Elixirr

파이썬 데이터분석

- 아르테미스 달탐사 -

인선미

수업 목차

- 오버 더 문과 달 탐사 비교
- 문제 정의

- Git Hub 접속 및 환경설정
- 암석 데이터 수집

- 파이썬 및 판다스 기초
- 수집된 암석 데이터 관찰

- 데이터프레임 컬럼값 변환
- 아폴로 임무별로 데이터를 가지는 missions 데이터프레임 만들기

- 암석 중량 데이터 추가
- 달 탐사선의 달모듈 중량 데이터 추가

- 명령모듈과 승무원영역 중량 데이터 추가
- 달 탐사선에서 암석이 차지하는 비율을 missions 데이터프레임에 추가

- 아르테미스 임무에서 가져올 예상 암석 중량 구하기

- 수집된 암석 중에서 남아 있는 양이 적은 암석을 구하는 데이터프레임 만들기

- 달에서 수집할 암석 종류를 알려주는 데이터프레임 만들기

- 최종 수집할 암석 종류와 갯수 구하기

데이터 분석의 5 단계

- 현재 상황에 따른 문제점 파악
- 미래 개선될 점에 대해서 분석 주제 정의

- 문제 정의에 따라서 분석에 필요한 데이터 선택
- 데이터 수집

- 확보한 데이터 관찰 및 기초 데이터 분석
- 분석에 필요한 데이터 컬럼 추가 및 가공

- 문제 정의를 반영한 데이터 분석
- 연관될 수 있는 다른 데이터의 수집, 연결 및 분석

- 데이터 분석을 반영할 수 있는 시각화
- 문제 정의에 맞춘 해결방안 도출

1.문제 정의

2.데이터 수집

3.데이터 가공

4.데이터 분석

5.시각화 및 탐색

1. 문제 정의

- 오버 더 문 소개
- 우주 암석과 달 암석의 중요성
- NASA의 달 탐사
- 달 암석 채취시의 문제점 파악 및 분석 주제 결정

데이터 분석의 5 단계

1.문제 정의

데이터 분석 단계에 맞추어 달 탐사 및 암석샘플 데이터 분석을 수행한다.

데이터 분석의 5단계

- **달 탐사 및 달 암석 채취시 문제점파악**
- **미래의 달 탐사 및 암석 채취시 개선될 점에 대해서 분석주제 정의**

- 문제 정의에 따라서 분석에 필요한 데이터 선택
- 모든 아폴로 달 탐사에서 채취된 암석 샘플 데이터 수집

- 암석 샘플 데이터 기초분석하고 필요한 컬럼 추가
- 아폴로 임무별 새로운 데이터 프레임 만들기

- 아르테미스 임무에서 채취할 암석
- 중량 구하기
- 수집해야 하는 암석 종류 구하기

- 달에서 수집할 암석 종류를 알려주는
- 데이터프레임 만들기
- 최종 수집할 암석 종류와 개수 구하기

Elixirr

Over The Moon 소개

오버더문 영화에서 나오는 달 탐사는 어떤 것인지 한번 볼까요?

오버 더 문 (Over the Moon)

오버 더 문은 페이페이라는 소녀가 달의 여신 항아를 만나기 위해 로켓을 만들어 모험을 떠나는 아름다운 이야기의 영화다.

영화 오버 더 문 줄거리

- 똑똑하고 호기심 넘치는 소녀 페이페이 는 엄마가 어린 시절 들려준 전설 속 달 의 여신 항아 이야기를 늘 마음에 품고 살 아간다.
- 일찍 하늘나라로 간 엄마를 그리워하며 살아가던 페이페이는 어느 날 항아의 전설이 지어낸 이야기일 뿐이라는 고모의 말에 울컥해, 항아의 존재를 증명하기 위해 달로 향 하는 로켓을 만들기로 결심한다.
- 수많은 시행착오 끝에 로켓 만들기에 성공 한 페이 페이는 예상치 못한 모험을 떠나 게 된다 .

오버 더 문 학습 경로

- 마이크로소프트의 오버 더 문 학습 경로는 영화 오버 더 문의 줄거리 에서 테마를 착안함
- 테마를 바탕으로 데이터 분석, 인공 지능 등을 공부할 수 있는 학습 모듈 을 제공

※테마		※학습 모듈
• 항아 이야기	→	• 유성우 예측
• 로켓 만들기	→	• 로켓 발사 연기 예측
• 달 탐사 모험	→	• 달 탐사 및 우주 암석
• 암석 식별 머신	→	• 달 암석 식별 머신

자료 : 네이버 영화 (Netflix의 Over the Moon), 이미지 및 줄거리 포함

페이 페이의 열정과 아이디어

1. 달 탐사 임무에서 가장 중요한 것
- 엄청난 열정, 독창성, 꾸준한 노력
- 철저한 준비
- 위의 것들은 페이페이와 아폴로 달 탐사 임무에 임했던 모든 사람들의 공통점이다.

2. 페이 페이가 로켓에 사용한 원리
- 자석 : 하늘로 올라가는 데 충분한 속도와 추진력 제공
- 폭죽 : 대기권을 통과하기 위한 마지막 추진력 제공

https://docs.microsoft.com/ko-kr/learn/modules/plan-moon-mission-using-python-pandas/1-introduction

페이 페이가 간과한 것은 무엇일까요?

오버 더 문 | 공식 예고편 #1 | Netflix & 펄 스튜디오 작품

1. 페이 페이가 우주 비행을 위해 정확히 계산한 것은 중량이다.
- 자신과 번지의 체중, 음식과 화장실 사용
- 이러한 철저한 중량 계산은 아폴로 달 탐사 임무와의 공통점이다.

2. 페이페이가 간과한 중량
- 우주선에 몰래 탑승한 동생 친의 중량
- 아폴로 13 탐사 - 5개 엔진 중 하나가 2분 먼저 중지되는 사건

3. 페이페이의 로켓과 아폴로 13의 공통점
- 중량 대비 불충분한 추진력 -> 지구 대기권 통과 실패

4. 우주 비행사들의 안전을 보장하기 위한 계획이 가장 최우선의 과제

NASA 임무 : 우주 암석 연구

1. NASA 임무
- 우주 암석 검색을 위해 우주 비행사와 탐사선을 우주로 보내는 것

2. NASA OSIRIS-Rex 임무

- 궤도로 우주선을 발사하고 Bennu 라는 소행성에서 표본을 수집하는 과정
- 소행성의 전토층은 태양계의 초창기 역사가 기록되어 있다.
- 궤도에 있는 동안 고화질 사진을 활영하고 표본 수집을 위해 착륙
- 사진은 지구에 있는 암석 전문가에게 전송되어 태양계에 대한 정보를 얻어낸다.

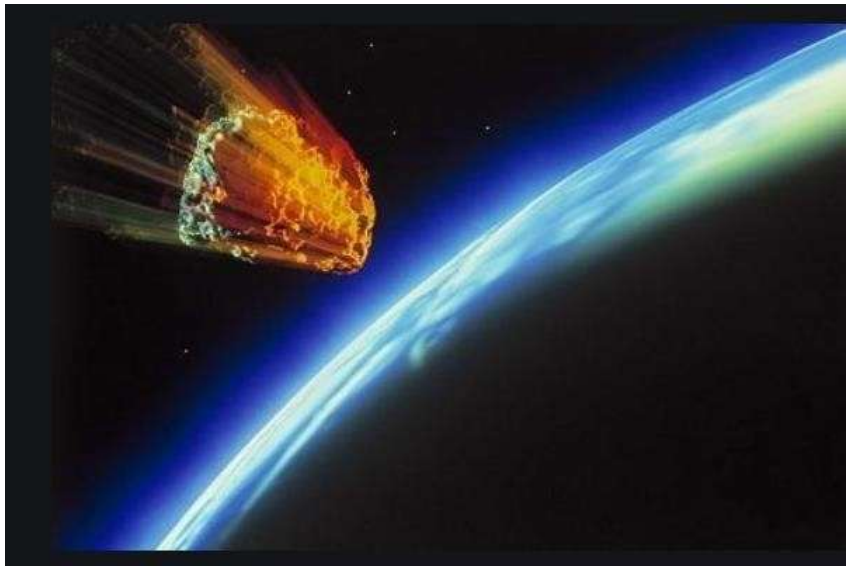
소행성의 전토층에 대한 연구는 인간이 어디에서 왔으며 어떤 운명이 기다리고 있는지 알려 줄 수 있어.

https://docs.microsoft.com/ko-kr/learn/modules/research- space-rocks-ai-nasa/6-example-mission

우주 암석

1. 우주 암석이란 ?

우주 암석은 달에서 발견되는 미세한 토양부터 우주를 떠다니는 행성 크기의 암석까지 다양하다. 우주에서 시작된 암석은 우주를 떠다니며 유성체 및 소행성으로 발견되거나 달, 행성, 심지어는 지구 표면에서 운석으로 발견된다.



https://images.app.goo.gl/eqL8MDdHV6XwGnXf8

https://images.app.goo.gl/zgXwSXcKEBXuKDFV7

2. 우주 암석 연구의 중요성

암석은 화산 분출과 같은 지질학적 이벤트를 기록하여 태양계의 역사에 대해 알려준다. 우주 암석은 인간보다 훨씬 오래되었으며 인간이 사라진 후에도 훨씬 더 오래 존재할 것이다.

우주 암석을 연구하는 이유

1. 우주 암석을 조사하여 태양계를 연구하는 방법의 한 가지 예는 화강암

화강암은 행성의 현재 구조와 행성이 형성된 방식에 대한 정보를 제공한다.

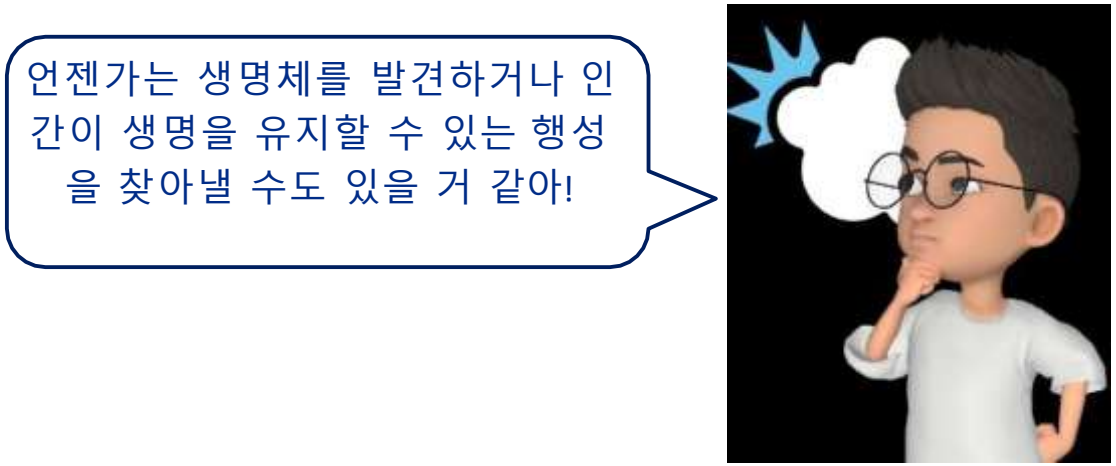
https://www.sedaily.com/News/NewsView/PhotoViewer?Nid=1OESP3LU8V&Page=1

2. 지구 이외의 행성에 생명이 있을까?

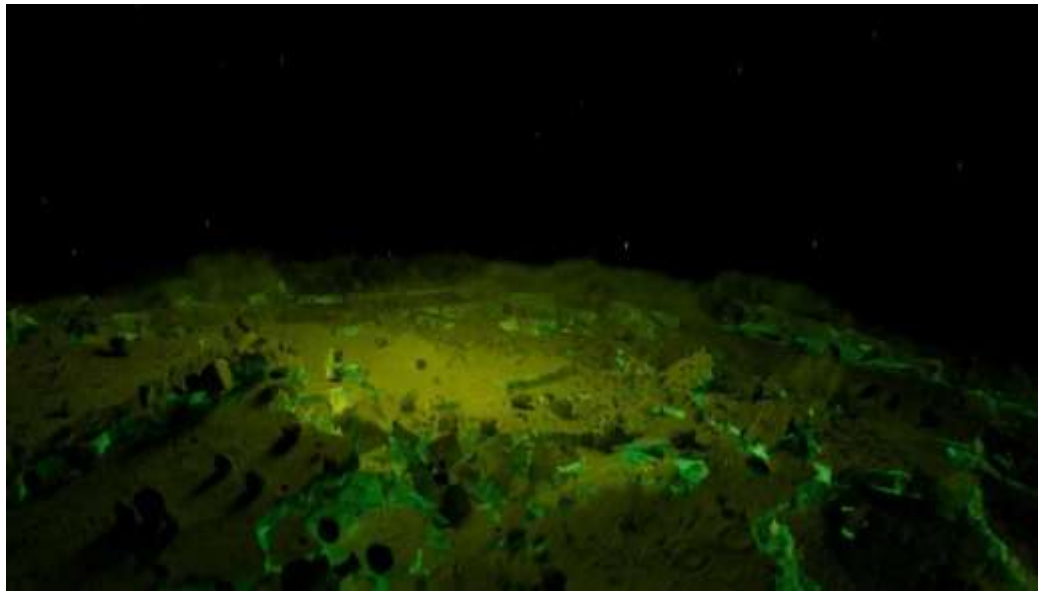
우주의 많은 암석은 수분을 함유하고 있어 이러한 암석에 대한 정보를 알고 있다면 물이 있는 환경을 가진 행성을 찾을 수 있다.

3. 미래의 자원 찾기

많은 우주 암석은 지구에서는 찾기 어려운 가스 및 화합물을 포함하고 있다. 이러한 암석을 사용하여 새로운 기술을 만들 수 있고 로켓 연료를 개발할 수 있을 것이다.



달 암석의 중요성



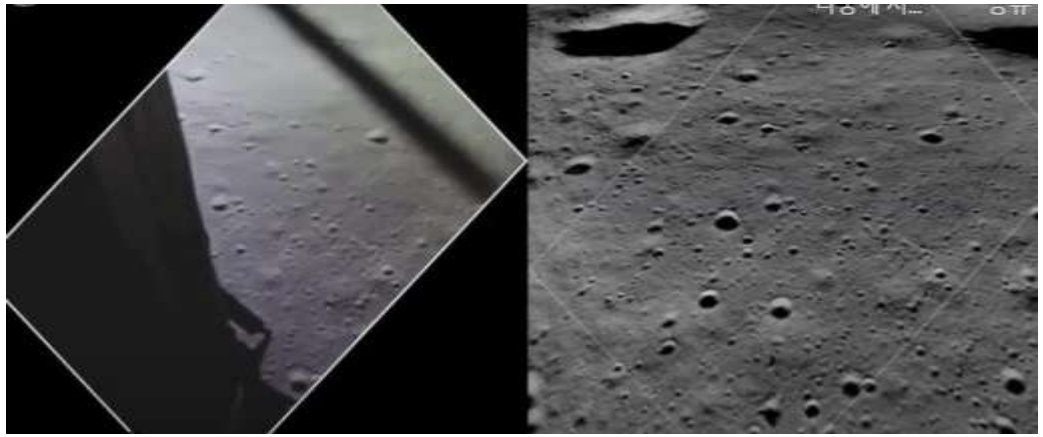
https://docs.microsoft.com/ko-kr/learn/modules/plan- moon-mission-using-python-pandas/2-rock-samples

1. 달 암석의 중요성
- 행성과 달의 형성 방법을 알려준다.
- 미래의 우주 탐사를 위한 많은 정보를 제공한다.

2. 달 암석 연구의 문제점
- 인간이 달 암석을 가져온 지 50년이 지났다
- 가져온 달 암석의 수는 한정되었는데 연구할 사항은 많다.

3. 달에서 새로운 암석 샘플을 가져오는 아르테미스 달 탐사를 준비 하게 되었다.

지금까지의 달 탐사에서 가져온 암석의 총 중량은 383킬로그램이구나

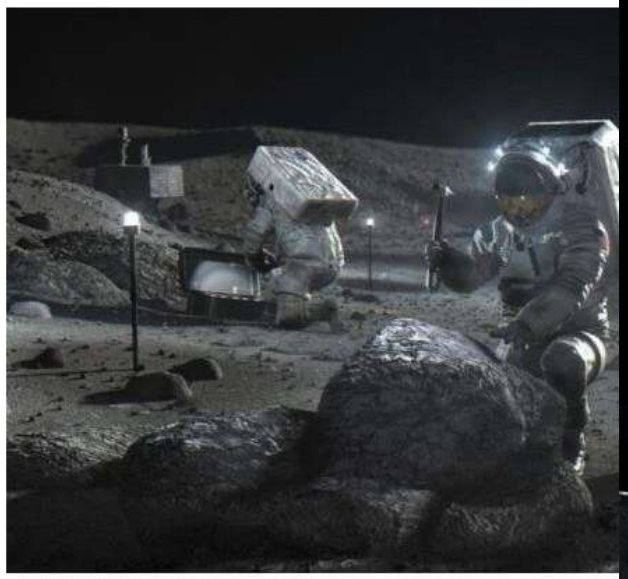


아르테미스 달 탐사

한국도 2024년 미국 달 탐사에 참여한다

곽노필 기자 +구독

아르테미스협정에 10번째 나라로 서명 마쳐
러 "지나치게 미국 중심" 불참...중국은 배제



아르테미스 유인 달탐사 상상도. 나사 제공

http://nuri.hani.co.kr/hanisite/mobile/index.html

1. NASA의 새로운 달 탐사 프로그램
2. 2024년에 최초의 여성 우주 비행사를 달에 보내는 임무가 포함되어 있다.
3. 궁극적으로 화성에 우주 비행사를 보내는 데 필요한 준비 작업을 하는 단계이다.
4. 인간과 AI를 활용한 머신이 은하계를 탐색하고 외계 행성에 오래 머무를 수 있는 방법을 찾아내는 것이다.
5. NASA는 아르테미스 달 탐사를 위해서 풍부한 발사 실험, 다양한 정보 수집 및 가설 테스트 후에 다른 행성을 탐사하기 위한 결론을 얻을 예정이다.

아르테미스 우주 비행사의 임무

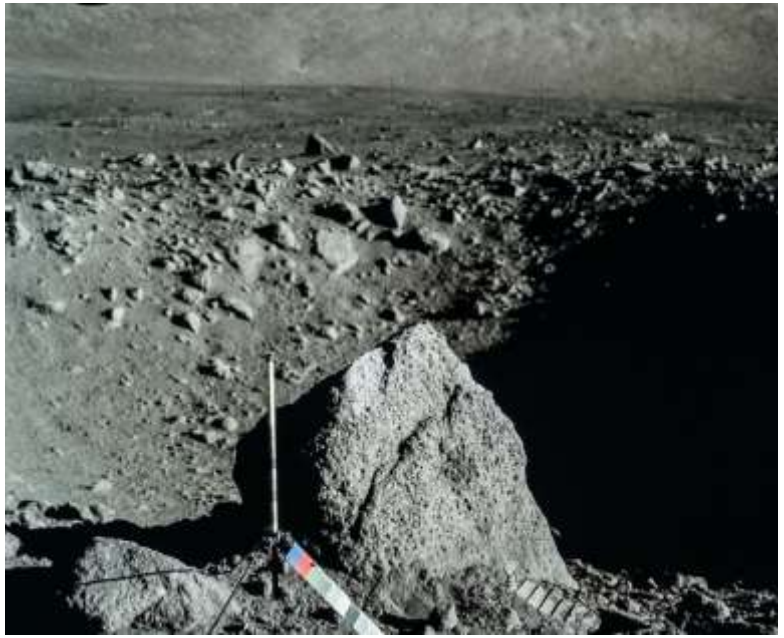
1. 우주 비행사의 임무는 달까지 안전하게 도달하고 달과 태양계 연구를 위한 암석 표본을 채취하는 것이다.

2. 달에 많은 암석은 현무암(Basalt)과 고지대 암석(Highland)이다. 이외에도 다음과 같은 암석이 있다.

전토층 : 충돌하는 개체의 영향으로 분쇄된 암석

각력암 : 부딪힌 다른 암석이 결합된 암석.

전토층과 각력암의 화학적 조성은 현무암과 고지대 암석과 유사하여 육안으로 식별이 힘들다.



이렇게 달 표면이 암석으로 잔뜩 덮여 있으니 우주 비행사가 암석 종류를 한 눈에 식별하는 것은 불가능해.

