자료구조 2주차 정리 (파이썬)

```
# 프로그램 = 자료구조 + 알고리즘

# 자료구조 : 컴퓨터에서 자료를 정리하고 조직화하는 다양한 구조
# 선형자료구조 : 항목들을 순서적으로 나열하여 저장하는 창고 : 리스트, 스택, 큐, 덱 등
# 비선형자료구조 : 항목들이 보다 복잡한 연결관계 : 트리, 그래프

# 알고리즘 : 컴퓨터로 문제를 풀기 위한 단계적 절차
# 알고리즘의 조건
# 입력 : 0개 이상의 입력이 존재
# 출력 : 1개 이상의 출력이 존재해야한다.
# 명백성 : 각 명령어의의미는 모호하지 않고 명백해야 한다.
# 유한성 : 한정된 수의 단계 후에는 반드시 종료되어야 한다.
# 유효성 : 각 명령어들은 실행 가능한 연산이어야 한다.

# 알고리즘의 기술 방법
# 자연어 : 읽기 쉽지만 단어들을 정확하게 정의하지 않으면 의미가 모호하다.
# 흐름도 : 직관적이며 이해하기 쉽다. 하지만 엄청나게 복잡한 알고리즘을 가지고 있다.
# 유사코드 : 알고리즘의 핵심적인 내용에만 집중가능하다.
# 특정언어 : 알고리즘의 가장 정확한 기술 가능, 단 구현시의 사항들이 알고리즘의 핵심적인 내용들의 이해를 방해

# 추상자료형(ADT) : 프로그래머가 추상적으로 정의한 자료형, 시스템의 정말 핵심적인 구조나 동작에만 집중
# 데이터 타입을 추상적(수학적)으로 정의한 것, 무엇인가를 정의하지만 어떻게 구현할 것인가는 정의하지 않음.

# 예) bag의 추상 자료형(ADT)
# Bag : 비어있는 가방을 새로 만든다.
# insert(e) : 가방에 항목 e를 넣는다.
# remove(e) : 가방에 e가 있는지 검사하여 있으면 이 항목을 꺼낸다.
# contains(e) : e가 들어있으면 T, 없으면 F를 반환한다.
# count() : 가방에 들어있는 항목들의 수를 반환한다.

# 예) bag의 추상 자료형의 구현(파이썬 사용)
def contains(bag, e):
    return e in bag

def insert(bag, e):
    bag.append(e)

def remove(bag, e):
    bag.remove(e)

def count(bag):
    return len(bag)
```

알고리즘의 성능분석 기법

실행 시간을 측정하는 방법 : 두 개의 알고리즘의 실제 실행 시간을 측정하는 것이며, 실제로 구현하는 것이 필요
time모듈을 사용하여 직접 실행

알고리즘의 복잡도를 분석하는 방법 : 직접 구현하지 않고서도 수행 시간을 분석하는 것, 일반적으로 연산의 횟수는 n이다.
시간 복잡도 분석 : 수행 시간 분석
기본적인 연산 고려, 알고리즘 수행에 필요한 연산의 개수를 계산, 시간 복잡도 함수 T(n)

공간 복잡도 분석 : 수행 시 필요로 하는 메모리 공간 분석

빅오표기법
차수가 가장 큰 항이 절대적인 영향
다른 항들은 상대적으로 무시
f(n)=5 -> O(1) : 상수형
f(n)=2n+1 -> O(n) : 선형
f(n)=3n*n+100 -> O(n*n) : 2차형
f(n)=5*2^n + n*n + 100 -> O(2^n) : 지수형

실행시간은 입력 집합에 따라 다를수 있다.
빅오메가 - 최선의 경우 : 수행 시간이 가장 빠른 경우, 의미가 없는 경우가 많다.
빅세타 - 평균의 경우 : 수행시간이 평균적인 경우, 계산하기가 상당히 어려움
빅오 - 최악의 경우 : 수행시간이 가장 늦은 경우, 가장 널리 사용되고, 계산하기 쉽고, 응용에 따라 중요한 의미를 가짐

순환과 반복의 알고리즘
순환 - O(log2n) : 순환적인 문제에서는 자연스러운 방법, 대부분의 순환은 반복으로 바꾸어 작성할 수 있음
반복 - O(n): 수행속도가 빠름

예) 거듭제곱 : 순환이 빠른 예시
순환 - O(log2n)
반복 - O(n)

