목차

회독

1. 형광펜 내용 복기
2. pandas, seaborn, matplotlib, wordcloud, numpy 함수들 안 보고 쓰기

1. 파이썬 기초

1. 기초 문법

컴퓨팅적 사고 수업 내용까지 공부해 두기.

1. for 반복문

list의 각 원소들이 순서대로 for 옆의 변수에 배정되며 루프가 도는 반복문.

```
for <변수> in <list>:
        명령문1
        명령문2
```

1) 형식

변수에는 일반적인 변수를 작성함.

들여쓰기로 포함되는 명령들을 구분함. (tab키)

아무 명령도 작성하지 않으면 오류 발생함.
대신 pass 키워드를 작성하면 됨.

2. while 반복문

```
while <조건> :
        명령문1
        명령문2
```

명시한 조건이 참이면 루프를 돌아 명령들을 수행하는 반복문.

1) 형식

들여쓰기로 포함되는 명령들을 구분함. (tab키)

조건에는 괄호를 씌워도 되고 안 씌워도 됨.

아무 명령도 작성하지 않으면 오류 발생함.
대신 pass 키워드를 작성하면 됨.

2) 무한 루프

조건에 1이나 True를 작성하여 무한루프를 만들 수 있음.

3. if 조건문

조건이 참인 경우 명령을 실행하는 조건문.

오른쪽에 명시한 if의 형식을 지키면서 c에서와 동일하게 else와 else if를 사용할 수 있음.
else는 그대로 else로, else if는 줄여서 elif로 표현함.

1) 형식

```
if <조건> :
        명령문1
        명령문2
```

들여쓰기로 포함되는 명령들을 구분함. (tab키)

조건에는 괄호를 씌워도 되고 안 씌워도 됨.

아무 명령도 작성하지 않으면 오류 발생함.
대신 pass 키워드를 작성하면 됨.

if 와 else, elif는 같은 레벨이어야 함. -> 들여쓰기 상에서 일치해야 의도대로 기능함.

4. 문자열 관련 함수들

1) len(<문자열>)

인자에 문자열을 지정하면 그 문자열의 길이를 리턴하는 함수.
(문자열을 직접 작성하는 경우 따옴표 씌움.)

2) <변수>.lower() / <변수>.upper() / <변수>.capitalize()

변수에 저장된 문자열의 대/소문자 전환 함수.

<변수>.lower() -> 소문자로 전환
<변수>.upper() -> 대문자로 전환
<변수>.capitalize() -> 문장 시작 단어의 첫 글자만 대문자로 전환.

메모 포함[이1]: 괄호 빠뜨리지 않도록 주의하기. 변수. 형태로 쓴다고 함수가 아닌 것은 아니다.

3) <변수>.strip()

변수에 저장된 문자열의 불필요한 공백을 제거하는 함수.

4) <변수>.replace(<문자열 1> , <문자열 2>)

변수에 저장된 문자열에서 문자열 1 을 문자열 2 로 바꾸는 함수.

메모 포함[이2]: 문자열이므로 당연히 따옴표로 묶어서 작성해야 한다.

5. 문자열 검색 문법들

1) in 연산자

해당 변수에 저장된 문자열에 지정한 문자/문자열이 있으면 1 을, 없으면 0 을 리턴하는 연산자.

```
<문자> in 변수
```

2) find() 함수

해당 변수에 저장된 문자열에서 지정한 문자가 몇 번째 인덱스에 위치하는지를 리턴하는 연산자.
인덱스 번호를 리턴함.

```
변수.find(<문자>)
```

6. 라이브러리 관련 기능들

1) pip list

현재 설치된 라이브러리들을 출력하는 명령어.

jupyter notebook, colab 에서 사용 가능.

2) pip install <라이브러리>

라이브러리를 설치하는 명령어.

2. list

1. list

파이썬의 기초 자료형. (linked list 로 구성되어 있음.)

한 번에 여러 개의 값들을 다룰 수 있게 함.

list 개념은 데이터 분석, AI 등에서 사용됨.

2. list 생성

수열(list)을 만드는 방법은 단순 명시, list(), range()로 세 가지가 있음. (수식까지 하면 4 가지)

list 는 출력 시 대괄호([])를 씌워서 출력됨.

list 안에 list 가 원소로 들어갈 수도 있음.

list 에는 문자열, 소수, 정수 등의 값이 들어갈 수 있음.
하나의 list 에 다른 자료형의 값들이 들어갈 수도 있음.

1) 단순 명시

별다른 형식 없이 명시하는 것만으로도 list 를 생성할 수 있음.
약식 표현임.

[<값>, <값>, . . .]

각괄호 안을 비워 두면 비어 있는 list 가 만들어짐.

2) list()

수열 생성 함수.
리스트를 생성하는 명령을 파라미터로 작성함.

list([<값>, <값>, . . .])
list(range(<값>, <값>, <값>))

list()의 파라미터로 아무것도 작성하지 않으면 비어 있는 list 가 만들어짐.

list() 함수를 사용하는 것이 정식 표현임.
list() 함수를 사용해야 하는 경우들이 있음.

메모 포함[이3]: 정확히 같은 지는 모르겠지만 일단 이 필기에서는 같은 것으로 취급한다.

메모 포함[이4]: 생성한 다음에 변수에 배정을 해 줘야 사용할 수 있다.

메모 포함[이5]: ex. [1, 2, [1, 2, 3]]

메모 포함[이6]: ex. [1, "Hello~!!", 2, 3.3]

메모 포함[이7]: 비어 있는 list를 만든 후에 append() 등의 함수를 사용하여 값을 넣어줄 수 있다.

메모 포함[이8]: ex.
range()함수로 생성한 list를 변수에 배정하려면 list()안에 넣어서 배정해야 한다..

3) range()

특정 패턴을 갖는 수열(리스트)을 만드는 함수.

range(시작값, 종료값, 증가값)

각 인자에는 정수를 적음. -> 실수는 적을 수 없음.

인자의 순서는 c 언어의 for 와 유사함.

시작값부터 시작해서 종료값보다 작은 값까지 증가값만큼 커지는 수열을 생성함.

종료값은 포함되지 않음.

시작값을 생략하면 default 로 0 이 적용됨.

증가값을 생략하면 default 로 1 이 적용됨.

종료값은 생략할 수 없음.

이를 이용해서 range(<숫자>) 형식으로 자주 쓰임.

인자를 두 개만 작성하면 각각 앞부터 적용되어 시작값과 종료값으로 취급됨.

메모 포함[이9]: ex. range(1, 11, 1)
-> [1, 2, 3, 4, 5, 6, ,7 ,8 9, 10]

range(a, b, 1)은 c언어의
for(int i = a; i < b; i++)과 유사함.