현무암

하이랜드 암석

https://docs.microsoft.com/ko-kr/learn/modules/research- space-rocks-ai-nasa/4-gather-space-rocks

우주 암석 채취

1. 암석 샘플을 적재할 로켓 공간의 제한성

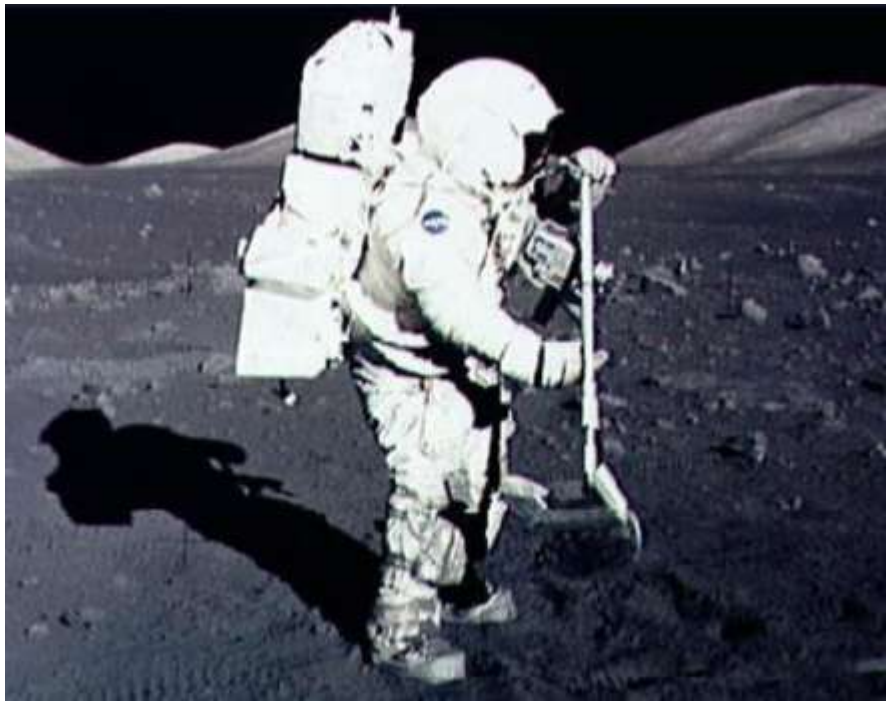
우주 비행사는 필요한 암석 종류에 대해 정확한 중량을 채취해야 한다.

2. 암석 채취의 방법

우주 비행사에게 해당 지역을 정확히 대표할 수 있는 암석 표본을 채취하도록 지시한다.

3. 암석 채취의 어려운 점

- 유사한 화학 조성을 갖는 우주 암석의 종류가 많고 우주 비행사가 육안으로 우주 암석의 종류를 식별할 수 없다.
- 달의 암석은 먼지와 흙으로 덮여 있고 조명은 밝지 않으며 음영이 널리 퍼져 있어 주변과 비슷해 보인다. 달 지표면의 암석은 같은 종류의 암석인 경우에도 실험실에 있는 암석 사진과 다르게 보인다.



https://www.yna.co.kr/view/AKR20200911034500075

문제 정의

아폴로 임무의 데이터 분석을 통해서 2024년 아르테미스 임무로 달에 착륙할 때 우주 비행사가 쉽게 암석 샘 플을 찾는 방법을 찾아보자!

달 암석이 중요해도 로켓에 실을 수 있는 암석 중량에 한계가 있어서 중요한 암석을 필요한 만큼 가져와야 할거야.

해리

우주 비행사가 달에서 암석 채취할 때 필요한 암석과 그 중량을 정확히 채취할 수 있을까?

제니

여섯 번의 아폴로 달 탐사에서 가져온 암석을 데이터 분석해서 우주 비행사에게 필요한 암석 종류와 개수를 알려줄 수 있는지 알아보자.

론

1. 문제정의

2. 데이터 수집

3.데이터 가공

4.데이터 분석

5.시각화 및 탐색

데이터 분석의 5 단계

데이터 분석 단계에 맞추어 달 탐사 및 암석샘플 데이터 분석을 수행한다.

데이터 분석의 5단계

- 달 탐사 및 달 암석 채취시 문제점 파악
- 미래의 달 탐사 및 암석 채취시 개선될 점에 대해서 분석주제 정의

- **암석 샘플 데이터에 관한 사이트를 알아보고 데이터를 수집**
- **깃허브에서 데이터를 다운로드**

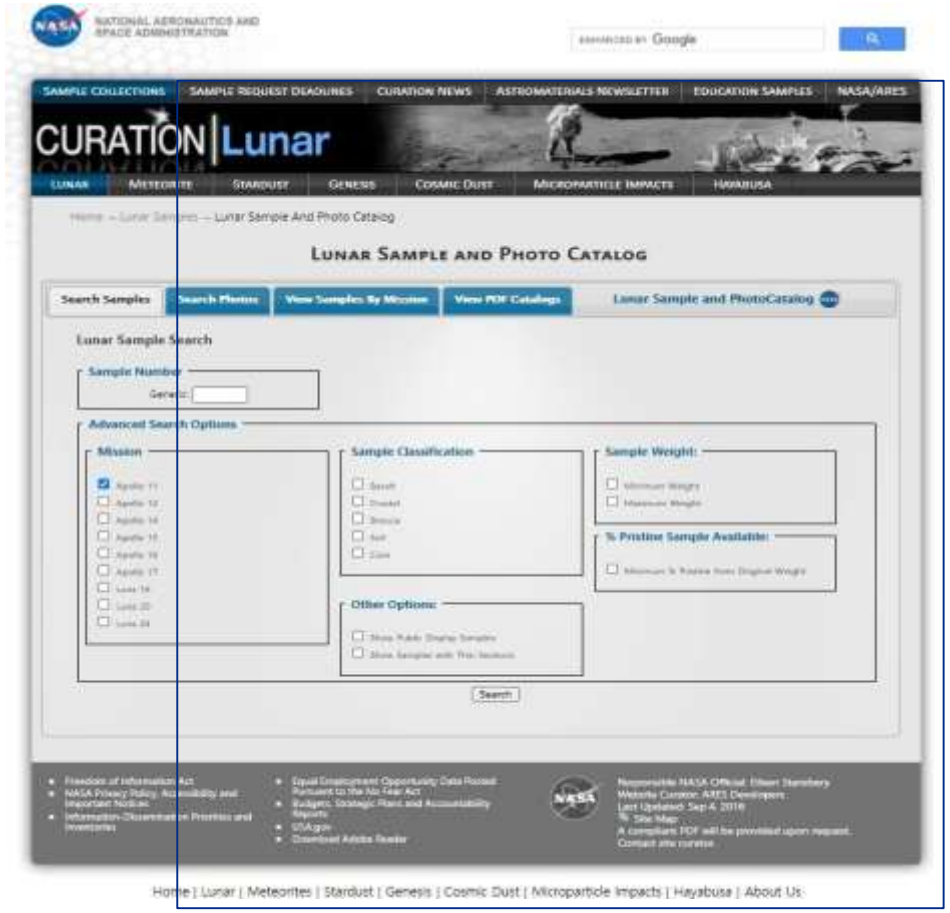
- 암석 샘플 데이터 기초분석하고 필요한 컬럼 추가
- 아폴로 임무별 새로운 데이터 프레임 만들기

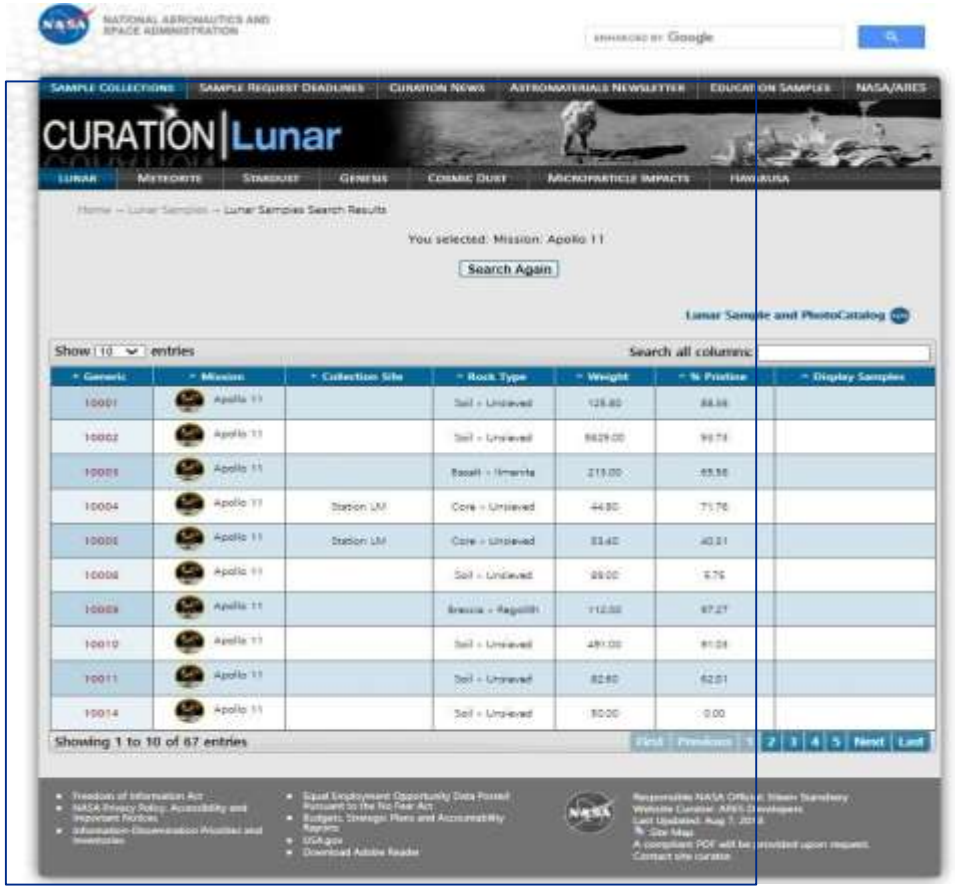
- 아르테미스 임무에서 채취할 암석 중량 구하기
- 수집해야 하는 암석 종류 구하기

- 달에서 수집할 암석 종류를 알려주는 데이터프레임 만들기
- 최종 수집할 암석 종류와 개수 구하기

암석 샘플 데이터 가져오기

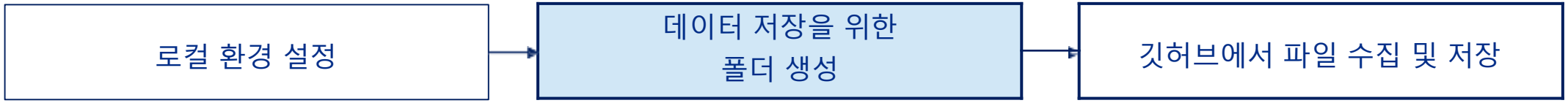
https://curator.jsc.nasa.gov/lunar/samplecatalog/index.cfm 사이트에 들어가서 필요한 컬럼에 체크한다.





데이터 수집 절차

분석에 사용할 데이터 파일을 깃허브에서 수집하여 로컬 폴더에 저장한다.



- 데이터 분석 작업 사전에 파이썬 개발 환경을 설정

- 파이썬
- 비주얼 스튜디오 코드 (Visual Studio Code / VS Code)
- 미니콘다 또는 아나콘다

- 분석에 사용할 데이터 파일을 저장할 로컬 폴더 생성

- 경로명 : OVER-THE-MOON/sample-return/data

- 깃허브에서 파일을 수집 및 로컬 폴더인 OVER-THE-MOON/sample-return/data 에 저장

- 달 탐사 학습 모듈 데이터 (깃허브)

- https://github.com/drguthals/learnwithdrg/tree/main/OverTheMoon/sample-return/data

1. 문제 정의
2. 데이터 수집
3. **데이터 가공**
4. 데이터 분석
5. 시각화 및 탐색

데이터 분석의 5 단계

데이터 분석 단계에 맞추어 달 탐사 및 암석샘플 데이터 분석을 수행한다.

데이터 분석의 5단계

- 달 탐사 및 달 암석 채취시 문제점 파악
- 미래의 달 탐사 및 암석 채취시 개선될 점에 대해서 분석주제 정의

- 문제 정의에 따라서 분석에 필요한 데이터 선택
- 모든 아폴로 달 탐사에서 채취된 암석 샘플 데이터 수집

- **암석 샘플 데이터 읽어들이고 관찰**
- 컬럼 변환
- 아폴로 임무별 데이터프레임 만들기
- 필요한 컬럼추가

- 문제 정의를 반영한 데이터 분석
- 연관될 수 있는 다른 데이터의 수집, 연결 및 분석

- 데이터 분석을 반영할 수 있는 시각화
- 문제 정의에 맞춘 해결방안 도출

여기서 배울 내용은 ?

1.문제정의

2.데이터수집

3.데이터 가공

4.데이터 모델링

5.시각화 및 탐색

단계 1 : 분석할 데이터프레임 읽어 들이고 둘러보기 – rock_samples

단계 2 : 데이터프레임 컬럼 변환하기

단계 3 : 분석주제에 맞는 새로운 데이터프레임 만들기 - missions

단계 4 : missions 데이터프레임에 새로운 컬럼 추가하기

1. 아폴로 임무별로 암석 중량 데이터와 달모듈 중량 데이터 추가

2. 아폴로 임무별로 명령모듈 중량 데이터와 승무원영역 중량 데이터 추가

3. 아폴로 임무별로 승무원영역과 암석샘플이 차지하는 비율 추가

판다스Pandas 라이브러리

데이터 분석에 필요한 판다스 라이브러리를 import 문을 이용하여 읽어 들인다.

import pandas as pd

[판다스 라이브러리]

- 데이터를 수집하고 정리하는 데 유용한 오픈 소스
- 판다스는 시리즈(Series, 1차원 배열)와 데이터프레임(DataFrame, 2차원 배열)이라는 구조화된 데이터 형식을 제공

[파이썬 명령어]

- 라이브러리와 모듈을 가져다 쓰겠다는 명령어

[라이브러리 (library)]

- 다양한 자료형, 함수, 모듈들을 모아놓은 곳
  예) 데이터프레임 : 판다스 내에 있는 자료형
- 파이썬 설치 시 기본적으로 제공

달 탐사에서 고려할 사항

'오버더문' 영화의 달 탐사에서 제시된 문제점은 무엇일까?	예상에 없던 동생 친의 중량!!
'오버더문'의 달 탐사와 아폴로 임무 모두 정확한 중량을 계산하는 것이 가장 중요하다.	아폴로 임무의 달 탐사선도 정확한 중량을 계산해서 만들어졌다
아폴로 임무에서 수집된 암석 샘플들의 종류와 양을 무제한으로 가져올 수 있을까?	달 탐사선과 암석 샘플의 중량을 모두 고려해서 결정해야 한다.
달에서 수집된 암석 샘플은 달 탐사선에 실려 온다.	달 탐사선의 구조와 중량을 알아야 한다.
아폴로 임무의 달 탐사선은 2개의 모듈로 되어 있다.	달 모듈과 명령 모듈의 중량을 구한다.

아폴로 임무에서 수집된 암석 샘플 데이터프레임

아폴로 임무에서 수집된 암석 샘플 데이터프레임 : rock_samples

아폴로 임무에서 수집된 암석 샘플 데이터를 읽어 들인다.	pd.read_csv()
읽어 들인 데이터프레임을 둘러본다.	df.head(), df.tail()
rock_samples 데이터프레임의 정보를 알아본다.	df.info(), 판다스의 자료형
rock_samples 데이터프레임의 요약통계를 알아본다.	df.describe()
rock_samples안의 중량 단위를 'g'에서 'kg'으로 변경한다.	df.apply(), lambda함수, df.rename()

데이터프레임으로 읽어 들이기 : pd.read_csv( )

판다스 라이브러리를 불러온 후 CSV 파일을 읽어서 데이터프레임 rock_samples를 생성한다.

rock_samples = pd.read_csv(파일명)

실제 코딩

작은 따옴표 ' ' 안에 디렉토리를 포함한 파일명

```
rock_samples = pd.read_csv('data/rocksamples.csv')
```

※ 변수 = 읽어들인 데이터프레임을 저장하는 공간

- pd. : 판다스 (as pd) 라이브러리에 속한
- read_csv( ) : 명령으로 CSV 파일을 읽고
- rock_samples 라는 공간 (= 변수) 에 저장함

데이터프레임 rock_samples 둘러보기

읽어들인 데이터프레임을 살펴본다.

rock_samples.head()

❶ 데이터프레임 : 행과 컬럼으로 이루어진 이차원 데이터

열 : 컬럼 ❸

행 : 인덱스 ❷

	ID	Mission	Type	Subtype	Weight (g)	Pristine (%)
0	10001	Apollo11	Soil	Unsieved	125.80	88.36
1	10002	Apollo11	Soil	Unsieved	5629.00	93.73
2	10003	Apollo11	Basalt	Ilmenite	213.00	65.56
3	10004	Apollo11	Core	Unsieved	44.80	71.76
4	10005	Apollo11	Core	Unsieved	53.40	40.31

❹ 값(value)

데이터프레임의 처음과 마지막 5개 행 보여주기

rock_samples.head()

데이터프레임의 처음 5개 행을 보여준다.

```
rock_samples.head()
```

	ID	Mission	Type	Subtype	Weight (g)	Pristine (%)
0	10001	Apollo11	Soil	Unsieved	125.8	88.36
1	10002	Apollo11	Soil	Unsieved	5629.0	93.73
2	10003	Apollo11	Basalt	Ilmenite	213.0	65.56
3	10004	Apollo11	Core	Unsieved	44.8	71.76
4	10005	Apollo11	Core	Unsieved	53.4	40.31

rock_samples.tail()

데이터프레임의 끝의 5개 행을 보여준다.

```
rock_samples.tail()
```

	ID	Mission	Type	Subtype	Weight (g)	Pristine (%)
2224	79528	Apollo17	Breccia	Regolith	2.38	100.0
2225	79529	Apollo17	Breccia	Regolith	1.84	100.0
2226	79535	Apollo17	Breccia	Regolith	1.69	100.0
2227	79536	Apollo17	Breccia	Regolith	1.66	100.0
2228	79537	Apollo17	Breccia	Regolith	1.05	100.0

데이터프레임 관찰하기 : df.info()

rock_samples.info()

```
rock_samples.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2229 entries, 0 to 2228
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   ID            2229 non-null   int64
 1   Mission       2229 non-null   object
 2   Type          2229 non-null   object
 3   Subtype       2226 non-null   object
 4   Weight (g)    2229 non-null   float64
 5   Pristine (%)  2229 non-null   float64
dtypes: float64(2), int64(1), object(3)
memory usage: 104.6+ KB
```

1. 데이터프레임 **전체 행수**를 알려준다 **:rock_samples.shape[0]**
2. 데이터프레임 **index**를 알려준다 **: rock_samples.index**
3. 데이터프레임 **컬럼 개수**를 알려준다 **: rock_samples.shape[1]**
4. 데이터프레임 **컬럼**을 알려준다 **: rock_samples.columns**
5. 데이터프레임의 **각 컬럼의 데이터타입**을 알려준다 **: rock_samples.dtypes**
6. 데이터프레임의 **메모리 사용량을 알려준다.**
7. 데이터프레임의 **전체 행과 컬럼의 수를 알려준다 : rock_samples.shape : (2229, 6)**
   rock_samples.shape[0] : 2229
   rock_samples.shape[1] : 6

데이터프레임 컬럼 살펴보기


```
rock_samples.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2229 entries, 0 to 2228
Data columns (total 6 columns):
ID                2229 non-null int64
Mission           2229 non-null object
Type              2229 non-null object
Subtype           2226 non-null object
Weight (g)        2229 non-null float64
Pristine (%)      2229 non-null float64
dtypes: float64(2), int64(1), object(3)
memory usage: 104.6+ KB
```

- Apollo 임무를 통해 2,229개 샘플이 있다.
- ID - NASA에서 샘플을 추적하는데 사용되는 고유 ID
- Mission – 샘플이 수집된 아폴로 임무
- Type - 암석 샘플 종류
- Subtype – 구체적인 암석 샘플 종류
- Weight (g) – 암석 샘플 중량 (g)
- 'Pristine (%)' - 남아 있는 암석 샘플 백분율 (일부 샘플이 연구에 사용됨).