예) 피보나치 수열 : 순환이 느린 예시
순환 - O(2^n)
반복 - O(n)

```
# 하노이의 탑 - 순환을 이용
def Hanoi_tower(n, fr, tmp, to): # 하노이의 탑 순환 함수

    if (n==1): # 종료 조건
        print("원판 1: %s --> %s" % (fr, to)) # 가장작은 원판을 옮김
    else:
        Hanoi_tower(n-1, fr, to, tmp) # n-1개를 to를 이용해 tmp로
        print("원판 %d: %s --> %s" % (n, fr, to)) # 하나의 원판을 옮김
        Hanoi_tower(n-1, tmp, fr, to)  # n-1개를 fr를 이용해 to로

if __name__ == "__main__":

    # 예) bag의 추상 자료형의 구현(파이썬 사용)
    myBag = []
    insert(myBag, '휴대폰')
    insert(myBag, '지갑')
    insert(myBag, '손수건')
    insert(myBag, '빗')
    insert(myBag, '자료구조')
    insert(myBag, '야구공')
    print('가방속의 물건', myBag)

    insert(myBag, '빗') # 중복이라도 입력
    remove(myBag, '손수건') # 손수건 제거
    print('가방속의 물건', myBag)

    # 하노이의 탑 - 순환을 이용
    print("\n\n하노이의 탑")
    Hanoi_tower(4, 'A', 'B', 'C')

'''
가방속의 물건 ['휴대폰', '지갑', '손수건', '빗', '자료구조', '야구공']
가방속의 물건 ['휴대폰', '지갑', '빗', '자료구조', '야구공', '빗']


하노이의 탑
원판 1: A --> B
원판 2: A --> C
원판 1: B --> C
원판 3: A --> B
원판 1: C --> A
원판 2: C --> B
원판 1: A --> B
원판 4: A --> C
원판 1: B --> C
원판 2: B --> A
원판 1: C --> A
원판 3: B --> C
원판 1: A --> B
원판 2: A --> C
원판 1: B --> C
'''
```

```
# 파이썬이란

# 파이썬의 예약어 (yield, nonlocal, del 등)
# 자료형(수치, 시퀀스, 매핑, 집합)과 리터럴(자료형에 대한 값)
# 수치(int, float, complex, bool)
# 시퀀스(str, list, tuple)
# 매핑(dict)
# 집합(set)

# 프로그램 = 자료구조 + 알고리즘

# 파이썬의 연산
# / : 실수의 나눗셈(결과가 실수)
# // : 정수 연산
# ** : 이항 연산자 추가
# ++, -- : 단항연산자 제공되지 않음, 복합분으로 대체하여 사용
# in 과 not in 연산자
# !, &&, || : 불리언 연산자

#  키보드 입력과 출력 함수

name = input("당신의 이름을 입력하세요.")
habby = input("취미가 무엇입니까?")
age = int(input("나이가 몇살 입니까?"))
score = float(input("평균학점이 얼마입니까?"))

print("\ninput over", end=" ")
print("\n%s의 취미는 %s이고 나이는 %d이며 평균학점은 %f임니다.\n" % (name, habby, age, score))

'''
당신의 이름을 입력하세요.jinseok
취미가 무엇입니까?study
나이가 몇살 입니까?24
평균학점이 얼마입니까?4.12

input over
jinseok의 취미는 study이고 나이는 24이며 평균학점은 4.120000임니다.
'''
```

```
# 반복 : looping

# 구구단 출력 ( for문 사용 )
dan = int(input("구구단 단 입력: "))

for n in range(2,10,1):
    print("%2d * %2d = %3d" % (dan, n, dan*n))

'''
구구단 단 입력: 5
 5 *  2 =  10
 5 *  3 =  15
 5 *  4 =  20
 5 *  5 =  25
 5 *  6 =  30
 5 *  7 =  35
 5 *  8 =  40
 5 *  9 =  45
'''

# 구구단 출력 ( while문 사용)
dan = int(input("구구단 단 입력: "))
n=1
while n<10:
    print("%2d * %2d = %3d" % (dan, n, dan*n))
    n += 1

'''
구구단 단 입력: 5
 5 *  2 =  10
 5 *  3 =  15
 5 *  4 =  20
 5 *  5 =  25
 5 *  6 =  30
 5 *  7 =  35
 5 *  8 =  40
 5 *  9 =  45
'''
```

```
# range()
for value in [11, 22, 33, 44, 55, 66]:
    print("값=%3d" % value)

'''
값= 11
값= 22
값= 33
값= 44
값= 55
값= 66
'''

for c in "game over!":
    print("결과 : ",c)

'''
결과 :  g
결과 :  a
결과 :  m
결과 :  e
결과 :
결과 :  o
결과 :  v
결과 :  e
결과 :  r
결과 :  !
'''

myset = set([12, 33, 52, 26, 99])
for e in myset:
    print("값 : ",e)

'''
값 :  33
값 :  99
값 :  12
값 :  52
값 :  26
'''

mydict = {'A':12 , 'B':22 , 'C':33 , 'D':44 , 'E':55 }
for e in mydict:
