|  |  |
| --- | --- |
| 교육 제목 | 개발기초 역량교육\_디지털 트랜스포메이션\_2 |
| 교육 일시 | 2021.09.24 |
| 교육 장소 | 자택(비대면) |
| **교육 내용** | |
| 오전 | 1. 함수의 활용(재귀함수, 메모화, 조기 리턴)   1-1. 재귀함수:f(n)=F(f(n-1),f(n-2),…,f(1))  1-2 팩토리얼(n!=n\*(n-1)\*…\*1), 피보나치 수열이 교과서에 나온 예제고, Combination이나 Permutation, sum, 경우의 수 문제 등, 특히 수열, 확률과 통계에 관련되어 응용할 수 있다.  1-3 메모화는 재귀함수 특성상 구했던 함수를 자주 사용하게 되는데 그런 중복되는 계산을 막아주는 역할을 하며, 참조할 dictionary를 정의하고 특정 조건이 만족되면 값을 추가하며 이를 꾸준히 인용하도록 알고리즘을 만든다.  1-4 조기 리턴은 조건문이 앞에 있을 때 else 구문 앞에 리턴값을 주어 들여쓰기 단을 줄이는 효과가 있다. else가 생략됐다고 해서 헷갈리지 말 것.   1. 함수 고급(튜플, 람다, with)   2-1. 튜플은 순서쌍으로 괄호를 생략해도 되고, 변수 값 교환, 함수의 리턴 값 등으로 용이하게 사용된다.  2-2. 람다는 간단한 함수를 쉽게 정의하는 방법으로 매개변수와 리턴값만으로 정의할 수 있다.  2-3. 함수를 매개변수로 전달할 수 있는데, 대표적으로 map 함수와 filter 함수가 있다.  2-4 파일 처리 시에는 open으로 파일을 열고, w(write), a(append), r(read)모드를 지정하여 입력하며, 반드시 close함수로 닫아줘야 한다.  2-5 with 키워드는 close함수를 축약할 수 있게 해주며, with open() as 파일 객체: 식으로 선언한다.  2-6 제너레이터는 함수 내부에 yield가 있으면 제너레이터고, yield 뒤의 값이 반환된다. 해당 제너레이터는 yield에서 멈추며, 그 이후를 실행하려면 next()로 실행 가능하다. 제너레이터는 1회용 함수다. |
| 오후 | 1. 구문 오류와 예외(구문 오류, 예외, 기본 예외 처리, try except 구문)   3-1오류는 크게 실행 전에 발생하는 오류와 실행 후에 발생하는 오류로 나뉘며, 실행 전에 발생하는 것을 syntax error(구문 오류), 실행 중에 발생하는 오류를 runtime error(런타임 오류), 혹은 exception(예외)라고 부른다.  3-2 syntax error는 큰따옴표나 괄호, 콜론, 세미콜론 등 문법적으로 맞지 않는 경우에 발생한다.  3-3 runtime error는 허용되지 않는 인덱스, 등 문법상으로는 맞지만 논리적으로 옳지 않을 때 발생한다.  3-4 기본 예외 처리는 조건문을 사용하거나 try 구문을 사용한다. 그러나 조건문만으로는 다양한 예외 처리를 모두 다루기 힘들다는 점에서 try 구문의 활용도가 있다.  3-5 try 구문으로 예외가 발생할 가능성이 있는 코드를 작성하고, except로 예외시 실행할 구문을 작성한다.  3-6 except와 pass를 묶어서 실행하면 예외시 별 처리하지 않고 넘어가게 된다.  3-7 try except에 else를 붙여 예외가 발생하지 않았을 때 실행할 코드를 추가할수도 있으며, 여기에 finally까지 붙여 무조건 실행할 코드를 붙일수도 있다. |