데이터의 자료형

개별 컬럼은 정수, 실수, 문자, 날짜, 부울값 등의 자료형 (데이터 타입datatype)을 가집니다.

object

- 문자 또는 문자열 (작은 따옴표로 구분) 예 : '월', '화', '중랑센터', '7', '0.5'

int64

- 정수 예 : 0, 1, 5, -14, 21504

float64

- 실수 예 : 0.23, 0.00024, -0.125

datetime64

- 날짜 (작은 따옴표로 구분) 예 : '2019-06-03 08:33:22', '2019-06-26'

boolean

- 참(True) / 거짓(False)을 나타냄 예 : True(1), False(0)

데이터프레임 요약 통계 : df.describe()

rock_samples.describe()

```
rock_samples.describe()
```

	ID	Weight (g)	Pristine (%)
count	2229.000000	2229.000000	2229.000000
mean	52058.432032	168.253024	84.512764
std	26207.651471	637.286458	22.057299
min	10001.000000	0.000000	0.000000
25%	15437.000000	3.000000	80.010000
50%	65527.000000	10.200000	92.300000
75%	72142.000000	93.490000	98.140000
max	79537.000000	11729.000000	180.000000

1. 데이터타입이 숫자인 컬럼의 **전체 갯수**
2. 데이터타입이 숫자인 컬럼의 **평균** 데이터
3. 타입이 숫자인 컬럼의 **표준편차** 데이터타
4. 입이 숫자인 컬럼의 **최소값** 데이터타입이
5. 숫자인 컬럼의 **하위 25% 값** 데이터타입이
6. 숫자인 컬럼의 **하위 50% 값**
7. 데이터타입이 숫자인 컬럼의 **하위 75% 값**
8. 데이터타입이 숫자인 컬럼의 **최대값**

데이터프레임 관찰보기 : 각 컬럼 누락값 검사

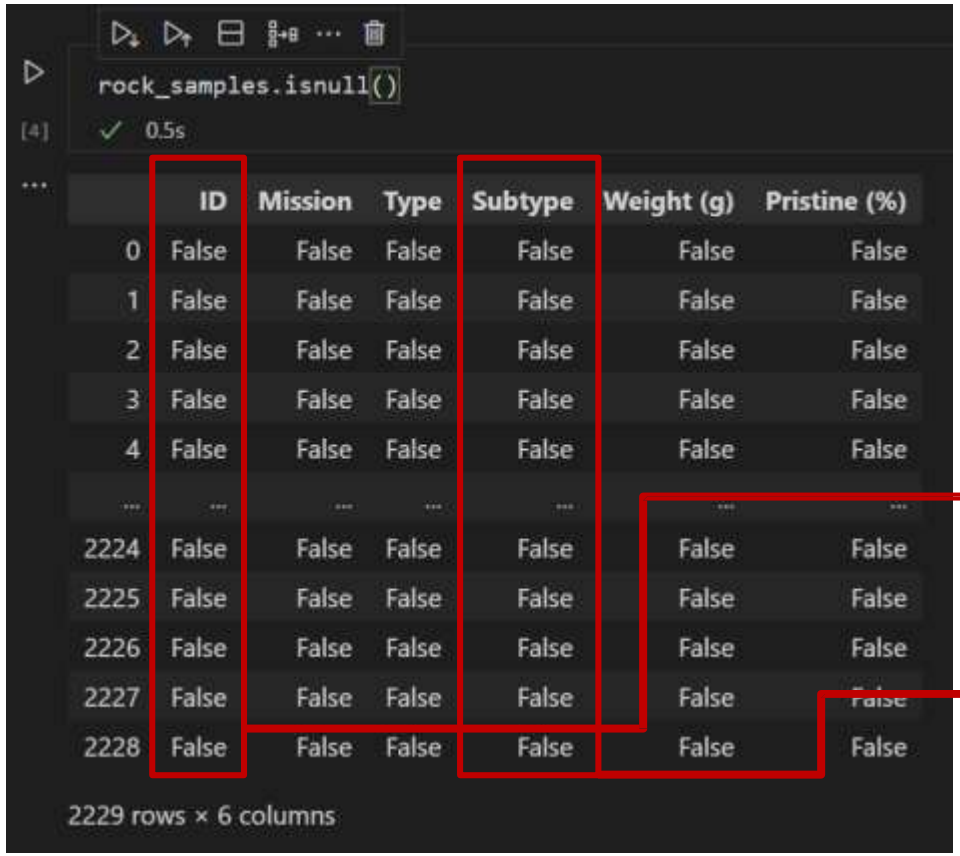
rock_samples.isnull().sum()
bike_ride 데이터프레임의 각 컬럼의 갖는 누락값들의 합을 보여준다.

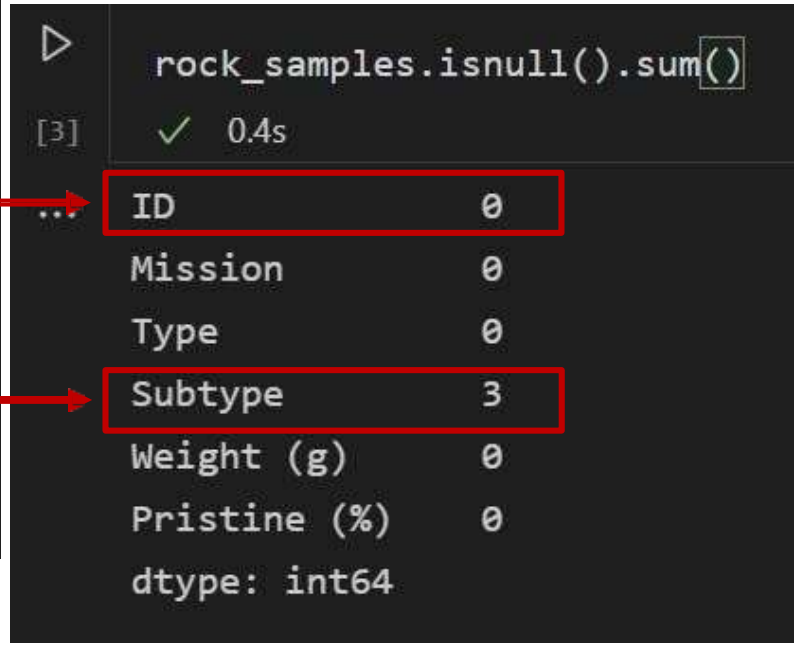
rock_samples.isnull()

- 해당 값이 null이면 True
- 해당 값이 null이 아니면 False

rock_samples.isnull().sum()

- True는 1, False는 0
- sum()을 수행해서 0보다 큰 수가 나오면 이 수가 해당 컬럼에서 null값의 개수가 된다.

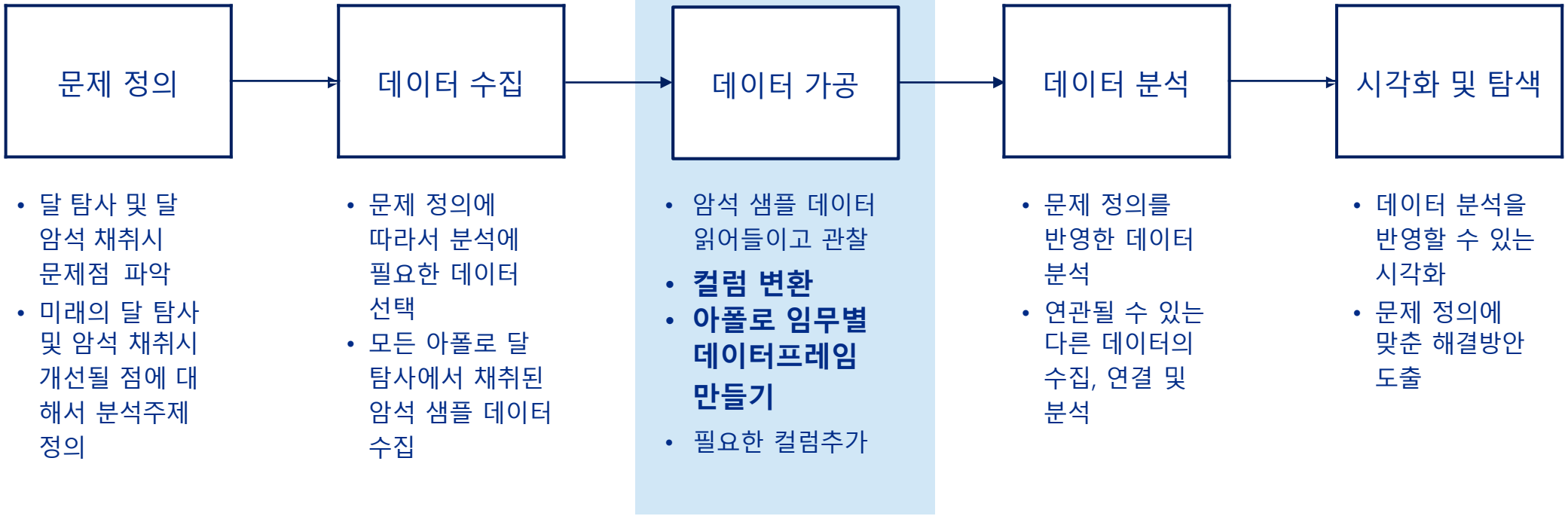




데이터 분석의 5 단계

데이터 분석 단계에 맞추어 달 탐사 및 암석샘플 데이터 분석을 수행한다.

데이터 분석의 5단계



여기서 배울 내용은 ?

1.문제정의

2.데이터수집

3.데이터 가공

4.데이터 모델링

5.시각화 및 탐색

단계 1 : 분석할 데이터프레임 읽어 들이고 둘러보기 – rock_samples

단계 2 : 데이터프레임 컬럼 변환하기

단계 3 : 분석주제에 맞는 새로운 데이터프레임 만들기 - missions

단계 4 : missions 데이터프레임에 새로운 컬럼 추가하기

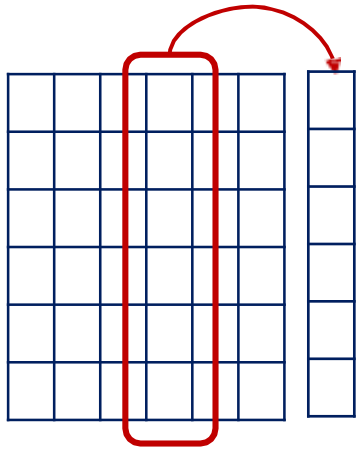
1. 아폴로 임무별로 암석 중량 데이터와 달모듈 중량 데이터 추가

2. 아폴로 임무별로 명령모듈 중량 데이터와 승무원영역 중량 데이터 추가

3. 아폴로 임무별로 승무원영역과 암석샘플이 차지하는 비율 추가

열 단위 데이터 추출 : [ ] 인덱스 연산자

필요한 컬럼명을 대괄호 [ ] 인덱스 연산자에 기술하여 선택할 수 있다. 한 컬럼만 입력하면 시리즈, [[ ]] 이 중으로 선택하면 데이터프레임이 된다.



시리즈 : 하나의 값으로만 구성되는 데이터 구조

시리즈

```
sample_series = rock_samples['Weight (g)']
sample_series
```

✓ 0.4s

```
0        125.80
1       5629.00
2        213.00
3         44.80
4         53.40
        ...
2224       2.38
2225       1.84
2226       1.69
2227       1.66
2228       1.05
Name: Weight (g), Length: 2229, dtype: float64
```

데이터프레임

```
sample_df = rock_samples[['Type', 'Weight (g)']]
sample_df
```

✓ 0.9s

	Type	Weight (g)
0	Soil	125.80
1	Soil	5629.00
2	Basalt	213.00
3	Core	44.80
4	Core	53.40
...	...	...
2224	Breccia	2.38
2225	Breccia	1.84
2226	Breccia	1.69
2227	Breccia	1.66
2228	Breccia	1.05

2229 rows × 2 columns

- 시리즈와 데이터프레임 모두 판다스에서 자주 사용되는 데이터 구조
- 시리즈는 하나의 값으로 구성 되므로 별도로 컬럼이 없음

컬럼에 변환 적용하기 > df.apply ( )

데이터프레임의 컬럼 , 시리즈 또는 데이터프레임 전체에 대해 함수를 적용하게 해주는 명령어이다.

df.apply(함수)

컬럼이나 데이터프레임에 적용할 함수

```
rock_samples['Weight (g)']
```

```
0       125.80
1      5629.00
2       213.00
3        44.80
4        53.40
5        89.00
6       112.00
7       491.00
8        82.60
9        50.00
10        0.40
```

```
rock_samples['Weight (g)'].apply(lambda x : x * 0.001)
```

```
0      0.12580
1      5.62900
2      0.21300
3      0.04480
4      0.05340
5      0.08900
6      0.11200
7      0.49100
8      0.08260
9      0.05000
10     0.00040
```

파이썬 문법 > lambda 함수 – (1)

함수(Function) : 입력값을 받아서 어떤 일을 수행한 뒤 그 결과값을 돌려 주는 구문들의 모음

함수를 사용하는 이유 : 동일한 일을 반복적으로 수행해야 할 때 구문들을 매번 작성할 필요가 없다.

람다lambda 함수 : 함수를 재사용하지 않고 즉시 실행이 필요한 경우 익명 함수일회성 함수를 사용한다.
특히 판다스의 데이터프레임에 새로운 컬럼을 추가하거나 특정 컬럼값을 변형시킬 때 apply 명령어와 결합하여 사용하면 유용하다.

람다lambda 함수의 표현 : lambda 입력값 : 결과값

예) rock_samples['Weight (g)'].apply(lambda x : x * 0.001)

rock_samples['Weight (g)'] 컬럼값들이 처음부터 차례로 x에 입력된다.

입력된 값에 0.001을 곱한 값들이 출력된다.

파이썬 문법 > lambda 함수 – (1)

- 간단한 기능을 일반적인 함수와 같이 정의해두고 쓰는 것이 아니고 필요한 곳에서 즉시 사용하고 버리는 함수임. 따라서 이름을 작성할 필요가 없이 사용 가능함
- 람다 함수는 return 문을 사용하지 않고 반환 값을 만드는 표현식이 있음
- 람다 함수의 몸체는 문이 아닌 하나의 식이다 -> 한줄로 작성
- 람다 함수는 함수를 함수 인자로 넘길 때 유용함
- 람다 함수의 장점은 코드의 간결함으로 이한 메모리의 절약

```
def add( x , y) :
    return x + y
```

```
lambda x , y : x + y
```

lambda <인수들> : <반환할식>

파이썬 문법 > lambda 함수 - (2)

람다lambda 함수의 표현 : lambda 입력값 : 결과값

예) rock_samples['Weight (g)'].apply(lambda x : x * 0.001)

rock_samples['Weight (g)'] 컬럼값들이 처음부터 차례로 x에 입력된다.

입력값에 0.001이 곱해진 값들이 출력된다.

```
rock_samples['Weight (g)']
```

```
0      125.80
1     5629.00
2      213.00
3       44.80
4       53.40
5       89.00
6      112.00
7      491.00
8       82.60
9       50.00
10       0.40
```

중량을 g -> kg으로 변환

```
rock_samples['Weight (g)'].apply(lambda x : x * 0.001)
```

```
0     0.12580
1     5.62900
2     0.21300
3     0.04480
4     0.05340
5     0.08900
6     0.11200
7     0.49100
8     0.08260
9     0.05000
10    0.00040
```

파이썬 문법 > 딕셔너리dictionary

정의 : 일상생활에서 사용하는 사전처럼 이름과 그 이름에 대응되는 내용이 하나의 항목으로 연결되어 있는 파이썬 자료구조이다. 이 때 이름을 키(key), 대응되는 내용을 값(value)라고 한다.

만들기 : 딕셔너리 이름 = { key1 : value1, key2 : value2, ... } 예) dict_ex = {'Weight (g)' : 'Weight (kg)'}
키(key) 값(value)

항목 추출하기 : 딕셔너리 이름[항목의 키]

예) val = dict_ex['Weight (g)']
val에 할당되는 값은 딕셔너리 dict_ex 에서 키인 Weight (g)에 대응되는 값인 Weight(kg)이 할당된다.

파이썬의 딕셔너리는 사전과 아주 비슷하네

컬럼명 변경하기 : df.rename()

rock_samples.rename(columns={ 변경전 컬러명 : 변경후 컬럼명 }, inplace=True)

1 **2** **3**

1. rock_samples.rename : 데이터프레임의 컬럼이름을 바꾸는 명령어
2. columns={ 변경전 컬럼명 : 변경후 컬럼명 } : 컬럼명을 바꾸고 싶으면 columns를 써준다.
   변경전 컬럼명과 변경후 컬럼명을 딕셔너리 형식으로 입력한다.
3. inplace=True : 변경된 내용을 rock_samples 데이터프레임에 고정시킨다.

```
rock_samples.rename(columns={'Weight (g)':'Weight (kg)'}, inplace=True)
rock_samples.head()
```

	ID	Mission	Type	Subtype	Weight (kg)	Pristine (%)
0	10001	Apollo11	Soil	Unsieved	0.13	88.36
1	10002	Apollo11	Soil	Unsieved	5.63	93.73
2	10003	Apollo11	Basalt	Ilmenite	0.21	65.56
3	10004	Apollo11	Core	Unsieved	0.04	71.76
4	10005	Apollo11	Core	Unsieved	0.05	40.31

rock_samples로부터 새로운 데이터프레임 생성

rock_samples 데이터프레임

```
rock_samples.head()
```

	ID	Mission	Type	Subtype	Weight (g)	Pristine (%)
0	10001	Apollo11	Soil	Unsieved	0.13	88.36
1	10002	Apollo11	Soil	Unsieved	5.63	93.73
2	10003	Apollo11	Basalt	Ilmenite	0.21	65.56
3	10004	Apollo11	Core	Unsieved	0.04	71.76
4	10005	Apollo11	Core	Unsieved	0.05	40.31

수집된 암석 샘플마다 하나의 행을 구성

missions 데이터프레임

```
missions
```

	Mission	Sample weight (kg)	Weight diff
0	Apollo11	21.55	0.00
1	Apollo12	34.34	12.79
2	Apollo14	41.83	7.49
3	Apollo15	75.40	33.57
4	Apollo16	92.46	17.06
5	Apollo17	109.44	16.98

아폴로 임무마다 하나의 행을 구성

missions 데이터프레임이 필요한 이유

우리는 각각의 고유 ID를 가진 암석 데이터가 있는 rock_samples 데이터프레임에서 새로운 missions 데이터프레임을 만들 예정입니다. 왜 missions 데이터프레임을 만들어야 할까요 ?

2024년에 발사될 아르테미스 달탐사는 아직 진행중이라서 달탐사선의 정확한 데이터가 없고 모두 예측치만 발표되고 있어.

해리

6번의 아폴로 달탐사에서 가져온 암석 중량, 달탐사선의 중량과 암석이 차지하는 비율은 확정된 값들이야.

제니

아폴로 달탐사의 확정된 값들에 대한 데이터 분석을 사용해서 아르테미스 달탐사에서 필요한 암석 중량과 달탐사선 중량을 예측할 수 있을 것 같아.

론

아폴로 임무별로 요약된 데이터프레임

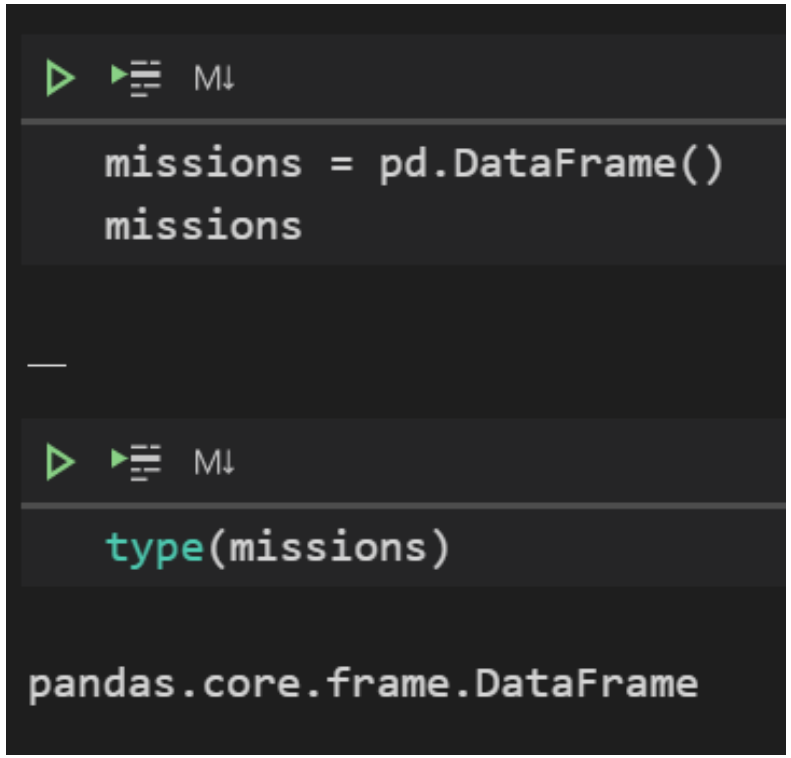
아폴로 임무별로 요약된 데이터프레임 : missions

빈 데이터프레임을 만들고 missions라는 변수에 할당한다.	pd.DataFrame()
rock_samples에 있는 아폴로 임무 6개의 값을 추출한다.	rock_samples['Mission'].unique()
아폴로 임무별로 지구로 가져온 총 샘플 중량 합계를 구한다.	rock_samples.groupby(' ')[' '].sum()
아폴로 임무와 각 임무의 샘플 총 중량 합계를 병합한다.	pd.merge(), df.rename()
아폴로 임무 간의 총 샘플 중량 합계 차이를 구한다.	Series.diff(), Series.fillna()

missions 데이터프레임 만들기

missions = pd.DataFrame()

pd.DataFrame()은 빈 데이터프레임을 만드는 명령어이고
이것을 변수 missions에 할당 만들어진 missions의 타입은 데이터프레임



missions 데이터프레임 만들기

unique()

: 데이터에 고유값들이 어떠한 종류들이 있는지 알고 싶을 때 사용하는 명령어

rock_samples['Mission'].unique()

rock_samples['Mission']의 타입은 시리즈이다.
Series.unique()는 시리즈에서 고유한 값들을 찾아주는 명령어이다.

```
type(rock_samples['Mission'])
```

```
pandas.core.series.Series
```

```
rock_samples['Mission'].unique()
```

```
array(['Apollo11', 'Apollo12', 'Apollo14', 'Apollo15', 'Apollo16',
       'Apollo17'], dtype=object)
```

변수 할당 : = 연산자

= 연산자를 사용하여 pd.read_csv(), pd.DataFrame() 등의 명령어로 생성된 데이터프레임을 새로운 변수에 할당할 수 있다. 분석 주제를 쉽게 표현하는 변수는 이해와 코딩에 편리하다.