메모 포함[이10]: ex. range(5) 하면 [0, 1, 2, 3, 4]가 됨.

3. list 의 인덱스

인덱스를 사용하여 변수에 배정한 list 의 여러 원소 중 하나를 특정하여 사용하거나 배정 수 있음.

c 의 배열처럼 인덱스는 0 부터 시작함.

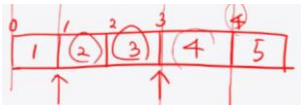
<변수이름>[<인덱스>]

4. slicing

list를 쪼개서 사용하는 것.

<변수이름>[<인덱스>:<인덱스>]

명시한 인덱스 부분만큼의 원소들만 가져와서 사용
앞에 명시한 인덱스부터 시작해서 뒤에 인덱스 바로 앞 것까지 포함함.
-> 인덱스 적용 방식은 오른쪽 그림 참고.
-> range에서도 종료값은 포함하지 않는 것처럼, 여기서도 종료 값을 포함하지 않음.



인덱스는 0부터 시작하는 것 유의.

앞에 인덱스를 생략하면 리스트의 맨 앞부터 적용됨.
뒤에 인덱스를 생략하면 리스트의 끝까지 적용됨.

5. list 에 데이터 추가하기

list 에 데이터를 추가하는 함수에는 Append, insert, extend 등이 있음.

1) append()

list의 맨 뒤에 항목을 추가하는 함수.

<변수이름>.append(<값>)

변수 이름에 list를 배정한 변수의 이름을 작성함.
값에 추가할 값을 작성함.

2) insert()

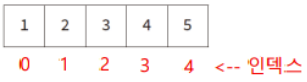
list 중간에 항목을 추가하는 함수.

<변수이름>.insert(<위치>, <값>)

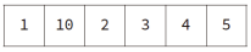
작성한 값이 list 의 중간에 끼워지게 됨.
명시한 인덱스 위치에 값이 들어가고, 기존에 저장되어 있던 값들은 해당 위치부터 한 칸 씩 뒤로 밀림.

위치로 인덱스를 명시함.
slicing 과 동일한 인덱스 적용 방식을 가짐.

(ex.




```
append(1, 10)
```



)

3) extend()

두 개의 list 을 연결하는 함수.
해당 변수의 값을 확장하는 것.

<변수1>.extend(<변수2>)

변수 1 에 저장된 list 의 뒤에 변수 2 에 저장된 list 가 연결되어 변수 1 에 저장됨.

6. list 의 데이터 지우기

1) <변수>.remove(<값>)

해당 변수에 저장되어 있는 list 에서 인자에 작성한 값을 찾아서 제거하는 함수.

해당 값이 2 개 이상인 경우, 앞의 값만 제거됨.

2) <변수>.pop(<인덱스 값>)

해당 변수에 저장되어 있는 list 에서 인자에 작성한 인덱스 값에 해당하는 위치의 값을 제거하는 함수.

+ list 와 사칙연산

1. 두 list 더하기

더하기 수식은 두 list 를 연결한 값을 가짐.

2. 상수 곱하기

list 에 상수를 곱한 수식은 해당 list 가 상수만큼 반복되는 형태의 값을 가짐.

빼기와 나누기는 안 되는 것으로 보임.

(ex.

```
>>> a + b
[10, 20, 30, 11, 22, 33]

>>> a * 2
[10, 20, 30, 10, 20, 30]
```

)

7. c 언어 배열과 파이썬 list 의 차이점

1) c 언어의 배열

장점

1. 구현이 쉬움.
2. 인덱스가 있어 빠르게 조회할 수 있음.
3. 연속된 메모리 공간이 할당되어 접근이 빠름
4. 참조를 위한 추가 메모리 할당이 필요 없음

단점

1. 삽입/삭제 시 뒤에 요소들을 이동해야 해서 비효율적임.
2. 선언 시에 지정한 메모리 크기를 변경할 수 없음.
3. 메모리 재사용이 불가능함. (초기 사이즈만큼 할당되고, 데이터가 없더라도 메모리를 차지하고 있음.)

2) 파이썬의 list

장점

1. 삽입/삭제 시 전후 노드의 참조 관계만 수정하면 되기 때문에 효율적임.
2. 크기가 고정적이지 않음.
3. 메모리를 재사용할 수 있음. (삭제 시 해당 노드의 참조가 사라져 GC 에 의해 가용 메모리로 전환됨.)

단점

1. 구현이 상대적으로 복잡함.
2. 연속된 메모리 공간이 할당되지 않아 검색이 비효율적임. (처음부터 순차적으로 검색해야 함)
3. 참조를 위한 메모리가 필요함.

3) 결론

배열 -> 저장할 데이터의 개수가 정해져 있음.
-> 삽입, 삭제 작업이 적음.
-> 특정 위치의 데이터를 조회하는 작업이 많음.

list -> 저장할 데이터의 개수가 미정임.
-> 삽입, 삭제 작업이 많음.
-> 특정 위치 데이터를 조회하는 경우가 별로 없음.

2. AI 기초

1. AI 기초

1. AI 의 발전 역사

1950 년 전후 -> 컴퓨터, 인공지능의 시작.
1950 년대 -> 머신러닝/인공지능 용어 등장.
1960 년 전후 -> 인공지능 황금기(엄청난 투자), 수많은 알고리즘 개발(ex. 퍼셉트론)
1970 년 전후 -> 인공지능 침체기
1970 년대 초반 -> 다양한 머신러닝 알고리즘 개발(ex. 진화 알고리즘)
1980 년 전후 -> 전문가 시스템 시작.
1980 년대 후반 -> 신경망의 탄생.
2000 년 전후 -> 경영분야에 데이터마이닝 등 적용.
2006 년 -> 딥러닝 알고리즘 개발
2015 년 ~ -> 인공지능이 현실 곳곳에 녹아 듦.

컴퓨터 시작 -> 인공지능 투자 상승 -> 인공지능 침체 -> 인공지능 부활

메모 포함[이11]: 인공 신경망 알고리즘.
전문적인 알고리즘으로 개발된 것이 아닌, 이론으로만 발표되었다.
딥러닝의 초기 모델.

메모 포함[이12]: 특정 분야들에 특화된 인공지능 개발.

메모 포함[이13]: 퍼셉트론 알고리즘을 개선한 것.
딥러닝 관련 알고리즘.

2. 인공지능에 기여한 과학자

1) 앨런 튜링

컴퓨터과학/인공지능의 아버지.

