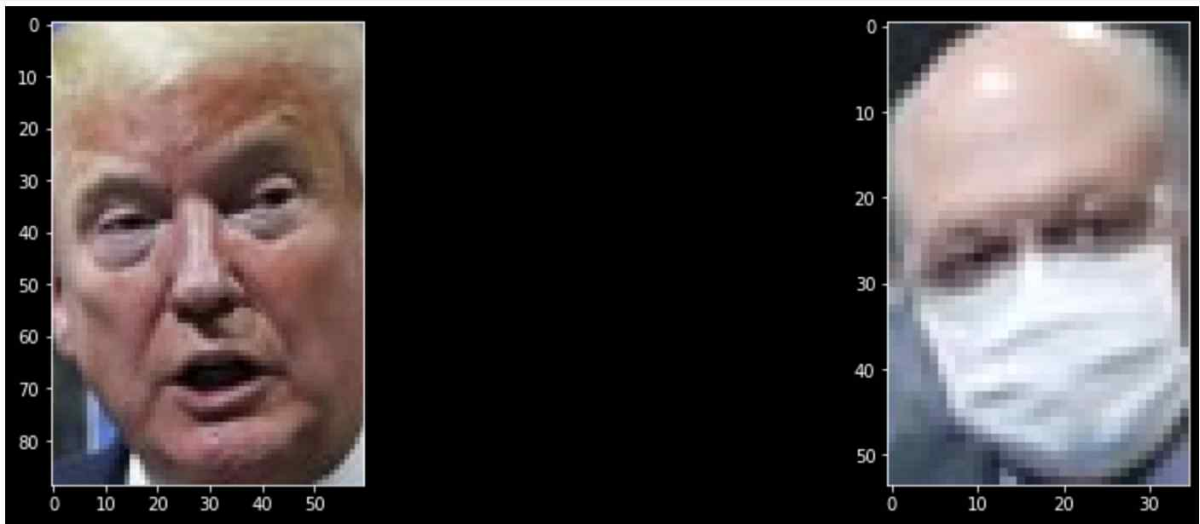
```
17. faces = []

18. for i in range(dets.shape[2]):
19.     confidence = dets[0, 0, i, 2]
20.     if confidence < 0.5:
21.         continue

22.     x1 = int(dets[0, 0, i, 3] * w)
23.     y1 = int(dets[0, 0, i, 4] * h)
24.     x2 = int(dets[0, 0, i, 5] * w)
25.     y2 = int(dets[0, 0, i, 6] * h)

26.     face = img[y1:y2, x1:x2]
27.     faces.append(face)
28.
29. plt.figure(figsize=(16, 5))
30.
31. for i, face in enumerate(faces):
32.     plt.subplot(1, len(faces), i+1)
33.     plt.imshow(face[:, :, :-1])
```

<실행결과>



17번줄은 faces라는 이름으로 빈 리스트를 생성한다. 18번줄의 dets에 저장된 것을 for문을 돌리면서 여러개의 얼굴을 확인한다. 확인한 결과를 19번줄에서 확인하고 20번줄에서는 결과값이 0.5이하이면 continue 문을 통해서 통과하게 된다.

22번 ~ 25번줄에서는 x와 y의 범위 값을 가져온다. 그 이유는 사진중에서 얼굴을 자르기 위함이며, 26번 ~ 27번줄에서 잘라낸 얼굴의 이미지를 face에 저장하게 된다.

29번줄에서는 저장된 이미지의 사이즈가 16, 5로 설정하며, 31번줄에서는 저장된 이미지의 값이 잘 저장되었는지 확인하게 된다.

5. 얼굴 이미지에서 마스크 추출하기

```
34. plt.figure(figsize=(16, 5))

35. for i, face in enumerate(faces):
36.     face_input = cv2.resize(face, dsize=(224, 224))
37.     face_input = cv2.cvtColor(face_input, cv2.COLOR_BGR2RGB)
38.     face_input = preprocess_input(face_input)
39.     face_input = np.expand_dims(face_input, axis=0)

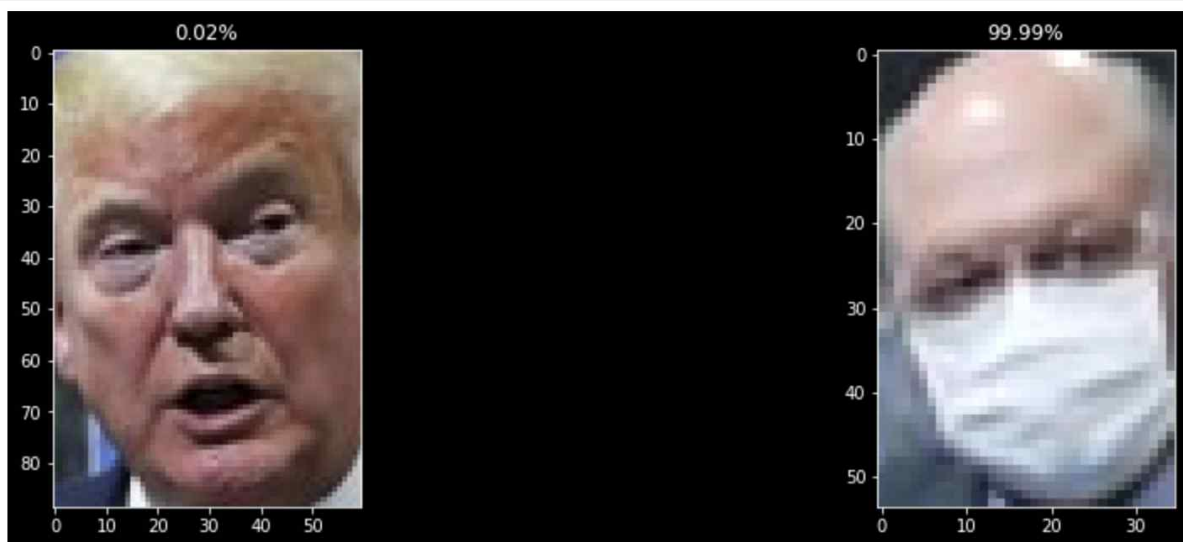
40.     mask, nomask = model.predict(face_input).squeeze()

41.     plt.subplot(1, len(faces), i+1)
42.     plt.imshow(face[:, :, ::-1])
43.     plt.title('%0.2f%%' % (mask * 100))
```

