



Class



Class는 변수와 함수를 하나로 묶은 덩어리이다.



변수와 함수를 한 덩어리로 묶을 수 있다.

- `potato`, `hambug`라는 변수와 `eat`이라는 함수를 `SetMenu` 클래스라는 하나의 덩어리로 묶었다.

```
class SetMenu:
    def __init__(self):
        self.potato = 100
        self.hambug = 200

    def eat(self):
        print('포테이토 맛있다')
        print(f'음 {self.hambug + self.potato} 만큼 맛있다!')
```



Class 구성요소 - 생성자, 필드

- `__init__(self)`
 - 언더바(_) 2개씩 사용하고, 고정된 Keyword의 특별한 함수이다.
 - 클래스 안에서 사용할 수 있는 변수를 이곳에서 생성한다. 이러한 변수를 **필드**라고 한다.
 - class 안에서 `__init__` 함수를 **생성자** 라고 한다.

```
class SetMenu:  
    def __init__(self):  
        self.potato = 100  
        self.hamburg = 200  
  
    def eat(self):  
        print('포테이토 맛있다')  
        print(f'음 {self.hamburg + self.potato} 만큼 맛있다!')
```



Class 구성요소 - 메서드

- 함수 여러개 생성 가능
 - 클래스 내부에 만들어진 함수를 “메서드” 라고 한다.
 - 생성자**와 **필드**, **메서드** 라는 용어를 암기해두자.
 - 생성자** : `__init__` 함수, 이곳에서 클래스 안에서 사용할 변수를 만든다.
 - 필드** : 클래스 안에서 사용할 변수
 - 메서드** : 클래스 안에서 만든 함수

```
class SetMenu:
    def __init__(self):
        self.potato = 100
        self.hamburg = 200

    def eat(self):
        print('포테이토 맛있다')
        print(f'음 {self.hamburg + self.potato} 만큼 맛있다!')
```

Class 사용하는 방법

제작한 Class를 사용하는 방법

- sanghai = SetMenu() 코드를 통해,
sanghaio 변수를 사용하여 제작한 Class를 사용할 수 있다.

```
class SetMenu:
    def __init__(self):
        self.potato = 100
        self.hambug = 200

    def eat(self):
        print('포테이토 맛있다')
        print(f'음 {self.hambug + self.potato} 만큼 맛있다!')

sanghai = SetMenu()
sanghai.eat()
```

포테이토 맛있다
음 300 만큼 맛있다!

[도전] 저글링 클래스 만들기



◎ 저글링 클래스 만들기

▪ 필드 (생성자에서 구현)

- hp = 20
- mana = 50

▪ run()메서드

- “똥다” 라는 Text 출력
- hp가 1 줄어든다.
- mana가 1 올라간다.

▪ show_status() 메서드

- hp와 mana를 출력한다.

모든 메서드에
항상 **self**를
넣어주어야한다.

```
class Zergling :  
    def __init__(self):  
        pass  
  
    def run(self):  
        pass  
  
    def show_stats(self):  
        pass
```

[도전] 저글링 클래스를 사용하는 코드 작성하기



◎ 앞에서 제작한 저글링 클래스 사용하는 코드 만들기

- 저글링 두 마리를 만든다.
 - 변수 z1, z2 에 생성
- 저글링을 각각 동작시킨다.
 - z1은 run 이후 show_status 호출
 - z2는 run 5회 이후 show_status 호출

[도전] Coin Game 만들기



- ◎ 앞에서 제작한 저글링 클래스 사용하는 코드 만들기
 - 저글링 두 마리를 만든다.
 - 변수 z1, z2 에 생성
 - 저글링을 각각 동작시킨다.
 - z1은 run 이후 show_status 호출
 - z2는 run 5회 이후 show_status 호출

[도전] GameMachine Class 제작 (5분)



input_coin(집어 넣을 코인 수)

- 코인은 최대 5 개까지 넣을 수 있음
- 입력된 코인이 10 보다 초과될 수 없음

play_game()

- 1 코인 씩 감소됨

집어넣은 코인이 얼마나 되는지
확인할 수 있어야 함

- 메서드 추가 생성 필요

```
gm = GameMachine()  
gm.input_coin(2)  
gm.show_status()  
gm.play_game()  
gm.show_status()
```