세계 2 차대전 당시 독일군의 암호를 해독하여 전쟁 종결에 기여함.
초기 컴퓨터 개발에 크게 기여함.
1950 년에 컴퓨터가 지능을 가지게 될 것이라고 전망, 튜링 테스트를 개발함.

메모 포함[이14]: 컴퓨터의 지능을 측정하는 기법.

3. 인공지능 분야에서 일어난 사건들

1) Shakey

1966 년에 개발된 최초의 인공지능 자율주행 로봇.

메모 포함[이15]: 인공지능 분야에 막대한 투자가 이루어지던 시기.

2) 체스, 퀴즈, 바둑

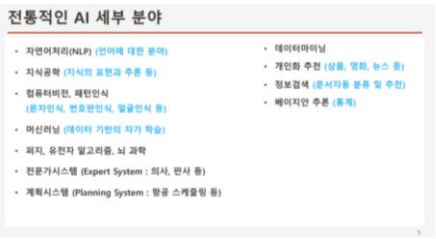
1997 년에 개발된 'IBM 딥 블루'는 세계 체스 챔피언과의 대국에서 승리.

2011 년에 개발된 'IBM 왓슨'은 미국의 유명 퀴즈쇼에 출연하여 우승.

2016 년에 개발된 'Google 알파고'는 이세돌 9 단과의 대국에서 승리.

4. 인공지능 관련 세부분야

메모 포함[이16]: 암기 X.



+ 컴퓨터과학의 분야들

메모 포함[이17]: 암기 X.

DB, OS, Software Engineering, 임베디드, 보안, AI 등.

5. 인공지능 관련 용어

1) 인공지능(AI, Artificial Intelligence)

컴퓨터가 사람처럼 생각하고 배우고 스스로 판단하고 행동하는 능력을 갖는 것. 또는 갖도록 연구하는 학문.

2) Strong AI, Weak AI, Super AI

Strong AI (강인공지능) -> 초창기 인공지능 연구자들이 목표했던 수준.
-> 로봇이 사람처럼 생각하고 행동하는 수준.

Weak Ai (약인공지능) -> 특정 분야에서만 문제를 해결할 수 있는 정도의 수준.
-> 현재 개발된 모든 AI 시스템은 Weak AI 에 포함됨.

Super Ai (초인공지능) -> 인간의 수준을 뛰어 넘는 수준.
-> AI 가 인간의 수준을 뛰어넘는 시점을 '특이점'이라고 함.

6. 머신러닝 알고리즘의 분류

1) 지도학습(교사학습) (Supervised Learning)

입력 데이터를 제공하고, Label(정답)이 존재하는 학습 방법.

2) 비지도학습(비교사학습) (Unsupervised Learning)

입력 데이터를 제공하고, Label(정답)이 존재하지 않는 학습 방법.

데이터를 구성된 방식으로 군집화함.

3) 강화학습(Reinforcement Learning)

주어진 환경에 따라서 반복 경험을 통해 데이터를 스스로 생성해 학습하는 학습 방법.

입력 데이터를 제공하지 않고, Label(정답)이 존재하는 학습 방법.

메모 포함[이18]: 지도학습을 마치면 비지도학습으로 넘어갈 수 있다.

메모 포함[이19]: 비지도학습에서는 결과를 맞히지는 않지만, 표본을 계속 쌓으며 개선되기 때문에 학습인 것이다.

2. 데이터 기초

1. 머신러닝 과정에서 데이터의 역할

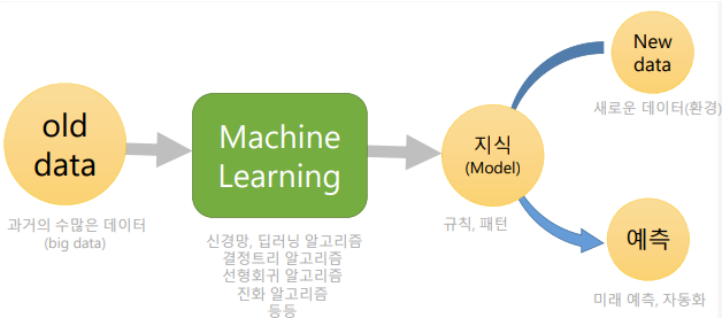
1) 머신러닝 과정

머신러닝 알고리즘을 구동하는 것.

머신러닝 알고리즘을 구동하기 위해서는 수많은 데이터 입력이 필요함.
머신러닝 알고리즘을 통해 만들어진 '지식'에 새로운 데이터를 입력하면 지식에 따라 답을 출력함.

2) 머신러닝 과정에서 데이터의 역할

1. 머신러닝 알고리즘으로의 입력으로 사용됨.
2. 머신러닝 알고리즘을 통해 출력된 '지식'에 새로운 데이터를 입력하는 데 사용됨.



2. 핵심 용어와 데이터의 구조

1) 핵심 용어

레코드(행) : 하나의 단위로 취급되는 데이터의 집합.

칼럼(열) : 표의 세로축, 열에 해당되는 호칭.

수치형 칼럼 : 숫자로 이루어진 칼럼.

범주형 칼럼 : 숫자 외의 텍스트로 이루어진 칼럼.

행 : 가로

열 : 세로.

(ex.

이름	나이	주소
Kim	20	서울시
Lee	25	경기도
Park	22	제주도
Choi	19	강원도
Song	21	인천시

5개의 레코드, 3개의 칼럼

5개의 행(row), 3개의 열(column)

)

2) 데이터의 구조

데이터는 레코드, 칼럼 등으로 구성됨.

+ 분야에 따른 핵심 용어

분야에 따라 의미는 같지만 핵심 용어는 다를 수 있음.

(ex.

)

메모 포함[이20]: 제목도 하나의 레코드로 인식하기도 한다. 하지만 이것은 특수한 경우이고, 별 말이 없으면 제목도 레코드로 치진 않는다.

메모 포함[이21]: 칼럼은 '필드'라고도 한다.

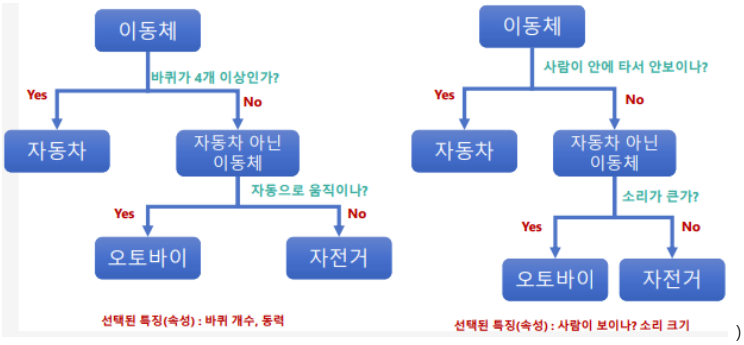
메모 포함[이22]: 수업에서 어떤 구조를 설명해 주지는 않았다.
수업에서 계속 등장하는 표를 기억해 두면 될 듯 하다.

