|  |  |
| --- | --- |
| 교육 제목 | **클래스기본,클래스의 추가적이 문구,판다스 자료구조(시리즈,데이터프레임,인덱스활용)** |
| 교육 일시 | 2021.9.28 |
| 교육 장소 | C06 |
| **교육 내용** | |
|  | C를 제외한 모든 프로그래밍 언어 = 객체 지향 프로그래밍 언어  -객체지향 프로그램 언어: 객체를 우선으로 생각해서 프로그래밍 하는 것  클래스 기반의 객체지향 프로그래밍 언어는 클래스를 시반으로 객체 만들고 그러한 객체를 우선으로 생각하며 프로그래밍 함  **- 객체: 여러가지 속성을 가질 수 있는 모든 대상**  **- 클래스: 객체를 좀더 효율적으로 생성하기 위해 만들어지 구문**  - 클래스 선언  Class 클래스 이름:  클래스 내용  -인스턴스: 생성자가 사용하여 이러한 클래스 기반으로 만들어진 객체  인스턴스 이름(변수 이름) = 클래스 이름() -> 생성자 함수라 부름  -생성자: 클래스 이름과 같은 함수, 클래스 내부에 \_ \_ init \_ \_ 라는 함수 만들면 객체를 생성할 때 처리할 내용을 작성할 수 있다.  Class 클래스 이름:  def \_ \_ init \_ \_(self,추가적인 매개변수):  pass  -메소드: 클래스가 가지고 있는 함수  Class 클래스 이름:  Def 메소드 이름 (self,추가적인 매개변수):  Pass |
|  | -상속: 어떤 클래스를 기반으로 속성(변수)과 기능(함수)을 물려받아 새로운 클래스를 만들다  -다중상속 : 다른 누군가가 만든 형태들을 조립해서 내가 원하는 걸 만드는 것  -instance()함수: 상속 관계에 따라서 객체가 어떤 클래스를 기반으로 만들어졌는지 확인  instance(인스턴스, 클래스)  **클래스 변수와 메소드**  인스턴스가 속성의 기능을 가질 수도 있지만 , 클래스가 속성과 기능을 가질 수도 있다  -클래스 변수:  만들기 class 클래스 이름: 접근하기 클래스 이름.변수이름  클래스 변수 = 값  -클래스 함수: 클래스가 가진 함수,’클래스가 가진 기능’라 고 명시적으로 나타내는 것  \* classmethod = 데코레이터  만들기 class 클래스 이름: 호출하기 클래스 이름.함수이름(매개변수)  @classmethod  Def 클래스 함수 (친, 매개변수):  Pass    -가비지 커넥터: 더 사용할 가능성이 없는 데이터를 메모리에서 제거하는 역할  -프라이 빗 변수: 변수를 마음대로 사용하는 것을 막아야 하고 클래스 외부에서 사용하려고 할 때 속성이 없다는 오류 발생  -게터 /세터: 프라이 빗 변수의 값을 추출하고나 변경할 목적으로, 간접적으로 속성에 접근하도록 해주는 함수    판다스 데이터 분석  판다스 자료구조  **-시리즈:데이터가 순차적으로 나열된 1차원 배열의 형태**  인덱스는 데이터 값(value) 와 일대일 대응(=파이썬 딕셔너리와 비슷한 구조)  -만들기: 딕셔너리를 시리즈로 변환하는 방법을 많이 사용  딕셔너리 -> 시리즈 변환: pandas.Series(딕셔너리)    -인덱스 구조:자기와 짝을 이루는 데이터 값의 순서와 주소를 저장한다  인덱스 배열: Serial객체.index, 데이터 값 배열: Serial.value  -원소 선택: 원소의 위치를 나타내는 주소 역할을 하는 인덱스를 이용하여 시리즈의 원소를 선택한다  정수형 인덱스 = [] 안에 위치를 나타내는 숫자  인덱스 이름을 사용할 때 []안에 ‘’ or “” 사용    **-데이터 프레임:데이터프레임은 2차원 배열이다 , 여러 개의 시리즈(열)을 모아 놓음**  -데이터 프레임 만들기: 딕셔너리 ->데이터프레임 변환, pandas.DataFrame()  - 행 인덱스/ 열 이름설정: pandas.DataFrame(2차월 배열,  Index= 행 인덱스 배열,  Columns=열 이름 배열)  - 행/열 이름 변경: DataFrame 객체.index= 새로운 행 인덱스 배열  (전체) DataFrame 객체.columns= 새로운 열 이름 배열  -행/열 이름 변경(일부): rename():이름의 일부를 선택하여 변경  DataFrame 객체.rename( index={기존인덱스: 새 인덱스,})  DataFrame 객체.rename( columns={기존이름: 새 이름,})  - 행 /열 삭제: drop() 메소드  DataFrame 객체.drop (행 인덱스 또는 배열, axis=0) 가로 행  DataFrame 객체.drop (열 이름 또는 배열, axis=1) 세로 열  - 행 선택: 행 데이터 선택 loc(인덱스 이름을 기준으로 행 선택) ,  iloc인덱스 사용(정수형 위치 인덱스를 사용할 때)  - 열 선택: 1개만 선택= DataFrame 객체 [“열이름”] or DataFrame 객체. 열이름  n개만 선택 = DataFrame 객체 [[열1,열2,열3,….,열n]]    - 원소 선택: 인덱스 이름: DataFrame객체.loc[행 인덱스, 열이름]  정수 위치 인덱스: DataFrame객체.iloc[행 번호, 열번호]  -열 추가: DataFrame객체[‘추가하려는 열 이름’] = 데이터 값  -행 추가: DataFrame.loc [‘새로운 행 이름’] = 데이터 값 (또는 배열)  -원소 값 변경: DataFrame객체의 일부분 또는 원소를 선택= 새로운 값  -행, 열 위치 바꾸기: DataFrame객체.transpose() 또는 DataFrame객체.T      -인덱스 활용(2021.9.29일 내용) |