남아있는 코인은 2 입니다
게임 재밌다
남아있는 코인은 1 입니다

클래스 코드 자세히보기

◎ 클래스 생성코드 동작 순서 이해 1

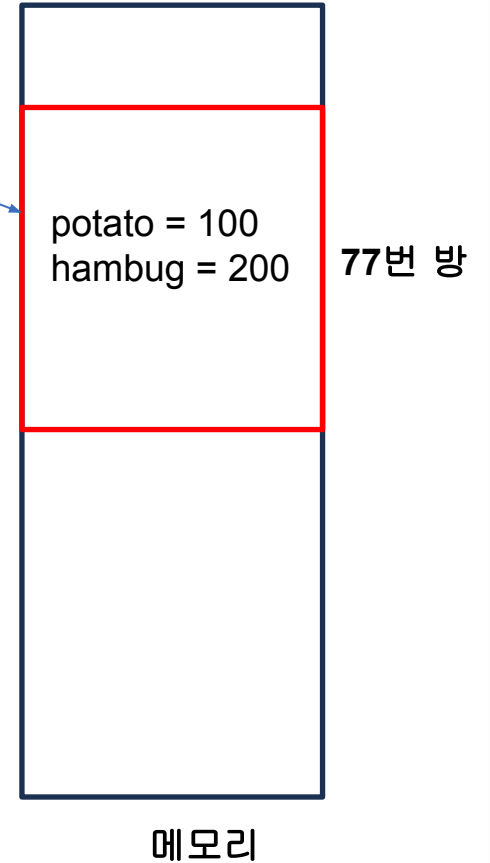
- 먼저 SetMenu()가 실행된다.
 - 메모리 공간에 SetMenu() 를 위한 공간이 만들어진다.
 - 이 공간을 **인스턴스**라고 부른다.

```
class SetMenu:
    def __init__(self):
        self.potato = 100
        self.hambug = 200

    def eat(self):
        print('포테이토 맛있다')
        print(f'음 {self.hambug + self.potato} 만큼 맛있다!')

sanghai = SetMenu()
sanghai.eat()
```

이렇게 생성된
메모리 공간을
인스턴스라고 한다.



클래스 코드 자세히보기

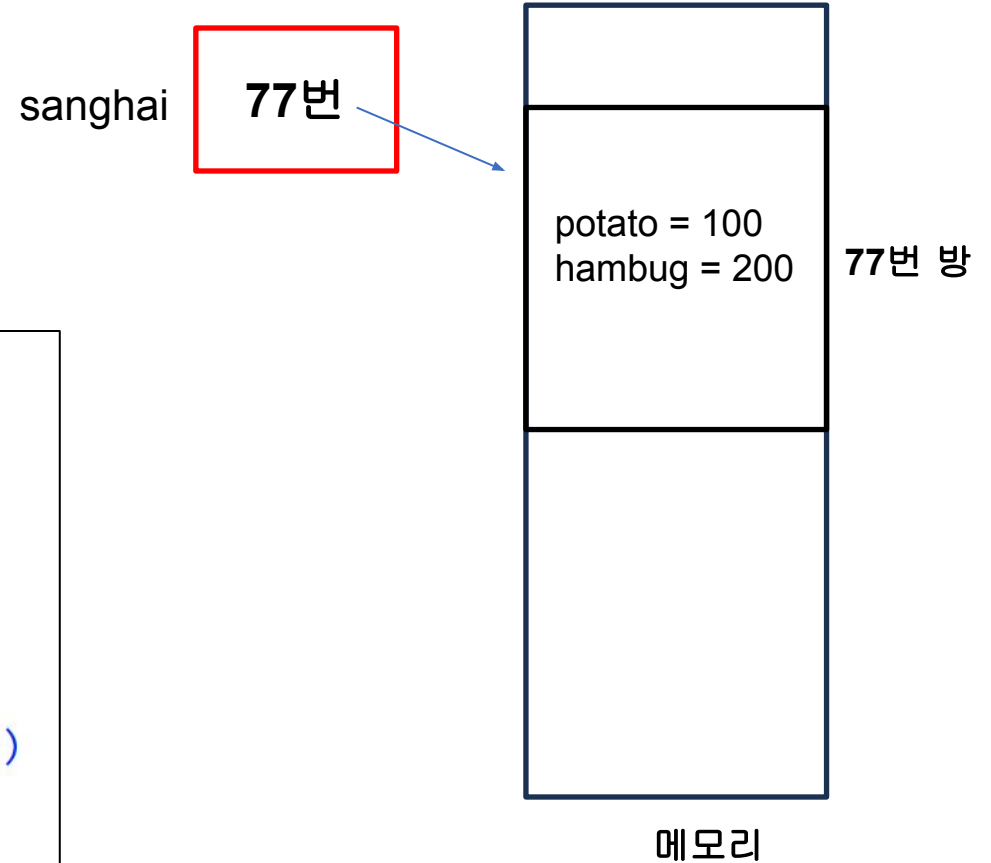
◎ 클래스 생성코드 동작 순서 이해 2

- 인스턴스의 방번호가 변수에 대입된다.
- 즉, 변수는 인스턴스를 가리키게 되는 것이다.
- “인스턴스를 **할당하였다.**” 라고 표현한다.

```
class SetMenu:
    def __init__(self):
        self.potato = 100
        self.hambug = 200

    def eat(self):
        print('포테이토 맛있다')
        print(f'음 {self.hambug + self.potato} 만큼 맛있다!')

sanghai = SetMenu()
sanghai.eat()
```



객체와 변수 차이 이해하기



- ◎ 파이썬에서는
수, 문자열, 리스트, 클래스의 인스턴스들을
모두 변수를 만들고 할당하는 방식 으로 코딩하게 된다.
- ◎ 파이썬에서는
수, 문자열, 리스트, 클래스의 인스턴스들을 **객체** 라고 한다.
 - 즉, 파이썬에서는 모든 데이터를 객체로 취급한다.
- ◎ **모든 객체는 변수에 할당한 후에, 그 변수를 이용하여 객체를 제어한다.**
 - 변수는 객체의 이름표이다.