3. 머신러닝의 원리 (Decision Tree 알고리즘)

Decision Tree 알고리즘에서, 머신러닝을 통해 얻은 지식은 Tree의 형태로 저장됨.
방식에 따라서 Tree의 형태는 다를 수 있음.

데이터를 제공하고, 그것을 분석하는 과정을 반복함으로써 지속적으로 Tree를 생성하고 수정함.

(ex.



)

1) Colab에서 Decision Tree 알고리즘의 구현 과정

1. 데이터 전처리
2. 코드 작성 (암기 X.)

+ 주피터 노트북과 Colab

1. 주피터 노트북
주피터노트북 실행 시 뜨는 웹 창에 명령을 작성하면 그 명령들이 콘솔 창으로 보내지고, 수행되어 결과가 웹으로 다시 보내져, 웹에서 출력되는 방식이다.

2. Colab
주피터노트북 실행 시 뜨는 웹 창의 역할을 구글이 인터넷에서 해 줌.
설치가 필요하지 않아 편리할 수 있지만, 로컬파일 업로드가 굉장히 불편함.

3. Pandas

1. Pandas (Panel Data Analysis)

데이터 분석을 위한 라이브러리.

import pandas as pd

1. 데이터를 프레임화(일반화)하는 기능을 가짐.
2. 대용량 데이터를 빠르게 자동 분석하고 처리할 수 있게 함.

아나콘다를 설치하면 함께 자동 설치됨.

메모 포함[이23]: 데이터를 특정 프레임으로 재구성한다.
특히 표로 재구성한다.

메모 포함[이24]: 엑셀 등을 사용할 때는 데이터가 많으면 속도가 크게 느려진다.

2. Pandas에 데이터 직접 입력하기

실제로는 직접 입력해서 사용하진 않음.

1. import pandas as pd
2. 변수에 데이터 생성. -> (오른쪽 그림) (레코드 단위로 작성)
3. <변수1> = pd.DataFrame(<변수2>)

```
stdData = [ ['kim', 20, '서울시'],
            ['lee', 25, '경기도'],
            ['park', 22, '제주도'],
            ['choi', 19, '강원도'],
          ]
```

1) pd.DataFrame(<데이터>)

매개변수로 작성한 변수에 저장된 데이터를 DataFrame 자료형으로 바꾸어 리턴하는 함수.
리턴값을 다른 변수(주로 식별자는 df)에 저장하여 사용함.

3. Pandas 로 데이터 준비하기

1) 데이터 관련 함수들

pd.read_excel('<파일 경로>')

엑셀 파일을 불러와서 DatatFrame 자료형으로 데이터를 리턴하는 함수.

pd. read_csv('<파일 경로>')

csv 파일을 불러와서 데이터를 리턴하는 함수.

맨 뒤 인자에 header=none을 작성. -> 데이터의 첫 줄에 칼럼명(헤더)이 존재하지 않음을 의미.
-> 첫 줄부터 들어감.

맨 뒤 인자에 delimiter='<값>'을 작성 -> 해당 값으로 데이터들을 구분함.

2) 파일 경로의 사용

경로는 해당 파일을 우클릭해 '속성' 메뉴로 알 수 있음.

속성에는 해당 파일이 존재하는 폴더까지만 표시되기 때문에 '/<파일>.<확장자>'를 추가로 작성해줘야 함.
파이썬에서 사용할 때는 ₩기호를 /로 바꿔 줘야 함.

앞에 있는 c:도 그대로 붙여 줘야 함.

4. 데이터 출력하기

seaborn 등의 데이터도 이 함수들로 출력이 가능함.

<변수>.info() -> 해당 변수에 저장된 데이터에 대한 상세정보를 출력하는 함수.

<변수>.describe() -> 해당 변수에 저장된 데이터 중 분석이 가능한 데이터의 통계를 출력하는 함수.

<변수>.head() -> 데이터의 앞쪽 레코드들만 출력하는 함수.
-> 기본 5개, 매개변수에 정수 입력하면 해당 개수만큼 출력함.

<변수>.tail() -> 데이터의 뒤쪽 레코드들만 출력하는 함수.
-> 기본 5개, 매개변수에 정수 입력하면 해당 개수만큼 출력함.

메모 포함[이25]: <변수>.to_excel("<파일 경로>")에서도 그렇듯이, 파일 경로를 작성할 때는 따옴표로 묶어 준다.

메모 포함[이26]: 파일 내에서 데이터는 , / tab 등 다양한 기준으로 구분되어 있기 때문에 어떤 것을 사용하고 있는지를 명시해 줘야 한다.

메모 포함[이27]: = 수치형 데이터.

5. 특정 데이터만 출력하기

특정 레코드 출력과 특정 칼럼 출력은 혼용 가능함.
(ex. d[2:10]['나이'], d[2:10].나이)

1) 특정 레코드만 출력하기

데이터가 들어 있는 변수를 슬라이싱하여 특정 레코드들만 출력할 수 있음.
슬라이싱하면 레코드 단위로 나뉨.

2) 특정 칼럼만 출력하기

데이터가 들어 있는 변수명 뒤에 .<칼럼명> 또는 ['<칼럼명>']을 작성해서 특정 칼럼만 출력할 수 있음.

.<칼럼명> -> 약식.

['<칼럼명>'] -> 정식. 칼럼명에 띄어쓰기가 있는 경우 정식 표기법만을 사용할 수 있음.

-> 칼럼명이 파이썬의 기본 식별자인 경우, 정식 표기법을 사용해야 함. (ex. class)

+ <변수명>.dropna()

해당 변수의 NaN 들을 제거하는 함수.

엑셀 파일을 불러와서 저장할 때 빈 칸이 있으면 그 칸은 NaN 이라는 값이 들어감.
정수만을 사용해야 하는데 NaN 이 있으면 오류가 발생함.
.dropna() 함수를 사용하여 NaN 들을 제거할 수 있음.

.<칼럼명> 등을 사용한 변수에도 그냥 제일 뒤에 작성하면 됨.

+ 이터레이터 객체로 데이터 가져오기 (csv)

이터레이터 객체 : for문을 써서 데이터를 하나씩 처리할 수 있는 데이터.

csv로 가져온 데이터는 이터레이터 객체로 가져와 짐.

f = open("<파일 경로>") -> 파일을 열기.
d = csv.reader(f) -> 연 파일을 csv로 읽기.
next(d) -> 첫 번째 인덱스 값을 지우는 것. (칼럼명 제외 필요.)

d의 인덱스에는 레코드 단위로 데이터가 저장되어 있음. -> d[0]는 첫 번째 레코드. d[1]은 두 번째 레코드.

