

04 엑셀을 위한 기본 함수 구현하기



경고 : 본 강의자료는 연세대학교 학생들을 위해 수업 목적으로 제작.게시된 것이므로, 수업목적 이외의 용도로 사용할 수 없으며, 다른 사람과 공유할 수 없습니다. 위반에 따른 법적 책임은 행위자 본인에게 있습니다.



연세대학교
YONSEI MIRAE
CAMPUS

- ◆ 패키지 설치방법 및 **pandas** 기본 사용방법 학습
- ◆ 엑셀의 텍스트 함수, 수학 및 통계 함수 구현

4.1 파이썬으로 엑셀 파일 다루기

4.2 텍스트 함수

4.3 수학 및 통계 함수

- 파이썬은 기능을 확장할 수 있는 다양한 패키지들을 제공하고 있음

- 패키지 관련 용어

구분	주 요 내 용
모듈(module)	특정 기능들(함수, 변수, 클래스 등)이 구현되어 있는 파이썬 파일(.py)
패키지(package)	특정 기능과 관련된 여러 모듈을 하나의 폴더에 넣어 구성
라이브러리(library)	유사한 모듈과 패키지를 묶어서 구성, 사용자가 별도의 설치나 import할 필요 없음

- 엑셀 관련 패키지

패키지	특 징
pandas	테이블 형태의 데이터를 다룰 수 있는 대표적인 파이썬 패키지
openpyxl	엑셀 워크시트 및 셀 편집 , excel 2010 버전 이후 xlsx 파일 편집 가능 (7장에서 사용)
xlwings	엑셀에서 Visual Basic for Application(VBA)를 대체하여 파이썬으로 매크로 구현

◆ Pandas

- "Panel Data Analysis"의 약어
- panel : 다차원 구조화된 데이터
- 오픈소스 데이터 분석 라이브러리 (데이터 분석, 크롤링, 모델링 등)
- 웨스 맥키니(Wes Mckinney)가 2009년 금융 데이터 분석을 위해 설계 개발 시작
- 시계열 등 다양한 금융 데이터 처리 기능을 제공
- R 혹은 Matlab 사용자가 접근하기 용이 (DataFrame과 R의 data.frame 유사)

◆ pandas 자료구조

- **Series**(1차원), **DataFrame**(2차원), **Panel**(3차원)
- Panel은 3D 라벨 배열 (여러 DataFrame을 포함)
- 가장 많이 사용하는 구조는 **DataFrame**(2차원)

Object Type	Indexers
Series	s.loc[indexer]
DataFrame	df.loc[row_indexer, column_indexer]
Panel	p.loc[item_indexer, major_indexer, minor_indexer]

Pandas Series & DataFrame

◆ Series

- 1차원 배열 모습의 자료구조이며, 각 요소는 NumPy의 데이터 타입
- `numpy.ndarray` 의 서브클래스
- 값과 값에 대한 인덱스(index)로 구성
- 리스트, 딕셔너리 혹은 다른 시리즈로 부터 생성
- **index, values**로 구성

◆ DataFrame

- 다양한 데이터를 포함할 수 있는 2차원 자료구조
- 시리즈(Series) 객체들의 딕셔너리
- 스프레드시트와 유사
- DataFrame 은 **columns(열)** 기반. **rows(행)**의 처리 방법과 달리함
 - ✓ **df.columns**
 - ✓ **df.index**
 - ✓ **df.values**



- ◆ 파이썬으로 어떤 특정한 일을 하려면 먼저 패키지를 설치해야 한다.
- ◆ www.pypi.org 에서 다운로드
- ◆ 패키지 설치 명령어

▪ **pip**(Python Install Package) : PyPI(Python Package Index) 를 이용하기 위한 인터페이스

명령어	내용
pip list	현재 환경에서 설치된 패키지를 보여줌
pip install 패키지명	패키지 설치, 일반적으로 압축파일 형태로 특정 사이트에서 제공
pip install 패키지명==버전번호	특정 버전의 패키지 설치 ex) pip install pandas==3.0.0
pip install --upgrade 패키지명	특정 패키지 업그레이드
pip uninstall 패키지명	특정 패키지 삭제



● 설치된 패키지 확인 및 pandas 패키지 설치하기

!pip list # 설치된 패키지 확인하기

openpyxl	3.0.10
outcome	1.1.0
packaging	21.3
pandas	1.4.2
pandoc	2.2
pandocfilters	1.5.0
parso	0.7.1
pickleshare	0.7.5
Pillow	9.1.1
pip	22.1.1
pluggy	1.0.0
plumbum	1.7.2

!pip install pandas # pandas 설치하기

● 모듈 불러오기

➤ **import** 명령어로 패키지 내에 포함된 **모듈**을 불러와야 함. 일반적으로 pandas는 **pd**라는 별명으로 불러옴

pandas 패키지를 pd 라는 별명으로 불러오기

import pandas as pd

● 데이터 프레임 생성하기

- pandas에서는 **Series()**, **DataFrame()** 자료형을 제공. 데이터 프레임은 기본적으로 행(row)과 열(column)로 구성
- **딕셔너리**로 열 이름과 데이터를 정의한 후 **DataFrame()** 함수를 실행

● 이름, 출생년도, 점수로 구성된 데이터 프레임 **df**를 생성

딕셔너리 유형으로 data를 만든 후 데이터 프레임 생성하기

```
Import pandas as pd
```

```
# 모듈 불러오기
```

```
data = {"이름" : ["홍길동", "이순신", "강감찬", "임꺽정", "이성계"],  
        "출생년도" : [1980, 1986, 1990, 1985, 1988],  
        "점수" : [1.5, 1.7, 3.6, 2.4, 2.9]}
```

```
df = pd.DataFrame(data)
```

```
df
```

행 방향 인덱스
0부터 시작

열 이름

	이름	출생년도	점수
0	홍길동	1980	1.5
1	이순신	1986	1.7
2	강감찬	1990	3.6
3	임꺽정	1985	2.4
4	이성계	1988	2.9

- DF의 열 이름을 변경하려면 **rename()** 속성에 변경하려는 열 이름을 **딕셔너리 형**으로, 변경할 축을 **1("columns")** 로 설정
- df의 두 번째 열 이름인 [출생년도]를 [출생]으로 변경

df의 열 이름 변경하기

```
df = df.rename({"출생년도": "출생"}, axis = "columns") # axis = 1  
df
```

	이름	출생	점수
0	홍길동	1980	1.5
1	이순신	1986	1.7
2	강감찬	1990	3.6
3	임꺽정	1985	2.4
4	이성계	1988	2.9

	이름	출생년도	점수
0	홍길동	1980	1.5
1	이순신	1986	1.7
2	강감찬	1990	3.6
3	임꺽정	1985	2.4
4	이성계	1988	2.9

- **axis** : 특정 메소드 또는 함수가 DataFrame에 적용되는 방향을 지정
- **axis = 0(index, row)** : 함수가 행 단위로 적용. 각 컬럼의 모든 행에 대해 작동
- **axis = 1(columns)** : 열 단위로 적용. 모든 컬럼에 대해 작동