[도전] 객체와 변수 구분하기



◎ 개수 맞추기

- 아래 코드에서 등장하는 변수의 개수는?
- 아래 코드에서 등장하는 객체의 개수는?
- 할당 횟수는?

```
a = 3
b = [1, 4, 2, 5, 4]
c = SetMenu()
a = 15
b[3] = 2
a = SetMenu()
```

[도전] Class를 이용한 구현 방법 소개



파이썬에서는 다음과 같이 클래스를 활용할 수 있으니,
다음 코드를 한번 이해해보자.

- 무엇을 집어 넣느냐에 따라 나오는 결과가 달라진다.
이 것을 **다형성**이라고 한다.

```
class Duck:
    def quack(self):
        print("꽹꽹")

class Person:
    def quack(self):
        print("사람이 꽹꽹 소리를 냅니다!")

def make_quack(who):
    who.quack()

duck = Duck()
person = Person()

make_quack(duck)
make_quack(person)
```



상속이란?

- 기존 클래스의 필드와 메서드를 물려받으면서, 새로운 클래스를 만드는 것을 상속이라고 한다.
- 부모클래스 보다 더 많은 필드와 메서드를 추가할 수 있음

```
class Person:  
    def walk(self):  
        print("사람이 걷는다")
```

```
class SuperMan(Person):  
    def fly(self):  
        print('난다')
```

```
a = SuperMan()
```

```
a.fly()
```

```
a.walk()
```

난다
사람이 걷는다

상속 - 오버라이딩

오버라이딩

- 부모의 메서드를 자식이 변경할 수 있음
- 이를 “재정의한다” 또는, “**Overriding 한다**”라고 표현한다.

```
class Person:
    def walk(self):
        print("사람이 걷는다")

class SuperMan(Person):
    def fly(self):
        print('난다')

    def walk(self):
        print('슈퍼맨이 두박두박')
```

walk 는 오버라이딩 된 메서드이다.

```
a = SuperMan()
a.fly()
a.walk()
```

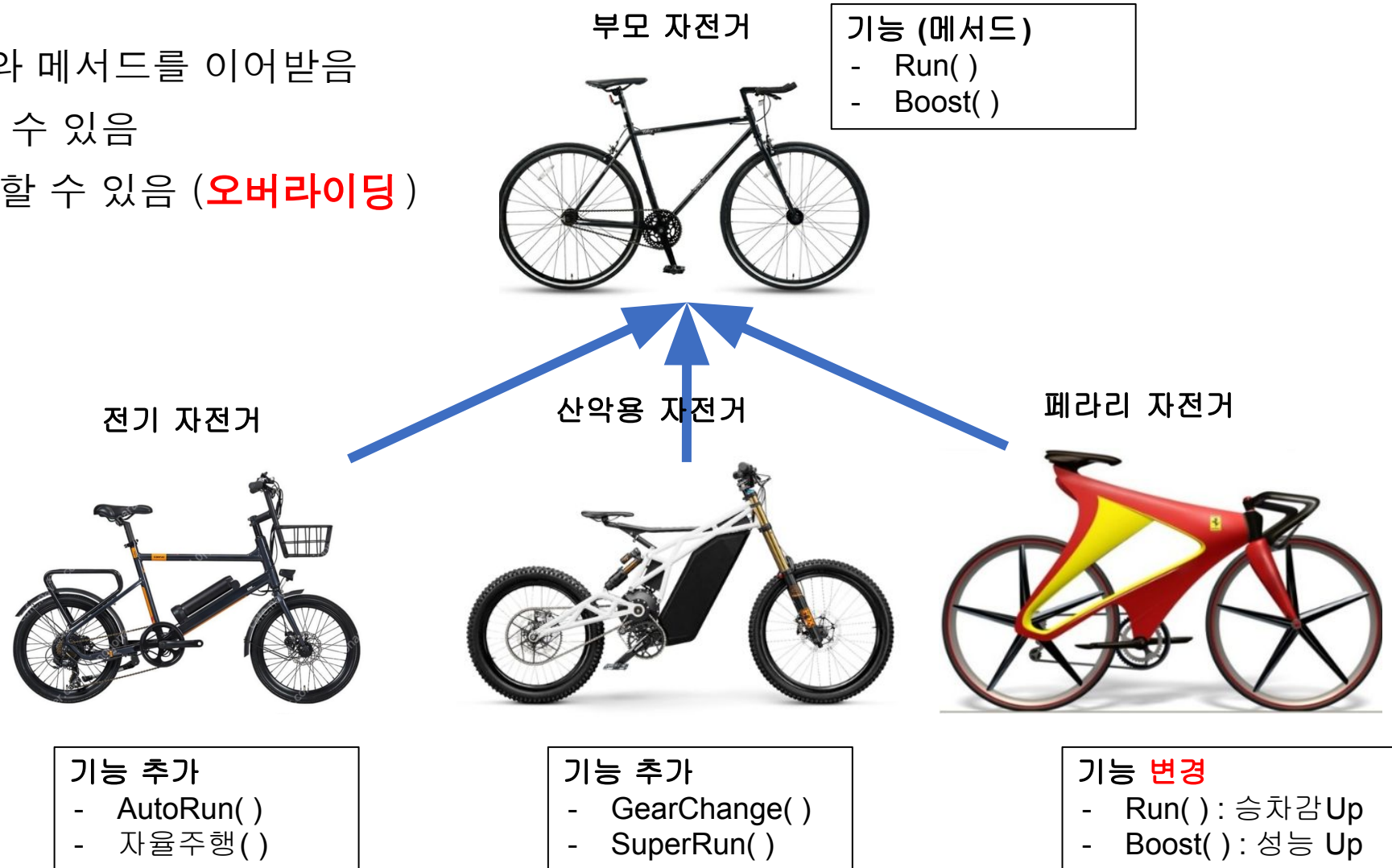
난다
슈퍼맨이 두박두박

상속 - 자전거 예시 살펴보기



상속으로 할 수 있는 것

- 부모 클래스의 필드와 메서드를 이어받음
- 메서드를 더 추가 할 수 있음
- 부모 메서드를 교체 할 수 있음 (**오버라이딩**)





간단하게 상속이 가능

- 마운틴 클래스는 super 라는 클래스를 상속받는다.
- 따라서 마운틴 클래스는
총 4개의 메서드를 가지고 있다.