    print("키 : %c, 값 : %2d" % (e,mydict[e]))

'''
키 : A, 값 : 12
키 : B, 값 : 22
키 : C, 값 : 33
키 : D, 값 : 44
키 : E, 값 : 55
'''
```

```
# 문자열(str)
msg = 'game over'
hi = "hello world"
sum = "예전엔" + hi + "이제는" + msg
print(sum)

print(msg, '의 첫 글자는 ', msg[0])
print(msg, '의 끝 글자는 ', msg[-1])

hobby = "테니스"
age = 21
score =  4.5
msg1 = "당신의 학점은 %4.1f입니다." % score
msg2 = "취미=%s, 나이=%d, 학점=%f" % (hobby, age, score)
print(msg1)
print(msg2)

'''
예전엔hello world이제는game over
game over 의 첫 글자는  g
game over 의 끝 글자는  r
당신의 학점은  4.5입니다.
취미=테니스, 나이=21, 학점=4.500000
'''

# 리스트(list)

# append(item) : 맨 뒤에 str 삽입
# extend(lst) : 맨 뒤에 list삽입
# count(item)
# index(item, 시작, 종료) : 시작부터 종료까지 item이 있는 곳의 가장작은 인덱스를 반환
# insert(pos, item) : pos위치에 item를 삽입한다.
# pop(pos) : pos 위치에 항목 item을 삽입한다.
# remove(itme)
# reverse()
# sort([key])

big3 = []
lotto = [23, 34, 11 , 42, 9]
big4 = ['제이플라', '도티', '대도서관', '보람튜브']

print("lotto[1] = ",lotto[1])
big4[2] = '블랙핑크'
big3.append("알라딘")
big3.append("엘사")
big3.append("안나")

'''
lotto[1] =  34
'''
```

```
# 튜플(tuple)

# 리스트와 동일하지만 크기나 값을 변경할 수 없다.
# 메모리는 효율적으로 사용가능

t = (0, 3, 7)
a = (2)
b = ('game' , 1 , 3.14 , 2019)

print("취미=%s, 나이=%d, 학점=%f " % (hobby, age, score))

'''
취미=테니스, 나이=21, 학점=4.500000
'''

# 딕셔너리(dict)

# 키와 관련된 값으로 이루어진 항목들의 집합
map = {'김연아':'피겨', '쿠드롱':'당구', '메시':'축구'}
print(map)
print("쿠드롱이 뭐하는 사람이지?", map['쿠드롱'])

map['나달'] = '테니스'
map.update({'최민영':'여자야구' , '고진영':'골프'})
print(map)

'''
{'김연아': '피겨', '쿠드롱': '당구', '메시': '축구'}
쿠드롱이 뭐하는 사람이지? 당구
{'김연아': '피겨', '쿠드롱': '당구', '메시': '축구', '나달': '테니스', '최민영': '여자야구', '고진영': '골프'}
'''

# 셋(set)
s1 = {1,2,3}
s2 = {2,3,4,5}
s3 = s1.union(s2) # Set()의 메서드에서 union 제공
s4 = s1.intersection(s2) # Set()의 메서드에서 intersection 제공
s5 = s1 - s2

print("s1 : ", s1)
print("s2 : ", s2)
print("s3 : ", s3)
print("s4 : ", s4)
print("s5 : ", s5)

s5 = { 31.4}
map = { 3.14 : 'phi'}

'''
s1 :  {1, 2, 3}
s2 :  {2, 3, 4, 5}
s3 :  {1, 2, 3, 4, 5}
s4 :  {2, 3}
s5 :  {1}
'''
```

```
# 파이썬 내장 함수 : type(), len(), ord() 등
# 사용자 정의함수

# 여러개의 값의 반환
# 예) 최소, 최대 값 찾는 함수
def find_min_max(A):
    min = A[0]
    max = A[0]
    for i in range(1, len(A)):
        if max < A[i] : max = A[i]
        if min > A[i] : min = A[i]
    return min, max

data = [ 5, 3, 8, 4, 9, 1, 6, 2, 7]
x, y = find_min_max(data)
print("(min, max) = ", (x, y))

'''
(min, max) =  (1, 9)
'''

# 디폴트 인수
# 예) 더하는 함수
def sum_range(begin, end, step=1): # 매개변수 step이 기본값을 가짐
    sum = 0
    for n in range(begin, end, step):
        sum += n
    return sum
print("sum = ", sum_range(1, 10)) # step은 디폴트 값(1)으로 처리됨)
print("sum = ", sum_range(1, 10, 2))

'''
sum =  45
sum =  25
'''

# 키워드 인수
print("sum = ", sum_range(step=3, begin=1 , end=10))
print("game ", end=" " ) # 라인피드가 발생하지 않음(키워드 인수 사용)

'''
sum =  12
game
'''
```

```
# 변수의 범위
# 내장 범위 : 프로그램의 어디에서나 사용 할 수 있다.
# 전역 범위 : 프로그램 어디서나 사용 가능
# 지역 범위 : 함수나 클래스의 멤버함수(메소드) 안에서 생성
# 인스턴스 범위 : 클래스의 데이터 멤버로 생성된 변수

# 전역변수
# 예) 반지름

def calc_perimeter(radius) :