3. 데이터 시각화

1. seaborn

import seaborn as sns -> seaborn 을 import 할 때는 matplotlib 도 함께 import 하는 것이 일반적임.

메모 포함[이28]: seaborn 라이브러리는 통상적으로 sns로 바꾸어 사용한다.

1. seaborn 예시 데이터

seaborn 라이브러리에서 tips, iris, titanic 등의 seaborn 예시 데이터를 불러와서 사용할 수 있음.

메모 포함[이29]: 데이터를 시각화해주는 기능을 가지고 있는 라이브러리이다. 테스트를 위한 여러 기본 데이터들도 내장하고 있다.

1) 예시 데이터 불러오기

<변수> = sns.load_dataset("<데이터명>")

메모 포함[이30]: seaborn 내의 특정 데이터를 불러오는 명령이다.

tips, iris, titanic, flights, fmri 등의 데이터명 작성 가능
<변수>에 데이터가 저장됨.

+ APT 실거래가 데이터 수집 방식

1. 데이터 확보 -> 인터넷 다운로드.
2. 데이터 전처리 -> 분석이 가능하도록 데이터 다듬기.
3. 데이터 분석 -> 데이터 시각화.

+ seaborn 과 matplotlib

데이터 시각화 라이브러리로는 주로 matplotlib 과 seaborn 을 사용한다.
seaborn 은 특히 AI 학습 시 많이 사용한다. 데이터 시각화에는 matplotilb 이 더 효과적임.

2. seaborn 관련 함수들

1) sns.relplot(data=<변수명>, x='<칼럼명>', y='<칼럼명>');

작성한 변수명에 저장된 데이터에서, 두 칼럼 사이의 관계를 여러 요소들을 고려하여 scatter(기본) 차트를 생성하는 함수.
데이터에 해당 칼럼이 존재해야 함.

; 를 작성하여 출력을 완료함. (plt.show()와 동일한 기능.)
함수는 내부적으로 matplotlib 을 사용하기 때문에, matplotlib 을 import 하여 plt.show()를 사용하는 것이 일반적임.

고급 옵션들. (맨 뒤 인자에 작성하는 것들.)

1. hue='<칼럼명>' -> 해당 칼럼을 기준으로 색을 나눔.
2. style='<칼럼명>' -> 해당 칼럼을 기준으로 marker 를 임의의 모양으로 구분함.
3. size='<칼럼명>' -> 해당 칼럼을 기준으로 marker 를 임의의 크기로 구분함.
4. kind='<차트 종류>' -> 지정한 종류로 차트를 생성함. 지정하지 않는다면 scatter 차트가 생성됨.
   -> line 을 작성하면 line 차트를 생성함.
   -> 차트 종류는 따옴표로 씌워서 지정해야 함. (문자열임.)

2) sns.lineplot(data=<변수명>, x='<칼럼명>', y='<칼럼명>');

작성한 변수명에 저장된 데이터에서, 두 칼럼 사이의 관계를 line 차트로 생성하는 함수.

+ 데이터를 엑셀 파일로 저장하기

<변수명>.to_excel('<파일명>')으로 저장할 수 있음.

해당 변수에는 데이터가 저장되어 있음.
파일명 뒤에는 .xlsx 등 확장자를 명시해 줘야 함.
파일명에 전체 경로를 지정하면 해당 위치에 저장되고, 파일명만을 지정하면 기본 폴더에 저장됨.

+ 기본 폴더

C 드라이브 -> 사용자 -> wnsx0

메모 포함[이31]: <seaborn 함수 공통>

기본적으로 seaborn 차트 생성 함수들은 데이터를 제공하고, 그 데이터에서 칼럼을 선택해 출력한다.

칼럼명을 지정할 때는 따옴표를 씌워줘야 한다.

여기서 seaborn의 차트 생성 함수들은 크게 두 가지로 나뉜다. 하나는 x, y를 지정해 주는 것이고, 다른 하나는 칼럼 하나만 지정해 주는 것이다. 전자에서는 특히 칼럼명에 따옴표를 씌워주는 것을 조심해야 한다. 후자에서는 약식으로 작성 시 따옴표를 씌우지 않는다.

seaborn의 차트 생성 함수들은 생성 이후 ; 또는 plt.show()를 작성해 주는 것이 기본이다.

메모 포함[이32]: relationship plot
-> 두 데이터 사이의 관계를 보여주는 차트를 생성하는 함수.

메모 포함[이33]: 칼럼명에도 따옴표 씌워야 한다.

메모 포함[이34]:

메모 포함[이35]: relationship plot
-> 두 데이터 사이의 관계를 보여주는 차트를 생성하는 함수.

메모 포함[이36]: 칼럼명에도 따옴표 씌워야 한다.

3) displot(<변수명>.<칼럼명>)

칼럼 하나의 분포를 차트로 생성하는 함수.

.<칼럼명> 표기법은 약식이므로, 띄어쓰기 등이 있을 때는 ['<칼럼명>'] 형식으로 작성해야 함.

query() 함수 등을 사용해 특정 값 근처만 출력할 수도 있음.

고급 옵션들. (맨 뒤 인자에 작성하는 것들.)
1. bins=<정수> -> 생성할 막대의 개수를 지정함.

; 를 작성하여 출력을 완료함. (plt.show()와 동일한 기능.)
함수는 내부적으로 matplotlib 을 사용하기 때문에, matplotlib 을 import 하여 plt.show()를 사용하는 것이 일반적임.

4) sns.countplot(<변수명>.<칼럼명>)

범주형 데이터를 막대 차트로 생성하는 함수.

한 번에 하나의 칼럼만을 넣을 수 있음. -> barplot() 함수를 사용하면 두 개의 칼럼을 사용할 수 있음.

; 를 작성하여 출력을 완료함. (plt.show()와 동일한 기능.)
함수는 내부적으로 matplotlib 을 사용하기 때문에, matplotlib 을 import 하여 plt.show()를 사용하는 것이 일반적임.

5) sns.barplot(data=<변수명>, x='<칼럼명>', y='<칼럼명>')

해당 변수에 저장된 범주형 데이터 하나와 수치형 데이터 하나로 막대 차트를 생성하는 함수.

x 에는 범주형 데이터를, y 에는 수치형 데이터를 지정해야 함.

; 를 작성하여 출력을 완료함. (plt.show()와 동일한 기능.)
함수는 내부적으로 matplotlib 을 사용하기 때문에, matplotlib 을 import 하여 plt.show()를 사용하는 것이 일반적임.