1. run()
2. boost()
3. gear()
4. superrun()

- 페라리 클래스는 몇개의 메서드를 가지고 있는가?

```
bbq.py x
1 class super():
2     def run(self): print("RUN")
3     def boost(self): print("BOOST")
4
5 class elec(super):
6     def autorun(self): print("부양")
7     def automove(self): print("승")
8
9 class mountain(super):
10    def gear(self): print("충")
11    def superrun(self): print("영차")
12
13 class ferrari(super):
14    def run(self): print("RUNRUN")
15    def boost(self): print("BBB000ST")
16
17 f = ferrari()
18 f.run()
19 f.boost()
20
21 m = mountain()
22 m.run()
23 m.superrun()
```

RUNRUN
BBB000ST
RUN
영차

[도전] 상속 구현하기



타이어 구현하기

부모 타이어 메서드

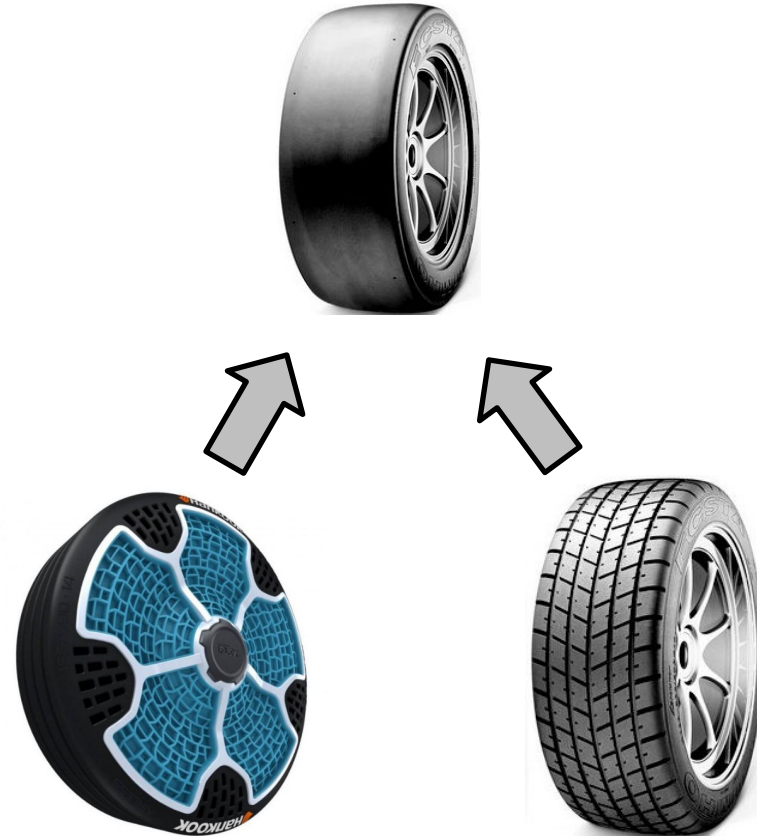
- run (“런” 출력)

자식 타이어1 메서드

- ran (“랜” 출력) – 메서드 추가

자식 타이어2 메서드

- run (“런런” 출력) – 오버라이딩
- ron (“론” 출력) – 메서드 추가





Project 02

파이썬과 Pandas를 사용한 데이터 처리 (데이터 사이언스 기초)