35번줄은 faces 개수를 매개변수를 받아서 그 개수만큼 반복하게 된다. 35번줄 ~ 39번줄은 데이터 즉 이미지를 전처리하는 부분으로 36번줄의 이미지 사이즈 224, 224로 바꾸어 준다. 37번줄은 cvtColor 매소드를 통해서 BGR을 RGB로 바꾸어 준다. 39번줄은 우리가 일반적인 데이터셋은 (224, 224, 3)인 데이터 셋을 차원을 하나 더 추가하여 (1, 224, 224, 3)으로 바꾸어 준다.

40번줄은 마스크를 쓸 확률과 마스크를 쓰지 않을 확률을 예측하는 함수를 생성하여 넣어준다.

<실행결과>



실행결과를 통해 첫 번째 사진의 마스크를 쓰지 않을 확률은 0.02%로 나오고 있으며 두 번째 사진의 실행결과는 99.99%로 나오는 것을 확인할 수 있다.

6.5. 머신러닝을 이용한 예측

◦ 당뇨병 예측해 보기

1. pandas 라이브러리 호출

```
1. import pandas as pd
2. import matplotlib.pyplot as plt
3. import seaborn as sns

4. df = pd.read_csv('diabetes.csv', names = ["pregnant", "plasma", "pressure",
        "thickness", "insulin", "BMI", "pedigree", "age", "class"])
```

1번줄은 파이썬의 pandas를 pd로 호출하게 된다. 2번줄은 matplotlib를 통해서 그래프로 표현하게 된다. 3번줄은 seaborn은 상관관계를 나타내주는 라이브러리이다. 4번줄은 데이터셋의 위치를 말하며 확장자를 csv로 갖는 파일이다.

2. 데이터 확인

```
5. print(df.head(5))
6. print(df.info())

7. print(df.describe())
8. print(df[['plasma', 'class']])
```

5번줄은 처음 다섯줄의 실행결과를 확인해보고자 입력하였다. 실행결과를 다음과 같다.

<실행결과>

	pregnant	plasma	pressure	thickness	insulin	BMI	pedigree	age	class
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

5번줄은 데이터의 전반적인 정보를 확인해 보고자 입력한 명령어이다. 총 768개의 데이터셋을 가지고 있으며, 열(W)에는 pregnant, plasma, pressure, thickness, insulin, BMI, pedigree, age, class 로 구성되어 있다. 실행결과와 다음과 같다.

<실행결과>

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   pregnant        768 non-null    int64
1   plasma          768 non-null    int64
2   pressure        768 non-null    int64
3   thickness       768 non-null    int64
4   insulin         768 non-null    int64
5   BMI             768 non-null    float64
6   pedigree        768 non-null    float64
7   age             768 non-null    int64
8   class           768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
None
```

7번줄은 데이터의 전반적인 정보를 6번줄보다 더 자세히 확인해 보고자 입력한 명령어이다. 총 768개의 데이터셋을 가지고 있으며, 평균값, 표준편차, 최소값, 최대값의 순서로 8 X 9 행렬에 표현해 준다.

<실행결과>

	pregnant	plasma	pressure	...	pedigree	age	class
count	768.000000	768.000000	768.000000	...	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	...	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	...	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	...	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	...	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	...	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	...	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	...	2.420000	81.000000	1.000000

[8 rows x 9 columns]

8번줄은 데이터의 전반적인 정보 중 임신 정보와 클래스만을 출력하는 명령어이다.

<실행결과>

	plasma	class
0	148	1
1	85	0
2	183	1
3	89	0
4	137	1
..
763	101	0
764	122	0
765	121	0
766	126	1
767	93	0

[768 rows x 2 columns]

3. 상관관계 그래프 확인

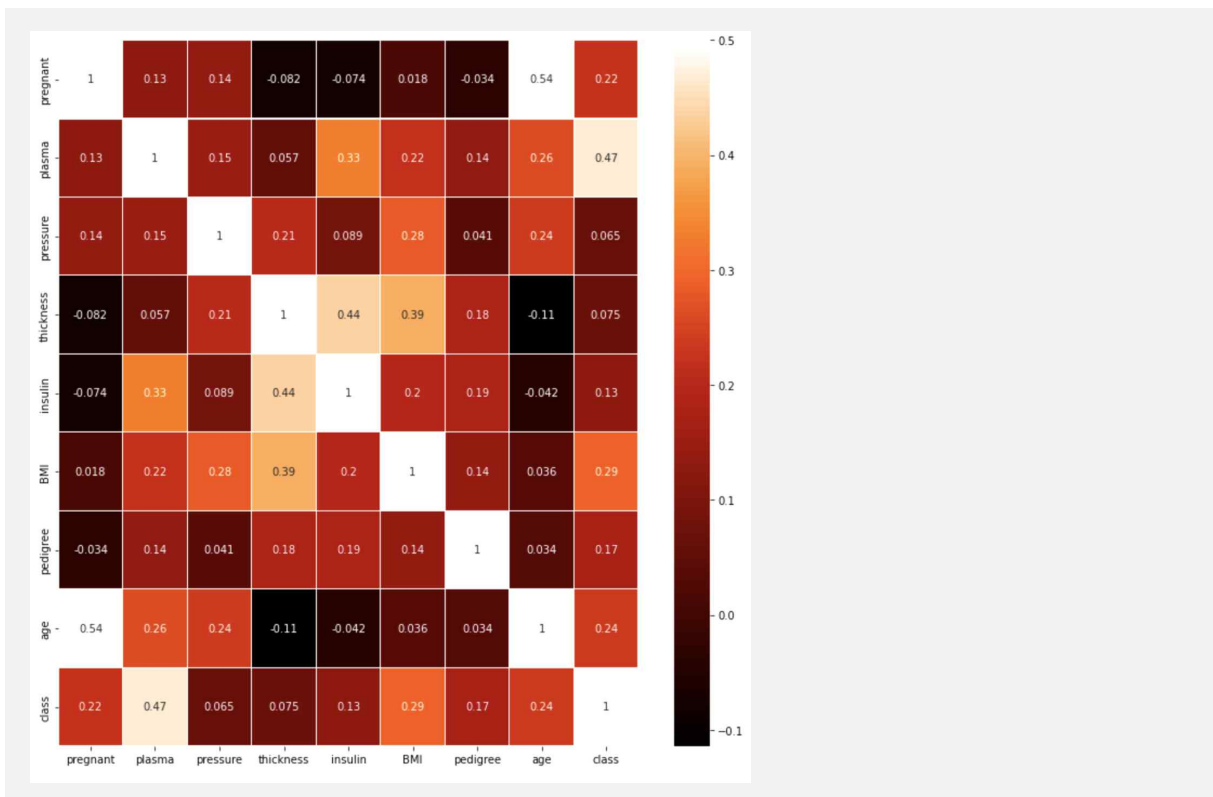
```
9. colormap = plt.cm.gist_heat
10. plt.figure(figsize=(12,12))

