|  |  |
| --- | --- |
| 교육 제목 | **재귀함수, 튜플, lambda, file, with구문, 예외구문, try/except/else/finally** |
| 교육 일시 | 21.09.24 |
| 교육 장소 | 온라인 (집) |
| **교육 내용** | |
| 오전 | **▣ 함수 활용**  **∇ 재귀함수, 메모리**  \* 팩토리얼, 피보나치 수 함수처럼 계산이 많이 필요한 함수이면,  dictionary 사용해서 계산값을 저장했다 불러오는 것이 계산속도 빠름.  **Ex))**  dictionary = { 1: 1, 2: 1 }  def fibonacci3(n):  if n in dictionary:  return dictionary[n]  else:  output = fibonacci3(n-1) + fibonacci3(n-2)  dictionary[n] = output  return output  -------------------------------------------------------------------------------------------  **∇ 튜플, lambda 매개변수: 매개변수조건**  \* 튜플: 1대1 대응해서 리스트 비슷하게 사용.  값을 한번 지정하면 변결할 수 없음  \* Map(함수명, 리스트): 리스트를 변수로 하는 함수 간단하게 사용  \* Filter(함수명, 리스트): 특정 조건이 들어간다면 filter 사용  \* lambda 매개변수: 매개변수 조건 -> 사용해서 더 간단한 문장 구현  **Ex.1))**  numbers = list(range(1, 10+1))  print(list(filter(lambda x: x%2 != 0, numbers)))  **Ex.2)) map()와 lambda 사용**  output\_a = map(lambda x: x\*x, list\_a)  print("# map() 함수의 결과 version.2")  print("map: ", list(output\_a))  **Ex.3)) filter()와 lambda 사용**  output\_b = filter(lambda x: x<3, list\_a)  print("#filter() 함수의 결과 version.2")  print("filter: ", list(output\_b))  -------------------------------------------------------------------------------------------  **∇ 파일 열고 닫기, with 구문**  \* open(“파일이름.확장자”, “모드”) -> r (read), a (append), w (write) 모드 존재  \* 마지막에 반드시 close를 해줘야 함.!!!!  \* with 구문: with 구문 사용 시 따로 close 안해둬도 되는 간편함.  with open("파일명.확장자", "모드") as 파일변수이름:  문장  **Ex.1)) 파일 열고 닫기**  file\_name1 = open("python.txt", "w")  file\_name1.write("hello world")  file\_name1.close()  **Ex.2)) with 구문 사용**  with open("test.txt", "w") as file\_name2:  file\_name2.write("Hello world...!")  -------------------------------------------------------------------------------------------  **∇ 제너레이터, yield**  \* yield 키워드를 넣으면 제너레이터가 됨.  \* yield는 return과 같은 역할, 값을 반환해줌. 단, 계속 반복 실행  \* next() 함수: next함수를 써줘야 제너레이터 함수 호출가능  **Ex))**  def test():  print("함수호출")  yield 1  print("시작 00 : ")  next(test())  ------------------------------------------------------------------------------------------- |
| 오후 | **▣ 예외구문**  **∇ try 구문**  \* 예외가 발생할 가능성이 있는 코드를 try를 사용해서 오류 미연에 방지  \* Try: 예외가 발생할 가능성이 있는 코드  \* except: 예외 발생시 실행할 코드.  \* else: 예외가 발생하지 않았을 때 실행할 코드.  \* finally: 무조건 실행되는 코드.  **Ex))**  pi = 3.14  try:  input\_num = int(input("정수입력ㄱㄱ: "))  except:  print("정수 입력 안함")  else:  print("원의 반지름: ", input\_num)  print("원의 둘레: ", 2\*input\_num\*pi)  print("원의 넓이: ", input\_num\*\*2\*pi)  finally:  print("무조건 실행")  ------------------------------------------------------------------------------------------- |