새로운 변수 = 생성된 데이터프레임

```
rock_samples = pd.read_csv('data/rocksamples.csv')
```

변수 rock_samples에는 pd.read_csv()로 읽어들인 데이터프레임이 있다.

```
missions = pd.DataFrame()
missions
```

변수 missions에는 빈 데이터프레임이 있다.

데이터프레임에 새로운 컬럼 추가

[ ] 연산자를 사용하여 데이터프레임에 새로운 컬럼을 추가할 수 있다.

데이터프레임['새로운 컬럼이름'] = 값 / 리스트 / 배열(array)

```
rock_samples['Mission'].unique()
```

```
array(['Apollo11', 'Apollo12', 'Apollo14', 'Apollo15', 'Apollo16',
       'Apollo17'], dtype=object)
```

```
missions['Mission'] = rock_samples['Mission'].unique()
missions
```

	Mission
0	Apollo11
1	Apollo12
2	Apollo14
3	Apollo15
4	Apollo16
5	Apollo17

```
missions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 1 columns):
Mission    6 non-null object
dtypes: object(1)
memory usage: 128.0+ bytes
```

여기서 배울 내용은 ?

1.문제정의

2.데이터수집

3.데이터 가공

4.데이터 모델링

5.시각화 및 탐색

단계 1 : 분석할 데이터프레임 읽어 들이고 둘러보기 – rock_samples

단계 2 : 데이터프레임 컬럼 변환하기

단계 3 : 분석주제에 맞는 새로운 데이터프레임 만들기 - missions

단계 4 : missions 데이터프레임에 새로운 컬럼 추가하기

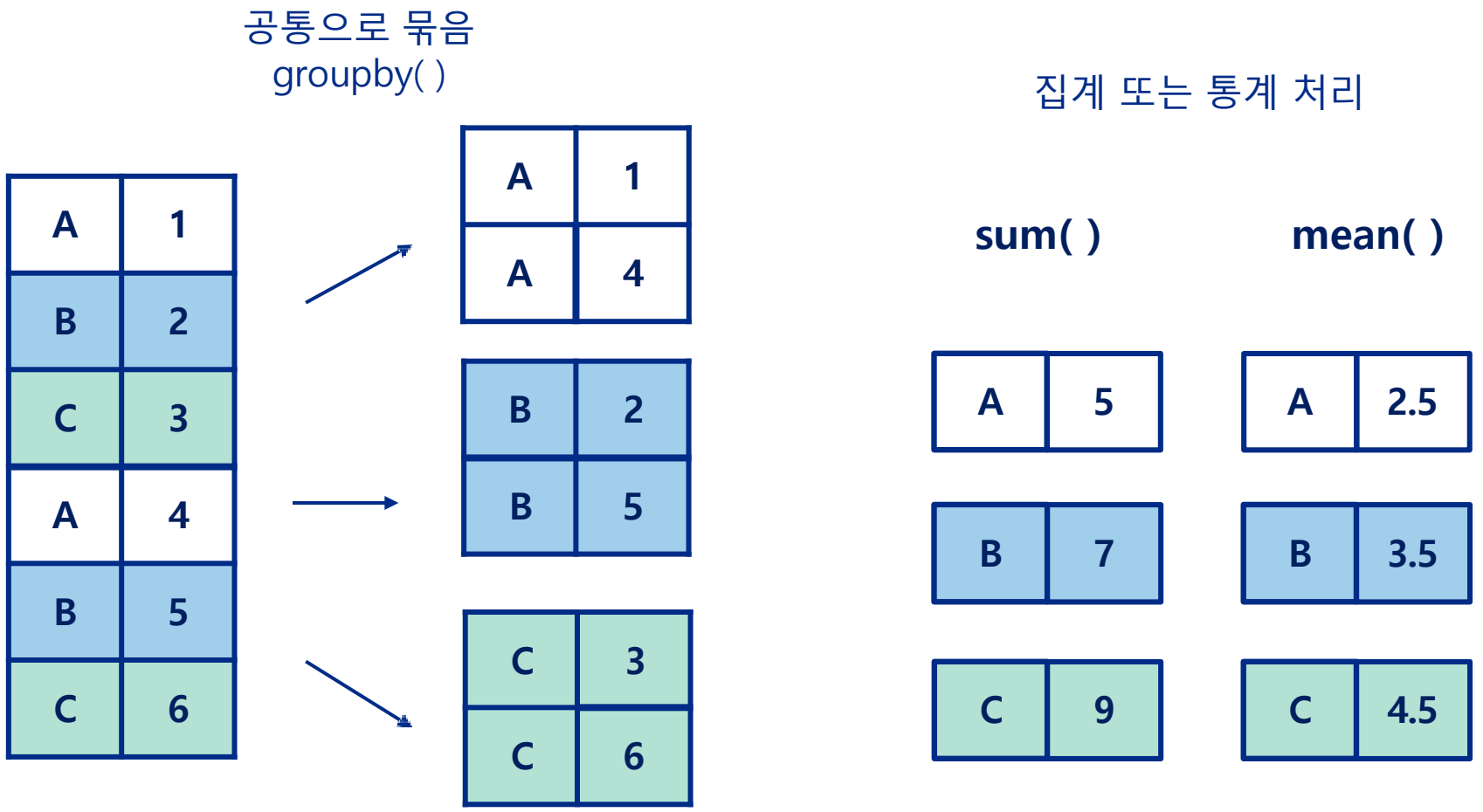
1. 아폴로 임무별로 암석 중량 데이터와 달모듈 중량 데이터 추가

2. 아폴로 임무별로 명령모듈 중량 데이터와 승무원영역 중량 데이터 추가

3. 아폴로 임무별로 승무원영역과 암석샘플이 차지하는 비율 추가

아폴로 임무별 샘플 중량 총합 : groupby – (1)

컬럼의 고유값에 따라 묶어서 집계 또는 통계 처리를 할 때 그룹바이 명령을 적용한다.



아폴로 임무별 샘플 중량 총합 : groupby – (2)

그룹 단위로 아폴로 임무를 사용할 수 있습니다. 그룹 단위로 합계, 평균, 최소, 최대, 개수 등 다양한 집계 및 통계 처리가 가능하다.

```
rock_samples.groupby('Mission' )['Weight (kg)'].sum()
```

그룹별 단위

- Apollo11
- Apollo12
- Apollo14
- Apollo15
- Apollo16
- Apollo17

집계 명령어를 사용할 컬럼

집계 명령어를 사용할 컬럼은

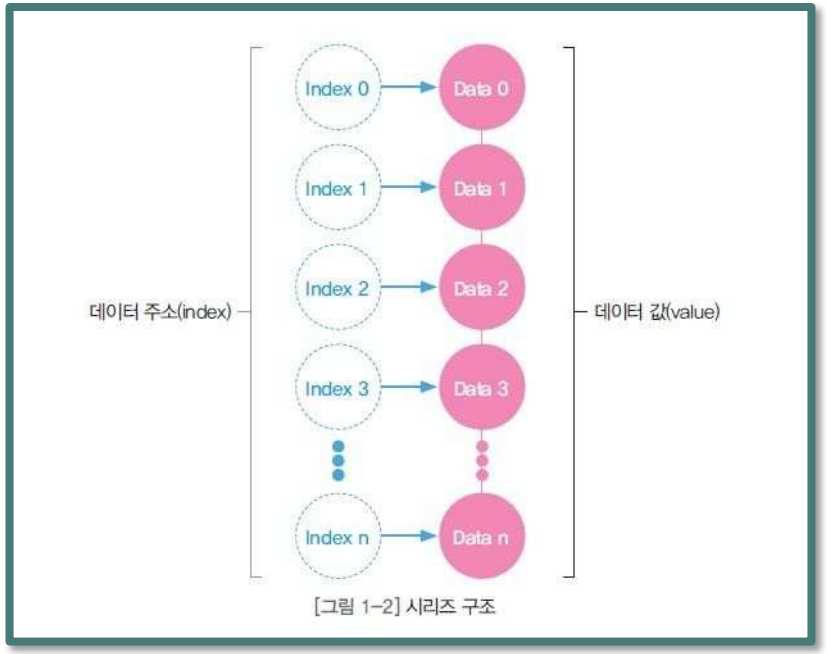
rock_samples['Weight (kg)']

집계 명령어

- sum( )
- mean( )
- min( ), max( ), median( )
- count( )
- first( )

아폴로 임무별 샘플 중량 총합 : groupby – (3)

- 시리즈Series – 데이터가 순차적으로 나열된 1차원 배열
- 인덱스Index와 데이터 값value 가 1:1 대응이 된다.
- 컬럼명이 없다.
- 인덱스Index : 데이터 값의 위치를 나타내는 이름표(데이터주소) 역할을 한다.



[그림 1-2] 시리즈 구조

< 출처: 파이썬 머신러닝 판다스 데이터분석- 한빛출판사>

```
sample_total_weight = rock_samples.groupby('Mission')['Weight (kg)'].sum
()
sample_total_weight
```

```
Mission
Apollo11     21.55
Apollo12     34.34
Apollo14     41.83
Apollo15     75.40
Apollo16     92.46
Apollo17    109.44
Name: Weight (kg), dtype: float64
```

```
type(sample_total_weight)
```

```
pandas.core.series.Series
```

rock_samples.groupby('Mission')['Weight (kg)'].sum()

- 결과 : 시리즈
- 인덱스 : 아폴로 임무
- 값 : 임무별 총 샘플 중량의 총합

데이터프레임과 시리즈 연결하기 : pd.merge()

pd.merge(df, s, on='시리즈 인덱스와 같은 데이터프레임의 컬럼명')

1 **2** **3**

1. **pd.merge** : 데이터프레임의 컬럼명 중에서 시리즈의 인덱스명과 같은 컬럼을 기준으로 데이터프레임과 시리즈를 병합시킨다.
2. **df, s** : 연결할 데이터프레임과 시리즈
3. **on='컬럼명'** : 시리즈의 인덱스명과 같은 데이터프레임의 컬럼명

df.merge(s,on='시리즈 인덱스와 같은데이터프레임의 컬럼명')

```
missions
```

	Mission
0	Apollo11
1	Apollo12
2	Apollo14
3	Apollo15
4	Apollo16
5	Apollo17

```
sample_total_weight
```

```
Mission
Apollo11     21.55
Apollo12     34.34
Apollo14     41.83
Apollo15     75.40
Apollo16     92.46
Apollo17    109.44
Name: Weight (kg), dtype: float64
```

```
missions = pd.merge(missions, sample_total_weight, on='Mission')
missions
```

	Mission	Weight (kg)
0	Apollo11	21.55
1	Apollo12	34.34
2	Apollo14	41.83
3	Apollo15	75.40
4	Apollo16	92.46
5	Apollo17	109.44

컬럼명 변경하기 : df.rename()

missions.rename(columns={ 변경전 컬럼명 : 변경후 컬럼명 }, inplace=True)

1 **2** **3**

1. missions.rename : 데이터프레임의 컬럼이름을 바꾸는 명령어
2. columns={ 변경전 컬럼명 : 변경후 컬럼명 }: 컬럼명을 바꾸고 싶으면 columns를 써준다

   변경전 컬럼명과 변경후 컬럼명을 딕셔너리 형식으로 입력한다.
3. inplace=True : 변경된 내용을 missions 데이터프레임에 고정시킨다.

```
missions.rename(columns={'Weight (kg)':'Sample weight (kg)'}, inplace=True)
missions
```

[15] ✓ 0.4s

	Mission	Sample weight (kg)
0	Apollo11	21.55424
1	Apollo12	34.34238
2	Apollo14	41.83363
3	Apollo15	75.39910
4	Apollo16	92.46262
5	Apollo17	109.44402

임무간 중량 차이 구하기

Series.diff()

- 시리즈의 값들에서 그 이전 값과 차이를 구한다.
- 시리즈의 맨처음 값은 이전 값이 없기 때문에 nan(not a number)값이 된다.

```
missions['Weight diff'] = missions['Sample weight (kg)'].diff()
missions
```

	Mission	Sample weight (kg)	Weight diff
0	Apollo11	21.55	nan
1	Apollo12	34.34	12.79
2	Apollo14	41.83	7.49
3	Apollo15	75.40	33.57
4	Apollo16	92.46	17.06
5	Apollo17	109.44	16.98

df.fillna(value='채울값', inplace=True)

- 데이터프레임에서 nan값들을 특정 값으로 채워준다.
- Value에 채울값을 써주고 inplace=True를 하면 데이터프레임에 변경된 것이 고정된다.

```
missions.fillna(value=0, inplace=True)
missions
```

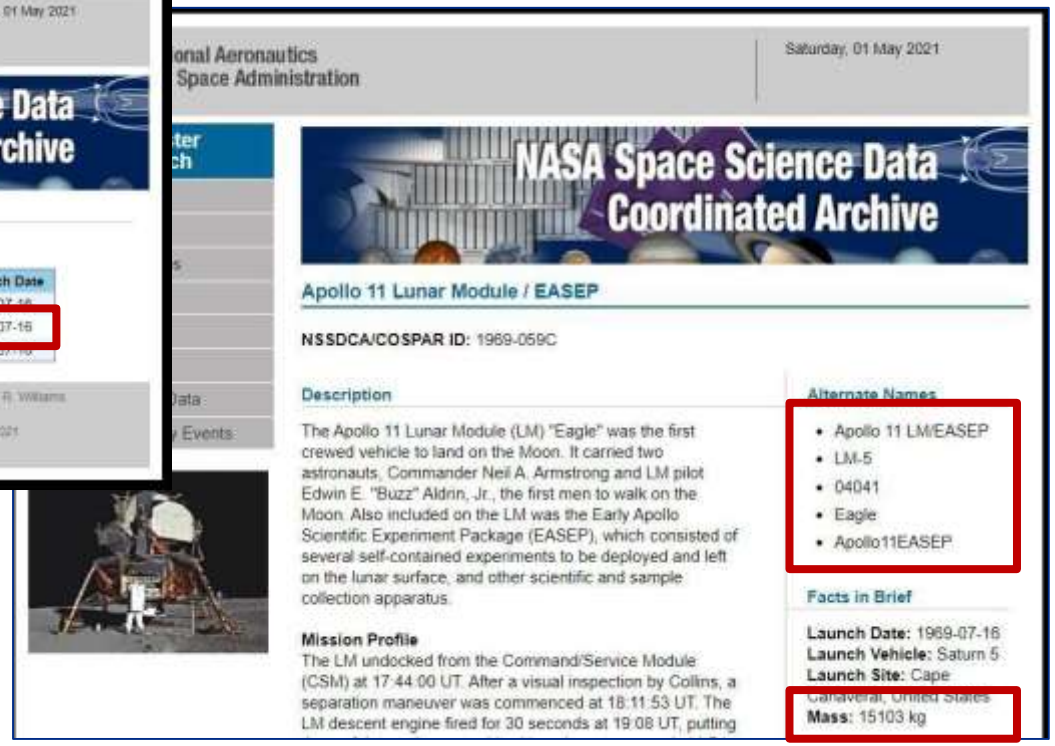
	Mission	Sample weight (kg)	Weight diff
0	Apollo11	21.55	0.00
1	Apollo12	34.34	12.79
2	Apollo14	41.83	7.49
3	Apollo15	75.40	33.57
4	Apollo16	92.46	17.06
5	Apollo17	109.44	16.98

달 탐사선 중량 데이터 추가

https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1969-059C 사이트에 들어가서 달 모듈과 명령 모듈의 데이터를 수집한다.

Name: 칸에 Apollo11을 입력하고 'enter'를 친다.

Apollo11 Lunar Module를 클릭한다.



붉은색 칸에 있는 모듈명과 중량데이터를 얻는다.

달 탐사선 중량 데이터 추가

Apollo 달탐사 임무에 사용된 Saturn V 로켓에는 두 가지 모듈이 있다.

달 모듈 (Lunar Module) : 달궤도에 도달한 후 명령 모듈에서 분리되는 모듈로서 달 표면에 착륙하고 우주 비행사를 수송한다. 우주비행사는 달에서 수집된 암석 샘플을 여기에 싣고 명령 모듈로 귀환한다.

명령 모듈 (Command Module) : 우주 비행사가 생활하는 모듈로서 우주비행사와 수집된 암석 샘플이 이 모듈에 실려 지구로 귀환한다.

승무원 영역 Crewed Area

암석이 실리는 부분인 달 탐사선 중량도 고려해야겠네.

새로운 정보로 추가

missions 데이터프레임에 아폴로 임무별로 달 모듈과 명령 모듈의 정보를 추가해야 달 탐사선을 위한 정확한 중량 계산이 가능해진다.

달 탐사선 중량 데이터 추가

missions 데이터프레임에 다음의 데이터를 가진 새로운 컬럼을 추가한다.

파이썬 문법 > 리스트list

정의 : 여러 데이터들을 잘 관리하기 위해서 묶어서 목록으로 관리할 수 있는 파이썬 자료형

만들기 : 리스트의 이름 = [항목1, 항목2, ...]

예) 달모듈 = ['Eagle', 'Intrepid', 'Antares', 'Falcon', 'Orion', 'Challenger']

인덱싱 : 리스트에 있는 여러 항목들은 모두 각각 그 위치가 0부터 시작하는 숫자로 매겨져 있다.

예) 달모듈 = ['Eagle', 'Intrepid', 'Antares', 'Falcon', 'Orion', 'Challenger']

인덱스 -> 0 1 2 3 4 5

리스트내의 항목 추출하기 : 리스트의 이름[항목의 인덱스]

예) 달모듈 = ['Eagle', 'Intrepid', 'Antares', 'Falcon', 'Orion', 'Challenger']

인덱스 -> 0 1 2 3 4 5

달모듈[0] = 'Eagle', 달모듈[1] = 'Intrepid', 달모듈[2] = 'Antares',

달 탐사선 중량 데이터 추가 : 달 모듈

달 모듈의 이름과 중량 데이터를 missions 데이터프레임에 새로운 컬럼으로 추가한다. 앞에서 배운 diff()과 fillna() 명령어를 사용해서 달 모듈간 중량 차이를 나타내는 컬럼을 추가한다.

```
missions['Lunar module (LM)'] = ['Eagle (LM-5)', 'Intrepid (LM-6)',
'Antares (LM-8)', 'Falcon (LM-10)', 'Orion (LM-11)', 'Challenger (LM-12)
']
missions['LM mass (kg)'] = [15103, 15235, 15264, 16430, 16445, 16456]
missions['LM mass diff'] = missions['LM mass (kg)'].diff()
missions['LM mass diff'] = missions['LM mass diff'].fillna(value=0)
```

missions

	Mission	Sample weight (kg)	Weight diff	Lunar module (LM)	LM mass (kg)	LM mass diff
0	Apollo11	0.021554	0.000000	Eagle (LM-5)	15103	0.0
1	Apollo12	0.034342	0.012788	Intrepid (LM-6)	15235	132.0
2	Apollo14	0.041834	0.007491	Antares (LM-8)	15264	29.0
3	Apollo15	0.075399	0.033565	Falcon (LM-10)	16430	1166.0
4	Apollo16	0.092463	0.017064	Orion (LM-11)	16445	15.0
5	Apollo17	0.109444	0.016981	Challenger (LM-12)	16456	11.0

임무간 달모듈 중량 차이 구하기

Series.diff()

- 시리즈의 값들에서 그 이전 값과 차이를 구한다.
- 시리즈의 맨처음 값은 이전 값이 없기 때문에 nan(not a number)값이 된다.

```
missions['LM mass diff'] = missions['LM mass (kg)'].diff()
missions
```

✓ 0.3s

	Mission	Sample_Weight (kg)	weight_diff	Lunar Module (LM)	LM mass (kg)	LM mass diff
0	Apollo11	21.55424	0.00000	Eagle (LM-5)	15103	NaN
1	Apollo12	34.34238	12.78814	Intrepid (LM-6)	15235	132.0
2	Apollo14	41.83363	7.49125	Antares (LM-8)	15264	29.0
3	Apollo15	75.39910	33.56547	Falcon (LM-10)	16430	1166.0
4	Apollo16	92.46262	17.06352	Orion (LM-11)	16445	15.0
5	Apollo17	109.44402	16.98140	Challenger (LM-12)	16456	11.0

df.fillna(value='채울값', inplace=True)

- 데이터프레임에서 nan값들을 특정 값으로 채워준다.
- value에 채울값을 써주고 inplace=True를 하면 데이터프레임에 변경된 것이 고정된다.

```
missions.fillna(value=0, inplace=True)
missions
```

✓ 0.5s

	Mission	Sample_Weight (kg)	weight_diff	Lunar Module (LM)	LM mass (kg)	LM mass diff
0	Apollo11	21.55424	0.00000	Eagle (LM-5)	15103	0.0
1	Apollo12	34.34238	12.78814	Intrepid (LM-6)	15235	132.0
2	Apollo14	41.83363	7.49125	Antares (LM-8)	15264	29.0
3	Apollo15	75.39910	33.56547	Falcon (LM-10)	16430	1166.0
4	Apollo16	92.46262	17.06352	Orion (LM-11)	16445	15.0
5	Apollo17	109.44402	16.98140	Challenger (LM-12)	16456	11.0

여기서 배울 내용은 ?