11. sns.heatmap(df.corr(),linewidths=0.1,vmax=0.5, cmap=colormap,
    linecolor='white', annot=True)
12. plt.show()
```

9번 ~ 10번줄은 데이터간의 상관관계를 그래프로 그리는 명령어로 그래프의 색상을 먼저 정한 후 그래프의 크기를 12 X 12로 출력한다.

11번줄은 그래프의 속성을 결정하는 구문으로 vmax값을 0.5로 지정한 후 0.5에 가까울수록 밝은 색으로 표현한다.

<실행결과>



12 X 12 그래프의 가로와 세로의 항목들이 서로 상관관계에 따라서 색깔로 표시되고 있다. 상관관계가 높을수록 밝은 색을 나타내기 때문에 plasma와 class항목이 가장 상관관계가 높다는 것을 알 수 있다.

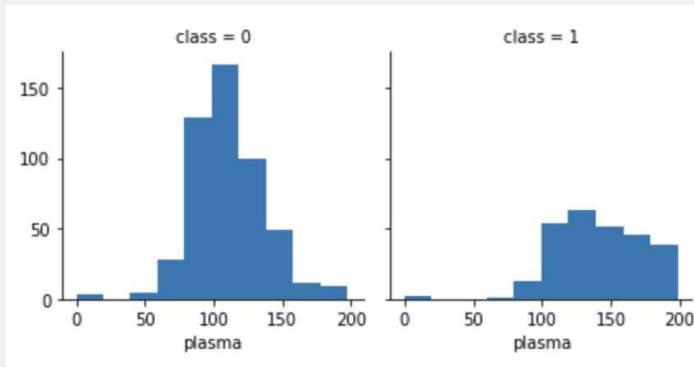
4. 그래프 속성 입력

```
13. grid = sns.FacetGrid(df, col='class')
14. grid.map(plt.hist, 'plasma', bins=10)

15. plt.show()
```


13번 class를 열로 나열하게 된다. class=1로 데이터 전처리 과정을 통해서 150이상의 값을 갖는 plasma항목의 수치가 많다는 것을 알 수 있다.

<실행결과>



5. 딥러닝 구현 환경 설정

```
16. from tensorflow.keras.models import Sequential
17. from tensorflow.keras.layers import Dense

18. import numpy
19. import tensorflow as tf

20. numpy.random.seed(3)
21. tf.random.set_seed(3)

22. dataset = numpy.loadtxt("diabetes.csv", delimiter=",")
23. X = dataset[:,0:8]
24. Y = dataset[:,8]

25. model = Sequential()
26. model.add(Dense(12, input_dim=8, activation='relu'))
27. model.add(Dense(8, activation='relu'))
28. model.add(Dense(1, activation='sigmoid'))
29. model.compile(loss='binary_crossentropy', optimizer='adam',
    metrics=['accuracy'])

30. model.fit(X, Y, epochs=200, batch_size=10)
31. print("\n Accuracy: %.4f" % (model.evaluate(X, Y)[1]))
```

16번 ~ 17번줄은 딥러닝을 구동하는데 필요한 케라스 함수를 호출한다.

18번줄은 numpy라이브러리와 tensorflow를 불러온다.

19번줄은 실행할 때마다 같은 결과 형태를 출력하기 위해서 설정하게 된다. random()함수를 통해서 컴퓨터 안의 미리 내장되어 있는 랜덤 테이블 중에서 하나를 호출해서 그 표 순서대로 숫자를 보여 주게 된다. seed는 랜덤 테이블 중에서 몇 번째 테이블을 호출할지를 결정하기 위함이다.

22번, 23번, 24번줄은 데이터셋을 불러오는 구문으로 1번부터 8번까지의 항목을 속성으로 정하고, 9번 항목으로 클래스라고 정하고 특별하게 데이터를 관리하게 된다.

25번줄은 모델을 설정하는 부분으로 설정한 딥러닝 모델에 은닉층을 추가하여 한 층씩 추가하는 구문이다. 추가할 때마다 model.add()함수를 활용하며 relu함수와 sigmoid함수를 활용하여 추가한다.

29번줄은 이항 분류를 하기 위하여 오차 함수를 binary_crossentropy사용하고 adam을 최적화 함수로 사용한다.

30번줄은 학습이 진행되며 총 200번동안 학습을 진행하게 된다. 학습이 진행되면서 정확도가 약 77%까지 올라가는 것을 확인할 수 있다.

<실행결과>