● 열과 행 데이터 가져오기

- `df["열이름"]` 또는 `df.열이름` 으로 해당 열의 데이터를 선택
- 2개 이상의 열을 선택할 때는 `df[["열 이름", "열 이름"]]` 이라고 입력

df의 열 데이터 가져오기

`df[["이름", "출생"]]` # [이름]과 [출생] 열 데이터 가져오기

	이름	출생
0	홍길동	1980
1	이순신	1986
2	강감찬	1990
3	임꺽정	1985
4	이성계	1988

`df.이름`

[이름] 열의 데이터만 선택

`#df["이름"]`

같은 결과

```
0 홍길동
1 이순신
2 강감찬
3 임꺽정
4 이성계
Name: 이름, dtype: object
```

- 행 데이터는 `df[시작 인덱스:끝 인덱스]`로 범위를 지정해서 데이터를 가져올 수 있음
 - 인덱스는 0 부터 시작하며, 범위는 시작 인덱스부터 끝 인덱스 -1 까지 임

df 행 데이터 가져오기

`df[1:3]` # 1행부터 2행까지 데이터 가져오기

	이름	출생	점수
1	이순신	1986	1.7
2	강감찬	1990	3.6

● loc[] 과 iloc[] 으로 행 데이터 가져 오기

함 수	내 용
<code>df.loc[0:1, ["이름", "출생"]]</code>	<ul style="list-style-type: none"> • 행, 열 이름으로 범위 지정, 행 이름이 없으면 인덱스를 이름으로 사용. • 인덱스를 행 이름으로 사용할 경우 0:1 은 0 행과 1 행을 의미함
<code>df.iloc[0:2, 0:2]</code>	<ul style="list-style-type: none"> • 인덱스 번호로 선택, 행과 열 범위는 시작 인덱스부터 끝인덱스-1 임. • 행과 열에 지정한 0:2 는 0~1 행, 0~1 열을 의미함

➤ **loc**은 행,열 이름으로, **iloc**은 인덱스로 행과 열을 선택하거나 범위를 지정할 수 있음

df에서 [이름]과 [출생] 열의 0행과 1행을 가져오기

`df.loc[0:1, ["이름", "출생"]]` # [0:1] 행 선택, [이름], [출생] 열 선택

`#df.iloc[0:2, 0:2]` # 위와 동일한 결과

	이름	출생
0	홍길동	1980
1	이순신	1986

● 열 추가 및 삭제하기

➤ 열 추가는 열 이름을 지정하고 리스트 형태로 데이터를 입력하거나, 다른 열과의 연산을 통해 새로운 열을 만들 수 있음

[보너스] 열 추가

```
df["보너스"] = df["점수"] * 5
```

df

	이름	출생	점수	보너스
0	홍길동	1980	1.5	7.5
1	이순신	1986	1.7	8.5
2	강감찬	1990	3.6	18.0
3	임꺽정	1985	2.4	12.0
4	이성계	1988	2.9	14.5

[지역] 열 데이터 추가

```
df["지역"] = ["서울","서울","부산","대구","인천"]
```

df

	이름	출생	점수	보너스	지역
0	홍길동	1980	1.5	7.5	서울
1	이순신	1986	1.7	8.5	서울
2	강감찬	1990	3.6	18.0	부산
3	임꺽정	1985	2.4	12.0	대구
4	이성계	1988	2.9	14.5	인천

➤ 열 삭제는 **del** df["보너스"]로 df의 [보너스] 열을 삭제

[보너스] 열 삭제

```
del df["보너스"]
```

df

	이름	출생	점수	지역
0	홍길동	1980	1.5	서울
1	이순신	1986	1.7	서울
2	강감찬	1990	3.6	부산
3	임꺽정	1985	2.4	대구
4	이성계	1988	2.9	인천

● 행 추가 및 삭제하기

➢ 행을 추가할 때도 loc[] 사용

```
df.loc[5] = ["김순신", 1980, 3.3, "광주"]
```

df

인덱스가 5인 행을 추가

	이름	출생	점수	지역
0	홍길동	1980	1.5	서울
1	이순신	1986	1.7	서울
2	강감찬	1990	3.6	부산
3	임꺽정	1985	2.4	대구
4	이성계	1988	2.9	인천
5	김순신	1980	3.3	광주

➢ 특정 행과 열에 있는 데이터 값만 변경

```
df.iloc[4, 1] = 1999
```

```
df.drop(5, inplace = True)
```

df

4행 1열의 데이터를 1999로 변경

5행을 삭제

	이름	출생	점수	지역
0	홍길동	1980	1.5	서울
1	이순신	1986	1.7	서울
2	강감찬	1990	3.6	부산
3	임꺽정	1985	2.4	대구
4	이성계	1999	2.9	인천

● 특정 행 삭제하기

➤ index() 함수로 특정 조건을 만족하는 인덱스를 구한 후 삭제

```
df1 = df.copy()
id = df1[df1["점수"] <= 2.0].index
df1.drop(id, inplace = True)
df1
```

```
# 원래의 데이터 보존을 위해 df를 df1로 복사
# df1["점수"] <= 2.0인 행의 인덱스를 id에 저장 [0, 1]
# id에 저장된 인덱스로 행 삭제
```

	이름	출생	점수	지역
2	강감찬	1990	3.6	부산
3	임꺽정	1985	2.4	대구
4	이성계	1999	2.9	인천

	이름	출생	점수	지역
0	홍길동	1980	1.5	서울
1	이순신	1986	1.7	서울
2	강감찬	1990	3.6	부산
3	임꺽정	1985	2.4	대구
4	이성계	1999	2.9	인천

- pandas는 다양한 형태의 파일들을 행과 열로 구성된 테이블 형태로 불러올 수 있는 함수를 제공
 - pandas는 엄밀히 따지면 클래스로 구성되어 있지만, 편의상 '**패키지**'로 부르기도 한다.
 - 데이터 프레임의 파일 읽고 쓰기 함수

함 수	내 용
to_csv	데이터 프레임을 쉼표(,) 로 구분된 csv (comma-separated values) 파일로 저장
to_excel	데이터 프레임을 엑셀 로 저장
read_csv	쉼표(,) 로 구분된 데이터를 읽어 옴
read_excel	엑셀(XLS, XLSX) 에서 표 형식 의 데이터를 읽어 옴
read_clipboard	클립보드에 있는 데이터를 읽어 옴(웹 페이지 에서 표를 읽어 올 때 유용)

작업을 위한 디렉토리 만들기



◆ 현재 폴더 아래 **chapter04** 폴더 만들기

! dir	# 현재 폴더의 모든 내용을 보여줌. ! 붙음
mkdir chapter04	# chapter04 폴더 만들기
cd chapter04	# chapter04 폴더로 이동. Change Directory
cd ..	# 상위 폴더로 이동
rmdir chapter04	# 하위에 있는 chapter04 폴더 remove
pwd	# 현재 폴더 경로 정보

- pandas는 다양한 형태의 파일들을 행과 열로 구성된 테이블 형태로 불러올 수 있는 함수를 제공

➤ df 저장 및 읽어 오기

```
# df를 chapter04 디렉토리에 "명단.xlsx"로 저장하기
df.to_excel("chapter04/명단.xlsx")
```

```
# "명단.xlsx" 파일을 불러와 df12에 저장하기
df12 = pd.read_excel("chapter04/명단.xlsx")
# 첫 번째 열을 index로 지정
df12 = pd.read_excel("chapter04/명단.xlsx", index_col = 0)
```

❖ 파일명에 **경로 입력시** 백슬래시(\)를 이스케이프 코드로 인식해 경로명을 제대로 인식하지 못하는 경우가 있음

- `df.to_excel(r"d:\works\명단.xlsx")` 처럼 경로 문자열 앞에 **r** 을 붙이면 해결됨

❖ \ 대신 / (슬래시)를 이용하면 문제 없음

- 문자열내 숫자나 특정 문자를 추출하거나, 불 필요한 문자를 제거
 - 엑셀에서 제공하는 다양한 텍스트 함수를 파이썬 **pandas** 패키지에서 구현할 수 있음
- 직원정보.xlsx 엑셀 파일을 불러와 데이터 프레임 **info**에 저장

```
# pandas를 pd라는 이름으로 불러오기
```

```
import pandas as pd
```

```
# 텍스트 함수 실습을 위한 "직원 정보.xlsx" 파일을 불러와 info에 저장하기
```

```
info = pd.read_excel("D:/Users/2022-여름 집중교육/Programs/chapter04/직원 정보.xlsx", sheet_name = "Sheet1")
```

```
#info = pd.read_excel("chapter04/직원 정보.xlsx", sheet_name = "Sheet1")
```

```
#info = pd.read_excel(r"chapter04\직원 정보.xlsx", sheet_name = "Sheet1")
```

```
info
```

