|  |  |
| --- | --- |
| 교육제목 | 데이터 기반 인공지능 시스템 엔지니어 양성 과정 |
| 교육일시 |  |
| 교육장소 | 자택(디스코드 사용 온라인 학습) |
| **교육내용** | |
| 오전 | 1. 재귀 함수   : 팩토리얼, 피보나치 등  팩토리얼 사용 예)  def factorial(n):  output = 1  for i in range(1, n+1):  output \*= i  return output  print("1! : ", factorial(1))  print("2! : ", factorial(2))  print("3! : ", factorial(3))  print("4! : ", factorial(4))   * 피보나치 함수(연산 느림)   # 변수 선언  counter = 0  # 함수 선언  def fibonacci(n):  #어떤 피보나치 수를 구하는지 출력.  print("fibonacci({})를 구합니다.".format(n))  global counter # global 변수 : 많이 쓰진 않지만 외부에서 변수를 쓸 경우  counter += 1  # 피보나치 수 구하기.  if n == 1:  return 1  if n == 2:  return 1  else :  return fibonacci(n-1) + fibonacci(n -2)  #함수 호출  fibonacci(10)  print("---")  print("fibonacci(10) 계산에 활용된 덧셈 횟수는 {}번 입니다.".format(counter))   * 피보나치 함수(위의 함수에 비해 빠름)   dictionary = {  1: 1,  2: 2  }  # 함수 선언. 피보나치함수.  def fibo(n):  if n in dictionary:  #메모 되어 있으면 메모값 리턴  return dictionary[n]  else:  #메모 안돼있으면 값 구함  output = fibo(n-1) + fibo(n-2)  dictionary[n] = output  return output    # 함수 호출  print("fibo(10):", fibo(10))  print("fibo(20):", fibo(20))  print("fibo(30):", fibo(30))  print("fibo(40):", fibo(40))  print("fibo(50):", fibo(50))   1. 리스트 평탄화 : 중첩된 리스트가 있을 때 중첩을 모두 제거하고 풀어서 1차원 리스트로 만드는 것.   def flatten(data):  # 최종 결과를 저장할 리스트 작성  output=[]  # 리스트 각각의 ltem을 가져와서 작업  for item in data:  # 데이터의 타입이 list이면 flatten함수 호출  if type(item) == list:  output += flatten(item)  # 아니면 output 리스트에 추가  else:  output.append(item)  return output    example = [[1, 2, 3],[4,[5,6]],7,[8,9]]  print("원본 :", example)  print("변환 :", flatten(example)) # 평탄화 함수 호출   1. **함수 고급**   튜플 : 함수와 함께 많이 사용되는 리스트와 비슷한 자료형으로 **한번 결정된 요소를 바꿀 수 없다.** 튜플은 (), 리스트는 [].  람다 : 함수를 간단하고 쉽게 선언하는 방법,   * lambda 매개변수 : 리턴 값 * map(lambda x:x\*x,리스트)   [a, b] = [10,20]  (c, d) = (30,40)  print(a,b,c,d)  a = 40  a  c = (1,2,3) # 튜플 값 변경 불가  #c[0]=4  a,b,c = 10,20,30  print(a,b,c)  #값을 교환  a,b = 10,20  print("before : ", a,b)  a,b = b,a  print("after : ", a,b)  문) 두 수를 입력받아 큰 수에서 작은 수를 빼는 함수를 작성하세요(num1-num2)  # 함수 정의  def minus\_func(num1, num2):  return num1 - num2  def trans\_func(num1, num2):  if num1 < num2:  return (num2, num1)  else:  return (num1, num2)    input\_numbers = input("두 수를 입력 > ").split()  (num1, num2) = trans\_func(int(input\_numbers[0]),int(input\_numbers[1]))  print(minus\_func(num1,num2))   1. filter함수와 map함수   map(함수, 리스트) filter(함수, 리스트)  # 함수 선언  def power(item):  return item \* item  def under\_3(item):  return item < 3  # 변수 선언  list\_input\_a = [1,2,3,4,5]  # map()함수 사용  output\_a = map(power, list\_input\_a)  print("# map() 함수의 실행 결과")  print("map(power, list\_input\_a):", output\_a)  print("map(power, list\_input\_a):", list(output\_a))  print()  # for i in 리스트: -> map(함수명, 리스트)  # 함수명(i) -> filter(함수명, 리스트):함수 조건에 맞는것만  # 리스트에서 계산  # filter()함수 사용  # filter(함수, 리스트) -> 조건에 맞는 요소만 실행. 