```
77/77 [=====] - 0s 1ms/step - loss: 0.4782 - accuracy: 0.7643
Epoch 188/200
77/77 [=====] - 0s 2ms/step - loss: 0.4758 - accuracy: 0.7721
Epoch 189/200
77/77 [=====] - 0s 1ms/step - loss: 0.4657 - accuracy: 0.7760
Epoch 190/200
77/77 [=====] - 0s 1ms/step - loss: 0.4687 - accuracy: 0.7760
Epoch 191/200
77/77 [=====] - 0s 1ms/step - loss: 0.4675 - accuracy: 0.7669
Epoch 192/200
77/77 [=====] - 0s 1ms/step - loss: 0.4675 - accuracy: 0.7565
Epoch 193/200
77/77 [=====] - 0s 1ms/step - loss: 0.4743 - accuracy: 0.7799
Epoch 194/200
77/77 [=====] - 0s 1ms/step - loss: 0.4765 - accuracy: 0.7695
Epoch 195/200
77/77 [=====] - 0s 2ms/step - loss: 0.4664 - accuracy: 0.7682
Epoch 196/200
77/77 [=====] - 0s 2ms/step - loss: 0.4741 - accuracy: 0.7669
Epoch 197/200
77/77 [=====] - 0s 1ms/step - loss: 0.4734 - accuracy: 0.7747
Epoch 198/200
77/77 [=====] - 0s 1ms/step - loss: 0.4668 - accuracy: 0.7578
Epoch 199/200
77/77 [=====] - 0s 1ms/step - loss: 0.4679 - accuracy: 0.7760
Epoch 200/200
77/77 [=====] - 0s 1ms/step - loss: 0.4662 - accuracy: 0.7721
24/24 [=====] - 0s 1ms/step - loss: 0.4587 - accuracy: 0.7708

Accuracy: 0.7708
```

◦ 꽃의 품종 예측해 보기

1. 딥러닝 환경 설정을 위한 기본 라이브러리 호출

```
1. from tensorflow.keras.models import Sequential
2. from tensorflow.keras.layers import Dense
3. from sklearn.preprocessing import LabelEncoder

4. import pandas as pd
5. import seaborn as sns
6. import matplotlib.pyplot as plt
7. import numpy as np
8. import tensorflow as tf
```

1번줄은 텐서플로의 케라스 함수를 호출하게 된다. 케라스 라이브러리 중 Sequential() 함수와 Dense() 함수를 호출한다. Sequential() 함수는 딥러닝의 구조를 한층 한층 쉽게 쌓아 올릴 수 있다. Sequential() 함수를 선언하고 model.add() 함수를 사용해서 필요한 층을 차례로 추가하게 된다.

3번줄은 수치데이터를 변환하기 위한 라이브러리 호출이다.

4번 ~ 8번줄은 pandas를 호출하며 앞으로 pd로 줄여서 구현하게 된다. 마찬가지로 seaborn 데이터 시각화를 위한 라이브러리를 호출하며 마찬가지로 matplotlib 라이브러리 역시 그래프를 표현하기 위한 라이브러리를 호출하게 된다.

2. 출력 형태 설정

```
9. np.random.seed(3)
10. tf.random.set_seed(3)
```

9번줄은 실행할 때마다 같은 결과 형태를 출력하기 위해서 설정하게 된다. random()함수를 통해서 컴퓨터 안의 미리 내장되어 있는 랜덤 테이블 중에서 하나를 호출해서 그 표 순서대로 숫자를 보여 주게 된다. seed는 랜덤 테이블 중에서 몇 번째 테이블을 호출할지를 결정하기 위해서이다.

3. 데이터 입력

```
11. df = pd.read_csv('iris.csv', names = ["sepal_length", "sepal_width",  
      "petal_length", "petal_width", "species"])
```

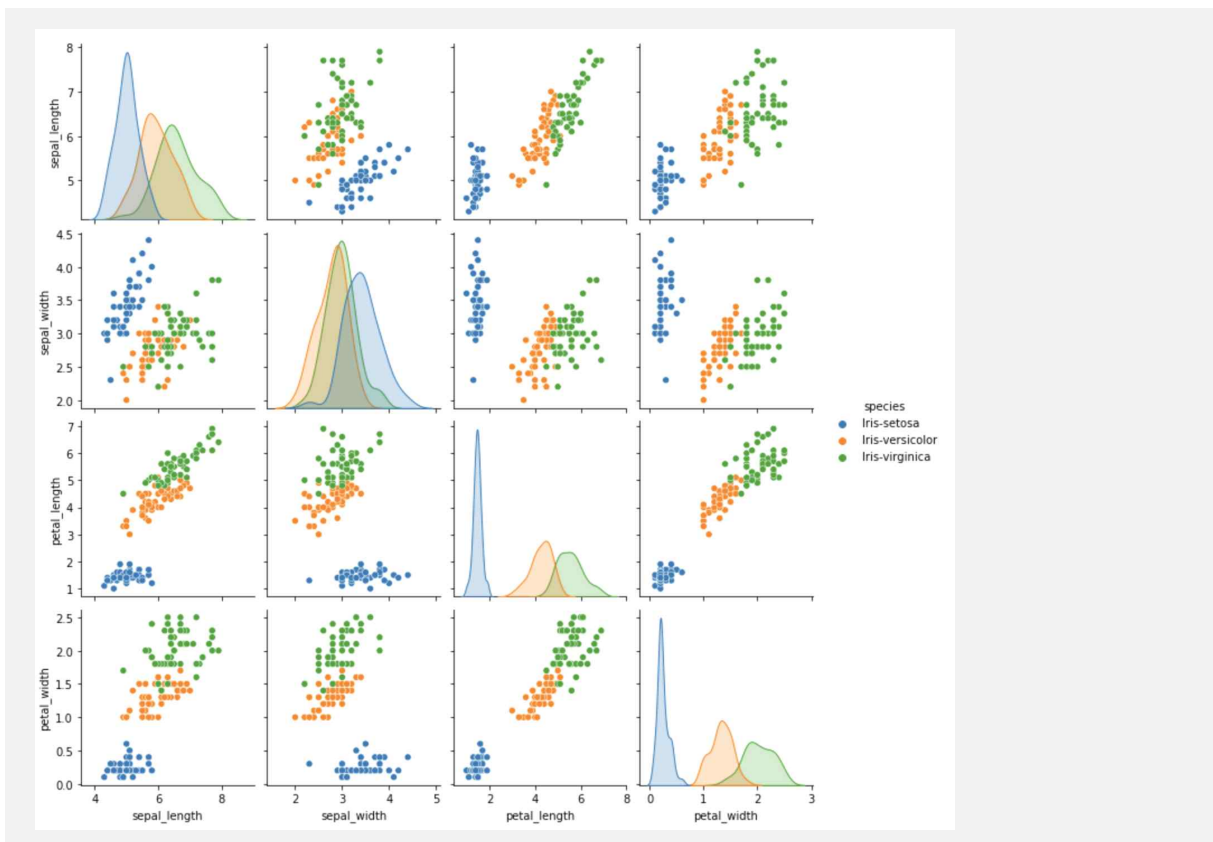
11번줄은 데이터셋의 위치를 말하며 확장자를 csv로 갖는 파일이다.

4. 입력 데이터 그래프로 확인