※ 총 10행 출력

	순번	성	이름	영문명	사원번호	주소	전화번호
0	1	김	철수	kim cheolsu	2005-1478345	강서구 공항로 20455	010-1000-8735
1	2	박	종수	park jongsu	2010-1345972	강서구 대저중앙로 3009	010-1000-8736
2	3	김	하나	kim hana	2012-1487234	강서구 하덕로 1002	010-1000-8737
3	4	이	백만	lee baekman	2001-3747234	기장군 기장을 연화100길	010-1000-8738
4	5	백	오십	baek osip	2002-4972944	기장군 기장을 차성로	010-1000-8739

여러 셀의 문자 합치기

- 성과 이름이 분리되어 있거나 주소가 여러 개의 셀로 구분되어 있는 경우 하나의 셀로 결합

➢ 엑셀에서는 concatenate() 함수를, 파이썬에서는 더하기(+) 연산자 또는 sum() 함수를 사용

엑셀

➢ 성과 이름 셀을 결합하기 위해 [H2] 셀에 =concatenate(B2, C2)를 입력, 결과를 [H2] 셀에 저장

직원정보 - Excel

파일 홈 삽입 페이지 레이아웃 수식 데이터 검토 보기 수행할 작업을 알려 주세요.

붙여넣기 글꼴 맞춤 표시 형식

일반 텍스트 줄 바꿈 병합하고 가운데 맞춤

조건부 서식 표 스타일

H2 : =CONCATENATE(B2,C2)

	A	B	C	D	E	F	G	H
1	순번	성	이름	영문명	사원번호	주소	전화번호	
2	1	김	철수	kim cheolsu	2005-1478345	강서구 공항로 20455	010-1000-8735	김철수
3	2	박	종수	park jongsu	2010-1345972	강서구 대저중앙로 3009	010-1000-8736	박종수
4	3	김	하나	kim hana	2012-1487234	강서구 하덕로 1002	010-1000-8737	김하나
5	4	이	백만	lee baekman	2001-3747234	기장군 기장을 연화100길	010-1000-8738	이백만
6	5	백	오십	baek osip	2002-4972944	기장군 기장을 차성로	010-1000-8739	백오십
7	6	영웅	재준	youngwoong jaejun	2011-2382747	기장군 기장을 기장해안로	010-1000-8740	영웅재준
8	7	현	빈	hyun bin	2000-2395634	남구 지게골로 10-2	010-1000-8741	현빈
9	8	장	하나	jang hana	2015-1626816	동래구 온천장로107-100	010-1000-8742	장하나
10	9	유	두울	yoo dooul	2016-4323930	동래구 동래로116	010-1000-8743	유두울
11	10	채	일	chae il	2012-3967845	북구 효열로 2502	010-1000-8744	채일
12								
13								

=concatenate(B2, C2)

여러 셀의 문자 합치기

파이썬

- 문자열 결합을 위해 더하기(+) 연산자 또는 **sum()** 함수를 이용
 - **sum()** 함수 속성 : **axis = 0**이면 **행** 방향으로 각 **열의 합**을, **axis = 1**이면 **열** 방향으로 각 **행의 합**을 계산

```
pd["열3"] = pd["열1"] + pd["열2"]
pd["열3"] = pd[["열1", "열2"]].sum(axis = 0 or 1)
```

```
info = pd.read_excel(r"chapter04/직원 정보.xlsx", sheet_name = "Sheet1")
info["성명"] = info["성"] + info["이름"]           # 데이터 프레임 열 연산
info["성명1"] = info[["성","이름"]].sum(1)        # 열 방향으로 각 행을 결합
info                                              # 데이터 프레임 info 출력
```

※ 총 10행 출력

	순번	성	이름	영문명	사원번호	주소	전화번호	성명	성명1
0	1	김	철수	kim cheolsu	2005-1478345	강서구 공항로 20455	010-1000-8735	김철수	김철수
1	2	박	종수	park jongsu	2010-1345972	강서구 대저중앙로 3009	010-1000-8736	박종수	박종수
2	3	김	하나	kim hana	2012-1487234	강서구 하덕로 1002	010-1000-8737	김하나	김하나
3	4	이	백만	lee baekman	2001-3747234	기장군 기장을 연화100길	010-1000-8738	이백만	이백만
4	5	백	오십	baek osip	2002-4972944	기장군 기장을 차성로	010-1000-8739	백오십	백오십

몇 개의 문자만 추출하기

- 특정 위치나, 범위에 있는 문자열을 추출하여 활용
 - 엑셀에서는 left(), right() 함수를, 파이썬에서는 str() 함수를 사용

엑셀

- 사원번호에서 앞 네자리와 전화번호에서 뒤 네자리 추출

직원정보 - Excel

파일 홈 삽입 페이지 레이아웃 수식 데이터 검토 보기 수행할 작업을 알려 주세요.

클립보드 글꼴 맞춤 표시 형식 스타일

H2 : =LEFT(E2,4)

	A	B	C	D	E	F	G	H	I
1	순번	성	이름	영문명	사원번호	주소	전화번호		
2	1	김	철수	kim cheolsu	2005-1478345	강서구 공항로 20455	010-1000-8735	2005	8735
3	2	박	종수	park jongsu	2010-1345972	강서구 대저중앙로 3009	010-1000-8736	2010	8736
4	3	김	하나	kim hana	2012-1487234	강서구 하덕로 1002	010-1000-8737	2012	8737
5	4	이	백만	lee baekman	2001-3747234	기장군 기장을 연화100길	010-1000-8738	2001	8738
6	5	백	오십	baek osip	2002-4972944	기장군 기장을 차성로	010-1000-8739	2002	8739
7	6	영웅	재준	youngwoong jaejun	2011-2382747	기장군 기장을 기장해안로	010-1000-8740	2011	8740
8	7	현	빈	hyun bin	2000-2395634	남구 지게골로 10-2	010-1000-8741	2000	8741
9	8	장	하나	jang hana	2015-1626816	동래구 온천장로107-100	010-1000-8742		8742
10	9	유	두울	yoo dooul	2016-4323930	동래구 동래로116	010-1000-8743		8743
11	10	채	일	chae il	2012-3967845	북구 효열로 2502	010-1000-8744		8744
12									
13									

=left(E2, 4)는 [E2] 셀의 왼쪽부터
지정한 개수 4만큼 추출

=right(G6, 4)는 [G6] 셀의
오른쪽부터 4만큼의 문자열 추출

몇 개의 문자만 추출하기

파이썬

➤ str[]을 사용하여 특정 문자열을 추출

- 위치는 0부터 시작하며, 범위를 지정할 경우 시작 위치와 **끝위치 -1**을 지정

```
pd["열"].str[위치]
pd["열"].str[시작위치:끝위치]
```

```
info = pd.read_excel(r"chapter04/직원 정보.xlsx", sheet_name = "Sheet1")
info["사원번호앞4자리"] = info["사원번호"].str[0:4]          # 사원번호 앞에서 4자리 추출
info["전화번호뒤4자리"] = info["전화번호"].str[9:13]        # 전화번호 뒤에서 4자리 추출
info
```

※ 총 10행 출력

	순번	성	이름	영문명	사원번호	주소	전화번호	사원번호앞4자리	전화번호뒤4자리
0	1	김	철수	kim cheolsu	2005-1478345	강서구 공항로 20455	010-1000-8735	2005	8735
1	2	박	종수	park jongsu	2010-1345972	강서구 대저중앙로 3009	010-1000-8736	2010	8736
2	3	김	하나	kim hana	2012-1487234	강서구 하덕로 1002	010-1000-8737	2012	8737
3	4	이	백만	lee baekman	2001-3747234	기장군 기장을 연화100길	010-1000-8738	2001	8738
4	5	백	오십	baek osip	2002-4972944	기장군 기장을 차성로	010-1000-8739	2002	8739