1.문제정의

2.데이터수집

3.데이터 가공

4.데이터 모델링

5.시각화 및 탐색

단계 1 : 분석할 데이터프레임 읽어 들이고 둘러보기 – rock_samples

단계 2 : 데이터프레임 컬럼 변환하기

단계 3 : 분석주제에 맞는 새로운 데이터프레임 만들기 - missions

단계 4 : missions 데이터프레임에 새로운 컬럼 추가하기

1. 아폴로 임무별로 암석 중량 데이터와 달모듈 중량 데이터 추가

2. 아폴로 임무별로 명령모듈 중량 데이터와 승무원영역 중량 데이터 추가

3. 아폴로 임무별로 승무원영역과 암석샘플이 차지하는 비율 추가

달 탐사선 중량 데이터 추가 : 명령 모듈

명령 모듈의 이름과 중량 데이터를 missions 데이터프레임에 새로운 컬럼으로 추가한다. 앞에서 배운 diff()과 fillna() 명령어를 사용해서 명령 모듈간 중량 차이를 나타내는 컬럼을 추가한다.

```
missions['Command module (CM)'] = ['Columbia (CSM-107)', 'Yankee
Clipper (CM-108)', 'Kitty Hawk (CM-110)', 'Endeavor (CM-112)', 'Casper
(CM-113)', 'America (CM-114)']
missions['CM mass (kg)'] = [5560, 5609, 5758, 5875, 5840, 5960]
missions['CM mass diff'] = missions['CM mass (kg)'].diff()
missions['CM mass diff'] = missions['CM mass diff'].fillna(value=0)
```

missions

	Mission	Sample weight (kg)	Weight diff	Lunar module (LM)	LM mass (kg)	LM mass diff	Command module (CM)	CM mass (kg)	CM mass diff
0	Apollo11	0.021554	0.000000	Eagle (LM-5)	15103	0.0	Columbia (CSM-107)	5560	0.0
1	Apollo12	0.034342	0.012788	Intrepid (LM-6)	15235	132.0	Yankee Clipper (CM-108)	5609	49.0
2	Apollo14	0.041834	0.007491	Antares (LM-8)	15264	29.0	Kitty Hawk (CM-110)	5758	149.0
3	Apollo15	0.075399	0.033565	Falcon (LM-10)	16430	1166.0	Endeavor (CM-112)	5875	117.0
4	Apollo16	0.092463	0.017064	Orion (LM-11)	16445	15.0	Casper (CM-113)	5840	-35.0
5	Apollo17	0.109444	0.016981	Challenger (LM-12)	16456	11.0	America (CM-114)	5960	120.0

임무간 명령모듈 중량 차이 구하기

Series.diff()

- 시리즈의 값들에서 그 이전 값과 차이를 구한다.
- 시리즈의 맨처음 값은 이전 값이 없기 때문에 nan(not a number)값이 된다.

```
missions['Command mass diff'] = missions['Command mass (kg)'].diff()
missions
```

✓ 0.5s Python

	Mission	Sample_Weight (kg)	weight_diff	Lunar Module (LM)	LM mass (kg)	LM mass diff	Command module (CM)	Command mass (kg)	Command mass diff
0	Apollo11	21.55424	0.00000	Eagle (LM-5)	15103	0.0	Columbia (CSM-107)	5560	NaN
1	Apollo12	34.34238	12.78814	Intrepid (LM-6)	15235	132.0	Yankee Clipper (CM-108)	5609	49.0
2	Apollo14	41.83363	7.49125	Antares (LM-8)	15264	29.0	Kitty Hawk (CM-110)	5758	149.0
3	Apollo15	75.39910	33.56547	Falcon (LM-10)	16430	1166.0	Endeavor (CM-112)	5875	117.0
4	Apollo16	92.46262	17.06352	Orion (LM-11)	16445	15.0	Casper (CM-113)	5840	-35.0
5	Apollo17	109.44402	16.98140	Challenger (LM-12)	16456	11.0	America (CM-114)	5960	120.0

df.fillna(value='채울값', inplace=True)

- 데이터프레임에서 nan값들을 특정 값으로 채워준다.
- value에 채울값을 써주고 inplace=True를 하면 데이터프레임에 변경된 것이 고정된다.

```
missions.fillna(value=0, inplace=True)
missions
```

✓ 0.1s Python

	Mission	Sample_Weight (kg)	weight_diff	Lunar Module (LM)	LM mass (kg)	LM mass diff	Command module (CM)	Command mass (kg)	Command mass diff
0	Apollo11	21.55424	0.00000	Eagle (LM-5)	15103	0.0	Columbia (CSM-107)	5560	0.0
1	Apollo12	34.34238	12.78814	Intrepid (LM-6)	15235	132.0	Yankee Clipper (CM-108)	5609	49.0
2	Apollo14	41.83363	7.49125	Antares (LM-8)	15264	29.0	Kitty Hawk (CM-110)	5758	149.0
3	Apollo15	75.39910	33.56547	Falcon (LM-10)	16430	1166.0	Endeavor (CM-112)	5875	117.0
4	Apollo16	92.46262	17.06352	Orion (LM-11)	16445	15.0	Casper (CM-113)	5840	-35.0
5	Apollo17	109.44402	16.98140	Challenger (LM-12)	16456	11.0	America (CM-114)	5960	120.0

데이터프레임DataFrame > 산술연산

데이터프레임과 데이터프레임의 산술연산은 각 데이터프레임의 같은 행, 같은 컬럼 위치에 있는 원소끼리 계산합니다.

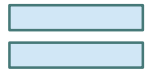
데이터프레임의 연산자 활용 : df1 + 산술연산자(+, -, *, /) + df2

df1

	0	1
0	32	17.25
1	48	81.2833
2	36	17.9250
3	45	63.1000
4	45	18.0500

df2

	0	1
0	10	10
1	20	20
2	30	NaN
3	NaN	40
4	50	50



df3

	0	1
0	42	27.25
1	68	101.283
2	66	NaN
3	NaN	103.1000
4	95	68.0500

승무원영역 중량 데이터 추가 : 달 모듈 + 명령 모듈

승무원영역은 달 모듈과 명령 모듈을 합한 것이다. 따라서 달 모듈과 명령 모듈의 중량 합과 중량 차이의 합을 나타내는 새로운 컬럼을 missions 데이터프레임에 추가한다.

```
missions['Total weight (kg)'] = missions['LM mass (kg)'] + missions['CM mass (kg)']
missions['Total weight diff'] = missions['LM mass diff'] + missions['CM mass diff']
```

승무원영역의 중량은 달모듈과 명령모듈의 중량을 합한 것이야.

M↓

```
missions
```

Mission	Sample weight (kg)	Weight diff	Lunar module (LM)	LM mass (kg)	LM mass diff	Command module (CM)	CM mass (kg)	CM mass diff	Total weight (kg)	Total weight diff
Apollo11	0.021554	0.000000	Eagle (LM-5)	15103	0.0	Columbia (CSM-107)	5560	0.0	20663	0.0
Apollo12	0.034342	0.012788	Intrepid (LM-6)	15235	132.0	Yankee Clipper (CM-108)	5609	49.0	20844	181.0
Apollo14	0.041834	0.007491	Antares (LM-8)	15264	29.0	Kitty Hawk (CM-110)	5758	149.0	21022	178.0
Apollo15	0.075399	0.033565	Falcon (LM-10)	16430	1166.0	Endeavor (CM-112)	5875	117.0	22305	1283.0
Apollo16	0.092463	0.017064	Orion (LM-11)	16445	15.0	Casper (CM-113)	5840	-35.0	22285	-20.0
Apollo17	0.109444	0.016981	Challenger (LM-12)	16456	11.0	America (CM-114)	5960	120.0	22416	131.0

페이로드Payload

페이로드는 로켓 안에 실리는 물건의 하중을 말한다. 우주선 안에 실리는 물건으로 화물, 승무원, 과학 장비, 실험 장치 등이 있다.

달모듈과 명령모듈을 합친 승무원 영역은 페이로드에 속하는구 나.

제니

암석 샘플이 실리는 곳도 승무원 영역이 면서 페이로드에 속해.

론

그럼 페이로드에서 승무원 영역이 차지하는 비율 과 샘플이 차지하는 비 율을 구해볼까?

해리

NASA SLS와 Orion Modules

NASA Factsheet on the Space Launch System(SLS)와 Orion Modules 의 일부 정보를 사용하여 중량 및 페이로드 관련 데이터를 수집한다.

데이터프레임DataFrame > 산술연산

데이터프레임의 모든 원소에 숫자를 산술연산 할 수 있습니다.

데이터프레임의 연산자 활용 : df + 산술연산자(+, -, *, /) + 숫자

df

	0	1
	age	fare
0	32	17.25
1	48	81.2833
2	36	17.9250
3	45	63.1000
4	45	18.0500

```
df + 10
```

	age	fare
0	42.0	27.2500
1	58.0	91.2833
2	46.0	27.9250
3	55.0	73.1000
4	55.0	28.0500

페이로드에서 승무원 영역이 차지하는 비율 구하기

아폴로 임무에 사용된 Saturn V 페이로드는 43,500 kg이고 모듈 중량은 임무마다 달랐다.

missions 데이터프레임에서 달 모듈과 명령 모듈을 '승무원 영역'으로 지칭했다. 우주선에서 승무원이 있을 수 있는 부분이고 샘플도 여기에 실릴 가능성이 높기 때문이다.

missions 데이터프레임에 페이로드 대한 승무원 영역('Crewed area : Payload') 컬럼을 다음과 같은 비율계산으로 추가한다. 이것은 페이로드에서 승무원 영역이 차지하는 비율을 의미한다.

```
# Sample-to-weight ratio
saturnVPayload = 43500
missions['Crewed area : Payload'] = missions['Total weight (kg)'] /
saturnVPayload
```

샘플이 차지하는 비율 구하기

샘플은 승무원 영역에 실릴 가능성이 높으므로 missions 데이터프레임에 승무원 영역에 대한 샘플 중량 비율을 구하고 이 데이터를 missions 데이터프레임에 추가한다. 이것은 승무원 영역에서 샘플이 차지하는 비율을 의미한다.

```
missions['Sample : Crewed area'] = missions['Sample weight (kg)'] /
missions['Total weight (kg)']
```

페이로드에 대한 샘플 중량 비율도 구하고 이 데이터를 missions 데이터프레임에 추가한다. 이것은 페이로드에서 샘플 중량이 차지하는 비율을 의미한다.

```
missions['Sample : Payload'] = missions['Sample weight (kg)'] /
saturnVPayload
```

완성된 missions 데이터프레임

missions

Lunar module (LM)	LM mass (kg)	LM mass diff	Command module (CM)	CM mass (kg)	CM mass diff	Total weight (kg)	Total weight diff	Crewed area : Payload	Sample : Crewed area	Sample : Payload
Eagle (LM-5)	15103	0.00	Columbia (CSM-107)	5560	0.00	20663	0.00	0.48	0.00	0.00
Intrepid (LM-6)	15235	132.00	Yankee Clipper (CM-108)	5609	49.00	20844	181.00	0.48	0.00	0.00
Antares (LM-8)	15264	29.00	Kitty Hawk (CM-110)	5758	149.00	21022	178.00	0.48	0.00	0.00
Falcon (LM-10)	16430	1166.00	Endeavor (CM-112)	5875	117.00	22305	1283.00	0.51	0.00	0.00
Orion (LM-11)	16445	15.00	Casper (CM-113)	5840	-35.00	22285	-20.00	0.51	0.00	0.00
Challenger (LM-12)	16456	11.00	America (CM-114)	5960	120.00	22416	131.00	0.52	0.00	0.00

missions 데이터프레임 내용

- 암석샘플 중량 데이터
- 달 모듈과 명령 모듈의 중량 데이터
- 승무원영역 중량 데이터
- 승무원영역 / 페이로드 비율
- 암석샘플 / 승무원영역 비율
- 암석샘플 / 페이로드 비율

이제 드디어 데이터 가공이 끝났다!!!

나 지금 어느 단계를 공부하는 거지?

1.문제정의

2.데이터수집

3.데이터 가공

4.데이터 모델링

5.시각화 및 탐색

단계 1 : 분석할 데이터프레임 읽어 들이고 둘러보기 – rock_samples

단계 2 : 데이터프레임 컬럼 변환하기

단계 3 : 분석주제에 맞는 새로운 데이터프레임 만들기 - missions

단계 4 : missions 데이터프레임에 새로운 컬럼 추가하기

1. 아폴로 임무별로 암석 중량 데이터와 달모듈 중량 데이터 추가

2. 아폴로 임무별로 명령모듈 중량 데이터와 승무원영역 중량 데이터 추가

3. 아폴로 임무별로 승무원영역과 암석샘플이 차지하는 비율 추가

1. 문제 정의
2. 데이터 수집
3. 데이터 가공
4. **데이터 분석**
5. 시각화 및 탐색

데이터 분석의 5 단계

데이터 분석 단계에 맞추어 달 탐사 및 암석샘플 데이터 분석을 수행한다.

데이터 분석의 5단계

- 달 탐사 및 달 암석 채취시 문제점 파악
- 미래의 달 탐사 및 암석 채취시 개선될 점에 대해서 분석주제 정의

- 문제 정의에 따라서 분석에 필요한 데이터 선택
- 모든 아폴로 달 탐사에서 채취된 암석 샘플 데이터 수집

- 암석 샘플 데이터 기초분석하고 필요한 컬럼 추가
- 아폴로 임무별 새로운 데이터 프레임 만들기

- **아르테미스 임무에서 채취할 암석 중량 구하기**
- 수집해야 하는 암석 종류 구하기

- 데이터 분석을 반영할 수 있는 시각화
- 문제 정의에 맞춘 해결방안 도출

여기서 배울 내용은 ?

1.문제정의

2.데이터수집

3.데이터 가공

4.데이터 분석

5.시각화 및 탐색

단계 1 : 아르테미스 임무 데이터프레임 만들기 : artemis_mission

단계 2 : 아르테미스 임무에서 수집할 수 있는 예상 암석샘플 중량 구하기

artemis_mission['Estimated sample weight (kg)']

단계 3 : 수집해야 하는 암석샘플 데이터프레임 만들기

low_samples, needed_samples

아르테미스(Artemis) 임무

2024년에 시작할 NASA의 두 번째 임무로서 Moon to Mars(달-화성) 프로그램의 첫 번째 단계이다.

Artemis 임무에서 우주 비행사는 달에서 추가로 암석 샘플을 가져올 수 있다.

아르테미스 달 탐사에서 달 암석을 또 가져 올 수 있겠다.

아르테미스 달 탐사선

현재 2024년에 발사될 아르테미스 달 탐사선의 정확한 구조와 중량 데이터가 없다.

아직까지 아르테미스 달 탐사에 사용될 달 탐사선 정보가 없어서 어떡하지?

세번의 아르테미스 달 탐사가 있을 예정이고 예상 페이로드와 예상 승무원 영역 값은 발표되어 있어.

그 값들과 아폴로 달탐사 데이터 분석에서 얻은 비율을 사용해서 아르테미스 달탐사의 암석 샘플 예상 중량을 구해보자

아르테미스(Artemis) 임무에서 고려할 사항

아직 2024년 실시될 예정인 아르테미스 임무에 사용할 우주선의 전체 사양을 모른다.

NASA Factsheet on the Space Launch System(SLS)와 Orion Modules 의 일부 정보를 사용하여 아르테미스 달탐사의 예상 승무원 영역값과 예상 페이로드값을 수집하고 아폴로 달탐사 데이터 분석에서 얻은 비율을 사용한다.

페이로드는 로켓이 대기권을 통과하여 우주로 나갈 수 있는 총중량을 말하며 이것이 각 모듈의 정확한 중량보다 더 정확할 수 있다.

아폴로 임무에 사용된 Saturn V 페이로드는 43,500 kg이고 모듈 중량은 임무마다 달랐다.

아르테미스 임무와 관련한 예측에 사용할 비율을 결정하기 위해 아폴로 임무에 사용된 Saturn V 페이로드, 아폴로 각 임무의 암석샘플 중량, 모듈 중량, 승무원영역 중량 데이터를 사용하겠다.

아르테미스 임무 데이터프레임 만들기

아르테미스 임무 데이터프레임 : artemis_mission

아르테미스 임무는 세번 있을 예정이고 아폴로 임무와 일치시키기 위해 승무원영역 3개에 집중한다.
세번의 아르테미스 예상 승무원 영역의 값은 동일하며 값은 26520이다.

NASA SLS(우주 발사 시스템)가 발표한 아르테미스 임무의 예상 페이로드는 증가할 것으로 보이며
각각 26988, 37965, 42955 이다.

예상 데이터를 사용해 artemis_mission 데이터프레임 만든다.	pd.DataFrame(), 딕셔너리
missions 데이터프레임에 있는 세개의 비율의 평균값을 구한다.	s.mean()
위의 평균값을 사용해 예상 암석샘플 중량을 구한다.	데이터프레임의 산술연산

딕셔너리를 사용하여 데이터프레임 만들기

딕셔너리의 값(value)에는 리스트를 사용할 수 있다.

데이터프레임 df

	c0	c1	c2	c3
0	1	4	7	10
1	2	5	8	11
2	3	6	9	12

```
# 열이름을 key로 하고, 리스트를 value로 갖는 딕셔너리 정의(2차원 배열)
dict_data = {'c0':[1,2,3], 'c1':[4,5,6], 'c2':[7,8,9], 'c3':[10,11,12]}

# 판다스 DataFrame() 함수로 딕셔너리를 데이터프레임으로 변환. 변수 df에 저장.
df = pd.DataFrame(dict_data)
```

```
artemis_crewedArea = 26520
artemis_mission = pd.DataFrame({'Mission':['artemis1','artemis1b',
'artemis2'],
                                'Total weight (kg)':
[artemis_crewedArea,artemis_crewedArea,artemis_crewedArea],
                                'Payload (kg)':[26988, 37965, 42955]})
```

```
artemis_mission
```

	Mission	Total weight (kg)	Payload (kg)
0	artemis1	26520	26988
1	artemis1b	26520	37965
2	artemis2	26520	42955

컬럼별 기초 통계

데이터프레임의 각각의 컬럼의 최대, 최소, 평균, 중앙값 등을 집계함수를 사용해서 구할 수 있다..

최소값 min

```
missions['Crewed area : Payload'].min()
```

```
0.4750114942528736
```

- missions['Crewed area : Payload']
  컬럼값들 중 최소값

최대값 max

```
missions['Crewed area : Payload'].max()
```

```
0.5153103448275862
```

- missions['Crewed area : Payload']
  컬럼값들 중 최대값

평균 mean

```
missions['Crewed area : Payload'].mean()
```

```
0.4963026819923371
```

- missions['Crewed area : Payload']
  컬럼값들의 평균값

중앙값 median

```
missions['Crewed area : Payload'].median()
```

```
0.4977816091954023
```

- missions['Crewed area : Payload']
  컬럼값들 중 중위값

아르테미스 암석샘플 중량 예측을 위한 비율

여섯 번의 아폴로 임무에서 암석샘플이 실릴 곳의 컬럼을 mean() 명령어를 사용하여 평균을 구한다. 이것을 사용해서 아르테미스 암석샘플 중량 예측에 사용한다.

```
missions
```

Lunar module (LM)	LM mass (kg)	LM mass diff	Command module (CM)	CM mass (kg)	CM mass diff	Total weight (kg)	Total weight diff	Crewed area : Payload	Sample : Crewed area	Sample : Payload
Eagle (LM-5)	15103	0.00	Columbia (CSM-107)	5560	0.00	20663	0.00	0.48	0.00	0.00
Intrepid (LM-6)	15235	132.00	Yankee Clipper (CM-108)	5609	49.00	20844	181.00	0.48	0.00	0.00
Antares (LM-8)	15264	29.00	Kitty Hawk (CM-110)	5758	149.00	21022	178.00	0.48	0.00	0.00
Falcon (LM-10)	16430	1166.00	Endeavor (CM-112)	5875	117.00	22305	1283.00	0.51	0.00	0.00
Orion (LM-11)	16445	15.00	Casper (CM-113)	5840	-35.00	22285	-20.00	0.51	0.00	0.00
Challenger (LM-12)	16456	11.00	America (CM-114)	5960	120.00	22416	131.00	0.52	0.00	0.00

```
missions['Sample weight (kg)'].sum()
```

```
375.03599000000014
```

여섯번의 아폴로 임무에서 구해온 암석샘플 중량의 총합을 구한다.

```
crewedArea_payload_ratio = missions['Crewed area : Payload'].mean()
sample_crewedArea_ratio = missions['Sample : Crewed area'].mean()
sample_payload_ratio = missions['Sample : Payload'].mean()
print(crewedArea_payload_ratio)
print(sample_crewedArea_ratio)
print(sample_payload_ratio)
```

```
0.4963026819923371
0.002848764392685612
0.0014369195019157093
```

왼쪽 붉은색 상자의 값들을 평균한 값을 구한다.