```
12. sns.pairplot(df, hue='species');  
13. plt.show()
```

12번줄은 데이터의 입력값들을 그래프로 시각화 하여 확인하는 부분이다. pairplot() 함수를 활용하여 데이터 전체를 한번에 보는 그래프를 다음과 같이 출력하게 된다.

<실행결과>



꽃잎과 꽃받침의 크기와 너비에 따라 차이가 있음을 확인 할 수 있다. 속성별로 어떤 연관이 있는지 상관도를 보여주는 그래프이다.

5. 데이터 분류

```
14. dataset = df.values
15. X = dataset[:,0:4].astype(float)
16. Y_obj = dataset[:,4]
```

14번줄은 입력 파일의 데이터 값을 불러와 데이터셋에 저장한 후 1번부터 4번까지의 속성을 4개의 정보로 분류하고 5번 값을 클래스로 분류하게 된다. 데이터 셋 안에 문자열이 포함되어 있기 때문에 pandas로 데이터를 불러와 X와 Y값으로 구분한다.

6. 문자열 숫자로 변환

```
17. e = LabelEncoder()
18. e.fit(Y_obj)
19. Y = e.transform(Y_obj)
20. Y_encoded = tf.keras.utils.to_categorical(Y)
```

17번줄은 Y값이 문자열이기 때문에 LabelEncoder() 함수를 통해서 문자열을 숫자로 바꾸어준다. 이렇게 하면 array([1, 2, 3])로 바뀐다. 또한 활성화함수를 적용하려면 Y값이 숫자 0과 1로 이루어져 있어야 한다.

7. 모델의 설정

```
21. model = Sequential()
22. model.add(Dense(16, input_dim=4, activation='relu'))
23. model.add(Dense(3, activation='softmax'))
```

21번 ~ 22번줄은 최종 출력값이 3개 중 하나로 나와야 하기 때문에 Dense의 노트수를 3으로 설정한다. 활성화함수로 softmax를 사용한다. softmax는 가중치의 합이 항상 1이 나오도록 해주는 함수이다.

8. 모델 컴파일

```
24. model.compile(loss='categorical_crossentropy', optimizer='adam',  
    metrics=['accuracy'])
```

24번줄은 카테고리별로 분류를 하기 위하여 오차 함수를 binary_crossentropy 사용하고 adam을 최적화 함수로 사용한다.

9. 모델 실행 및 결과 출력

```
25. model.fit(X, Y_encoded, epochs=50, batch_size=1)  
26. print("\n Accuracy: %.4f" % (model.evaluate(X, Y_encoded)[1]))
```

25번줄은 모델을 학습하는 부분으로 총 50번을 학습하게 된다. 26번줄은 정확도를 출력하는 구문이다.

<실행결과>

```
Epoch 32/50
150/150 [=====] - 0s 1ms/sample - loss: 0.1179 - accuracy: 0.9667
Epoch 33/50
150/150 [=====] - 0s 1ms/sample - loss: 0.1203 - accuracy: 0.9667
Epoch 34/50
150/150 [=====] - 0s 1ms/sample - loss: 0.1137 - accuracy: 0.9800
Epoch 35/50
150/150 [=====] - 0s 986us/sample - loss: 0.1161 - accuracy: 0.9733
Epoch 36/50
150/150 [=====] - 0s 1ms/sample - loss: 0.1119 - accuracy: 0.9667
Epoch 37/50
150/150 [=====] - 0s 1ms/sample - loss: 0.1069 - accuracy: 0.9867
Epoch 38/50
150/150 [=====] - 0s 1ms/sample - loss: 0.1097 - accuracy: 0.9733
Epoch 39/50
150/150 [=====] - 0s 1ms/sample - loss: 0.1072 - accuracy: 0.9667
Epoch 40/50
150/150 [=====] - 0s 1ms/sample - loss: 0.0998 - accuracy: 0.9867
Epoch 41/50
150/150 [=====] - 0s 1ms/sample - loss: 0.1051 - accuracy: 0.9667
Epoch 42/50
150/150 [=====] - 0s 1ms/sample - loss: 0.1005 - accuracy: 0.9667
Epoch 43/50
150/150 [=====] - 0s 1ms/sample - loss: 0.0973 - accuracy: 0.9867
Epoch 44/50
150/150 [=====] - 0s 1ms/sample - loss: 0.0979 - accuracy: 0.9800
Epoch 45/50
150/150 [=====] - 0s 1ms/sample - loss: 0.1090 - accuracy: 0.9667
Epoch 46/50
150/150 [=====] - 0s 1ms/sample - loss: 0.0962 - accuracy: 0.9667
Epoch 47/50
150/150 [=====] - 0s 1ms/sample - loss: 0.0926 - accuracy: 0.9867
Epoch 48/50
150/150 [=====] - 0s 993us/sample - loss: 0.0924 - accuracy: 0.9733
Epoch 49/50
150/150 [=====] - 0s 1ms/sample - loss: 0.0884 - accuracy: 0.9667
Epoch 50/50
150/150 [=====] - 0s 999us/sample - loss: 0.0894 - accuracy: 0.9733
150/1 [=====]