몇 개의 문자만 추출하기

파이썬

➤ “주소”에서 원하는 문자열만 추출하기

- `split(" ")` 을 통해 공백을 분리하고, `str[0]` 으로 첫번째 문자열을 [구] 열로 생성

```
info = pd.read_excel(r"chapter04/직원 정보.xlsx", sheet_name = "Sheet1")
```

```
#info["구"] = info["주소"].str.split(" ").str[0]
```

```
info["구"] = info["주소"].str.split(" ").str[1]
```

```
info[["성", "이름", "전화번호", "구"]]
```

※ 총 10행 출력

	성	이름	전화번호	구
0	김	철수	010-1000-8735	강서구
1	박	종수	010-1000-8736	강서구
2	김	하나	010-1000-8737	강서구
3	이	백만	010-1000-8738	기장군
4	백	오십	010-1000-8739	기장군

- `str[1]` 으로 실행해야 결과가 나옴. “주소” 열이 공백으로 시작하기 때문.
- 뒤에서 `strip()` 함수를 쓰기 위함

영문 대소문자 바꾸기

- 영문자가 입력된 셀 전체를 대/소문자로 변환하거나 문자열의 첫 글자만 대문자로 변환
 - 엑셀에서는 upper(), lower(), proper() 함수를, 파이썬에서는 upper(), lower(), capitalize() 함수를 사용

엑셀

- 영문 이름을 모두 대문자로 변환해보고, 공백으로 분리된 첫 문자를 대문자로 변환

직원정보 - Excel

	A	B	C	D	E	F	G	H
1	순번	성	이름	영문명	사원번호	주소	전화번호	
2	1	김	철수	kim cheolsu	2005-1478345	강서구 공항로 20455	010-1000-8735	KIM CHEOLSU
3	2	박	종수	park jongsu	2010-1345972	강서구 대저중앙로 3009	010-1000-8736	PARK JONGSU
4	3	김	하나	kim hana	2012-1487234	강서구 하덕로 1002		KIM HANA
5	4	이	백만	lee baekman	2001-3747234	기장군 기장을 연화100길		LEE BAEKMAN
6	5	백	오십	baek osip	2002-4972944	기장군 기장을 차성로	010-1000-8739	BAEK OSIP
7	6	영웅	재준	youngwoong jaejun	2011-2382747	기장군 기장을 기장해안로	010-1000-8740	YOUNGWONG JAEJUN
8	7	현	빈	hyun bin	2000-2395634	남구 지계골로 10-2	010-1000-8741	HYUN BIN
9	8	장	하나	jang hana	2015-1626816	동래구 온천장로107-100	010-1000-8742	JANG HANA
10	9	유	두울	yoo dooul	2016-4323930	동래구 동래로116	010-1000-8743	YOO DOOUL
11	10	채	일	chae il	2012-3967845	북구 효열로 2502	010-1000-8744	CHAE IL

=upper(D2)는 [D2] 셀의 영문명을 모두 대문자로 변환

=proper(D3)은 [D3]셀의 문자열 중 공백 기준, 앞 글자를 대문자로 변환

=upper(D2)

=proper(D3)

파이썬

- 영문 소문자를 대문자로 변환하는 함수는 `str.upper()`, 대문자를 소문자로 변환하는 함수는 `str.lower()`
 - 영문의 첫 글자만 대문자로 변환하려면 `str.capitalize()` 함수 사용

```
pd["열"].str.upper()    | pd["열"].str.lower()  
pd["열"].str.capitalize()
```

```
info = pd.read_excel(r"chapter04/직원 정보.xlsx", sheet_name = "Sheet1")  
info["대문자"] = info["영문명"].str.upper()           # [영문명] 열을 대문자로 변환  
info["첫글자"] = info["영문명"].str.capitalize()      # [영문명] 열의 첫글자만 대문자로 변환  
info[["순번", "성", "이름", "영문명", "전화번호", "대문자", "첫글자"]]
```

※ 총 10행 출력

	순번	성	이름	영문명	전화번호	대문자	첫글자
0	1	김	철수	kim cheolsu	010-1000-8735	KIM CHEOLSU	Kim cheolsu
1	2	박	종수	park jongsu	010-1000-8736	PARK JONGSU	Park jongsu
2	3	김	하나	kim hana	010-1000-8737	KIM HANA	Kim hana
3	4	이	백만	lee baekman	010-1000-8738	LEE BAEKMAN	Lee baekman
4	5	백	오십	baek osip	010-1000-8739	BAEK OSIP	Baek osip

파이썬

➤ `swapcase()` 함수는 대문자를 소문자로, 소문자를 대문자로 변환

```
info = pd.read_excel(r"chapter04/직원 정보.xlsx", sheet_name = "Sheet1")
info["대문자"] = info["영문명"].str.swapcase()           # "영문명 " 을 모두 대문자로
info["소문자"] = info["대문자"].str.swapcase()          # 대문자로 바뀐 " 대문자 " 를 모두 소문자로
info
```

※ 총 10행 출력

	순번	성	이름	영문명	사원번호	주소	전화번호	대문자	소문자
0	1	김	철수	kim cheolsu	2005-1478345	강서구 공항로 20455	010-1000-8735	KIM CHEOLSU	kim cheolsu
1	2	박	종수	park jongsu	2010-1345972	강서구 대저중앙로 3009	010-1000-8736	PARK JONGSU	park jongsu
2	3	김	하나	kim hana	2012-1487234	강서구 하덕로 1002	010-1000-8737	KIM HANA	kim hana
3	4	이	백만	lee baekman	2001-3747234	기장군 기장을 연화100길	010-1000-8738	LEE BAEKMAN	lee baekman
4	5	백	오십	baek osip	2002-4972944	기장군 기장을 차성로	010-1000-8739	BAEK OSIP	baek osip

특정 문자 바꾸기

- 문자열 데이터에 포함된 하이픈("-")이나 공백(" ")을 다른 문자로 변경
 - 엑셀에서는 replace(), substitute() 함수를, 파이썬에서는 replace() 함수를 사용

엑셀

- 전화번호에 있는 특수문자 하이픈("-") 제거

직원정보 - Excel

파일 홈 삽입 페이지 레이아웃 수식 데이터 검토 보기 수행할 작업을 알려 주세요.

맑은 고딕 11

클립보드 글꼴 맞춤 표시 형식 스타일

H2 =REPLACE(G2,4,1,"")

	A	B	C	D	E	F	G	H
1	순번	성	이름	영문명	사원번호	주소	전화번호	
2	1	김	철수	kim cheolsu	2005-1478345	강서구 공항로 20455	010-1000-8735	0101000-8735
3	2	박	종수	park jongsu	2010-1345972	강서구 대저중앙로 3009	010-1000-8736	0101000-8736
4	3	김	하나	kim hana	2012-1487234	강서구 하덕로 1002	010-1000-8737	0101000-8737
5	4	이	백만	lee baekman	2001-3747234	기장군 기장을	010-1000-8738	0101000-8738
6	5	백	오십	baek osip	2002-4972944	기장군 기장을	010-1000-8739	0101000-8739
7	6	영웅	재준	youngwoong jaejun	2011-2382747	기장군 기장을	010-1000-8740	0101000-8740
8	7	현	빈	hyun bin	2000-2395634	남구 지계골로 10-2	010-1000-8741	0101000-8741
9	8	장	하나	jang hana	2015-1626816	동래구 온천장로107-100	010-1000-8742	0101000-8742
10	9	유	두울	yoo dooul	2016-4323930	동래구 동래로116	010-1000-8743	0101000-8743
11	10	채	일	chae il	2012-3967845	북구 효열로 2502	010-1000-8744	0101000-8744
12								