결과는 리스트.  output\_b = filter(under\_3, list\_input\_a)  print("#filter() 함수의 실행 결과")  print("filter(under\_3, list\_input\_a):",output\_b)  print("filter(under\_3, list\_input\_a):", list(output\_b))  -> 람다를 적용해서 코드를 줄일 수 있다  power = lambda x: x\*x #lambda 매개변수 : 리턴 값  under\_3 = lambda x: x < 3  # 변수 선언  list\_input\_a = [1,2,3,4,5]  # lambda 함수 사용, lambda로 함수를 직접 선언  output\_a = map(lambda x: x\*x, list\_input\_a)  print("map : ", list(output\_a))  output\_a = filter(lambda x: x<3, list\_input\_a)  print("filter : ", list(output\_a))   1. file을 이용한 데이터 저장(open/closed 함수)   open(파일명, 모드)  w : write 모드(새로 쓰기 모드, 기존 파일이 존재하면 기존 데이터 삭제 후 새로 작성)  a : append 모드(뒤에 이어서 쓰기 모드)  r : read 모드(읽기 모드, 읽기만 가능. 존재하지 않으면 error)  ex)  #file = open("python.txt",'w') #file open  file = open("python.txt",'a') #file open  file.write("Hello python programming!!!\n") #file write  file.close() # file close  #file.write("test") -> error 발생. close된 file이기때문  file = open("python.txt","r")  contents = file.read()  print("file read data :", contents)  file.close() with 키워드  * 조건문과 반복문에서 파일을 열고서 닫지 않는 실수가 생김 * with 구문 종료 시 파일을 자동으로 닫음 * with open(문자열:파일경로, 문자열:모드) as 파일 객체: 문장 * open 사용 시 file 객체 변환   ex) with open("python.txt",'a') as file:  file.write("Hello python programming!!!\n")  문)100명의 키와 몸무게를 이름과 같이 파일에 저장하기  # 랜덤한 숫자를 만들기 위해 가져옴  import random  # 한글 리스트 생성  hanguls = list("가나다라마바사아자차카타파하")  # 파일을 쓰기 모드로 연다  with open("info.txt", "w") as file:  for i in range(100):  # 랜덤한 값으로 변수 생성  name = random.choice(hanguls) + random.choice(hanguls)  weight = random.randrange(40,100)  height = random.randrange(140,200)    #텍스트를 쓴다  file.write("{},{},{}\n".format(name,height,weight))  with open("info.txt", "r") as file:  for line in file:  (name, height, weight) = line.split(',')  print("{} : {}, {}".format(name, height, weight)) 제너레이터  * 이터레이터를 직접 만들 때 사용하는 코드 * 함수 내부에 yield 키워드 사용하면 해당 함수는 제너레이터 함수 됨 * 일반 함수와 달리 호출해도 함수 내부 코드가 실행되지 않음  next() 제너레이터 함수 만들기 : yield 키워드를 넣으면 일반 함수가 제너레이터 함수가 됨  def test():  print("함수 호출 111")  yield 1  print("함수 호출 222")  yield 2  print("함수 호출 333")  yield "abc"  output = test()  print("시작 00 :", next(output)) # next() 함수를 사용해야 제너레이터 함수 호출  print("시작 11 :", next(output))  print("시작 22 :", next(output))  next(output) # 더이상 진행할 부분이 없음 error  예제)  print("# 홀수만 추출하기")  print(list(filter(lambda x: x%2,numbers)))  print()  print("# 3이상, 7 미만 추출하기")  print(list(filter(lambda x: 3 <= x < 7,numbers)))  print()  print("# 제곱해서 50 미만 추출하기")  print(list(filter(lambda x: x\*x < 50,numbers)))  예제) 조건문으로 예외 처리하기  PI = 3.14  # 예외처리 try:실행할 구문 입력 except: 예외 처리 구문 입력  try:  input\_number = int(input("정수 입력> ")) # 예외 발생 가능 코드  except:  print("정수를 입력하지 않음 ")  else:  print("원의 반지름 :", input\_number)  print("원의 둘레 :", input\_number\*2\*PI)  print("원의 넓이 :", input\_number\*input\_number\*PI)  finally:  print("무조건 실행") # 무조건 실행되는 부분 |
