|  |  |
| --- | --- |
| 교육 제목 | python 기초 228~309p |
| 교육 일시 | 2021.09.24 |
| 교육 장소 | 영우글로벌러닝 교육장(비대면) |
| **교육 내용** | |
| 오전 | 재귀함수 : 함수 내부에서 함수를 호출하는 형태  피보나치 수열 : 값이 커질수록 기하급수적으로 연산량이 많아짐  -> 이를 경량화 하기 위해 메모화를 함  메모화 : 한번 계산한 값들을 딕셔너리에 메모해 두어 다시 계산하지 않고 값을 확인함 -> 속도가 굉장히 빨라짐  \*\*global 키워드 - 함수내부에서 외부의 변수를 사용하기 위해 선언 |
| 오후 | 튜플 과 리스트의 차이점  1. 대괄호가 아닌 소괄호로 선언  2. 한번 선언하면 값을 바꿀 수 없음  3. 딕셔너리의 키로 리스트는 x 튜플 o  \*\*괄호가 없어도 튜플로 인식될 수 있다면 튜플임  \*\*몫, 나머지 = divmod(a, b) : 튜플로 간단하게 나머지와 몫을 동시에 구할 수 있다  \*\*요소가 하나일때 튜플을 출력하려면 요소뒤에 ,을 찍어줘야함  예) print((273, )) -> 튜플, print((273)) -> 그냥 정수  \*\*  map(함수명, 리스트)  filter(함수명, 리스트(조건)), lambda 매개변수: 리턴값  파일처리  o쓰기  1.새로쓰기(write) : w  2.있는 파일 뒤에(append) : a  o읽기(read) : r  file 처리하는 함수 : open close read write  file = open(텍스트파일.txt, "w")  file.write("안녕하세요")  file.close() \*오픈 후에는 꼭 클로즈를 해줘야함  with 구문을 쓰면 더 간단하게 코드작성 가능  with open(문자열: 파일 경로, 문자열: 모드) as 파일 객체:  제너레이터 - 함수내부에 yield 키워드를 쓰면 제너레이터 함수가 되고 함수 출력이 안됨 -> 실행하려면 next() 함수를 써줘야함    try: ~except: ~else: ~finally:  try: -단독사용 불가  예외 발생가능성 있는 구문  except 예외종류 as 예외 객체 활용할 변수:  예외가 발생했을 때 실행할 구문  raise 예외객체 - 구현되지 않은 부분을 예외를 발생시켜 잊어버리지 않도록 하는 기능  - 교수님께서 이전 강의내용 짚어준 포인트-  python : 자료의 형태 - 숫자, 문자열, 불리언  리스트 [], 딕셔너리 {}, 튜플 ()  list[반복문] : 리스트 내포  조건식 if ~ elif ~ else  반복문 for( 변수 in 반복자료) :  while  break, continue  fun\_a(10, 20) -> def func(d, b) : a,b가 매개변수 |