=replace(G2, 4, 1, "")은 [G2] 셀에 있는 문자열의 4번째부터 1개 문자(-)를 ""로 대체

=substitute(G3, "-", " ")는 [G3] 셀에 있는 "-"을 모두 공백으로 대체

파이썬

- `str.replace()` 함수는 찾은 문자를 새로운 문자로 대체하며, 값이 중복일 경우 몇 번째까지 바꿀 것인지를 변경 횟수로 지정, 변경 횟수를 생략하면 찾은 문자를 모두 새로운 문자로 대체

```
pd["열"].str.replace(찾을 문자, 변경할 문자, 변경 횟수)
```

```
info = pd.read_excel(r"chapter04/직원 정보.xlsx", sheet_name = "Sheet1")
info["phone"] = info["전화번호"].str.replace("-", "", 1)           # 첫 번째 하이픈(-) 제거
info["phone1"] = info["전화번호"].str.replace("-", " ")           # 모든 하이픈(-)을 공백으로 대체
info[["순번", "성", "이름", "전화번호", "phone", "phone1"]]
```

※ 총 10행 출력

	순번	성	이름	전화번호	phone	phone1
0	1	김	철수	010-1000-8735	0101000-8735	010 1000 8735
1	2	박	종수	010-1000-8736	0101000-8736	010 1000 8736
2	3	김	하나	010-1000-8737	0101000-8737	010 1000 8737
3	4	이	백만	010-1000-8738	0101000-8738	010 1000 8738
4	5	백	오십	010-1000-8739	0101000-8739	010 1000 8739

문자열 길이 구하기

● 데이터 분석 시 문자열 길이에 따라 서로 다른 작업을 수행할 때 활용

➢ 엑셀이나 파이썬 모두 **len()** 함수를 이용해 문자열의 길이를 구할 수 있음

엑셀

➢ 주소 문자열 길이 구하기

직원정보 - Excel

파일 홈 삽입 페이지 레이아웃 수식 데이터 검토 보기 수행할 작업을 알려 주세요.

맑은 고딕 11 가 가 텍스트 줄 바꿈 일반 % .00 .00 조건부 서식 표 서식 스타일 셀 삽입 삭제

붙여넣기 가 가 가 가 가 가 병합하고 가운데 맞춤 표시 형식 스타일

H2 : X ✓ fx =LEN(F2)

	A	B	C	D	E	F	G	H
1	순번	성	이름	영문명	사원번호	주소	전화번호	
2	1	김	철수	kim cheolsu	2005-1478345	강서구 공항로 20455	010-1000-8735	14
3	2	박	종수	park jongsu	2010-1345972	강서구 대저중앙로 3009	010-1000-8736	15
4	3	김	하나	kim hana	2012-1487234	강서구 하덕로 1002	010-1000-8737	13
5	4	이	백만	lee baekman	2001-3747234	기장군 기장을 연화100길	010-1000-8738	15
6	5	백	오십	baek osip	2002-4972944	기장군 기장을 차성로	010-1000-8739	12
7	6	영웅	재준	youngwoong jaejun	2011-2382747	기장군 기장을 기장해안로	010-1000-8740	14
8	7	현	빈	hyun bin	2000-2395634	남구 지계골로 10-2	010-1000-8741	13
9	8	장	하나	jang hana	2015-1626816	동래구 온천장로107-100	010-1000-8742	16
10	9	유	두울	yoo dooul	2016-4323930	동래구 동래로116	010-1000-8743	11
11	10	채	일	chae il	2012-3967845	북구 효열로 2502	010-1000-8744	12
12								

=len(F2)는 [F2] 셀에 있는
문자열(공백, 도트 포함)의 길이를 계산

문자열 길이 구하기



파이썬

➤ `str.len()` 함수로 문자열의 길이 계산

```
pd["열"].str.len()
```

```
info = pd.read_excel(r"chapter04/직원 정보.xlsx", sheet_name = "Sheet1")
info["주소길이"] = info["주소"].str.len()
info[["순번", "성", "이름", "주소", "주소길이"]].head()
```

[주소] 열의 문자열 길이 반환
head() : 상위 5행만 출력

	순번	성	이름	주소	주소길이
0	1	김	철수	강서구 공항로 20455	14
1	2	박	종수	강서구 대저중앙로 3009	15
2	3	김	하나	강서구 하덕로 1002	13
3	4	이	백만	기장군 기장읍 연화100길	15
4	5	백	오십	기장군 기장읍 차성로	12

문자열 공백 삭제하기

- 문자열 앞뒤에는 보이지 않는 공백이 존재하는데 공백을 제거해야 정확한 길이를 구할 수 있음
 - 엑셀에서는 trim() 함수를, 파이썬에서는 strip(), lstrip(),.rstrip() 함수를 활용

엑셀

- 주소 문자열의 공백 제거전 길이와 공백 제거후 길이 비교

직원정보 - Excel						
파일 홈 삽입 페이지 레이아웃 수식 데이터 검토 보기 수형할 작업을 알려 주세요.						
<div> <div> <div>붙여넣기</div> <div>클립보드</div> </div> <div> <div>글꼴</div> <div>맞춤</div> </div> <div> <div>표시 형식</div> <div>스타일</div> </div> <div> <div>조건부 서식</div> <div>표</div> <div>셀</div> </div> <div> <div>삽입</div> <div>삭제</div> <div>서식</div> </div> <div> <div>정렬 및 필터</div> <div>편집</div> </div> </div>						
H2	=LEN(F2)					
	B	C	D	E	F	G
1	성	이름	영문명	사원번호	주소	전화번호
2	김	철수	kim cheolsu	2005-1478345	강서구 공항로 20455	010-1000-8735
3	박	종수	park jongsu	2010-1345972	강서구 대저중앙로 3009	010-1000-8736
4	김	하나	kim hana	2012-1487234	강서구 하덕로 1002	010-1000-8737
5	이	백만	lee baekman	2001-3747234	기장군 기장을 연화100길	010-1000-8738
6	백	오십	baek osip	2002-4972944	기장군 기장을 차성로	010-1000-8739
7	영웅	재준	youngwoong jaejun	2011-2382747	기장군 기장을 기장해안로	010-1000-8740
8	현	빈	hyun bin	2000-2395634	남구 지게골로 10-2	010-1000-8741
9	장	하나	jang hana	2015-1626816	동래구 온천장로107-100	010-1000-8742
10	유	두울	yoo dooul	2016-4323930	동래구 동래로116	010-1000-8743
11	채	일	chae il	2012-3967845	북구 효열로 2502	010-1000-8744
12						

[I2] 셀에 =trim(F2)를 입력하여
공백이 제거된 주소 문자열을 추출

[J2] 셀에 =len(I2)를 입력하여
공백이 제거된 문자열의 길이를 구함

H열과 J열의 데이터를 비교해보면
공백 제거 전, 후의 문자열 길이가
다른 것을 확인할 수 있음

파이썬

➤ strip() 함수로 문자열의 양끝(왼쪽, 오른쪽) 공백 제거, lstrip()은 왼쪽 공백,.rstrip()은 오른쪽 공백을 제거

```
pd["열"].str.strip()    # 문자열 양끝(왼쪽,오른쪽) 공백 제거
pd["열"].str.lstrip()   # 문자열의 왼쪽 공백 제거
pd["열"].str.rstrip()   # 문자열의 오른쪽 공백 제거
```

```
info = pd.read_excel(r"chapter04/직원 정보.xlsx", sheet_name = "Sheet1")
info["주소길이"] = info["주소"].str.len()           # [주소] 열의 길이 구하기
info["공백제거"] = info["주소"].str.strip()         # [주소] 열의 문자열 앞뒤 공백 제거
info["공백제거후 길이"] = info["공백제거"].str.len() # 공백 제거 문자열의 길이 구하기
info[["순번", "주소", "주소길이", "공백제거", "공백제거후 길이"]].tail(6) # 뒤에서 6행 출력
```