프로젝트 목표



✓ 데이터 사이언스에 대한 이해

✓ 캐글에 대한 이해

✓ 데이터 분석 파이썬 패키지(Numpy, Pandas, Matplotlib) 이해하기

✓ 실습 - Google 주식 데이터 활용하기

✓ 도전 과제 - Netflix 주식 데이터 활용하기

준비사항



개발도구

- ✓ Python 3.9+
- ✓ Jupyter notebook



준비사항

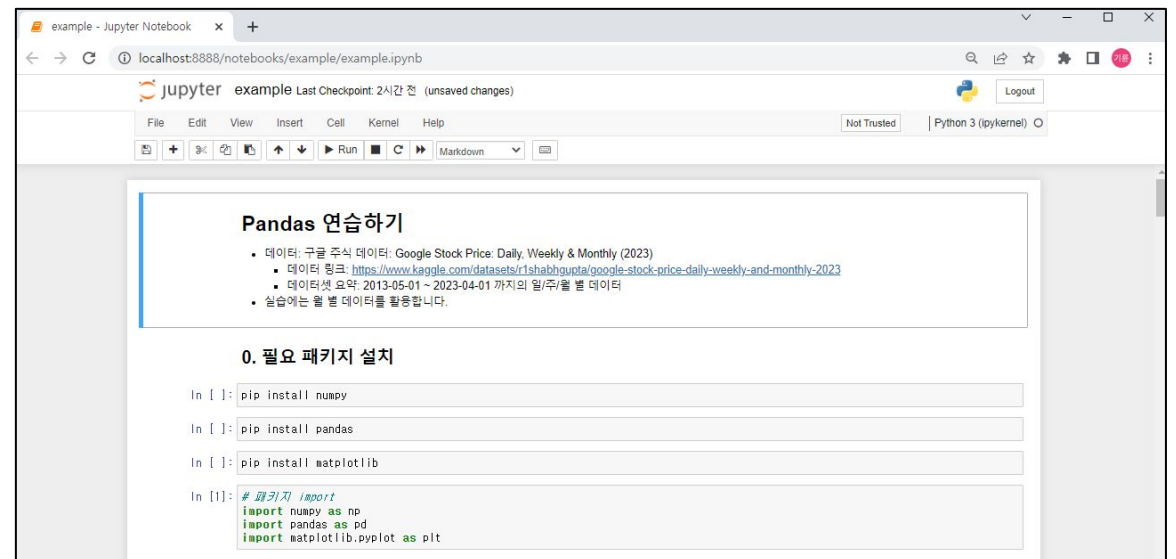


Jupyter notebook 이란 ?

- 데이터 사이언스 작업에 많이 활용되는 파이썬 개발 환경
- 웹 브라우저에서 실행
- 코드 실행, 텍스트 문서 작성, 시각화 등을 하나의 문서에 통합하여 작업 가능

데이터 사이언스 작업에 많이 쓰이는 이유

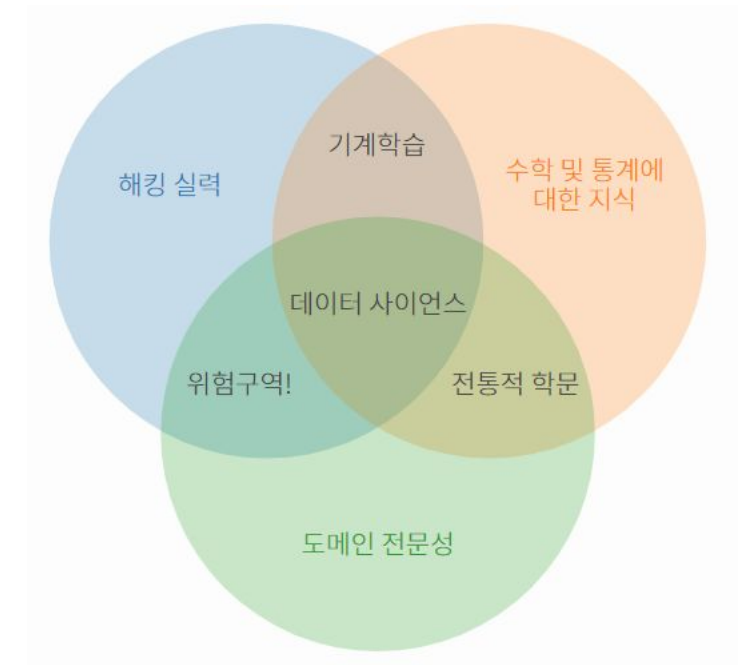
- 셀 단위 코드 실행으로 결과를 바로 확인할 수 있음
- 문서를 작성할 수 있는 마크다운 기능 제공
- 개별적인 코드 블록을 실행할 수 있음





데이터 사이언스

- 다양한 데이터로부터 문제 해결에 필요한 지식과 정보를 추출하기 위해 과학적 방법론, 프로세스, 알고리즘, 시스템을 동원하는 융합 분야
- 컴퓨터 과학, 통계학, 수학 등 다양한 학문의 원리와 기술을 활용





데이터 사이언스 프로세스

- 필요한 정보를 추출하는 5가지 단계
1. 문제 정의 : 해결하고자 하는 문제 정의
 2. 데이터 수집 : 문제 해결에 필요한 데이터 수집
 3. 데이터 전처리(정제) : 실질적인 분석을 수행하기 위해 데이터를 가공하는 단계
 4. 수집한 데이터의 오류 제거(결측치, 이상치), 데이터 형식 변환 등
 5. 데이터 분석 : 전처리가 완료된 데이터에서 필요한 정보를 추출하는 단계
 6. 결과 해석 및 공유 : 의사 결정에 활용하기 위해 결과를 해석하고 시각화 후 공유하는 단계

데이터 사이언스 프로세스 실습



프로세스 1. 문제 정의

- 실습에서 해결하고자 하는 문제는 다음과 같습니다.
- 구글의 주식 가격은 앞으로 어떻게 될까 ?





