|  |  |
| --- | --- |
| 교육 제목 | **Python 함수 리뷰 및 error 함수, 대처 방법** |
| 교육 일시 | 20211102 09:00 ~ 18:00 |
| 교육 장소 | 자택 온라인 수업 |
| **교육 내용** | |
| 오전 | "    # 함수 부분의 복습  # global 변수 정보 확인  # 평균 구하는 함수  def input\_data():  global dict\_score  input\_str = input("name score > ").split()  if "q" in input\_str:  return False  dic\_score[input\_str[0]] = int(input\_str[1])  return True  # 재귀함수 - factorial function  # 재귀함수 - fibonacci  def fibo(n): # 임의의 문자로 선언 가능  global count  count += 1  if n == 1:  return 1  if n == 2:  return 2  else:  return fibo(n-1) + fibo(n-2)  count = 0  fibo(35)  print("fibo({})에 계산에 활용된 덧셈 횟수는 {}: >".format(35, count)) # 굉장히 많은 시간/계산을 투여함.  dic\_fibo = {1:1, 2:2}  def fibo\_m(n):  global count\_fibo  count += 1  if n in dict\_fibo:  return dict\_fibo[n]  else:  output = fibo\_m(n-1) + fibo\_m(n-2)  dict\_fibo[n] = output  return output |
| 오후 | lcount\_fibo = 0  fibo\_m(35)  print("fibo({})에 계산에 활용된 덧셈 횟수는 {}: >".format(35, count\_fibo))  # memo 적용/ memo 함수  # 메모화  # 튜플 함수 : 리스트와 유사한 자료형 / 한 번 결정된 요소는 바꿀 수 없음.  # 람다 함수  # tuple : 값 변경 안됨 ( )  tuple\_var = (1, 2, 3)  print("tuple[0]".tuple\_var[0])  # tuple\_var[1] = 4 # error  a, b = 10, 20  print("{}, {}".format(a, b))  a, b = b, a  print("{}, {}".format(a, b))  for idx, value in enumerate([1, 2, 3, 4]):  print("{}, {}".format(idx, value))  # 람다 함수 : 함수를 좀 더 효율적으로 전달하기 위함. 함수를 간단히 정의하는 것.  # filter 함수와 map 함수  # power 함수는 제곱의 함수 (a를 대입해주면 a\*a를 output을 해주는 함수)  def power(x):  return x\*x  def uner3(x):  return x<3  print("map() : ".list(map(power, [1, 2, 3, 4, 5])))  print("filter() : ".list(filter(under3, [1, 2, 3, 4, 5])))  # 파일 처리 : 텍스트 파일의 처리 (파일 열기, 읽기, 쓰기)  # open 함수 :  # 파일 객체 = open(문자열 ...)  # file = open("basic.txt", "w")  # ./ 현재 작업 중인 디렉토리, ../ : 상위 드렉토리, ./디렉토리명 : 하위 디렉토리명  # File read  file = open("./basic.txt", "r") # 입력모드로, 존재하지 않으면 error  a = file.read()  a  file.close()  with open ('./basic.txt', 'r') as file: # 파일작성이 끝난 뒤 자동으로 파일을 닫아 줌. file.close() 자동으로 실행.  # 파일명은 score.txt로  # 이름, 성적을 입력받아 파일에 저장, 'q'가 입력되면 종료, 홍길동 90 (구분자는 '.', \n 으로)  # file.write() => print() 함수와 사용법이 유사 file.write({}, {}\n.format(name, score))  with open("./score") as score.file:  while True:  if not input\_data(score\_file)  break  score\_file.seek(0) # 파일 포인터를 처음으로 이동시킴  for line in score\_file: # 파일에서 한 라인씩 가져옴.  print(line)  print("------")  score\_file.seek(0) # 파일 포인터를 처음으로 이동시킴.  print(score\_file.read())  # generator 이터레이터를 직접 만들 때 사용하는 코드  # 함수 내부에 yield 키워드를 사용하면 해당 함수는 generator 가 됨.  # next 함수를 이용해 내부 코드 실행할 수 있도록 함.  def g\_func ():  print('test1')  yield "test1"    print("test2")  yield "test2"    print("test3")  yield "y test3"    func = g  # 구문 오류, 런타임 오류, 예외  # 구문 오류 : 실행 전에 오류가 발생하기에 중간에 수정/업데이트가 가능하다.  # 런타임 오류 : 실행 중에 오류가 발생하기에 중간 수정이 어렵다.  # except 함수 : 에러를 미리 예측하고 이에 대한 대책을 만들어 두기 위함. |