	순번	주소	주소길이	공백제거	공백제거후 길이
4	5	기장군 기장읍 차성로	12	기장군 기장읍 차성로	11
5	6	기장군 기장읍 기장해안로	14	기장군 기장읍 기장해안로	13
6	7	남구 지게골로 10-2	13	남구 지게골로 10-2	12
7	8	동래구 온천장로107-100	16	동래구 온천장로107-100	15
8	9	동래구 동래로116	11	동래구 동래로116	10
9	10	북구 효열로 2502	12	북구 효열로 2502	11

- 엑셀은 매우 편리하게 활용할 수 있는 수학 및 통계 함수를 제공하고 있는데 파이썬으로도 구현 가능
 - 데이터가 적을 때는 엑셀 함수를 활용하는 것이 편리하지만, 수천~수만 개의 행을 처리할 때는 파이썬이 유리
- 실습 데이터 불러오기
 - read_excel() 함수로 "성적 처리.xlsx" 파일을 불러온 후 데이터 프레임 score에 저장

pandas를 pd라는 이름으로 불러오기

import pandas as pd

수학 및 통계함수 실습을 위해 '성적 처리.xlsx' 엑셀을 불러와 score에 저장하기

score = pd.read_excel(r"chapter04/성적 처리.xlsx", sheet_name = "Sheet1")

score

※ 총 12행 출력

	반	성명	국어	영어	수학	사회	과학
0	1반	홍길동	93	80	94	73	64
1	2반	백일홍	93	63	76	84	92
2	3반	이삼상	94	74	86	90	70
3	1반	정말로	83	55	64	90	65
4	2반	한번도	87	95	66	75	60

데이터 합계 구하기

- 엑셀을 배울 때 처음 접하는 함수가 `sum()`임, 더하기가 가장 기본적인 연산이고 실무에서도 많이 쓰임
 - 엑셀과 파이썬 모두 `sum()` 함수 또는 더하기(+) 연산자를 활용해 합계를 구할 수 있음

엑셀

➢ 개인별 성적합계 계산

	A	B	C	D	E	F	G	H	I
1	반	성명	국어	영어	수학	사회	과학		
2	1반	홍길동	93	80	94	73	64	404	
3	2반	백일홍	93	63	76	84	92	408	
4	3반	이상상	94	74	86	90	70	414	
5	1반	정말로	83	55	64			357	
6	2반	한번도	87	95	66			383	
7	3반	이철수	53	81	59	88	69	350	
8	1반	김영자	71	71	51	84	57	334	
9	2반	다니엘	87	54	95	71	97	404	
10	3반	이미로	59	54	75	90	82	360	
11	1반	신성삼	64	66	59	91	86	366	
12	2반	케로로	56	76	52	64	65	313	
13	3반	장발장	85	51	64	80	68	348	
14									

[H2] 셀에 `=sum(C2:G2)`라고
입력하면 C2+D2+E2+F2+G2를
실행한 결과가 저장됨

파이썬

➤ **sum()** 함수로 데이터 프레임의 열, 행 방향의 합계를 구할 수 있음

- 속성을 **axis=0(기본)**으로 지정 시 **행** 방향으로 각 열을 더함, **axis=1**로 지정 시 **열** 방향으로 각 행의 합계를 산출

```
pd["열"].sum(axis=0, or 1)          # 행 방향 열(0) 또는 열방향 행(1) 합계 구하기
pd["열"] + pd["열"] + ... + pd["열"] # + 연산자로 직접 계산
```

```
score = pd.read_excel(r"chapter04/성적 처리.xlsx", sheet_name = "Sheet1")
total_korea = score["국어"].sum(0)    # 행 방향 [국어] 열의 데이터 합계 구하기
total_korea
```

1508

```
score["sum"] = score.iloc[:, 2:7].sum(1)    # 2열 ~ 6열(국어~과학) 각 행의 데이터 합계 구하기
score["sum1"] = score["국어"] + score["영어"] + score["수학"] + score["사회"] + score["과학"]
score.head()
```

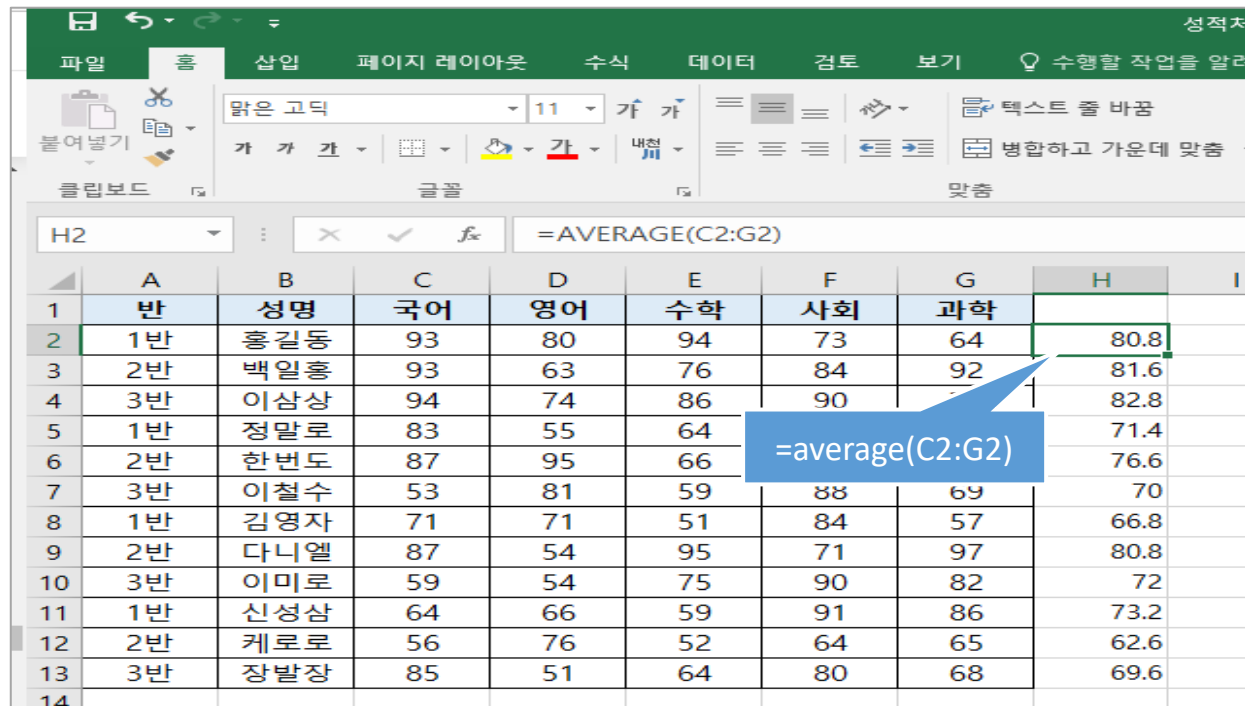
	반	성명	국어	영어	수학	사회	과학	sum	sum1
0	1반	홍길동	93	80	94	73	64	404	404
1	5반	백일홍	93	63	76	84	92	408	408
2	3반	이상상	94	74	86	90	70	414	414
3	4반	정달로	83	55	64	90	65	357	357
4	5반	한번도	87	95	66	75	60	383	383

데이터 평균 구하기

- 데이터의 평균을 구하는 것은 더하기(+) 연산과 마찬가지로 정형 데이터를 다룰때 가장 많이 사용
 - 엑셀에서는 average() 함수를, 파이썬에서는 **mean()** 함수를 이용해 평균을 구할 수 있음

엑셀

- 개인별 성적 평균 계산



The image shows an Excel spreadsheet with a green header bar. The formula bar at the top displays `=AVERAGE(C2:G2)`. The spreadsheet contains columns for class (반), name (성명), and scores for Korean (국어), English (영어), Math (수학), Social Studies (사회), and Science (과학). The average score for each student is calculated in column H. A blue callout box points to cell H2, containing the text `=average(C2:G2)`.