아르테미스 임무에서 예상 샘플 중량 구하기 – (1)

데이터프레임의 모든 원소에 숫자를 산술연산 할 수 있습니다.

데이터프레임의 연산자 활용 : df + 산술연산자(+, -, *, /) + 숫자

```
sample_crewedArea_ratio
```

```
0.0028487643926856l2
```

26520 * 0.002849 = 75.55
26520 * 0.002849 = 75.55
26520 * 0.002849 = 75.55

```
sample_payload_ratio
```

```
0.0014369195019157093
```

26988 * 0.001437 = 38.78
37965 * 0.001437 = 54.55
42955 * 0.001437 = 61.72

```
artemis_mission['Sample weight from total (kg)'] = artemis_mission
['Total weight (kg)'] * sample_crewedArea_ratio
artemis_mission['Sample weight from payload (kg)'] = artemis_mission
['Payload (kg)'] * sample_payload_ratio
```

아르테미스 임무에서 예상 샘플 중량 구하기 – (2)

마지막으로 두 예측의 평균을 구할 수 있습니다.

```
artemis_mission
```

	Mission	Total weight (kg)	Payload (kg)	Sample weight from total (kg)	Sample weight from payload (kg)
0	artemis1	26520	26988	75.55	38.78
1	artemis1b	26520	37965	75.55	54.55
2	artemis2	26520	42955	75.55	61.72

- Sample weight from total (kg)과 Sample weight from payload (kg) 의 평균을 구해서 최종 예상 샘플 중량인 Estimated sample weight (kg) 컬럼을 만든다.
- 세번의 아르테미스 예상 샘플 중량은 57kg, 65kg, 68kg 이다.

이제 세번의 아르테미스 임무에서 샘플 중량을 예측했는데 이제부터 어떤 종류의 암석을 우선 가져와야 하는지 예측해 볼까요?

```
artemis_mission['Estimated sample weight (kg)'] = (artemis_mission['Sample weight from payload (kg)'] + artemis_mission['Sample weight from total (kg)'])/2
artemis_mission
```

	Mission	Total weight (kg)	Payload (kg)	Sample weight from total (kg)	Sample weight from payload (kg)	Estimated sample weight (kg)
0	artemis1	26520	26988	75.55	38.78	57.16
1	artemis1b	26520	37965	75.55	54.55	65.05
2	artemis2	26520	42955	75.55	61.72	68.64

데이터 분석의 5 단계

데이터 분석 단계에 맞추어 달 탐사 및 암석샘플 데이터 분석을 수행한다.

데이터 분석의 5단계

- 달 탐사 및 달 암석 채취시 문제점 파악
- 미래의 달 탐사 및 암석 채취시 개선될 점에 대해서 분석주제 정의

- 문제 정의에 따라서 분석에 필요한 데이터 선택
- 모든 아폴로 달 탐사에서 채취된 암석 샘플 데이터 수집

- 암석 샘플 데이터 기초분석하고 필요한 컬럼 추가
- 아폴로 임무별 새로운 데이터 프레임 만들기

- 아르테미스 임무에서 채취할 암석 중량 구하기
- **남아 있는 양이 적은 암석을 구하는 데이터프레임 만들기**

- 데이터 분석을 반영할 수 있는 시각화
- 문제 정의에 맞춘 해결방안 도출

여기서 배울 내용은 ?

1.문제정의

2.데이터수집

3.데이터 가공

4.데이터 분석

5.시각화 및 탐색

단계 1 : 아르테미스 임무 데이터프레임 만들기 : artemis_mission

단계 2 : 아르테미스 임무에서 수집할 수 있는 예상 암석샘플 중량 구하기

artemis_mission['Estimated sample weight (kg)']

단계 3 : 수집해야 하는 암석샘플 데이터프레임 만들기

low_samples, needed_samples

남아 있는 양이 적은 샘플 데이터프레임

남아 있는 양이 적은 샘플 데이터프레임 : low_samples

rock_samples에서 샘플 중량과 남은 양의 백분율을 곱해서 암석 샘플의 현재 남은 양을 나타내는 컬럼을 만든다.	시리즈 사이의 산술연산
rock_samples에서 샘플의 남은 양이 적은 행을 추출해 low_samples 데이터프레임을 만든다.	불리언 인덱싱, 행 추출 연산자
low_samples에 있는 암석 유형을 구한다.	Series.unique()
low_samples에서 암석유형별로 개수를 센다.	df.groupby(' ')[' '].count()
low_samples에서 부족한 암석유형에 해당하는 행을 추출한다.	isin(리스트) 구문

low_samples 데이터프레임

샘플 중량이 0.16kg이상이고 남아 있는 샘플량이 50% 아래에 있는 샘플들을 추출한다. 이러한 샘플들이 어떤 암석인지 알아본다.

low_samples 데이터프레임의 정보를 알아본다.	low_samples.info()
low_samples에 있는 암석 종류들을 알아본다.	low_samples.Type.unique()
rock_samples에 있는 암석 종류들을 알아본다.	rock_samples.Type.unique()
rock_samples에는 6가지, low_samples에는 4가지가 있다.	
low samples에서 부족한 암석이 무엇인지 알아본다.	low_sample.groupby('Type') ['Weight (kg)].count()

암석 샘플 수집의 우선순위

rock_samples 데이터프레임을 살펴본다.

수집된 양과 샘플 백분율을 사용해서 아폴로 임무에서 가져온 각 샘플 중 남은 양을 구할 수 있다.

rock_samples['Pristine (%)']는 %단위로 표시되었으므로 정확한 kg양을 구하려면 0.01를 곱해야 한다.

```
rock_samples.head()
```

	ID	Mission	Type	Subtype	Weight (kg)	Pristine (%)
0	10001	Apollo11	Soil	Unsieved	0.13	88.36
1	10002	Apollo11	Soil	Unsieved	5.63	93.73
2	10003	Apollo11	Basalt	Ilmenite	0.21	65.56
3	10004	Apollo11	Core	Unsieved	0.04	71.76
4	10005	Apollo11	Core	Unsieved	0.05	40.31

```
rock_samples['Remaining (kg)'] = rock_samples['Weight (kg)'] *
(rock_samples['Pristine (%)'] * .01)
rock_samples.head()
```

	ID	Mission	Type	Subtype	Weight (kg)	Pristine (%)	Remaining (kg)
0	10001	Apollo11	Soil	Unsieved	0.13	88.36	0.11
1	10002	Apollo11	Soil	Unsieved	5.63	93.73	5.28
2	10003	Apollo11	Basalt	Ilmenite	0.21	65.56	0.14
3	10004	Apollo11	Core	Unsieved	0.04	71.76	0.03
4	10005	Apollo11	Core	Unsieved	0.05	40.31	0.02

암석 샘플 수집의 우선순위

암석의 'Remaining (kg)' 컬럼을 추가한후 요약통계를 알아본다.

```
rock_samples.describe()
```

	ID	Weight (kg)	Pristine (%)	Remaining (kg)
count	2229.00	2229.00	2229.00	2229.00
mean	52058.43	0.17	84.51	0.14
std	26207.65	0.64	22.06	0.53
min	10001.00	0.00	0.00	0.00
25%	15437.00	0.00	80.01	0.00
50%	65527.00	0.01	92.30	0.01
75%	72142.00	0.09	98.14	0.08
max	79537.00	11.73	180.00	11.17

평균적으로 각 샘플의 중량이 약 0.17kg이고 약 84%가 남아 있다. 이 정보를 사용하여 부족할 가능성이 높은 샘플만 추출할 수 있다.

적은 양이 남아 있는 행을 추출해서 low_samples 데이터프레임을 만든다.

rock_samples['Weight (kg)'] 0.16이상이고 rock_samples['Pristine (%)'] 50 이하인 행만 추출해서 low_samples에 할당한다.

Boolean indexing에 대해 알아본다.

행을 추출하는 loc[ ]에 대해 알아본다.

불리언 인덱싱

불리언 인덱싱은 조건문의 결과인 불리언값(True/False)을 가지는 1차원의 Boolearn array가 입력될 경우, True값을 가지는 행만 추출하는 인덱싱이다.

```
rock_samples['Weight (kg)'] >= .16
```

```
0      False
1      False
2      False
3      False
4      False
5      False
6      False
7      False
8      False
9      False
10     False
```

```
rock_samples['Pristine (%)'] <= 50
```

```
0      False
1      False
2      False
3      False
4       True
5       True
6      False
7      False
8      False
9       True
10      True
```

Boolean array의 전체 개수는 데이터프레임의 행의 수와 같아야 한다.

```
(rock_samples['Weight (kg)'] >= .16) & (rock_samples['Pristine (%)'] <= 50)
```

```
0      False
1      False
2      False
3      False
4      False
5      False
6      False
7      False
8      False
9      False
10     False
```

두 개 이상의 조건문을 &, | 등의 논리 연산자로 엮어서 인덱싱 할 수 있다.

행 단위 데이터 추출

행 단위 데이터 추출은 loc 명령어 및 대괄호 [ ]를 사용한다.

```
(rock_samples['Weight (kg)'] >= .16) & (rock_samples['Pristine (%)'] <= 50)
```

```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11     True
12    False
13    False
14     True
15     True
```

loc 명령어에 불리언 인덱싱을 할 경우에는 1차원 불리언 배열을 입력 받는데 불리언 배열의 전체 개수는 데이터프레임의 행의 수와 같다.

```
low_samples = rock_samples.loc[(rock_samples['Weight (kg)'] >= .16) & (rock_samples['Pristine (%)'] <= 50)]
low_samples.head()
```

	ID	Mission	Type	Subtype	Weight (kg)	Pristine (%)	Remaining (kg)
11	10017	Apollo11	Basalt	Ilmenite	0.97	43.71	0.43
14	10020	Apollo11	Basalt	Ilmenite	0.42	27.88	0.12
15	10021	Apollo11	Breccia	Regolith	0.25	30.21	0.08
29	10045	Apollo11	Basalt	Olivine	0.18	12.13	0.02
37	10057	Apollo11	Basalt	Ilmenite	0.92	35.15	0.32

불리언 인덱싱 값이 True인 행만 추출된다.

수집할 암석 샘플 유형

남아 있는 샘플량이 적은 암석만 추출해 놓은 low_samples 데이터프레임에서 암석 종류별 개수를 세어본다.

low_samples.groupby('Type')['Weight (kg)'].count()

```
low_samples.groupby('Type')['Weight (kg)'].count()

Type
Basalt     14
Breccia     8
Core        1
Soil        4
Name: Weight (kg), dtype: int64
```

low_samples 데이터프레임에는 Basalt(현무암), Breccia(각력암)이 Core나 Soil보다 많다.

아르테미스 임무때 수집할 암석 샘플을 알려줄 데이터프레임을 만들 때 Basalt와 Breccia를 집어 넣어야지.

isin(리스트)

isin은 데이터프레임의 특정 컬럼이 리스트의 값들을 포함하고 있는지에 따라 True/False값을 돌려준다.

low_samples['Type'].isin(['Basalt', 'Breccia'])

```
low_samples['Type'].isin(['Basalt', 'Breccia'])
```

```
11     True
14     True
15     True
29     True
37     True
39     True
52     True
59    False
68     True
69    False
```

low_samples.loc[low_samples['Type'].isin(['Basalt', 'Breccia'])]

```
low_samples.loc[low_samples['Type'].isin(['Basalt', 'Breccia'])]
```

	ID	Mission	Type	Subtype	Weight (kg)	Pristine (%)	Remaining (kg)
11	10017	Apollo11	Basalt	Ilmenite	0.97	43.71	0.43
14	10020	Apollo11	Basalt	Ilmenite	0.42	27.88	0.12
15	10021	Apollo11	Breccia	Regolith	0.25	30.21	0.08
29	10045	Apollo11	Basalt	Olivine	0.18	12.13	0.02
37	10057	Apollo11	Basalt	Ilmenite	0.92	35.15	0.32

isin(리스트)를 사용한 결과는 True/False값을 가지는 불리언 배컬럼이 되고 이것을 행 추출 명령어 loc에 사용하면 특정 행들을 추출할 수 있다.

아르테미스 임무에서 수집할 샘플 유형

low_samples 데이터프레임에서 Basalt(현무암)과 Breccia(각력암)에 해당하는 행을 추출해서 아르테미스 임무에서 수집할 샘플을 저장하는 데이터프레임을 만들 것이다.

```
needed_samples = low_samples.loc[low_samples['Type'].isin(['Basalt', 'Breccia'])]
needed_samples
```

```
needed_samples.info()
✓ 0.7s
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22 entries, 11 to 2183
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ID              22 non-null     int64
 1   Mission         22 non-null     object
 2   Type            22 non-null     object
 3   Subtype         22 non-null     object
 4   Weight (kg)     22 non-null     float64
 5   Pristine (%)    22 non-null     float64
 6   Remaining (kg)  22 non-null     float64
dtypes: float64(3), int64(1), object(3)
memory usage: 1.4+ KB
```

```
needed_samples.head()
```

	ID	Mission	Type	Subtype	Weight (kg)	Pristine (%)	Remaining (kg)
11	10017	Apollo11	Basalt	Ilmenite	0.97	43.71	0.43
14	10020	Apollo11	Basalt	Ilmenite	0.42	27.88	0.12
15	10021	Apollo11	Breccia	Regolith	0.25	30.21	0.08
29	10045	Apollo11	Basalt	Olivine	0.18	12.13	0.02
37	10057	Apollo11	Basalt	Ilmenite	0.92	35.15	0.32

과연 Basalt와 Breccia만 남아 있는 양이 적은 암석 샘플일까? 아예 아폴로 임무에서 많이 수집되지 못했던 암석 샘플은 없을까? 만약 그런 암석이 있다면 아르테미스 임무에서도 수집되어야 할 암석이겠군

.

아르테미스 임무에서 수집할 샘플 데이터프레임

아르테미스 임무에서 수집할 샘플 데이터프레임 : needed_samples

low_samples에서 부족한 암석유형에 해당하는 행을 추출해 needed_samples 데이터프레임을 만든다.	isin(리스트) 구문
needed_samples 에 있는 암석유형별 중량 총합을 구한다.	needed_samples.groupby(' ')[' '].sum()
rock_samples 에 있는 암석유형별 중량 총합을 구한다.	rock_samples.groupby(' ')[' '].sum()
위의 값들을 통해 처음부터 양이 적은 암석유형을 알아낸다.	
위에서 구한 암석유형도 needed_samples에 추가한다.	df.append()

아폴로 임무에서 수집이 안됐던 암석 샘플

아폴로 임무에서 처음부터 수집이 안됐던 암석 샘플이 있는지 알아보기 위해 need_samples와 rock_samples 데이터프레임에서 암석 종류별 총량을 알아보자.

```
needed_samples.groupby('Type')['Weight (kg)'].sum()
```

```
Type
Basalt    17.42
Breccia   10.12
Name: Weight (kg), dtype: float64
```

Basalt와 Breccia는 아폴로 임무에서 수집이 많이 됐고 사용도 많아서 남아 있는 샘플양이 현저히 줄었기 때문에 아르테미스 임무에서 수집되어야 해,

```
rock_samples.groupby('Type')['Weight (kg)'].sum()
```

```
Type
Basalt      93.14
Breccia    168.88
Core        19.94
Crustal      4.74
Soil        87.59
Special      0.74
Name: Weight (kg), dtype: float64
```

Crustal은 아폴로 임무에서 아예 수집이 현저히 안되어 있기 때문에 아르테미스 임무에서도 수집되어야 해.

데이터프레임에 샘플 추가 : df.concat()

	name	age	city
0	A	18	Seoul
1	B	30	Incheon
2	C	25	Seoul
3	D	42	Busan

	name	age	city	height
1	B	30	Incheon	150
2	C	25	Seoul	170
3	D	42	Busan	180
4	E	11	Suwon	135

```
concat1 = pd.concat([df1, df2])
```

	name	age	city	height
0	A	18	Seoul	NaN
1	B	30	Incheon	NaN
2	C	25	Seoul	NaN
3	D	42	Busan	NaN
1	B	30	Incheon	150.0
2	C	25	Seoul	170.0
3	D	42	Busan	180.0
4	E	11	Suwon	135.0

```
concat2 = pd.concat([df1, df2], ignore_index=True)
```

	name	age	city	height
0	A	18	Seoul	NaN
1	B	30	Incheon	NaN
2	C	25	Seoul	NaN
3	D	42	Busan	NaN
4	B	30	Incheon	150.0
5	C	25	Seoul	170.0
6	D	42	Busan	180.0
7	E	11	Suwon	135.0

데이터프레임에 샘플 추가 : df.concat()

```
concat3 = pd.concat([df1, df2], axis=1)
```

	name	age	city	name	age	city	height
0	A	18.0	Seoul	NaN	NaN	NaN	NaN
1	B	30.0	Incheon	B	30.0	Incheon	150.0
2	C	25.0	Seoul	C	25.0	Seoul	170.0
3	D	42.0	Busan	D	42.0	Busan	180.0
4	NaN	NaN	NaN	E	11.0	Suwon	135.0

```
concat4 = pd.concat([df1, df2], axis=1, join='inner')
```

	name	age	city	name	age	city	height
1	B	30	Incheon	B	30	Incheon	150
2	C	25	Seoul	C	25	Seoul	170
3	D	42	Busan	D	42	Busan	180

데이터프레임에 샘플 추가 : df.concat()

pd.concat([df1, df2])

df1,df2 두 데이터 프레임 합치기

```
needed_samples.head()
```

	ID	Mission	Type	Subtype	Weight (kg)	Pristine (%)	Remaining (kg)
11	10017	Apollo11	Basalt	Ilmenite	0.97	43.71	0.43
14	10020	Apollo11	Basalt	Ilmenite	0.42	27.88	0.12
15	10021	Apollo11	Breccia	Regolith	0.25	30.21	0.08
29	10045	Apollo11	Basalt	Olivine	0.18	12.13	0.02
37	10057	Apollo11	Basalt	Ilmenite	0.92	35.15	0.32

```
rock_samples.loc[rock_samples['Type'] == 'Crustal']
```

	ID	Mission	Type	Subtype	Weight (kg)	Pristine (%)	Remaining (kg)
497	15361	Apollo15	Crustal	Cataclastic	0.00	66.56	0.00
498	15362	Apollo15	Crustal	Cataclastic	0.00	56.88	0.00
499	15363	Apollo15	Crustal	Cataclastic	0.00	71.00	0.00
540	15415	Apollo15	Crustal	Anorthosite	0.27	67.07	0.18

```
needed_samples=pd.concat([needed_samples,crustal])
needed_samples
```

[63] ✓ 0.0s

	ID	Mission	Type	Subtype	Weight (kg)	Pristine (%)	Remaining (kg)
11	10017	Apollo11	Basalt	Ilmenite	0.97300	43.71	0.425298
14	10020	Apollo11	Basalt	Ilmenite	0.42500	27.88	0.118490
15	10021	Apollo11	Breccia	Regolith	0.25000	30.21	0.075525
29	10045	Apollo11	Basalt	Olivine	0.18500	12.13	0.022441
37	10057	Apollo11	Basalt	Ilmenite	0.91900	35.15	0.323028
...	...	...	...	...	...	...	...
2089	78238	Apollo17	Crustal	Norite	0.00000	86.03	0.000000
2092	78256	Apollo17	Crustal	Pristine	0.00000	0.00	0.000000
2126	78517	Apollo17	Crustal	Cataclasite	0.00182	82.97	0.001510
2130	78527	Apollo17	Crustal	Cataclasite	0.00516	82.05	0.004234
2189	79215	Apollo17	Crustal	Cataclasite	0.55380	93.33	0.516862

68 rows × 7 columns

나 지금 어느 단계를 공부하는 거지?

1.문제정의

2.데이터수집

3.데이터 가공

4.데이터 분석

5.시각화 및 탐색

단계 1 : 아르테미스 임무 데이터프레임 만들기 : artemis_mission

단계 2 : 아르테미스 임무에서 수집할 수 있는 예상 암석샘플 중량 구하기

artemis_mission['Estimated sample weight (kg)']

단계 3 : 수집해야 하는 암석샘플 데이터프레임 만들기

low_samples, needed_samples

1. 문제 정의
2. 데이터 수집
3. 데이터 가공
4. 데이터 분석
5. **시각화 및 탐색**

데이터 분석의 5 단계

데이터 분석 단계에 맞추어 달 탐사 및 암석샘플 데이터 분석을 수행한다.