    # global perimeter
    print("파이 값: ", pi)
    perimeter = 2 * pi * radius
    return perimeter

pi = 3.14159
perimeter = 0
True_perimeter = calc_perimeter(10)
print("원둘레(r=10) = ", perimeter)
print("원둘레(r=10) = ", True_perimeter)

'''
파이 값:  3.14159
원둘레(r=10) =  0
원둘레(r=10) =  62.8318
'''
```

```
from Function import find_min_max
from Function import sum_range

data = [5, 3, 8, 4, 9, 1, 6, 2, 7]
print("(min,max) = ", find_min_max(data))
print("sum = ", sum_range(1, 10))

'''
(min,max) =  (1, 9)
sum =  45
'''
from math import pow, sqrt # math모듈

result = pow(2, 10)
dist = sqrt(1000)

print(result)
print(dist)
'''
1024.0
31.622776601683793
'''
'''
# 이것도 가능
import Function

data = [5, 3, 8, 4, 9, 1, 6, 2, 7]
print("(min,max) = ", Function.find_min_max(data))
print("sum = ", Function.sum_range(1, 10))
'''
```

```
# 클래스

# car class 생성
class Car:
    def __init__(self, color, speed = 0): # 생성자 함수
        self.color = color
        self.speed = speed

    def speedUp(self): # 멤버 함수 구현
        self.speed += 10

    def speedDown(self): # 멤버 함수 구현
        self.speed -= 10

    def __eq__(self, carB): # 비교 연산자
        return self.color == carB.color

    def __str__(self): # 출력 연산자
        return "color = %s, speed = %d" % (self.color, self.speed)

car1 = Car('black', 0)
car2 = Car('red', 120)
car3 = Car('yellow', 30)
car4 = Car('blue', 0)
car5 = Car('green')
car6 = Car('purple')

car2.speedDown()
car4.speedUp()

car3.color = 'purple'
car5.speed = 100

# 모듈 사용 X
print("car2 == car6 : ", car2 == car6)
print("car3 == car6 : ", car3 == car6)

# 모듈 사용 O
print("car2 == car6 : ", Car.__eq__(car2, car6))
print("car3 == car6 : ", Car.__eq__(car3, car6))

print("[car3]", Car.__str__(car3))

'''
car2 == car6 :  False
car3 == car6 :  True
car2 == car6 :  False
car3 == car6 :  True
[car3] color = purple, speed = 30
'''
```

```
# 클래스의 상속

# 부모 클래스를 인수로 대입
class SuperCar(Car):

    # 상속받는 클래스의 생성자 선언, 부모의 생성자 변수 뿐만 아니라 상속 클래스의 생성자 변수도 추가하여 선언
    def __init__(self, color, speed=0, bTurbo = True):
        super().__init__(color, speed) # 부모 클래스의 생성자 호출
        self.bTurbo = bTurbo # 터보변수 생성, 상속 클래스의 생성자 호출

    # 터보 모드를 On/Off하는 메소드, 디폴트 값을 True로 입력
    def setTurbo(self, bTurbo = True):
        self.bTurbo = bTurbo

    # 메소드의 재정의(오버라이딩, Overriding)
    def speedUp(self):
        if self.bTurbo:
            self.speed += 50
        else:
            super().speedUp()

    def __str__(self): # 메소드의 재정의
        if self.bTurbo :
            return "[%s] [speed = %d] 터보모드" % (self.color, self.speed)
        else:
            return "[%s] [speed = %d] 일반모드" % (self.color, self.speed)


s1 = SuperCar("Gold", 0, True)
s2 = SuperCar("white", 0, False)

s1.speedUp()
s2.speedUp()
print("슈퍼카1 : ", s1)
print("슈퍼카2 : ", s2)

'''
슈퍼카1 :  [Gold] [speed = 50] 터보모드
슈퍼카2 :  [white] [speed = 10] 일반모드
'''
```