	A	B	C	D	E	F	G	H
1	반	성명	국어	영어	수학	사회	과학	
2	1반	홍길동	93	80	94	73	64	80.8
3	2반	백일홍	93	63	76	84	92	81.6
4	3반	이상상	94	74	86	90		82.8
5	1반	정말로	83	55	64			71.4
6	2반	한번도	87	95	66			76.6
7	3반	이철수	53	81	59	88	69	70
8	1반	김영자	71	71	51	84	57	66.8
9	2반	다니엘	87	54	95	71	97	80.8
10	3반	이미로	59	54	75	90	82	72
11	1반	신성삼	64	66	59	91	86	73.2
12	2반	케로로	56	76	52	64	65	62.6
13	3반	장발장	85	51	64	80	68	69.6
14								

[H2] 셀에 `=average(C2:G2)`를 입력하면
홍길동의 국어, 영어, 수학, 사회, 과학 점수의
평균을 구할 수 있음

파이썬

➤ **mean()** 함수로 데이터 프레임의 열, 행 방향의 평균을 구할 수 있음

- 속성을 **axis=0**으로 지정 시 **행** 방향으로 각 열의 수치형 데이터 평균을, **axis=1**로 지정 시 **열** 방향으로 각 행의 평균을 계산

```
pd["열"].mean(axis = 0 or 1)          # 행(0) 또는 열(1)방향 평균 구하기  
(pd["열"] + pd["열"] + ---- + pd["열"])/ 열 개수      # +, / 연산자로 직접 계산
```

```
score = pd.read_excel(r"chapter04/성적 처리.xlsx", sheet_name = "Sheet1")  
korea_avg = score["국어"].mean()      # [국어] 열의 행 방향 평균 구하기  
korea_avg
```

75.4

```
score["평균"] = score.iloc[:, 2:7].mean(1)          # 2열 ~ 6열의 열 방향 각 행의 평균 구하기  
score["평균1"] = (score["국어"] + score["영어"] + score["수학"] + score["사회"] + score["과학"])/5
```

score

	반	성명	국어	영어	수학	사회	과학	평균	평균1
0	1반	홍길동	93	80	94	73	64	80.8	80.8
1	5반	백일홍	93	63	76	84	92	81.6	81.6

조건에 따른 합계, 평균 구하기

● 데이터를 요약하여 보고할 때, 조건에 따라 항목별 합계를 구하거나 평균을 계산

➢ 엑셀에서는 sumif(), averageif() 함수를, 파이썬에서는 groupby() 함수와 sum(), mean() 함수를 결합하여 사용

엑셀

➢ 반별 과목 합계를 구하기

성적처리 - Excel

파일 홈 삽입 페이지 레이아웃 수식 데이터 검토 보기 수행할 작업을 알려 주세요.

클립보드 글꼴 맞춤 표시 형식 스타일

J2 : X ✓ f =SUMIF(\$A\$2:\$A\$13,\$I2,C\$2:C\$13)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
	반	성명	국어	영어	수학	사회	과학		반	국어	영어	수학	사회	과학
2	1반	홍길동	93	80	94	73	64		1반	311	272	268	338	272
3	2반	백일홍	93	63	76	84	92		2반	323	288	289	294	314
4	3반	이삼상	94	74	86	90	70		3반	291	260	284	348	289
5	1반	정말로	83	55	64	90	65							
6	2반	한번도	87	95	66	75	60							
7	3반	이철수	53	81										
8	1반	김영자	71	71										
9	2반	다니엘	87	54										
10	3반	이미로	59	54	75	90	82							
11	1반	신성삼	64	66	59	91	86							
12	2반	케로로	56	76	52	64	65							
13	3반	장발장	85	51	64	80	68							
14														

=SUMIF(\$A\$2:\$A\$13, \$I2, C\$2:C\$13)

[I1:N1] 각 셀에 반, 국어, 영어, 수학, 사회, 과학을, [I2:I4] 각 셀에 1반, 2반, 3반을 입력

[J2] 셀 에 =SUMIF(\$A\$2:\$A\$13, \$I2, C\$2:C\$13)을 입력해 1반의 국어 점수 합계를 구함

\$A\$2:\$A\$13은 조건을 탐색할 범위, \$I2는 조건, C\$2:C\$13은 합계를 구할 범위를 지정

조건에 따른 평균을 구하는 함수 =averageif()도 동일한 방법으로 사용

조건에 따른 합계, 평균 구하기

파이썬

➤ **groupby()** 함수로 조건에 따른 합계 및 평균을 구할 수 있음

- [조건 열]은 그룹화할 기준열을 의미하며, 계산 결과는 데이터 프레임으로 출력됨

```
pd.groupby(["조건 열"]).sum()      # 조건 열을 기준으로 그룹화 후 합계 계산
pd.groupby(["조건 열"]).mean()    # 조건 열을 기준으로 그룹화 후 평균 계산
```

```
score = pd.read_excel(r"chapter04/성적 처리.xlsx", sheet_name = "Sheet1")
score1 = score.groupby(["반"]).sum()      # [반] 열을 기준으로 그룹화 한 후 합계 계산
score1                                     # score1 출력
```

```
score2 = score.groupby(["반"]).mean()     # 반 열을 기준으로 그룹화 한 후 평균 계산
score2                                    # score2 출력
```

score1

	국어	영어	수학	사회	과학
반					
1반	286	298	272	287	275
2반	287	252	289	330	318
3반	307	289	286	294	277
4반	307	247	251	289	303
5반	321	309	325	286	327

score2

	국어	영어	수학	사회	과학
반					
1반	71.50	74.50	68.00	71.75	68.75
2반	71.75	63.00	72.25	82.50	79.50
3반	76.75	72.25	71.50	73.50	69.25
4반	76.75	61.75	62.75	72.25	75.75
5반	80.25	77.25	81.25	71.50	81.75

순위 구하기

- 수치 데이터일 경우 평균이나 총계를 구한 다음, 내림차순 또는 오름차순으로 순위를 구함
 - 엑셀, 파이썬 모두 rank() 함수를 활용하여 순위를 계산할 수 있음

엑셀

- 개인별 성적 평균값을 기준으로 내림차순, 오름차순 순위 구하기

	A	B	C	D	E	F	G	H	I	J
	반	성명	국어	영어	수학	사회	과학	평균	순위(내림차순)	순위(오름차순)
2	1반	홍길동	93	80	94	73	64	80.8	3	9
3	2반	백일홍	93	63	76	84	92	81.6	2	11
4	3반	이상상	94	74	86	90	70	83.0	1	12
5	1반	정말로	83	55	64				8	5
6	2반	한번도	87	95	66				5	8
7	3반	이철수	53	81	59				9	4
8	1반	김영자	71	71	51	84	57	66.8	1	2
9	2반	다니엘	87	54	95	71	97	80.8	3	9
10	3반	이미로	59	54	75	90	82			
11	1반	신성삼	64	66	59	91	86			
12	2반	케로로	56	76	52	64	65			
13	3반	장발장	85	51	64	80	68	69.6	10	3

[H2:J2] 범위에 평균, 순위(내림차순), 순위(오름차순)를 입력

[H2] 셀에 =AVERAGE(C2:G2)를 입력하여 개인별 평균을 계산,

[I2] 셀에 =RANK.EQ(H2, \$H\$2:\$H\$13, 0)을 입력. 내림차순 순위를 구하고,

[J2] 셀에 =RANK.EQ(H2, \$H\$2:\$H\$13, 1)을 입력하면 오름차순 순위를 구할 수 있음

파이썬

- 파이썬에서는 **rank()** 함수로 순위를 구할 수 있음

```
pd["열"].rank(method = "방법", ascending = True/False, pct = True/False)
```