고급 옵션들. (맨 뒤 인자에 작성하는 것들.)
1. orient='<형태>' -> 막대 차트를 수평 또는 수직으로 생성하도록 지정함.
  -> h 는 수평, v 는 수직(기본)임.
  -> 수평으로 지정할 경우, x, y 위치가 바뀌기 때문에 반대로 지정해 줘야 함.

메모 포함[이37]: distribute plot

메모 포함[이38]: displot(), countplot(), barplot() 모두 바 차트를 생성한다.

displot() -> basic한 바 차트 생성. 옵션 존재. 하나의 칼럼만 지정 가능.
countplot() -> 화려한 바 차트 생성. 하나의 칼럼만 지정 가능.
barplot() -> 두 가지 칼럼 지정 가능.

메모 포함[이39]:

메모 포함[40]:

메모 포함[이41]:

6) sns.boxplot(data=<변수명>, x='<칼럼명>')

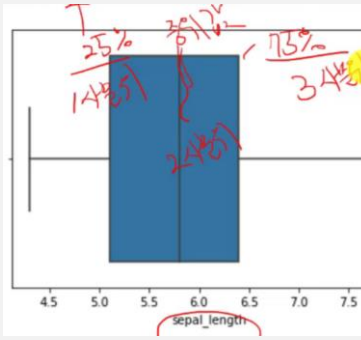
해당 변수에 저장된 수치형 데이터 하나의 분포를 박스형 차트로 생성하는 함수.

x 축만을 지정하면 하나의 박스를 생성함.
x, y 축을 지정하면 여러 개의 박스를 생성함.

박스 가운데 세로선은 중위값(2 사분위)을 나타냄,
박스의 왼쪽, 오른쪽 경계선은 각각 25%(1 사분위), 75%(3 사분위)를 나타냄.

고급 옵션들. (맨 뒤 인자에 작성하는 것들.)
1. orient='<형태>' -> 막대 차트를 수평 또는 수직으로 생성하도록 지정함.
-> h 는 수평(기본), v 는 수직임.

2. hue='<칼럼명>' -> 해당 칼럼을 기준으로 색을 나눔.



7) <변수명>.query('<조건>')

해당 변수에 저장된 데이터 중 조건에 맞는 값을 가진 데이터들만 모아서 리턴하는 함수.

조건은 "<칼럼명> < 10" 이런 식으로 지정할 수 있음.

8) sns.pairplot(<변수명>)

작성한 변수명에 저장된 데이터에 있는 모든 칼럼을 사용하여 여러 개의 차트로 표시하는 함수.

고급 옵션들. (맨 뒤 인자에 작성하는 것들.)
1. hue='<칼럼명>' -> 해당 칼럼을 기준으로 색을 나눔.

9) sns.set_style('<배경>')

차트의 배경 스타일을 지정하는 함수.

배경 종류
white -> 흰색 배경.
dark -> 검은색 배경.
grid -> 색깔 뒤에 작성하여 grid 를 추가할 수 있음. (ex. whitegrid, darkgrid)

+ 차트 크기 조정

matplotlib 의 plt.figure() 함수를 사용함.

2. Matplotlib

1. Matplotlib

데이터 시각화 라이브러리.

아나콘다에 자동 포함되어 있음.

import matplotlib.pyplot as plt

1) 기본 사용 방식

1. import matplotlib.pyplot as plt -> 라이브러리 import.
2. <변수> = [<값 1>, <값 2>, <값 3>, ...] -> 데이터 생성.
   -> Pandas 를 사용할 수도 있음.
3. plt.plot(<변수>) -> 차트 생성.
4. 함수들 -> 차트의 세부 내용을 설정하는 함수들.
5. plt.show() -> 차트 출력. (plot() 함수만 수행해도 출력되기도 함.)

+ jpg vs png (상식.)

둘 다 그림 압축 확장자임.

jpg : 손실압축방법 사용 -> 용량이 비교적 작고 품질이 떨어짐. 알파값(투명도) 미지원. (포토샵 등에서 사용 X.)

png : 비손실압축방법 사용 -> 용량이 비교적 크고 품질이 좋음. 알파값(투명도) 지원(포토샵 등에서 사용).

메모 포함[42]:

메모 포함[이43]: ex.
filter = first.query('fare <= 150')
-> fare 칼럼의 값이 150 이하인 레코드만 filter에 저장한다.

메모 포함[이44]: mat(math, 수학) + plot(도표) + lib(라이브러리)

2. matplotlib 관련 함수들

1) plt.plot(<데이터>, <데이터>)

차트 생성 함수.
기본적으로 선형 그래프가 생성됨.

매개변수에는 데이터를 작성함. (list 나 pandas 를 사용해서 생성한 데이터 묶음 등.)
데이터 묶음을 하나만 제공하면 y 축 값으로만 들어가고 x 축 값은 자동 설정됨.
데이터 묶음을 두 개 제공하면 첫 번째 인자의 데이터가 x 값으로, 두 번째 인자의 데이터가 y 값으로 들어감.

기본적으로 하나의 plt.plot() 함수에서는 하나의 데이터 묶음만 처리가 가능함.
여러가지 값들을 개별적으로 그래프로 그리려면 여러 개의 plot() 함수를 사용해야 함.
-> 개별적인 차트가 생성되고, 출력은 한꺼번에 됨.

고급 옵션들. (맨 뒤 인자에 작성하는 것들.)
이것들은 맨 오른쪽 매개변수에 작성함.

1. marker='<marker>' -> 차트의 선형 그래프의 나타난 값에 도형을 추가하는 옵션.
   -> marker 에는 여러 가지 종류가 있음. (암기 X)

2. markersize=<size> -> 차트 marker 의 크기를 설정함.

3. color='<색>' -> 차트의 색깔을 설정함.
   -> 색깔에는 여러 종류가 있음. (r, g, b 등) (암기 X)
   -> 특정 웹페이지에 들어가서 색 코드를 얻은 후 그것을 사용할 수도 있음.

4. label='<범례>' -> 차트의 범례를 설정함. (각 그래프가 어떤 값을 나타내는 것인지 이름 설정.)
   -> plt.legend() 함수로 출력해줘야 함.

5. linestyle='<형태>' -> 선의 스타일을 설정함.
   -> 따옴표 안에 아무것도 작성하지 않으면 선이 표시되지 않음.

