| 교육 제목 | 데이터 기반 인공지능 시스템 엔지니어 양성 과정\_파이썬&판다스 |
| --- | --- |
| 교육 일시 | 2021년 9월 28일 |
| 교육 장소 | YGL C-6 학과장 & 자택(디스코드 이용한 온라인) |
| **교육 내용** | |
| 오전 | 혼공파(혼자 공부하는 파이썬 8장 \_ 클래스   1. 클래스의 기본    * 객체(object) : 여러가지 속성을 가질 수 있는 대상    * 클래스 : 객체지향 프로그래밍 코드를 조금 더 효과적으로 만들기 위한 기능 ! 객체지향 프로그램의 핵심 -> 객체와 관련된 코드를 분리할 수 있게 하는 것    * ! class 클래스 이름:  \_\_(탭)\_\_ 클래스 내용 !!! 클래스이름은 캐멀 케이스(파스칼 케이스) 규칙으로 짓는 것이 좋다.  ! 인스턴스 이름(변수 이름) = 클래스 이름() → 생산자 함수 !!! 인스턴스(instance) : 클래스를 기반으로 만들어진 객체    * 생성자(constructor) : 클래스와 이름이 같은 함수 ! class 클래스 이름:  \_\_(탭)\_\_ def \_\_init\_\_(self, 추가적인 매개변수):  \_\_(탭)\_\_\_\_(탭)\_\_ pass    * 메소드(method) : 클래스가 가지고 있는 함수  ! class 클래스 이름:  \_\_(탭)\_\_ def 메소드 이름(self, 추가적인 매개변수):  \_\_(탭)\_\_\_\_(탭)\_\_ pass 2. 클래스의 추가적인 구문    * isinstance() : 객체(instance)가 어떤 클래스로부터 만들어졌는지 확인하는 함수 ! isinstance(인스턴스, 클래스) : 상속관계 확인도 가능함    * 특수한 이름의 메소드 : \_\_???\_\_() 형태로 되어 있으며 특수한 상황에서 자동으로 호출되도록 만들어졌있음.    * 클래스 변수와 메소드  - 클래스 변수는 class 구문 바로 아래의 단계에서 변수로 선언 ! class 클래스 이름:  \_\_(탭)\_\_ 클래스 변수 = 값 !!! 클래스 변수에 접근하기 : 클래스 이름.변수 이름  - 클래스 함수도 변수와 마찬가지로 클래스가 가진 함수임  ! class 클래스 이름:  \_\_(탭)\_\_ @classmethod  \_\_(탭)\_\_ def 클래스 함수(cls, 매개변수):  \_\_(탭)\_\_\_\_(탭)\_\_ pass   \*\*\* 기타 : 상속은 어떤 클래스를 기반으로 그 속성과 기능을 물려받아 새로운 클래스를 만드는 것을 말합니다. |
| 오후 | 파이썬 머신러닝 판다스 데이터 분석 1장 \_ 판다스 입문   1. 데이터 과학자가 판다스를 배우는 이유    * 판다스 라이브러리는 데이터를 수집하고 정리하는 데 최적화된 도구    * 데이터 분석 업무의 80~90%는 데이터를 수집하고 정리하는 일 2. 판다스 자료 구조    * 컴퓨터가 이해할 수 있도록 동일한 형식을 갖는 구조로 통합된 데이터를 준비해야 한다.    * 판다스의 구조화된 데이터 형식 - 시리즈(series) : 1차원 배열, 인덱스(index)와 데이터값(value)이 일대일 대응 ! 딕셔너리 → 시리즈 변환 : pandas.Series(딕셔너리) ! 인덱스 배열 : Series객체.index ! 데이터 값 배열 : Series객체.values - 데이터프레임(dataframe) : 2차원 배열, 열(column)은 공통의 속성을 갖는 일련의 데이터를 나타내고 행(row)은 개별 관측대상에 대한 다양한 속성 데이터들의 모음인 레코드(record)가 된다.  ! 딕셔너리 → 데이터프레임 변환 : pandas.DataFrame(딕셔너리 객체) ! 행 인덱스/열 이름 설정 : pandas.DataFrame(2차원 배열, index=행 인덱스 배열, columns=열 이름 배열) !! 행 인덱스 변경 : DataFrame 객체.index = 새로운 행 인덱스 배열 !! 열 이름 변경 : DataFrame 객체.columns = 새로운 열 이름 배열  ! rename() 메소드 적용법 : 행 인덱스 또는 열 이름 일부를 선택하여 변경할 수 있다. 새로운 데이터프레임 객체를 반환함으로 원본 객체를 변경하기 위해서는 inplace=True 옵션을 사용한다.  - 행 인덱스 변경: DataFrame 객체.rename(index={기존 인덱스:새 인덱스, … })  - 열 이름 변경: DataFrame 객체.rename(columns={기존 이름:새 이름, … })  ! drop() 메소드 적용법 : 데이커 프레임의 행 또는 열을 삭제하는 명령. 새로운 데이터프레임 객체를 반환함으로 원본 객체를 변경하기 위해서는 inplace=True 옵션을 사용한다.  - 행 삭제: DataFrame 객체.drop(행 인덱스 또는 배열, axis=0, inplace=True) - 열 삭제: DataFrame 객체.drop(열 이름 또는 배열, axis=1, inplace=True)  ! 행 선택 : loc[‘인덱스 이름’] vs iloc[정수형 위치 인덱스] : 데이터프레임의 행 데이터를 선택하기 위해 사용되는 인덱서  c.f> 범위 슬라이싱 : DataFrame 객체.iloc[시작 인덱스 : 끝 인덱스 : 슬라이싱 간격]  ! 열 1개 선택 : DataFrame 객체[‘열 이름'] or DataFrame 객체.열 이름 ! 열 n개 선택 : DataFrame 객체[[열1, 열2, … , 열n]] → 데이터프레임 생성됨   ! 원소 선택 : 2개 이상 선택할 때는 리스트 형식으로 입력하거나 슬라이싱을 이용한다.  - 이름 인덱스 : DataFrame 객체.loc[행 인덱스, 열 이름] - 정수 위치 인덱스 : DataFrame 객체.iloc[행 번호, 열 번호]  ! 열 추가 : DataFrame 객체[‘추가하려는 열 이름'] = 데이터 값 ! 행 추가 : DataFrame 객체.loc[‘새로운 행 인덱스'] = 데이터 값 (또는 배열)  ! 원소 값 변경 : DataFrame 객체의 일부분 또는 원소를 선택 = 새로운 값 |