- rank() 함수의 method 속성

method	내 용
average	동점 관측치 간의 그룹 내 평균 순위 부여(default) ex) 3위가 2개일 때 3.5 위로 순위 부여. 다음 순위는 5위부터 시작
min	동점 관측치 그룹 내 최소 순위 부여 ex) 3위가 2개일 때 3 위로 순위 부여. 다음 순위는 5위부터 시작
max	동점 관측치 그룹 내 최대 순위 부여 ex) 3위가 2개일 때 4 위로 순위 부여. 다음 순위는 5위부터 시작
first	동점 관측치 중 데이터 상에서 먼저 나타나는 관측치부터 순위 부여 ex) 3위가 2개일 때 처음 관측치는 3 위, 다음 관측치는 4 위
dense	최소값(min) 과 같은 방법으로 순위를 부여하나 그룹 간 순위가 1씩 증가 ex) 3위가 2개일 때 3 위로 순위 부여. 다음 순위는 4 위

파이썬

➤ rank() 함수

```
pd["열"].rank(method = "방법", ascending = True/False, pct = True/False)
```

- **ascending**은 True일 경우 오름차순, False일 경우 내림차순으로 정렬
- **pct**는 True일 때 순위를 백분율로 출력, False일 때는 순위로 출력

```
score = pd.read_excel(r"chapter04/성적 처리.xlsx", sheet_name = "Sheet1")
score["평균"] = score.iloc[:, 2:7].mean(1) # 2~6열 열 방향 각 행의 평균 구하기
score["순위_오름"] = score["평균"].rank(ascending = True) # 동점 시 평균 순위 부여, 오름 차순
score["순위_내림"] = score["평균"].rank(ascending = False) # 동점 시 평균 순위 부여, 내림 차순
score["순위_내림_min"] = score["평균"].rank(method = "min", ascending = False) # 동점 시 최소 순위
score
```

※ 일부만 출력

	반	성명	국어	영어	수학	사회	과학	평균	순위_오름	순위_내림	순위_내림_min
0	1반	홍길동	93	80	94	73	64	80.8	9.5	3.5	3.0
1	2반	백일홍	93	63	76	84	92	81.6	11.0	2.0	2.0
2	3반	이삼상	94	74	86	90	70	82.8	12.0	1.0	1.0
3	1반	정말로	83	55	64	90	65	71.4	5.0	8.0	8.0
4	2반	한번도	87	95	66	75	60	76.6	8.0	5.0	5.0

파이썬

- `sort_values()` 함수를 이용하여 데이터를 특정 열을 기준으로 정렬하기
 - `by = "열이름"` 속성으로 정렬할 열 이름을 지정하고, `ascending = True`(오름차순) / `False`(내림차순)

```
score.sort_values(by = "평균", ascending = False)
```

※ 일부만 출력

	반	성명	국어	영어	수학	사회	과학	평균	순위_오름	순위_내림	순위_내림_min
2	3반	이삼상	94	74	86	90	70	82.8	20.0	1.0	1.0
1	5반	백일홍	93	63	76	84	92	81.6	19.0	2.0	2.0
0	1반	홍길동	93	80	94	73	64	80.8	17.5	3.5	3.0
7	5반	다니엘	87	54	95	71	97	80.8	17.5	3.5	3.0
15	3반	조미료	86	68	97	56	85	78.4	16.0	5.0	5.0

최대값/최소값 구하기

● 수치형 데이터 처리 시 최대값/최소값을 기준으로 특정 조건에 따른 작업을 수행할 때 필요

➢ 엑셀, 파이썬 모두 max(), min() 함수를 활용하여 최대값/최소값을 계산

엑셀

➢ 개인 과목별 최대값, 최소값 구하기

The screenshot shows an Excel spreadsheet titled '성적처리 - Excel'. The data is organized in columns: A (반), B (성명), C (국어), D (영어), E (수학), F (사회), G (과학), H (최대값), and I (최소값). Rows 2-13 contain student data. The formula bar shows '=MAX(C2:G2)' for cell H2. A blue callout points to cell H2 with the text '=MAX(C2:G2)'. Another blue callout points to cell I2 with the text '=MIN(C2:G2)'.

	A	B	C	D	E	F	G	H	I
1	반	성명	국어	영어	수학	사회	과학	최대값	최소값
2	1반	홍길동	93	80	94	73	64	93	64
3	2반	백일홍	93	63	76	84	92	93	92
4	3반	이삼삼	94	74	86			94	70
5	1반	정말로	83	55	64			83	65
6	2반	한번도	87	95	66				60
7	3반	이철수	53	81	59	88	69		53
8	1반	김영자	71	71	51	84			57
9	2반	다니엘	87	54	95	71			87
10	3반	이미로	59	54	75	90	82	82	59
11	1반	신성삼	64	66	59	91	86	86	64
12	2반	케로로	56	76	52	64	65	65	56
13	3반	장발장	85	51	64	80	68	85	68
14									

[H1] 셀에 최대값, [I1] 셀에 최소값을 입력

[H2] 셀에 =MAX(C2:G2)를 입력하면
홍길동의 과목별 최대값을,

[I2] 셀에 =MIN(C2:G2)를 입력하면 과목별
최소값을 구할 수 있음

최대값/최소값 구하기

파이썬

- 파이썬에서는 `max()`, `min()` 함수로 최대값/최소값을 구할 수 있음
 - `axis=0`이면 행 방향으로 각 열의 최대값/최소값을, `axis=1`이면 열 방향으로 각 행의 최대값/최소값을 구함

```
pd.min(axis=0 or 1)    # 최소값
pd.max(axis=0 or 1)    # 최대값
```

```
score = pd.read_excel(r"chapter04/성적 처리.xlsx", sheet_name = "Sheet1")
score_row = score.iloc[ : , [2,3,4,5,6]]    # score에서 숫자열(국어~과학)만 추출하여 score_row에 저장
score["MIN"] = score_row.min(1)              # score_row의 행 방향 최소값을 score["MIN"]에 저장
score[ " MAX " ] = score_row.max(1)          # score_row의 행 방향 최대값을 score["MAX"]에 저장
score.head()                                # score의 앞쪽 5개 행만 출력
```

※ 일부만 출력

	반	성명	국어	영어	수학	사회	과학	MIN	MAX
0	1반	홍길동	93	80	94	73	64	64	94
1	2반	백일홍	93	63	76	84	92	63	93
2	3반	이삼상	94	74	86	90	70	70	94
3	1반	정말로	83	55	64	90	65	55	90
4	2반	한번도	87	95	66	75	60	60	95

파이썬

➤ describe() 함수로 데이터 프레임의 기초 통계량을 확인

- 각 열별 데이터 개수 및 평균, 표준편차, 최소값, 4분위수, 최대값이 데이터 프레임 형태로 출력

```
score = pd.read_excel(r"chapter04/성적 처리.xlsx", sheet_name = "Sheet1")
score.describe()          # score 데이터 프레임의 기초 통계량 확인
```

	국어	영어	수학	사회	과학
count	12.000000	12.000000	12.000000	12.000000	12.000000
mean	77.083333	68.333333	70.083333	81.666667	72.916667
std	15.512214	13.580289	15.174490	9.018500	13.041600
min	53.000000	51.000000	51.000000	64.000000	57.000000
25%	62.750000	54.750000	59.000000	74.500000	64.750000
50%	84.000000	68.500000	65.000000	84.000000	68.500000
75%	88.500000	77.000000	78.500000	90.000000	83.000000
max	94.000000	95.000000	95.000000	91.000000	97.000000