6. linewidth=<넓이> -> 선의 두께를 설정함.

color, marker, linestyle 은 축약형 표기 사용 가능.
'<color><marker><linestyle>'을 인자로 작성함. (ex. plt.plot(data, 'ro--'))

2) plt.bar(<데이터>, <데이터>)

두 데이터로 막대그래프를 생성하는 함수.

plt.plot()과 plt.bar()를 동시에 사용하여 선과 막대를 한꺼번에 출력할 수도 있음.

고급 옵션들. (맨 뒤 인자에 작성하는 것들.)
1. alpha=<정수> -> 투명도를 지정함. 1 이 가장 진한 값.

메모 포함[이45]: matplotlib 차트 함수들의 기본적 특징은 두 가지 데이터를 작성한다는 것이다. 첫 번째 데이터가 x축으로, 두 번째 데이터가 y축으로 들어간다.

마커	의미
o (소문자 O)	원
v (소문자 V)	역삼각형
^	삼각형
s (소문자 S)	정사각형(square
D (알파벳 d, D)	마름모
+	더하기(plus) 기
. (점)	작은 원

마커	의미
x (소문자 X)	x 표시
X (대문자 X)	진한 x 표시
_ (언더바)	_ (언더바) 표
*	별 표시

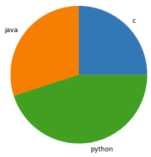
메모 포함[이46]: 이거 암기하기.

line style	사용
———	-
- - - - - -	--
-·-·-·-·-	-.
·····	:
표시안됨	빈문자

3) plt.pie(<데이터>, labels=<데이터>)

두 데이터로 pie 차트를 생성하는 함수.

첫 번째 인자로 들어가는 데이터는 값이고, 두 번째 인자 labels 에 들어가는 데이터는 해당 값의 이름임.




```
import matplotlib.pyplot as plt

lang = ['c', 'java', 'python']
count = [ 25, 30, 45]

plt.pie( count, labels=lang)
plt.show( )
```

4) plt.scatter(<데이터 1>, <데이터 2>)

scatter 차트 생성 함수.

scatter 차트는 점을 찍어서 나타내는 그래프임.

x 축의 데이터와 y 축의 데이터를 분리해서 제공해야 함.

x 축의 데이터(numpy)는 첫 번째 인자에, y 축의 데이터(numpy)는 두 번째 인자에 작성함.

고급 옵션들. (맨 뒤 인자에 작성하는 것들.)

1. c=<데이터> -> 데이터(numpy)로 정수 수열을 넣으면 수에 해당되는 색이 각각의 점에 부여됨.

2. s=<데이터> -> 데이터(numpy)로 정수 수열을 넣으면 수에 해당되는 크기가 각각의 점에 부여됨.

5) plt.legend()

plt.plot()에서 작성한 label 을 출력하는 함수.

이 함수를 수행해야 label 이 출력됨.

plt.plot()에서 label 을 지정하지 않으면 이 함수를 사용할 수 없음.

인자로 아무것도 작성하지 않으면 자동으로 위치를 선정해 출력함.

loc='<문구>'를 인자로 작성하면 해당하는 위치에 출력함.

문구 종류는 오른쪽 그림에 나와 있음.

메모 포함[이47]: 암기.

plt.plot() 함수에서 사용 가능.

코드	의미
best	적합한 곳 (자동)
upper right	상단의 오른쪽
upper left	상단의 왼쪽
upper center	상단의 중앙
lower right	하단의 오른쪽
lower left	하단의 왼쪽
lower center	하단의 중앙
center left	중간의 왼쪽
center right	중간의 오른쪽
center	정 중앙
right	오른쪽 (중간)

6) plt.xlabel('<x 축>') / plt.ylabel('<y 축>')

plot 의 x, y 축 이름을 작성하는 함수.

plt.plot() 함수에서 사용 가능.

7) plt.show()

차트 출력 함수.

실행 전, 명령 묶음의 제일 마지막에 작성해야 함.

이 함수 대신 ;를 써도 되기는 함.

8) plt.grid()

차트에 격자무늬 출력 함수.

매개변수로는 아무것도 작성하지 않아도 됨. (true 값으로 반영됨.)

plt.plot() 함수에서 사용 가능.

9) plt.title('<제목>')

차트 제목 출력 함수.

한글 제목을 작성하면 오류 발생함.

1. plt.rc('font', family='Gothic')
2. plt.rc('font', family='NanumGothic')
3. plt.rc('font', family='Malgun Gothic')

이때 세가지 중 하나의 명령을 사용해야 한글 제목을 사용할 수 있음.

plt.plot() 함수에서 사용 가능.

10) plt.savefig('<파일명>')

주피터 노트북의(주피터 노트북 사용 시.) 기본 폴더에 출력한 차트를 저장하는 함수.

단순히 마우스 오른쪽 키로 저장하면 사진 품질이 떨어짐.

c 드라이브 -> 사용자 -> jhun

11) plt.xlim(<시작>, <끝>) / plt.ylim(<시작>, <끝>)

plot 의 x, y 값의 출력 범위를 정하는 함수.

plt.plot() 함수에서 사용 가능.

12) plt.xticks(<데이터>, <데이터>)

데이터로 제공한 list 값 만을 x 축의 눈금으로 출력하는 함수.

plt.plot(), plt.bar() 함수에서 사용 가능. 근데 왜 데이터를 2 개 주지?

13) plt.figure(figsize=(<숫자 1>, <숫자 2>))

차트 출력 시 크기를 지정하는 함수.

숫자 1 에는 가로, 숫자 2 에는 세로를 지정함.

seaborn, matplotlib 모두에 사용이 가능함.

차트 생성 코드 앞쪽에 작성해야 함.
해당 위치를 확보해 두고, 차트를 그 안에 그림.

14) plt.annotate(<문자열>, xy=(<x 좌표>, <y 좌표>), xytext=(<x 좌표>, <y 좌표>))

차트에 화살표를 그려서 문자열을 표시하는 함수.

문자열에는 출력할 문자열을 지정함. (직접 작성하는 것이면 따옴표 씌워서.)
xy 에는 화살표가 가리킬 좌표를 지정함.
xytext 에는 문자열이 출력될 좌표를 지정함.

한글 문자열을 출력하면 오류 발생함.
1. plt.rc('font', family='Gothic')
2. plt.rc('font', family='NanumGothic')
3. plt.rc('font', family='Malgun Gothic')
이때 세가지 중 하나의 명령을 사용해야 한글 문자열을 사용할 수 있음.

고급 옵션들. (맨 뒤 인자에 작성하는 것들.)
1. arrowprops=('color:'<색깔>') -> 색깔 등 화살표의 속성 지정.
-> 색깔로 r 등을 지정할 수 있음.

+ 숫자 데이터를 통째로 문자열로 전환하는 방법

데이터가 숫자로 인식되어 축의 눈금이 많아지는 경우, 지정한 데이터로만 눈금을 구성하기 위한 방법임.