프로세스 2. 데이터 수집

- 주식 가격을 분석하기 위해서는 기간 별 주식 가격에 대한 데이터가 필요합니다.
 - 데이터 수집은 다양한 기술과 방법을 활용할 수 있습니다.
 - 웹 스크래핑 (Web Scraping): 웹 페이지에서 데이터를 추출하는 기술
 - 데이터 크롤링 (Data Crawling): 웹 페이지를 자동으로 탐색하고 데이터를 수집하는 기술
 - Open API 활용: 공개된 API 를 통해 데이터를 수집
 - 데이터 공유 플랫폼 활용: 다양한 사용자가 데이터를 공유하고 활용할 수 있는 온라인 플랫폼
- => 종류: 캐글(Kaggle), Data.world , 데이콘(Daicon), 공공데이터포털 등



프로세스 2. 데이터 수집 - 캐글(Kaggle)

- 데이터 분석 경진대회 플랫폼
- 기업 및 단체에서 데이터와 해결 과제를 등록하면, 데이터 과학자들이 이를 해결하는 방법을 개발하고 경쟁할 수 있는 플랫폼
- 경진 대회, 데이터셋 공유, 토론 등의 기능이 가능하며 많은 데이터 과학자와 분석가들이 활용함
- 실습을 위해 캐글의 구글 주가 데이터를 다운로드 받아 활용합니다.



프로세스 2. 데이터 수집 - 캐글(Kaggle)

- 회원가입 (<https://www.kaggle.com/>)
- 공식 사이트에 접속하여 회원가입을 진행합니다.

데이터 사이언스 프로세스 실습



프로세스 2. 데이터 수집 - 캐글(Kaggle)

- Google Stock Price 검색
- 실습에 활용할 데이터는 “Google Stock Price: Daily, Weekly & Monthly (2023)” 입니다.
- 데이터셋 요약: 2013-05-01 부터 최근까지 일/주/월 별 데이터

← Google Stock Price

people and society 159

packages 152


business 148

pandas 119

matplotlib 103

finance 117

google stock price

 Dataset

Google Stock Price: Daily, Weekly & Monthly (2023)

by r1shabhgupta

a month ago • 62 kB • ^ 35

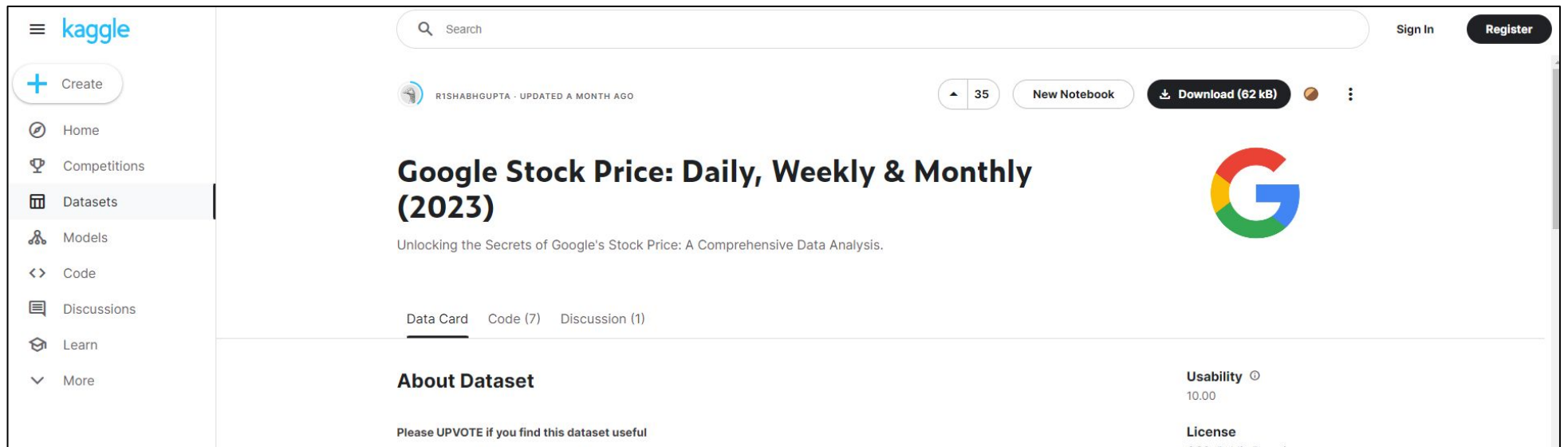
[Google Stock Price: Daily, Weekly & Monthly \(2023\)](#)

데이터 사이언스 프로세스 실습



프로세스 2. 데이터 수집 - 캐글(Kaggle)

- 데이터 다운로드
- 우측 상단의 **Download** 버튼을 클릭하여 데이터를 다운로드 받습니다.





프로세스 2. 데이터 수집 - 캐글(Kaggle)

- 압축 해제 후 실습을 진행할 폴더에 저장합니다.
- 파일 구조를 다음과 같이 만들어 줍니다.

```
실습 폴더 /  
  archive/  
    google-stock-dataset-Daily.csv/  
    google-stock-dataset-Monthly.csv/  
    google-stock-dataset-Weekly.csv/  
    google_stock_price_example.ipynb
```




[참고] csv란 ?

- 몇 가지 필드를 쉼표(,)로 구분한 텍스트 데이터 및 텍스트 파일
- 일반적으로 표 형식의 데이터를 CSV 형태로 많이 사용
- 저장, 전송 및 처리 속도가 빠르며, 처리 가능한 프로그램이 다양합니다.