데이터 분석의 5단계

- 달 탐사 및 달 암석 채취시 문제점 파악
- 미래의 달 탐사 및 암석 채취시 개선될 점에 대해서 분석주제 정의

- 문제 정의에 따라서 분석에 필요한 데이터 선택
- 모든 아폴로 달 탐사에서 채취된 암석 샘플 데이터 수집

- 암석 샘플 데이터 기초분석하고 필요한 컬럼 추가
- 아폴로 임무별 새로운 데이터프레임 만들기

- 아르테미스 임무에서 채취할 암석 중량 구하기
- 수집해야 하는 암석 종류 구하기

- **최종 데이터 분석 결과를 포함하는 데이터프레임 만들기**
- 최종 수집할 암석 종류와 개수 구하기

여기서 배울 내용은 ?

1.문제정의

2.데이터수집

3.데이터 가공

4.데이터 모델링

5.시각화 및 탐색

아르테미스 우주비행사에게 전달할 최종 데이터프레임
needed_samples_overview 완성하기

최종 데이터 분석 결과를 포함하는 데이터프레임 만들기

최종 수집할 암석 종류와 개수 구하기

최종 수집할 샘플 데이터프레임

최종 수집할 샘플 데이터프레임 : needed_samples_overview

최종 단계는 아르테미스 임무의 우주 비행사에게 쉽게 알려주는 needed_samples_overview 데이터프레임을 만든다.	pd.DataFrame()
암석유형과 암석유형별 중량 합계, 암석 크기를 알 수 있는 평균 중량 컬럼들을 만든다.	s.unique(), pd.merge() df.groupby()[].sum().reset_index()
암석유형별 개수와 암석이 차지하는 비율을 나타내는 컬럼을 만든다	df.groupby()[].mean().reset_index() pd.merge(), pd.rename()
아르테미스 임무를 예측하기 위해 artemis_mission 에서 예상 샘플 중량의 평균을 구한다.	s.mean()
암석이 차지하는 비율에 예상 샘플 중량의 평균을 곱해 최종 수집할 샘플 중량과 샘플 개수를 구한다.	데이터프레임 산술연산

needed_samples_overview 데이터프레임 만들기

needed_samples_overview = pd.DataFrame()

pd.DataFrame()은 빈 데이터프레임을 만드는 명령어이고 이것을 변수 needed_samples_overview에 할당한다. 만들어진 missions의 타입은 데이터프레임이다

```
# needed_samples_overview 데이터프레임을 만든다.

needed_samples_overview = pd.DataFrame()
needed_samples_overview
```

✓ 0.3s

```
type(needed_samples_overview)
```

✓ 0.7s

```
pandas.core.frame.DataFrame
```

needed_samples['Type'].unique()

needed_samples['Type']의 타입은 시리즈이다.
Series.unique()는 시리즈에서 고유한 값들을 찾아주는 명령어이다.

```
needed_samples['Type'].unique()
```

✓ 0.4s

```
array(['Basalt', 'Breccia', 'Crustal'], dtype=object)
```

```
# needed_samples 데이터프레임에서 중복되지 않은 암석유형을 추출한다.

needed_samples_overview['Type'] = needed_samples['Type'].unique()
needed_samples_overview
```

✓ 0.4s

	Type
0	Basalt
1	Breccia
2	Crustal

최종 샘플 예측 – needed_samples_overview 만들기

최종 단계는 아르테미스 임무의 우주 비행사에게 쉽게 알려주기 위해 모든 정보를 통합하는 것이다. 먼저 이미 수집이 확정된 각 암석 유형의 컬럼이 필요하다.

```
needed_samples_overview = pd.DataFrame()
needed_samples_overview['Type'] = needed_samples.Type.unique()
needed_samples_overview
```

	Type
0	Basalt
1	Breccia
2	Crustal

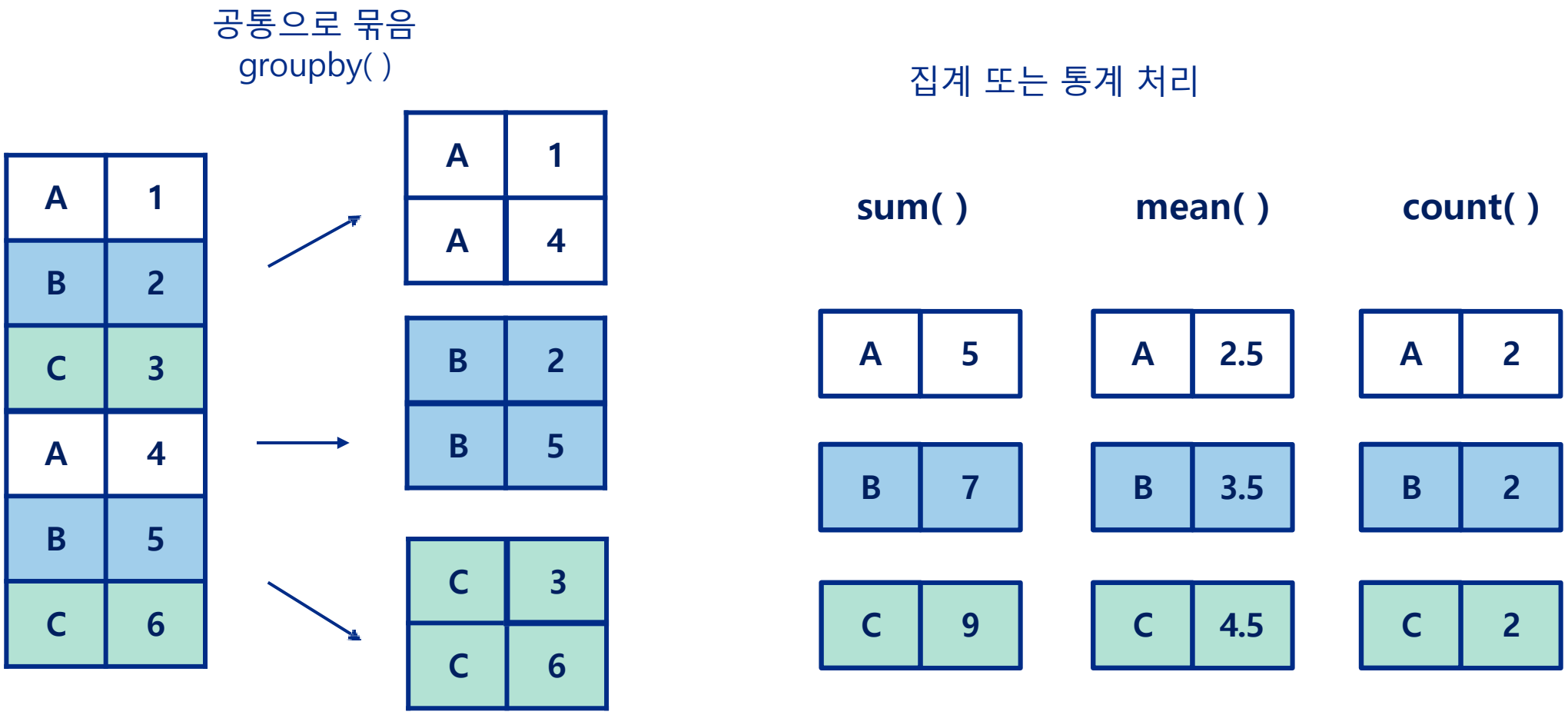
빈 데이터프레임을 만든다 : pd.DataFrame()

needed_samples에 있는 암석 종류들을 알아본다.
needed_samples.Type.unique()

needed_samples_overview에 Type 컬럼을 만든다.
needed_samples_overview.Type과
need_samples_overview['Type']은 같은 표시이다.

샘플 유형별 총중량 구하기 : groupby – (1)

컬럼의 고유값에 따라 묶어서 집계 또는 통계 처리를 할 때 그룹바이 명령을 적용한다.



샘플 유형별 총중량 구하기 : groupby – (2)

그룹 단위로 암석샘플 유형을 사용할 수 있다. 그룹 단위로 합계, 평균, 최소, 최대, 개수 등 다양한 집계 및 통계 처리가 가능하다.

```
needed_samples.groupby('Type' )['Weight (kg)'].sum()
```

그룹별 단위

- Basalt
- Breccia
- Crustal

집계 명령어를 사용할 컬럼

- 집계 명령어를 사용할 컬럼은 need_samples['Weight (kg)']

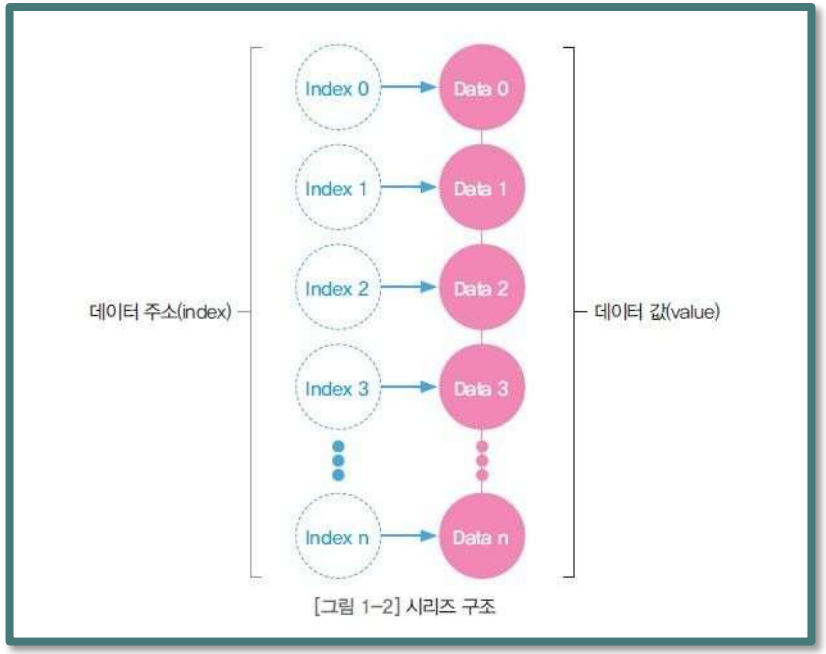
집계 명령어

- sum( )
- mean( )
- min( ), max( ), median( )
- count( )
- first( )

샘플 유형별 총중량 구하기 : groupby – (3)

- 시리즈Series – 데이터가 순차적으로 나열된 1차원 배열
- 인덱스Index와 데이터 값value 가 1:1 대응이 된다.
- 컬럼명이 없다.
- 인덱스Index : 데이터 값의 위치를 나타내는 이름표(데이터주소) 역할을 한다.

```
needed_samples.groupby('Type')['Weight (kg)'].sum()
✓ 0.4s
Type
Basalt      17.42340
Breccia     10.11850
Crustal      4.74469
Name: Weight (kg), dtype: float64
```



[그림 1-2] 시리즈 구조

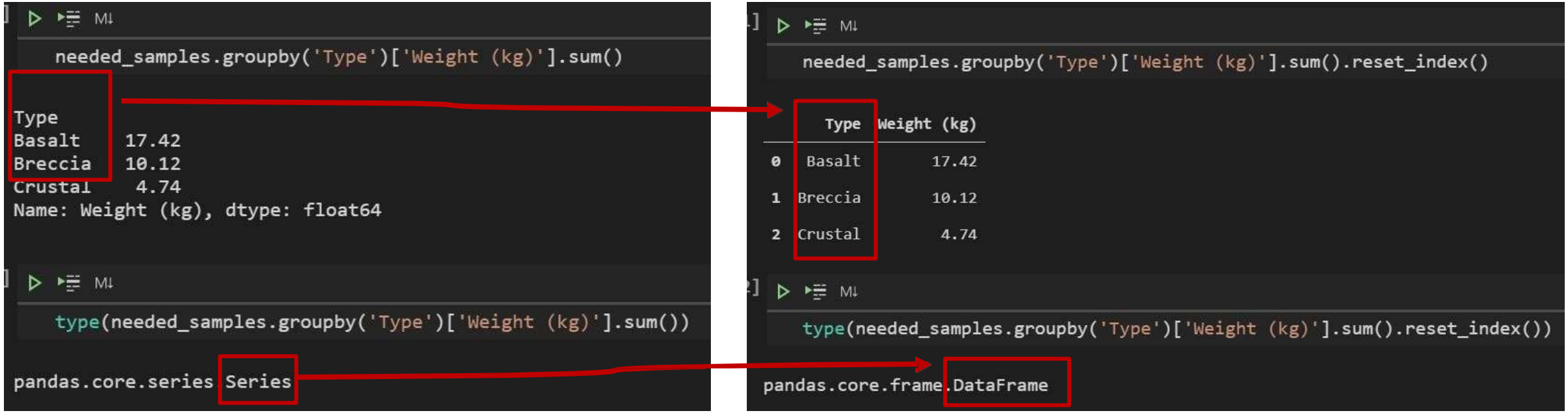
< 출처: 파이썬 머신러닝 판다스 데이터분석- 한빛출판사>

needed_samples.groupby('Type')['Weight (kg)'].sum()

- 결과 : 시리즈
- 인덱스 : 샘플유형
- 값 : 중량 총합

인덱스 재설정 : reset_index()

데이터프레임이나 시리즈의 인덱스를 컬럼으로 전송하며 새로운 정수 인덱스를 세팅한다. 시리즈를 데이터프레임으로 변환시킨다.



인덱스 → 컬럼

시리즈 → 데이터프레임

데이터프레임 연결하기 : pd.merge()

pd.merge(df1, df2, on='컬럼명')

1 **2** **3**

1. **pd.merge** : **두 개의** 데이터프레임에서 공통된 컬럼을 기준으로 동일한 값을 가지는 행을 각 데이터프레임에서 찾은 후, 이를 병합시킨다.
2. **df1, df2** : 연결할 데이터프레임 두개
3. **on='df1_컬럼명'** : df1과 df2 데이터프레임에 있는 컬럼명

```
needed_samples_overview
```

	Type
0	Basalt
1	Breccia
2	Crustal

```
needed_sample_weights = needed_samples.groupby('Type')['Weight (kg)']
.sum().reset_index()
needed_sample_weights
```

	Type	Weight (kg)
0	Basalt	17.42
1	Breccia	10.12
2	Crustal	4.74

공통된 컬럼

두개의 데이터프레임이 병합

```
needed_samples_overview = pd.merge(needed_samples_overview,
needed_sample_weights, on='Type')
needed_samples_overview
```

	Type	Weight (kg)
0	Basalt	17.42
1	Breccia	10.12
2	Crustal	4.74

컬럼명 변경하기 : df.rename()

needed_samples_overview.rename(columns={ 변경전 컬러명 : 변경후 컬럼명 }, inplace=True)

1 **2** **3**

1. needed_samples_overview.rename : 데이터프레임의 컬럼이름을 바꾸는 명령어
2. columns={ 변경전 컬럼명 : 변경후 컬럼명 } : 컬럼명을 바꾸고 싶으면 columns를 써준다.
   변경전 컬럼명과 변경후 컬럼명을 딕셔너리 형식으로 입력한다.
3. inplace=True : 변경된 내용을 missions 데이터프레임에 고정시킨다.

```
# 컬럼명을 'Total weight (kg)'으로 변경한다.

needed_samples_overview.rename(columns={'Weight (kg)' : 'Total weight (kg)'}, inplace=True)
needed_samples_overview
```

	Type	Total weight (kg)
0	Basalt	17.42340
1	Breccia	10.11850
2	Crustal	4.74469

샘플 평균 중량 구하기

아르테미스 우주비행사가 암석을 쉽게 식별하기 위해 크기 정보로서 암석의 평균 중량을 구해 추가한다.

1

```
# # needed_samples 데이터프레임에서 암석유형별 중량 평균을 구한다.

needed_samples.groupby('Type')['Weight (kg)'].mean()
```

✓ 0.5s

```
Type
Basalt      1.244529
Breccia     1.264812
Crustal     0.103145
Name: Weight (kg), dtype: float64
```

2

```
# reset_index를 해서 시리즈를 needed_sample_ave_weights 데이터프레임으로 만든다.

needed_sample_ave_weight = needed_samples.groupby('Type')['Weight (kg)'].mean().reset_index()
needed_sample_ave_weight
```

✓ 0.4s

	Type	Weight (kg)
0	Basalt	1.244529
1	Breccia	1.264812
2	Crustal	0.103145

3

```
needed_samples_overview와 needed_sample_ave_weights를 병합한다.

needed_samples_overview = pd.merge(needed_samples_overview, needed_sample_ave_weight, on='Type')
needed_samples_overview
```

✓ 0.5s

	Type	Total weight (kg)	Weight (kg)
0	Basalt	17.42340	1.244529
1	Breccia	10.11850	1.264812
2	Crustal	4.74469	0.103145

4

```
# 컬럼명을 'Average weight (kg)'으로 변경한다.

needed_samples_overview.rename(columns={'Weight (kg)' : 'Average weight (kg)'}, inplace=True)
needed_samples_overview
```

✓ 0.3s

	Type	Total weight (kg)	Average weight (kg)
0	Basalt	17.42340	1.244529
1	Breccia	10.11850	1.264812
2	Crustal	4.74469	0.103145

최종 샘플 예측 – 샘플 총중량과 평균중량 구하기

수집할 암석들의 현재 총중량을 구해서
needed_samples_overview 데이터프레임에 추가한다.

아르테미스 우주비행사가 암석을 쉽게 식별하기 위해 크기 정보로서 암석의 평균 중량을 구해 추가한다.

Crustal은 평균 중량이 매우 작으므로 발견하기가 어렵기 때문에 수집도 어려워 양이 적다.

```
needed_sample_weights = needed_samples.groupby('Type')['Weight (kg)']
.sum().reset_index()
needed_samples_overview = pd.merge(needed_samples_overview,
needed_sample_weights, on='Type')
needed_samples_overview.rename(columns={'Weight (kg)':'Total weight
(kg)'}, inplace=True)
needed_samples_overview
```

	Type	Total weight (kg)
0	Basalt	17.42
1	Breccia	10.12
2	Crustal	4.74

```
needed_sample_ave_weights = needed_samples.groupby('Type')['Weight (kg)
'].mean().reset_index()
needed_samples_overview = pd.merge(needed_samples_overview,
needed_sample_ave_weights, on='Type')
needed_samples_overview.rename(columns={'Weight (kg)':'Average weight
(kg)'}, inplace=True)
needed_samples_overview
```

	Type	Total weight (kg)	Average weight (kg)
0	Basalt	17.42	1.24
1	Breccia	10.12	1.26
2	Crustal	4.74	0.10

나 지금 어느 단계를 공부하는 거지?

1.문제정의

2.데이터수집

3.데이터 가공

4.데이터 모델링

5.시각화 및 탐색

아르테미스 우주비행사에게 전달할 최종 데이터프레임
needed_samples_overview 완성하기

최종 데이터 분석 결과를 포함하는 데이터프레임 만들기

최종 수집할 암석 종류와 개수 구하기

데이터 분석의 5 단계

데이터 분석 단계에 맞추어 달 탐사 및 암석샘플 데이터 분석을 수행한다.

데이터 분석의 5단계

- 달 탐사 및 달 암석 채취시 문제점 파악
- 미래의 달 탐사 및 암석 채취시 개선될 점에 대해서 분석주제 정의

- 문제 정의에 따라서 분석에 필요한 데이터 선택
- 모든 아폴로 달 탐사에서 채취된 암석 샘플 데이터 수집

- 암석 샘플 데이터 기초분석하고 필요한 컬럼 추가
- 아폴로 임무별 새로운 데이터프레임 만들기

- 아르테미스 임무에서 채취할 암석 중량 구하기
- 수집해야 하는 암석 종류 구하기

- 최종 데이터 분석 결과를 포함하는 데이터프레임 만들기
- **최종 수집할 암석 종류와 개수 구하기**

여기서 배울 내용은 ?