메모 포함[이48]: [1, 2, 3, 4, 5]를 ['1', '2', '3', '4', '5']로 바꾸는 방법.

방법 1. for 문으로 각 원소를 수정.

```
for x = range(len(year)) :
        year[x] = str(year[x])
```

방법 2. Numpy 와 plt.xticks() 함수 사용.

```
pos = np.arange(len(year))
plt.bar(year, birth)
plt.xticks(pos, year)
plt.show()
```

15) plt.imshow(<데이터>)

인자의 데이터로 이미지 차트를 생성하는 함수.

plt.plot() 함수와 동일하게, 출력 시에는 plt.show() 함수를 호출하는 것이 기본임.

word cloud 를 생성한 후 출력할 때 사용함.

16) plt.axis('off')

차트의 축을 설정하는 함수.

인자로 'off'를 지정하여 축을 지울 수 있음. (wordcloud 에서 사용.)

3. Word Cloud

1. World Cloud

주어진 텍스트의 단어들을 중요도에 따라 특정 모양으로 나타낸 것.

worldcloud, matplotlib 라이브러리 import 필요.

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

메모 포함[이49]: wordcloud 라이브러리 안에 있는 WordCloud를 가져오는 명령이다.

1) 기본 방식

1. worldcloud, matplotlib 라이브러리 import.
2. 데이터 생성. -> 직접 작성하거나 파일을 불러와서 생성.
3. word cloud 생성. -> <변수> = WorldCloud(width=1500, height=1000).generate(<데이터>)
4. 이미지 차트 생성. -> plt.imshow() 함수 사용.
5. 차트 출력. -> plt.show() 함수 사용.
6. 내용을 파일로 저장.

메모 포함[이50]: 여기서 지정하는 가로 세로는 픽셀의 크기이다.
출력되는 차트의 크기를 지정하려면 plt.figure() 함수를 사용해야 한다.

+ 파일 불러오기 (Colab)

```
from google.colab import files
files.upload()
```

이 두 코드를 작성하면 파일을 선택하는 창이 뜸.

이후 파일을 여는 코드를 작성하면 됨. (알 필요 X)

+ 파일로 저장하기 (Colab)

1. <변수>.to_file('<파일명>') -> word cloud 를 생성하여 저장한 변수를 작성해야 함.
2. files.download('<파일명>') -> 저장 이전에 form google.colab import files 를 작성해야 함.

+ 한글 Word Cloud 생성 (Colab)

한글 Word Cloud 생성을 위해서는 한글 폰트 설치가 필요함. -> ppt 에 한글 폰트 설치 명령어들이 있음.
matplotlib 의 한글화로는 word cloud 출력 시 한글 다 깨짐.

2. 이미지 차트 다듬기

차트 크기 조절 -> plt.figure() 함수 사용.

차트 축 제거 -> plt.axis() 함수 사용.

불용어 제거 -> from wordcloud import STOPWORDS

-> <변수> = STOPWORDS.union([<문자열 목록>])

-> WordCloud()의 괄호 안에 stopwords=<변수>로 마지막 인자에 지정.

메모 포함[이51]: 데이터 분석에서 신경 쓸 필요가 없는 단어.

메모 포함[이52]: ex. 'AI', 'human', 'experience' 와 같이, list처럼 작성한다.

메모 포함[이53]: [] 대신 {}를 사용해도 된다.

메모 포함[이54]: 변수에 문자열들을 넣어서 stopwords에 지정해도 되지만, STOPWORDS를 지정할 수도 있다.
이 경우, STOPWORDS의 기본 단어들이 불용어로 지정된다.

3. 이미지 마스크

word cloud 를 특정 이미지 모양으로 생성하는 방법.

1) 기본 방법

1. 이미지 파일 준비 -> 흑색과 백색으로만 이뤄진 이미지. 흑색 부분에 word cloud 가 생성됨.
   -> 위에 정리한 파일 불러오기 방법을 사용해서 불러옴.
2. 이미지를 행렬로 저장. (알 필요 X)
3. WordCloud()의 괄호 안에 해당 행렬 지정.

4. 데이터 연산

1. Numpy

1. Numpy (Numberical Python)

과학적 계산을 위한 외부 라이브러리.

import numpy as np

외부 라이브러리이기 때문에 설치가 필요함.
-> 아나콘다 설치 시에 자동 설치됨.

1) Numpy 의 장점

수학 관련 계산(행렬 등)을 간단하게 할 수 있음.
-> 데이터분석, 시각화, 머신러닝 등에 필수적임.

range 와는 달리, 수열 생성 시 실수를 지원함.

sin, cos 계산을 지원함.

2. Array 생성

1) np.arange(<시작값>, <종료값>, <증가값>)

numpy 의 수열 생성 함수.
range() 함수와 사용 방법은 동일함.

특징 1. range() 함수의 인자에는 소수를 작성할 수 없지만, np.arange() 함수에서는 가능함.
특징 2. np.arrange() 함수로 생성한 수열을 저장한 변수를 사칙 연산하면 각 원소에 개별적으로 사칙 연산한 것으로 취급됨.

메모 포함[이55]: 이 강의에서는 아나콘다의 주피터 노트북이나 Colab을 사용해서 실습한다.

2) np.array(<list>)

인자에 작성한 list 로 numpy array 를 형성하는 함수.

3) np.linspace(<시작값>, <종료값>, <구분값>)

시작값부터 종료값까지의 값을 구분값만큼의 개수로 구분하여 수열을 생성하는 함수.
값을 구분값만큼의 개수로 등분하는 것임.

시작값과 종료값을 모두 포함하여 수열을 생성함.

3. 그래프 그리기

1) np.sin(<값>) / np.cos(<값>)

인자에 작성한 값에 대응되는 sin/cos 값을 리턴하는 함수.

값으로 데이터가 저장되어 있는 변수를 작성해, 해당 데이터 전체에 적용할 수 있음.

+ array([<값 1>, <값 2>, <값 3>, . . .])

list 와는 달리, numpy 의 수열은 출력하면 array 가 앞에 추가로 출력됨.

4. 랜덤 값 생성

1) np.random.randint(<숫자 1>, <숫자 1>, <개수>)

숫자 1 과 숫자 2 사이의 정수 중에서 작성한 개수만큼 무작위 값을 추출하여 수열을 생성하는 함수.

2) np.random.normal(<숫자>, <표준편차>, <개수>)

해당 숫자를 기준으로 표준편차 이내에 있는 값 중에서 작성한 개수만큼 무작위 값을 추출하여 수열을 생성하는 함수.