|  |  |
| --- | --- |
| 교육 제목 | **함수의 활용,고급,예외처리,고급** |
| 교육 일시 | 2021.9.24 |
| 교육 장소 | 비대면 |
| **교육 내용** | |
|  | **함수의 활용**  재귀함수: 재귀 란 ‘자기자신을 호출하는 것’, 내부에 자기자신을 호출하는 함수  -반복문으로 팩토리얼 구하기  -재귀함수로 팩토리얼 구하기  재귀함수의 문제: 같은 것은 기하급수적으로 많이 반복함  해결법: 메모화(딕셔너리 를 사용해서 한번계산한 값을 저장함)  피보나치 함수  조기리턴: 함수의 흐름 중간에return키워드를 사용하여 코드 들여쓰기를 줄인다  **함수 고급**  -튜플: 함수와 함께 사용하는 리스트와 비슷한 자료형으로 리스트와 다른 점은 한번 결정된 요소는 바꿀 수 없다  -람다:매개변수로 함수를 전달하기 위해 함수 구문을 작성하는 것이 번거롭고,코드공간 낭비라고 생각될 때 함수를 간단하고 쉽게 선언  --함수의 매개변수로 함수 전달하기  --filter(함수명,리스트)함수  --map(함수명,리스트)함수  -- lambda 매개변수: 리턴값 (매개변수로 함수를 전달 하는게 번거 로울 때)    **파일처리**  열기open(문자열:파일경로,문자열:읽기모드), close ,읽기read,쓰기write  -open(파일명,모드)  - w :기존의 파일이 존재O =기존 데이터 삭제 후 새로 생성, 존재x = 새로생성  - a : 뒤이어서 쓰기( 마지막에 추가)  - r : 읽기만 가능, 존재하지 않으면 에러  -close()  **with 키워드**: 블록을 벗어날 때 close()함수를 자동으로 호출  조건문과 반복문에 들어가다 보면 파일을 열고 닫지 않은 실수하는 경우가 생길 수 있기 때문에 이를 방지하기 위해 생김with구문 종료 시 파일을 자동으로 닫음  텍스트 한 줄 씩 읽기 : for 한 줄을 나타내는 문자열 in 파일 객체: 처리      **제너레이터**  이더레이터를 직접 만들 때 사용  함수내에 yield 키워드 사용하면 해당 함수는 제너레이터 함수가 됨  일반함수와 달리 호출해도 함수 내부코드가 실행되지 않음  next 함수 사용해 내부코드 실행  yield 키워드 부분 까지만 실행하며, next() 함수 리턴값 으로 yield 키워드 뒤에 입력한 값이 출력 |
|  | **구문오류(error)와 예외**  오류의 종류  -구문오류: 프로그램실행전 문법적오류로 프로그램이 실행조차 안되게 함  -예외(런타임 에러): 프로그램 실행 중 발생하는 오류  기본 예외처리:예외를 해결하는 모든 것  -조건문을 사용하는 방법  -try 구문의 사용하는 방법  Try except  Try except else  try except finally  try except else finally  try: 예외가 발생할 가능성이 있는 코드  except : 예외가 발생했을 때 실행할 코드  else: 예외가 발생하지 않았을 때 실행할 코드  finally:  try finally  \*try는 단독으로 사용할 수 없다  \*else 구문은 반드시 except구문뒤 사용  예제)    예외객체: 예외와 관련된 모든 정보를 담고 있는 객체  **-Exception**  Try: 예외가 발생할 가능성이 있는 구문  Except 예외의 종류 as 예외 객체를 화룡할 변수 이름:  예외가 발생했을 때 실행할 구문  예제)    **Raise 구문**  -아직 구현되지 않은 부분에서 강제로 예외 발생시키기  -pass처럼 강제 종료 |