Accuracy: 0.9867
```

6.6. 머신러닝을 이용한 회귀

◦ 선형회귀를 통한 예측점수 계산하기

1. numpy 호출

```
1. import numpy as np
```

1번줄은 선형회귀를 실행하기 위한 Numpy 라이브러리를 호출한다.

2. 임의의 기울기와 절편입력

```
2. fake_a_b=[3,76]
```

2번줄은 임의의 기울기 3과 절편 76을 갖는 일차방정식 곡선을 만들어 준다.

3. X값과 Y값 입력

```
3. data = [[3, 88], [6, 95], [9, 96], [12, 97]]  
4. x = [i[0] for i in data]  
5. y = [i[1] for i in data]
```

3번줄은 임의의 데이터 X를 3, 6, 9, 12로 넣어주고, Y는 88, 95, 96, 97로 설정한다. data 변수에 X, Y를 쌍으로 입력을 한다. 4번, 5번줄에서는 쌍으로 입력받은 정수를 리스트에 첫 번째 값은 x, 두 번째 값은 y에 저장한다.

4. 결과 출력하는 함수 생성

```
6. def predict(x):  
7.     return fake_a_b[0]*x + fake_a_b[1]
```

6번, 7번줄은 predict() 함수를 만들고 $ax+b$ 라는 일차방정식을 만들어 준다.

5. 평균 제곱 오차 함수 생성

```
8. def mse(y, y_hat):  
9.     return ((y - y_hat) ** 2).mean()
```

8번, 9번줄은 예측값과 실제값을 y, y_hat에 입력하여 평균 제곱을 구한다.

6. 최종 값 반환 함수

```
10. def mse_val(y, predict_result):  
11.     return mse(np.array(y), np.array(predict_result))
```

10번, 11번줄은 y값과 prddict_result() 값을 매개변수로 입력하여 예측값과 실제값으로 mse() 함수에 입력하게 된다.

7. 빈 리스트 생성

```
12. predict_result = []  
  
13. for i in range(len(x)):  
14.     predict_result.append(predict(x[i]))  
15.     print("공부시간=%.f, 실제점수=%.f, 예측점수=%.f" % (x[i], y[i],  
        predict(x[i])))  
16.     print("MSE 최종값: " + str(mse_val(predict_result,y)))
```

12번, 결과값을 넣어줄 빈 리스트를 생성한다. 13번줄은 x의 길이만큼 반복하는 반복문을 생성한다. 14번줄은 결과에 해당되는 predict_result 리스트를 생성한다. 15번, 16번줄은 예측값을 출력하는 구문이다.

<실행결과>

```
공부시간=3, 실제점수=88, 예측점수=85  
공부시간=6, 실제점수=95, 예측점수=94  
공부시간=9, 실제점수=96, 예측점수=103  
공부시간=12, 실제점수=97, 예측점수=112  
MSE 최종값: 71.0
```

◦ 지역의 집값 예측하기

1. numpy 호출

```
1. from keras.models import Sequential
2. from keras.layers import Dense
3. from sklearn.model_selection import train_test_split

4. import numpy
5. import pandas as pd
6. import tensorflow as tf
```

1번 ~ 2번줄은 선형회귀를 실행하기 위한 Keras 라이브러리와 Numpy 라이브러리를 호출한다. 데이터 처리를 위한 pandas 라이브러리를 호출한다.

2. seed 값 설정

```
7. seed = 0
8. numpy.random.seed(seed)
9. tf.random.set_seed(3)

10. df = pd.read_csv("housing.csv", delim_whitespace=True, header=None)
```

7번줄은 seed 값을 설정하는 부분으로 초기화를 0으로 설정한다. seed는 랜덤 테이블 중에서 몇 번째 테이블을 호출할지를 결정하기 위함이다. 10번줄은 데이터셋을 업로드 한다.

3. X값과 Y값 입력

```
11. dataset = df.values

12. X = dataset[:,0:13]
13. Y = dataset[:,13]
```

11번줄은 임의의 데이터셋을 생성한다. dataset 변수에 X, Y를 쌍으로 입력을 한다. 컬럼수가 총 14개로 13개의 속성으로 이루어 1개의 클래스로 구성되어 있다.

4. 학습 데이터 전처리

```
14. X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,
    random_state=seed)
```

14번줄은 데이터를 전처리하는 과정으로 학습 사이즈는 0.3으로 정하고 seed() 함수를 랜덤으로 호출한다.

5. 선형회귀 실행

```
15. model = Sequential()
16. model.add(Dense(30, input_dim=13, activation='relu'))
17. model.add(Dense(6, activation='relu'))
18. model.add(Dense(1))
```

15번 ~ 18번줄은 선형회귀를 실행하기 위한 부분이다. relu함수를 사용하고 케라스의 model.add()를 활용하여 필요한 만큼 층을 빠르고 쉽게 쌓을 수 있다. model.add() 함수안에는 Dense() 함수가 포함되어 있다. 몇 개 층을 쌓을지는 데이터의 양에 따라서 결정된다.

6. 예측값과 실제값 비교

```
19. model.compile(loss='mean_squared_error', optimizer='adam')
20. model.fit(X_train, Y_train, epochs=200, batch_size=10)