1.문제정의

2.데이터수집

3.데이터 가공

4.데이터 모델링

5.시각화 및 탐색

아르테미스 우주비행사에게 전달할 최종 데이터프레임
needed_samples_overview 완성하기

최종 데이터 분석 결과를 포함하는 데이터프레임 만들기

최종 수집할 암석 종류와 개수 구하기

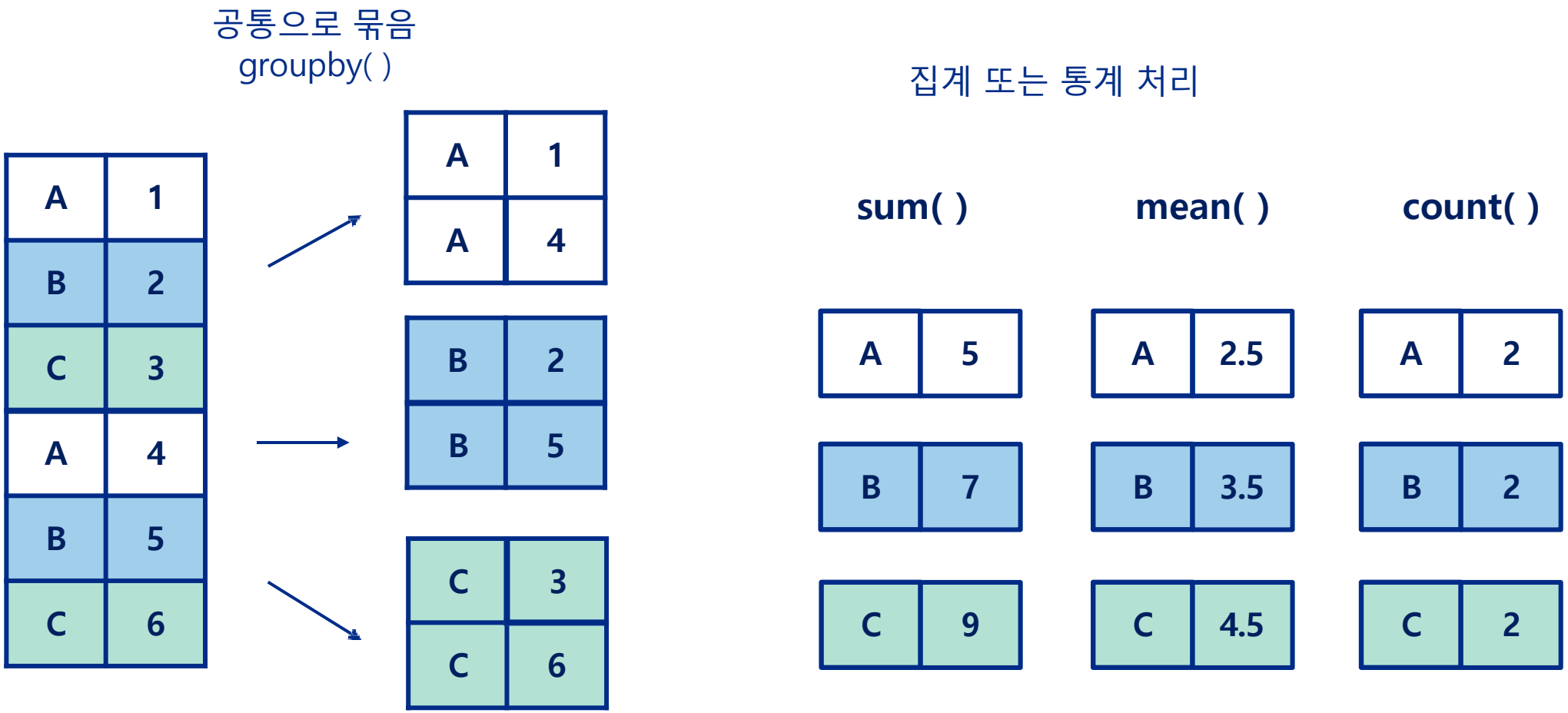
최종 수집할 샘플 데이터프레임

최종 수집할 샘플 데이터프레임 : needed_samples_overview

최종 단계는 아르테미스 임무의 우주 비행사에게 쉽게 알려주는 needed_samples_overview 데이터프레임을 만든다.	pd.DataFrame()
암석유형, 암석유형별 중량 합계, 암석 크기를 알 수 있는 평균 중량 컬럼들을 만든다.	s.unique(), pd.merge() df.groupby()[].sum().reset_index()
암석유형별 개수와 암석이 차지하는 비율을 나타내는 컬럼을 만든다	df.groupby()[].count().reset_index() pd.merge(), pd.rename()
아르테미스 임무를 예측하기 위해 artemis_mission 에서 예상 샘플 중량의 평균을 구한다.	s.mean()
암석이 차지하는 비율에 예상 샘플 중량의 평균을 곱해 최종 수집할 샘플 중량과 샘플 개수를 구한다.	데이터프레임 산술연산

샘플 유형별 개수 구하기 : groupby – (1)

컬럼의 고유값에 따라 묶어서 집계 또는 통계 처리를 할 때 그룹바이 명령을 적용한다.



샘플 유형별 개수 구하기 : groupby – (2)

그룹 단위로 암석샘플 유형을 사용할 수 있다. 그룹 단위로 합계, 평균, 최소, 최대, 개수 등 다양한 집계 및 통계 처리가 가능하다.

rock_samples.groupby('Type')['ID'].count()

그룹별 단위

- Basalt
- Breccia
- Crustal
- Soil
- Core
- Special

집계 명령어를 사용할 컬럼

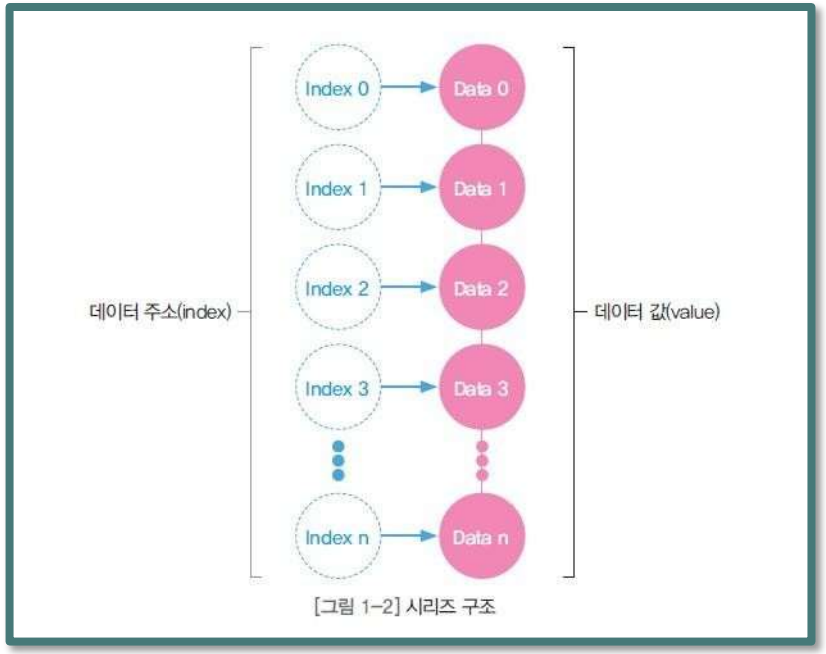
- 집계 명령어를 사용할 컬럼은 rock_samples['ID']
- count() 를 사용할 때는 컬럼에 상관없음.

집계 명령어

- sum( )
- mean( )
- min( ), max( ), median( )
- count( )
- first( )

샘플 유형별 개수 구하기 : groupby – (3)

- 시리즈Series – 데이터가 순차적으로 나열된 1차원 배열
- 인덱스Index와 데이터 값value 가 1:1 대응이 된다.
- 컬럼명이 없다.
- 인덱스Index : 데이터 값의 위치를 나타내는 이름표(데이터주소) 역할을 한다.



[그림 1-2] 시리즈 구조

< 출처: 파이썬 머신러닝 판다스 데이터분석- 한빛출판사>

```
rock_samples.groupby('Type')['ID'].count()
✓ 0.4s

Type
Basalt      351
Breccia     959
Core         56
Crustal      46
Soil        813
Special       4
Name: ID, dtype: int64
```

rock_samples.groupby('Type')['ID'].count()

- 결과 : 시리즈
- 인덱스 : 샘플유형
- 값 : 샘플유형별 개수

인덱스 재설정 : reset_index()

데이터프레임이나 시리즈의 인덱스를 컬럼으로 전송하며 새로운 정수 인덱스를 세팅한다. 시리즈를 데이터프레임으로 변환시킨다.

```
rock_samples.groupby('Type')['ID'].count()
```

✓ 0.4s

```
Type
Basalt      351
Breccia     959
Core         56
Crustal      46
Soil        813
Special       4
Name: ID, dtype: int64
```

```
total_rock_count = rock_samples.groupby('Type')['ID'].count().reset_index()
total_rock_count
```

✓ 0.7s

	Type	ID
0	Basalt	351
1	Breccia	959
2	Core	56
3	Crustal	46
4	Soil	813
5	Special	4

시리즈 → 데이터프레임

인덱스 → 컬럼

데이터프레임 연결하기 : pd.merge()

pd.merge(df1, df2, on='컬럼명')

1 **2** **3**

1. **pd.merge** : **두 개의** 데이터프레임에서 공통된 컬럼을 기준으로 동일한 값을 가지는 행을 각 데이터프레임에서 찾은 후, 이를 병합시킨다.
2. **df1, df2** : 연결할 데이터프레임 두개
3. **on='df1_컬럼명'** : df1과 df2 데이터프레임에 있는 컬럼명

두개의 데이터프레임이 병합

```
needed_samples_overview
✓ 0.5s
```

	Type	Total weight (kg)	Average weight (kg)
0	Basalt	17.42340	1.244529
1	Breccia	10.11850	1.264812
2	Crustal	4.74469	0.103145

```
total_rock_count = rock_samples.groupby('Type')['ID'].count().reset_index()
total_rock_count
✓ 0.7s
```

	Type	ID
0	Basalt	351
1	Breccia	959
2	Core	56
3	Crustal	46
4	Soil	813
5	Special	4

```
# needed_samples_overview와 total_rock_count를 'Type'에 따라 병합한다.needed_sample_ave_weight

needed_samples_overview = pd.merge(needed_samples_overview, total_rock_count, on='Type')
needed_samples_overview
✓ 0.6s
```

	Type	Total weight (kg)	Average weight (kg)	ID
0	Basalt	17.42340	1.244529	351
1	Breccia	10.11850	1.264812	959
2	Crustal	4.74469	0.103145	46

공통된 컬럼

컬럼명 변경하기 : df.rename()

needed_samples_overview.rename(columns={ 변경전 컬러명 : 변경후 컬럼명 }, inplace=True)

1 **2** **3**

1. needed_samples_overview.rename : 데이터프레임의 컬럼이름을 바꾸는 명령어
2. columns={ 변경전 컬럼명 : 변경후 컬럼명 } : 컬럼명을 바꾸고 싶으면 columns를 써준다.
   변경전 컬럼명과 변경후 컬럼명을 딕셔너리 형식으로 입력한다.
3. inplace=True : 변경된 내용을 missions 데이터프레임에 고정시킨다.

```
# 컬럼 이름을 'Number of samples'로 변경한다.

needed_samples_overview.rename(columns={'ID' : 'Number of rocks'}, inplace=True)
needed_samples_overview
```

	Type	Total weight (kg)	Average weight (kg)	Number of rocks
0	Basalt	17.42340	1.244529	351
1	Breccia	10.11850	1.264812	959
2	Crustal	4.74469	0.103145	46

최종 샘플 예측 – 샘플 유형별 암석개수

우주비행사가 수집할 세가지 암석의 개수를 표시한다. total_rock_count는 아폴로 임무에서 가져온 암석의 총개수를 나타내고 여기서 needed_samples_overview와 공통된 값인 Basalt, Breccia, Crustal 의 값만 추출된다.

```
total_rock_count = rock_samples.groupby('Type')['ID'].count()
.reset_index()
total_rock_count
```

total_rock_count

	Type	ID
0	Basalt	351
1	Breccia	959
2	Core	56
3	Crustal	46
4	Soil	813
5	Special	4

```
total_rock_count = rock_samples.groupby('Type')['ID'].count()
.reset_index()
needed_samples_overview = pd.merge(needed_samples_overview,
total_rock_count, on='Type')
needed_samples_overview.rename(columns={'ID':'Number of samples'},
inplace=True)
needed_samples_overview
```

	Type	Total weight (kg)	Average weight (kg)	Number of samples
0	Basalt	17.42	1.24	351
1	Breccia	10.12	1.26	959
2	Crustal	4.74	0.10	46

최종 샘플 예측 – 암석 비율

세가지 유형의 암석이 차지하는 비율을 구한다. 이는 needed_samples_overview['Number of sample']의 총합을 구하고 이것을 각 암석의 개수에 나누어 비율을 구할 수 있다.

```
total_rocks = needed_samples_overview['Number of samples'].sum()
total_rocks
```

```
1356
```

```
total_rocks = needed_samples_overview['Number of samples'].sum()
needed_samples_overview['Percentage of rocks'] =
needed_samples_overview['Number of samples'] / total_rocks
needed_samples_overview
```

	Type	Total weight (kg)	Average weight (kg)	Number of samples	Percentage of rocks
0	Basalt	17.42	1.24	351	0.26
1	Breccia	10.12	1.26	959	0.71
2	Crustal	4.74	0.10	46	0.03

위에서 구한 모든 정보를 artemis_mission['Estimated sample weight (kg)'] 과 연결한다.

최종 샘플 예측 – 수집할 암석 중량

데이터프레임의 모든 원소에 숫자를 산술연산 할 수 있습니다.

각각의 암석이 수집되어야 할 중량 = 평균 암석 중량 * 각각의 암석비율

```
artemis_mission
```

	Mission	Total weight (kg)	Payload (kg)	Sample weight from total (kg)	Sample weight from payload (kg)	Estimated sample weight (kg)
0	artemis1	26520	26988	75.55	38.78	57.16
1	artemis1b	26520	37965	75.55	54.55	65.05
2	artemis2	26520	42955	75.55	61.72	68.64

```
artemis_ave_weight = artemis_mission['Estimated sample weight (kg)']
.mean()
artemis_ave_weight
```

```
63.617134115797946
```

```
needed_samples_overview['Weight to collect'] = needed_samples_overview
['Percentage of rocks'] * artemis_ave_weight
needed_samples_overview['Rocks to collect'] = needed_samples_overview
['Weight to collect'] / needed_samples_overview['Average weight (kg)']
needed_samples_overview
```

	Type	Total weight (kg)	Average weight (kg)	Number of samples	Percentage of rocks	Weight to collect	Rocks to collect
0	Basalt	17.42	1.24	351	0.26	16.47	13.23
1	Breccia	10.12	1.26	959	0.71	44.99	35.57
2	Crustal	4.74	0.10	46	0.03	2.16	20.92

63.6171 * 0.26 = 16.47

63.6171 * 0.71 = 44.99

63.6171 * 0.03 = 2.16

최종 샘플 예측 – 수집할 암석 개수

데이터프레임의 모든 원소에 숫자를 산술연산 할 수 있습니다.

수집할 암석 개수 = 각각의 수집되어야 할 암석 중량 / 암석의 평균 중량

```
needed_samples_overview['Weight to collect'] = needed_samples_overview
['Percentage of rocks'] * artemis_ave_weight
needed_samples_overview['Rocks to collect'] = needed_samples_overview
['Weight to collect'] / needed_samples_overview['Average weight (kg)']
needed_samples_overview
```

	Type	Total weight (kg)	Average weight (kg)	Number of samples	Percentage of rocks	Weight to collect	Rocks to collect
0	Basalt	17.42	1.24	351	0.26	16.47	13.23
1	Breccia	10.12	1.26	959	0.71	44.99	35.57
2	Crustal	4.74	0.10	46	0.03	2.16	20.92

16.47 / 1.24 = 13.23

44.99 / 1.26 = 35.57

2.16 / 0.10 = 20.92

최종 샘플 예측 – needed_samples_overview 완성

앞에서 공부한 아르테미스 임무에 관한 데이터프레임을 다시 보고, 여기서 'Estimated sample weight (kg)의 평균을 구한다.

'Estimated sample weight (kg)의 평균을 각 암석이 차지하는 비율에 곱해서 수집할 암석의 중량을 구한다. 그 다음 이 값들을 각 암석의 평균 중량으로 나누어서 우주비행사가 수집할 암석 수를 구한다.

```
artemis_mission
```

	Mission	Total weight (kg)	Payload (kg)	Sample weight from total (kg)	Sample weight from payload (kg)	Estimated sample weight (kg)
0	artemis1	26520	26988	75.55	38.78	57.16
1	artemis1b	26520	37965	75.55	54.55	65.05
2	artemis2	26520	42955	75.55	61.72	68.64

```
artemis_ave_weight = artemis_mission['Estimated sample weight (kg)']
.mean()
artemis_ave_weight
```

```
63.617134115797946
```

```
needed_samples_overview['Weight to collect'] = needed_samples_overview
['Percentage of rocks'] * artemis_ave_weight
needed_samples_overview['Rocks to collect'] = needed_samples_overview
['Weight to collect'] / needed_samples_overview['Average weight (kg)']
needed_samples_overview
```

	Type	Total weight (kg)	Average weight (kg)	Number of samples	Percentage of rocks	Weight to collect	Rocks to collect
0	Basalt	17.42	1.24	351	0.26	16.47	13.23
1	Breccia	10.12	1.26	959	0.71	44.99	35.57
2	Crustal	4.74	0.10	46	0.03	2.16	20.92

우주 암석 연구의 AI 솔루션

관련된 수업 소개

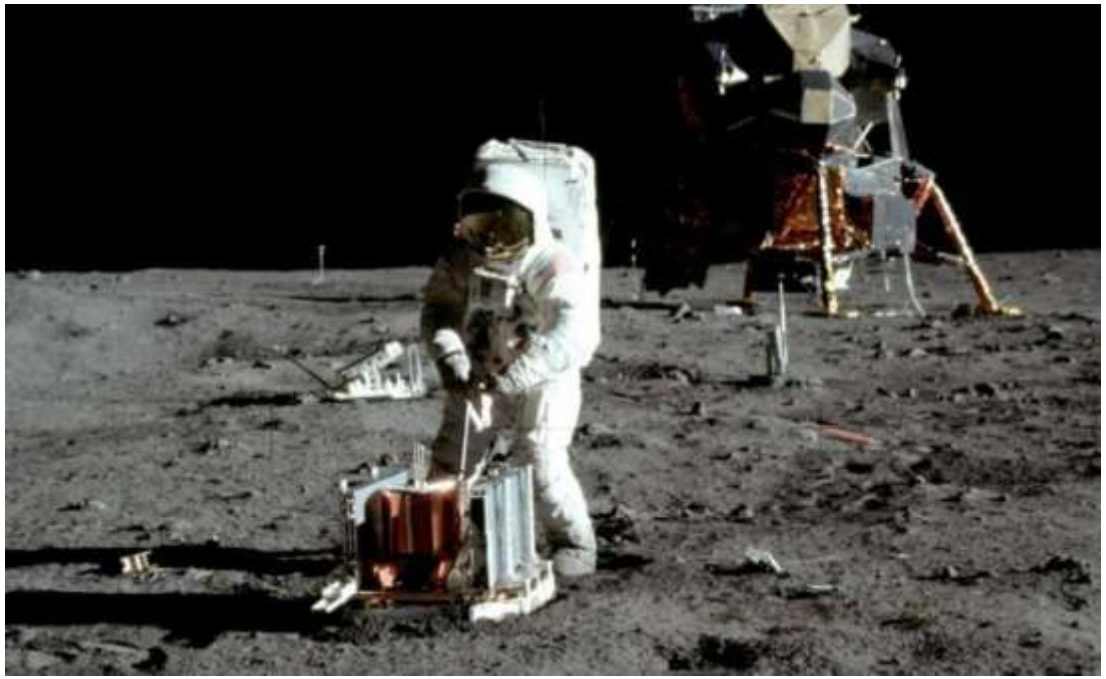
우주 암석 식별에 AI를 통합하면 인간의 우주 암석 수집 프로세스를 개선할 수 있다.

1. 암석 사진을 촬영할 수 있는 컴퓨터를 갖춘 우주 비행사

컴퓨터가 우주 비행사에게 암석 유형을 보여주고 우주 비행사는 해당 암석 종류를 수집해야 하는지 확인한 후 채취 여부를 결정할 수 있다.

2. AI를 통합하면 우주 비행사는 지구로 가져와야 하는 암석을 좀 더 빠르고 정확하게 찾고 식별할 수 있다.

이 컴퓨터는 위치, 온도 및 노광과 같은 메타데이터를 수집할 수 있다. 우주 비행사와 지구의 과학자가 피드백을 통해 암석을 식별하는 AI 모델을 개선시킬 수 있다.



우주비행사 닐 암스트롱이 달 표면에서 임무를 수행하고 있는 모습. - 미국항공우주국 제공

미래의 임무에서는 달 표면을 자율 주행하고 연구가 필요한 암석을 검색하는 AI 컴퓨터를 장착한 탐사선을 만들 수 있지 않을까?

암석 식별 머신

관련된 수업 소개

아르테미스 임무의 우주 비행사에게 암석 식별을 쉽게 할 수 있도록 어떤 도움을 줄 수 있을까 ?

달 표면에는 화학적 조성이 유사해서 비슷하게 보이는 우주 암석이 너무도 많아.

해리

우주 비행사가 어두운 달 표면에서 우주복을 입은 상태로 암석을 만지지 않고 필요한 암석을 식별하는 것은 불가능할 것 같아.

제니

우주 비행사가 암석 유형을 쉽게 파악하기 위해서 딥러닝의 도움을 받는 것은 어떨까?

론

감사합니다 ☺