| 오후 | 1. 오류처리 try, except, else, finally  * try + except * try + except + else * try + except + finally * try + except + else + finally * try + finally   ex)  def write\_func(filename, text):  try:  file = open(filename, 'r')  return file # return이 중간에 존재  file.write(text) # 실행 안됨  except Exception as error:  print("file error", error)  finally: # 중간에 return이 되어도 무조건 실행  print("try end... ")  #file.close() # 얘는 실행됨    write\_func("file\_close1.txt", "hello file")  #file.closed # True   * 모든 예외 잡기   # 변수 선언  list\_number = [52, 273, 32, 72, 100]  #try except 구문으로 예외 처리  try:  #숫자 입력  number\_input = int(input("몇번째 요소를 찾을거임? > "))  #리스트 요소 출력  print("{}번째 요소: {}".format(number\_input, list\_number[number\_input]))  print("value error ", list\_number.index(number\_input)) # index error  예외.커몽()  except ValueError as exception:  # ValueError 발생하는 경우  print("정수 입력!!!")  print(type(exception), exception)  except IndexError as exception:  # IndexError 발생하는 경우  print("리스트의 인덱스를 벗어났음!!!")  print(type(exception), exception)  except Exception as exception:  # 이외의 예외 발생  print("이건 머선 오류고")  print(type(exception), exception)  오늘 배운 함수 총 정리 문제)  파일명을 입력받아 "w"모드로 open  이름과 성적을 입력받아 파일에 저장  이름에 "end"가 입력되면 file을 close한 후  "r"모드로 파일을 다시 open  파일에서 자료를 읽어 list에 저장한 후  키보드로부터 검색할 이름을 검색 한 후 있으면 이름과 성적 출력  없으면 "Not found" 출력  파일을 open하는 함수 file\_open() 작성  #파일명 입력  file\_name =input("파일명 입력 >")  file\_mode =input("모드 입력 >")  file = file\_open(file\_name, "'"+file\_mode+"'") #파일open 함수 호출  file\_write(file) #파일에 저장하는 함수 호출  file.close() #파일 close  #파일에서 자료를 가져와서 변수에 저장  names, scores = [],[]  with file\_open(file\_name,"r") as file:  for item in file:  values = item.split(',')  names.append(values[0])  scores.append(int(values[1]))    #검색할 이름 입력  name = input("find name >")  #자료 검색 : 존재하면 출력, 없으면 "Not found"  if name in names:  print("{} : {}".format(name,scores[names.index(name)]))  else:  print("Not Found")  #파일open  def file\_open(file\_name, file\_mode):  file = open(file\_name, file\_mode)  return file    def file\_write(file): # 자료 입력 (이름, 성적). 이름에 'end'입력되면 입력 종료  while True:  name = input("이름 입력 : 'end'입력시 종료 >")  if name == 'end' :  break    while True:  try:  score = int(input("성적 입력 > "))  break  except:  print("숫자 입력 ")  context = name + ',' + str(score) + '\n'  file.write(context) # file에 자료 저장  # 전체 성적의 합과 평균, 인원수 출력  print(sum(scores)) |