21. Y_prediction = model.predict(X_test).flatten()

22. for i in range(10):
23.     label = Y_test[i]
24.     prediction = Y_prediction[i]
25.     print("실제가격: {:.3f}, 예상가격: {:.3f}".format(label, prediction))
```

19번줄은 모델을 실행하는 부분으로 최적화 함수로 adam()을 사용한다. 21번줄의 flatten() 함수는 데이터 배열이 몇 차원이든 모두 1차원으로 변경하기 쉽게 해주는 함수이다. range('숫자')는 0부터 계속해서 증가시켜 주는 구문이다. 총 10번을 반복하게 된다. 출력 결과는 다음과 같다.

<실행 결과>

```
실제가격: 22.600, 예상가격: 20.133
실제가격: 50.000, 예상가격: 24.157
실제가격: 23.000, 예상가격: 28.158
실제가격: 8.300, 예상가격: 13.419
실제가격: 21.200, 예상가격: 22.280
실제가격: 19.900, 예상가격: 23.254
실제가격: 20.600, 예상가격: 20.012
실제가격: 18.700, 예상가격: 26.365
실제가격: 16.100, 예상가격: 18.521
실제가격: 18.600, 예상가격: 11.163
```

실제 가격과 예상가격이 비례하여 변화하는 것을 확인할 수 있다.

6.7. 딥러닝을 이용한 자연어처리

◦ 주어진 단어로 토큰화 하기

1. 딥러닝을 위한 기본 라이브러리 호출

```
1. import numpy
2. import tensorflow as tf

3. from numpy import array
4. from tensorflow.keras.preprocessing.text import Tokenizer
5. from tensorflow.keras.preprocessing.sequence import pad_sequences
6. from tensorflow.keras.models import Sequential
7. from tensorflow.keras.layers import Dense, Flatten, Embedding
```

1번 ~ 2번줄은 딥러닝을 구현하기 위한 라이브러리를 호출한다. 4번줄은 numpy와 tensorflow를 호출하고, 케라스의 Tokenizer() 함수를 사용하여 단어의 빈도수를 쉽게 계산할 수 있다. 텍스트 전처리에도 유용하게 사용되는 함수이다.

2. 전처리 함수 호출

```
8. from tensorflow.keras.preprocessing.text import text_to_word_sequence
9. text = '건강한 마음이 건강한 육체를 만든다 '
```

8번줄은 텍스트 전처리를 위한 함수 중 `text_to_word_sequence` 함수를 호출한다. `text_to_word_sequence` 함수를 사용하면 문장을 단어 단위로 쉽게 나눌 수 있다. 해당 함수를 사용하여 전처리를 하고자 하는 텍스트를 토큰화 한다.

9번줄은 토큰화한 텍스트를 저장하게 된다. 주어진 문장은 ‘건강한 마음이 건강한 육체를 만든다’ 이다.

3. 텍스트 토큰화

```
10. result = text_to_word_sequence(text)

11. print("\n원문:\n", text)
12. print("\n토큰화:\n", result)
```

10번줄은 `text` 변수에 입력되어 있는 값을 `result` 변수에 입력한다. 11번, 12번줄 아래의 실행결과와 같이 원문과 토큰화한 문장을 반환해 준다.

<실행결과>

```
원문:
건강한 마음이 건강한 육체를 만든다

토큰화:
['건강한', '마음이', '건강한', '육체를', '만든다']
```

4. 텍스트 토큰화

```
13. docs = ['먼저 텍스트의 각 단어를 나누어 토큰화 합니다.', '텍스트의 단어로 토큰화 해야 딥러닝에서 인식됩니다.', '토큰화 한 결과는 딥러닝에서 사용 할 수 있습니다.',
           ]

14. token = Tokenizer()
15. token.fit_on_texts(docs)

16. print("\n단어 카운트:\n", token.word_counts)
17. print("\n문장 카운트: ", token.document_count)

18. print("\n각 단어가 몇개의 문장에 포함되어 있는가:\n", token.word_docs)
19. print("\n각 단어에 매겨진 인덱스 값:\n", token.word_index)
```

13번줄은 전처리 하고자 하는 3개의 문장을 정하게 된다. 이후 14번줄에서 토큰화 함수를 통해서 전처리 하는 과정을 거치게 된다. 15번줄에서 토큰화 함수에 문장을 적용하게 된다.

<실행결과>

단어 카운트:

```
OrderedDict([('먼저', 1), ('텍스트의', 2), ('각', 1), ('단어를', 1), ('나누어', 1), ('토
```

문장 카운트: 3

각 단어가 몇개의 문장에 포함되어 있는가:

```
defaultdict(<class 'int'>, {'나누어': 1, '토큰화': 3, '합니다': 1, '먼저': 1, '텍스트의'
```

각 단어에 매겨진 인덱스 값:

```
{'토큰화': 1, '텍스트의': 2, '답러님에서': 3, '먼저': 4, '각': 5, '단어를': 6, '나누어': 7,
```

출력결과를 살펴보면 3문장에서 처리되는 출현빈도가 높은 것부터 나열해서 토큰화 3회, 답러님에서 2회, 텍스트의 2회로 결과를 출력하게 된다.

5. 텍스트 리뷰 자료 지정

```
20. docs = ["너무 재밌네요","최고예요","참 잘 만든 영화예요","추천하고 싶은 영화  
입니다","한번 더 보고싶네요","글쎄요","별로예요","생각보다 지루하네요","연기  
가 어색해요","재미없어요"]
```

```
21. classes = array([1,1,1,1,1,0,0,0,0,0])
```

20번줄은 텍스트를 읽고 긍정인지 부정인지 예측해 보는 구문이다. 긍정이면 1 클래스에 부정이면 0 클래스에 지정하게 된다.

6. 토큰화