예시

- 엑셀

이름	생년	월	일	성별	직업	사는 곳
홍길동	1992	7	17	남	강사	서울
희동이	1997	4	3	여	학생	대구
금땡구	1988	2	15	남	개발자	광주

이름,생년,월,일,성별,직업,사는 곳
홍길동,1992,7,17,남,강사,서울
희동이,1997,4,3,여,학생,대구
금땡구,1988,2,15,남,개발자,광주



프로세스 3. 데이터 전처리(정제)

- 데이터 전처리 단계를 분석을 진행하기 전 데이터를 정제하는 단계입니다.
- 다음과 같은 과정을 포함합니다.
- 불완전하거나 오류가 있는 데이터를 제거하여 데이터의 품질을 개선
- 중복 데이터 제거
- 분석하기 적절한 형식으로 데이터를 변환

데이터 전처리 및 분석에 사용되는 파이썬 패키지

- Numpy
- Pandas
- Matplotlib

데이터 사이언스 프로세스 실습



자주 활용되는 파이썬 패키지

- 데이터 전처리 및 데이터 분석 : Numpy, Pandas
- 데이터 시각화 : Matplotlib



Numpy

- 다차원 배열을 쉽게 처리하고 효율적으로 사용할 수 있도록 지원하는 파이썬 패키지

장점

- Numpy 행렬 연산은 데이터가 많을수록 Python 반복문에 비해 훨씬 빠르다.
- 다차원 행렬 자료 구조를 제공하여 개발하기 편하다.

특징

- CPython(공식 사이트의 Python)에서만 사용 가능
- 행렬 인덱싱(Array Indexing) 기능 제공

실습 파일: 1.Numpy_Basic.ipynb



Pandas

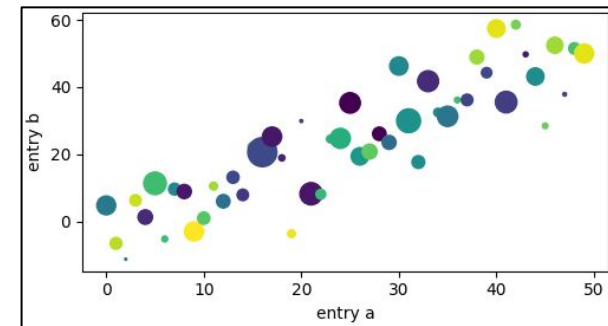
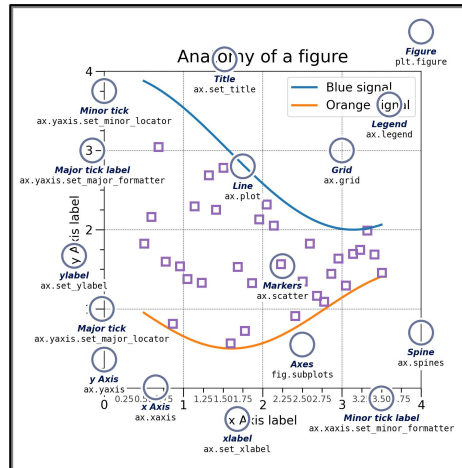
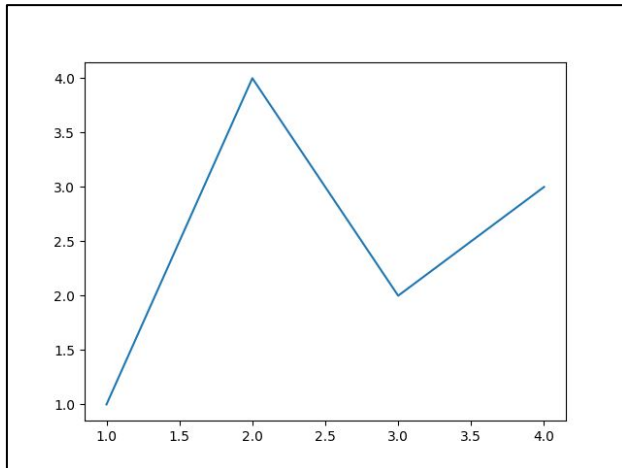
- Numpy 의 한계
 - => 유연성(데이터에 레이블을 붙이거나, 누락된 데이터로 작업)이 부족함
 - => 그룹화, 피벗 등 구조화가 부족함
- Pandas 는 마치 프로그래밍 버전의 엑셀을 다루듯 고성능의 데이터 구조를 만들 수 있음
- Numpy 기반으로 만들어진 패키지로, **Series**(1차원 배열) 과 **DataFrame**(2차원 배열) 이라는 효율적인 자료구조 제공

실습 파일: 2.Pandas_Basic.ipynb
3.Pandas_Advanced.ipynb



Matplotlib

- Python 에서 데이터 시각화를 위해 가장 널리 사용되는 라이브러리
- 다양한 종류의 그래프와 도표를 생성하고 데이터를 시각적으로 표현할 수 있습니다.



실습 파일: 4.matplotlib_basic.ipynb



실습

1. 캐글을 활용하여 데이터를 다운로드 받아 활용합니다.

- 데이터셋: “Google Stock Price: Daily, Weekly & Monthly (2023)(구글 주식 데이터)”
- 데이터셋 요약: 2013-05-01 ~ 최근까지의 일/주/월 별 데이터

2. 실습 파일명: `google_stock_price_example.ipynb`

3. 데이터 전처리 및 시각화 연습

요구사항 - 공통 요구사항



공통 요구사항

캐글을 활용하여 데이터를 다운로드 받아 활용합니다.