```
22. token = Tokenizer()
```

```
23. token.fit_on_texts(docs)
```

```
24. print(token.word_index)
```

```
25. x = token.texts_to_sequences(docs)
```

```
26. print("\n리뷰 텍스트, 토큰화 결과:\n", x)
```

22번 ~ 23번줄은 토큰화 과정을 진행하는 구문으로 케라스에서 제공하는 `Tokenizer()` 함수의 `fit_on_text`를 이용해 각 단어들을 하나의 토큰으로 변경한다. 24번줄의 `print()`함수를 통해서 단어들을 인덱싱하여 보여준다.

<실행결과>

```
{'너무': 1, '재밌네요': 2, '최고예요': 3, '참': 4, '잘': 5, '만든': 6, '영화예요': 7, '추천하고': 8, '싫은'  
리뷰 텍스트, 토큰화 결과:  
[[1, 2], [3], [4, 5, 6, 7], [8, 9, 10], [11, 12, 13], [14], [15], [16, 17], [18, 19], [20]]
```

각 단어별로 1부터 20까지의 숫자로 토큰화하여 구성되었다. 입력된 리뷰 데이터의 토큰수가 다르다는 것을 알 수 있는데, ‘참 잘 만든 영화예요’라는 토큰은 [4, 5, 6, 7]을 가지고 있다. 이러한 입력을 위해서는 학습 데이터의 길이가 동일해야 한다.

7. 패딩

```
27. padded_x = pad_sequences(x, 4)  
28. print("\n패딩 결과:\n", padded_x)
```

토큰화 작업 중에 길이가 일정하지 않아서 생기는 문제점을 방지하기 위해서 패딩 작업을 실시하는 구문이다. 27번줄은 서로 다른 길이를 데이터 4로 맞추어 준다.

<실행결과>

```
패딩 결과:  
[[ 0  0  1  2]  
 [ 0  0  0  3]  
 [ 4  5  6  7]  
 [ 0  8  9 10]  
 [ 0 11 12 13]  
 [ 0  0  0 14]  
 [ 0  0  0 15]  
 [ 0  0 16 17]  
 [ 0  0 18 19]  
 [ 0  0  0 20]]
```

이전에 1부터 20까지의 숫자로 토큰화 하였다. 이후 패딩 작업을 하기 위해서 케라스의 `pad_sequences()` 함수를 이용하여 원하는 길이보다 짧은 경우 0을 채워서 같은 길이로 맞추어 주는 작업이다.

8. 딥러닝 모델 구현

```
29. print("\n딥러닝 모델 시작:")
30. word_size = len(token.word_index) + 1

31. model = Sequential()
32. model.add(Embedding(word_size, 8, input_length=4))
33. model.add(Flatten())
34. model.add(Dense(1, activation='sigmoid'))

35. model.compile(optimizer='adam', loss='binary_crossentropy',
                 metrics=['accuracy'])
36. model.fit(padded_x, classes, epochs=20)
37. print("\n Accuracy: %.4f" % (model.evaluate(padded_x, classes)[1]))
```

최적화 함수로 adam()을 사용하고 오차 함수로 binary_crossentropy() 사용하여 이항 분류를 한다. 20번의 학습을 실행하고 정확도는 100%까지 올리는 것을 확인할 수 있다.

<실행결과>

```
딥러닝 모델 시작:
Epoch 1/20
1/1 [=====] - 0s 453ms/step - loss: 0.6964 - accuracy: 0.4000
Epoch 2/20
1/1 [=====] - 0s 8ms/step - loss: 0.6945 - accuracy: 0.6000
Epoch 3/20
1/1 [=====] - 0s 10ms/step - loss: 0.6927 - accuracy: 0.6000
Epoch 4/20
1/1 [=====] - 0s 8ms/step - loss: 0.6909 - accuracy: 0.7000
Epoch 5/20
1/1 [=====] - 0s 5ms/step - loss: 0.6891 - accuracy: 0.8000
Epoch 6/20
1/1 [=====] - 0s 6ms/step - loss: 0.6873 - accuracy: 0.8000
Epoch 7/20
1/1 [=====] - 0s 11ms/step - loss: 0.6855 - accuracy: 0.9000
Epoch 8/20
1/1 [=====] - 0s 4ms/step - loss: 0.6836 - accuracy: 0.9000
Epoch 9/20
1/1 [=====] - 0s 6ms/step - loss: 0.6818 - accuracy: 0.9000
Epoch 10/20
1/1 [=====] - 0s 5ms/step - loss: 0.6800 - accuracy: 0.9000
Epoch 11/20
1/1 [=====] - 0s 5ms/step - loss: 0.6782 - accuracy: 0.9000
Epoch 12/20
1/1 [=====] - 0s 6ms/step - loss: 0.6764 - accuracy: 0.9000
Epoch 13/20
1/1 [=====] - 0s 5ms/step - loss: 0.6746 - accuracy: 0.9000
Epoch 14/20
1/1 [=====] - 0s 6ms/step - loss: 0.6727 - accuracy: 1.0000
Epoch 15/20
1/1 [=====] - 0s 6ms/step - loss: 0.6709 - accuracy: 1.0000
Epoch 16/20
1/1 [=====] - 0s 5ms/step - loss: 0.6690 - accuracy: 1.0000
Epoch 17/20
1/1 [=====] - 0s 5ms/step - loss: 0.6672 - accuracy: 1.0000
Epoch 18/20
1/1 [=====] - 0s 5ms/step - loss: 0.6653 - accuracy: 1.0000
Epoch 19/20
1/1 [=====] - 0s 5ms/step - loss: 0.6635 - accuracy: 1.0000
Epoch 20/20
1/1 [=====] - 0s 5ms/step - loss: 0.6616 - accuracy: 1.0000
WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_test_function.<locals>.<
1/1 [=====] - 0s 106ms/step - loss: 0.6597 - accuracy: 1.0000

Accuracy: 1.0000
```

출력결과를 살펴보면 학습이 진행됨에 따라서 정확도는 계속해서 향상되고 있으며 손실은 계속해서 줄고 있는 것을 확인할 수 있다.