- 데이터셋: “Netfilx Stock Price Prediction(넷플릭스 주식 가격 데이터)”
- 데이터셋 요약: 2018-02-05 ~ 2022-02-04 까지의 일별 데이터

명시된 요구사항 이외에는 자유롭게 작성해도 무관합니다.



세부 요구사항

- A. 데이터 전처리 – 데이터 읽어오기
 - B. 데이터 전처리 – **2021년** 이후의 증가 데이터 출력하기
 - C. 데이터 분석 – **2021년** 이후 최고, 최저가 출력하기
 - D. 데이터 분석 – **2021년** 이후 월 별 평균 증가 출력하기
 - E. 데이터 시각화 – **2022년 1월** 이후 월 별 최고, 최저, 증가 시각화
-
- [참고] 출력 결과는 데이터를 다운로드 받는 시기에 따라 다르게 나올 수 있습니다.



A. 데이터 전처리 - 데이터 읽어오기

- Pandas 를 사용하여 csv 파일(NLFX.csv)을 DataFrame 으로 읽어옵니다.
- 이 때, ['Data', 'Open', 'High', 'Low', 'Close'] 필드만 읽어오도록 구성합니다.
- 출력 결과 예시

```
In [3]: # DataFrame 출력  
df
```

```
Out[3]:
```

	Date	Open	High	Low	Close
0	2018-02-05	262.000000	267.899994	250.029999	254.259995
1	2018-02-06	247.699997	266.700012	245.000000	265.720001
2	2018-02-07	266.579987	272.450012	264.329987	264.559998
3	2018-02-08	267.079987	267.619995	250.000000	250.100006
4	2018-02-09	253.850006	255.800003	236.110001	249.470001
...
1004	2022-01-31	401.970001	427.700012	398.200012	427.140015
1005	2022-02-01	432.959991	458.480011	425.540009	457.130005
1006	2022-02-02	448.250000	451.980011	426.480011	429.480011
1007	2022-02-03	421.440002	429.260010	404.279999	405.600006
1008	2022-02-04	407.309998	412.769989	396.640015	410.170013

1009 rows × 5 columns



B. 데이터 전처리 - 2021년 이후의 종가 데이터 출력하기

- csv 파일을 **DataFrame** 으로 읽어와 **2021년** 이후의 데이터만 필터링합니다.
=> [힌트] 필터링이 가능한 형식으로 데이터 타입을 변경한 후 필터링을 진행합니다.
=> **Pandas** 의 **to_datetime()** 을 활용합니다.
- 필터링이 완료된 **DataFrame** 의 종가 데이터를 **Matplotlib** 를 사용하여 시각화합니다.
- 출력 결과 예시





c. 데이터 분석 - 2021년 이후 최고, 최저 종가 출력하기

- csv 파일을 DataFrame 으로 읽어와 2021년 이후의 데이터만 필터링합니다.
- 종가(Close) 필드를 활용하여, 2021년 이후 가장 높은 종가와 가장 낮은 종가를 출력합니다.
- Pandas 의 내장 함수를 사용합니다.
- 출력 결과 예시

```
In [10]: print("최고 종가:", max_price)
         print("최저 종가:", min_price)
```

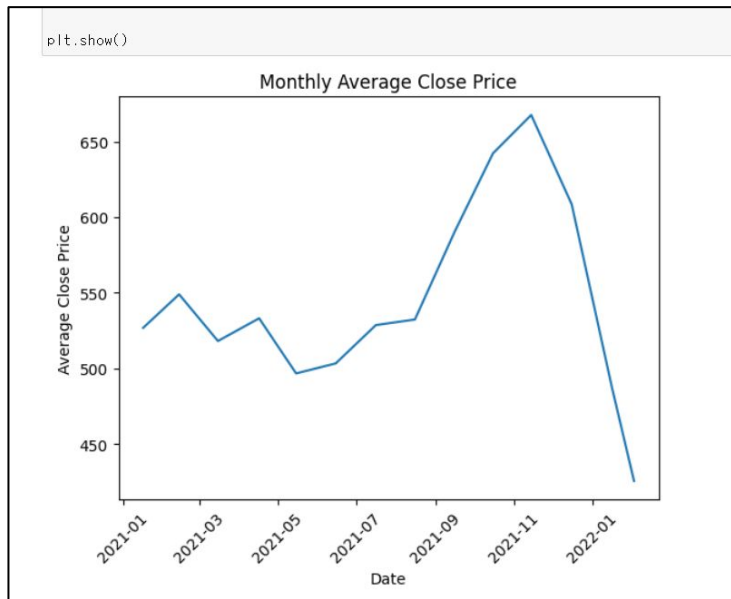
최고 종가: 691.690002

최저 종가: 359.700012



D. 데이터 분석 - 2021년 이후 월 별 평균 종가 출력하기

- csv 파일을 **DataFrame** 으로 읽어와 2021년 이후의 데이터만 필터링합니다.
- 월 별로 그룹화하여 평균 종가를 계산한 새로운 **DataFrame** 을 만들어 그래프로 시각화합니다.
- 출력 결과 예시





E. 데이터 시각화 - 2022년 이후 최고, 최저, 종가 시각화하기

- csv 파일을 `DataFrame` 으로 읽어와 2022년 이후의 데이터만 필터링합니다.
- `Matplotlib` 를 활용하여 3가지 필드를 한 번에 분석할 수 있도록 아래와 같이 시각화합니다.
- 출력 결과 예시

