Professional Coding Specialist

COS Pro 파이썬 1 급

5 강-10 강. 모의고사 1 차

1. 모의고사 1 차(1-10 번)

과정 소개

CosPro 1 급 파이썬 1 차 문제를 풀어보며 문제 유형을 익히고, 파이썬을 이용하여 알고리즘을 구현하기 위해 필요한 관련 지식을 익혀보도록 힌다.

학습 목차

학습 목표

1. YBM IT(www.ybmit.com) 에서 제공하는 COS Pro 1 급 파이썬 샘플 문제를 풀어보며 파이썬을 이용하여 주어진 문제를 해결하기 위한 알고리즘을 구성하는 능력을 배양한다.
2. 많이 등장하는 문제 유형을 익혀서 COS Pro 1 급 시험에 대비한다.

1. 문제 1

1) 문제 코드

```
from abc import *

class DeliveryStore(metaclass=ABCMeta):
    @abstractmethod
    def set_order_list(self, order_list):
        pass

    @abstractmethod
    def get_total_price(self):
        pass

class Food:
    def __init__(self, name, price):
        self.name = name
        self.price = price

class PizzaStore@@@:
    def __init__(self):
        menu_names = ["Cheese", "Potato", "Shrimp", "Pineapple", "Meatball"]
        menu_prices = [11100, 12600, 13300, 21000, 19500];
        self.menu_list = []
        for i in range(5):
            self.menu_list.append(Food(menu_names[i], menu_prices[i]))

        self.order_list = []

    def @@@:
        for order in order_list:
            self.order_list.append(order)

    def @@@:
        total_price = 0
        for order in self.order_list:
            for menu in self.menu_list:
                if order == menu.name:
                    total_price += menu.price
        return total_price

def solution(order_list):
    delivery_store = PizzaStore()

    delivery_store.set_order_list(order_list)
    total_price = delivery_store.get_total_price()
    return total_price

#The following is code to output testcase.
order_list = ["Cheese", "Pineapple", "Meatball"]
ret = solution(order_list)

#Press Run button to receive output.
print("Solution: return value of the function is", ret, ".")
```

2) 문제 개요

- 추상 클래스 DeliveryStore 를 상속 받는 PizzaStore 클래스를 정의하는 문제.
- 추상 클래스 DeliveryStore 에 있는 추상 메소드인 set_order_list( ), get_total_price( )를 자식 클래스인 PizzaStore 클래스에서 정의해야 PizzaStore 클래스에 대한 객체를 생성해서 사용할 수 있음.

3) 모듈, 패키지, 표준 라이브러리 사용하기

① 모듈/패키지/표준 라이브러리 구분

- 모듈 : 어떤 기능을 하는 함수를 정의해 놓은 파이썬 파일로 다른 파이썬 파일에서 import 해서 사용 가능.

< myadd.py >

```
def add(x,y):
    r=x+y
    q=r/2
    return r,q
```

< 모듈 사용하기.py >

```
import myadd

a,b=myadd.add(10,20)

print(a,b)
```

- 패키지 : 특정 기능과 관련된 어떤 모듈을 묶어 동일한 폴더에 모아 놓은 것.
- 표준 모듈 라이브러리 : 파이썬에 기본으로 설치된 모듈, 패키지, 함수를 통틀어 지칭.
- dir( ) 함수 : 모듈이 정의하는 이름을 확인하는 함수

② import 기본 사용법

- import 명령을 사용하여 math 모듈 참조하기

```
import math

math.pow(2, 3)
dir(math)
```

- math 모듈에 있는 pow( ) 함수 실행.
- math 모듈이 정의한 이름들을 문자열 리스트로 return.

- 패키지에 있는 특정 모듈 사용하기

```
sound/
    formats/
        wavread.py
        ...
    effects/
        echo.py
        ...
```

sound 폴더의 하위 폴더 effects 안에 echo.py 가 있는 경우.

예 1)

```
import sound.effects.echo
sound.effects.echo.echofilter(input, out, delay=0.7, atten=4)
```

➔ sound 폴더의 하위 폴더 effects 안에 있는 echo.py 를 import.

➔ echofilter( ) 함수를 사용하기 위해서 import 하는 경로명 전체를 적어줘야 함.

예 2)

```
from sound.effects. import echo
echo.echofilter(input, output, delay=0.7, atten=4)
```

➔ sound.effects 패키지로부터 echo 모듈만 import

➔ echofilter( ) 함수를 사용하기 위해서는 import 명령에서 언급한 echo 만 적음.

예 3)

```
from sound.effects.echo import echofilter
echofilter(input, output, delay=0.7, atten=4)
```

➔ sound.effects 패키지에 있는 echo 모듈로부터 echofilter( )만 import.

➔ echofilter( ) 함수를 사용할 때는 echofilter( )만 적음.

예 4)

```
from sound.effects import *
```

➔ sound.effects 패키지에 있는 모든 모듈을 import

- 추상 클래스 생성을 위한 모듈 import

```
from abc import *

class Animal(metaclass=ABCMeta):
    @abstractmethod
    def move(self):
        pass
```

- abc 패키지에 있는 모든 모듈을 import

```
from abc import ABCMeta,abstractmethod

class Animal(metaclass=ABCMeta):
    @abstractmethod
    def move(self):
        pass
```

- abc 패키지에서 ABCMeta, abstractmethod 만 import.

```
from abc import ABC,abstractmethod

class Animal(ABC):
    @abstractmethod
    def move(self):
        pass
```

- abc 패키지에서 ABC, abstractmethod 만 import.
- ABC : ABCMeta 를 메타클래스로 갖는 도우미 클래스.

4) 클래스(Class) : 객체를 생성하기 위한 틀

```
class 클래스 이름:
    멤버변수
    def __init__(self, 매개변수들):
    def __del__(self, 매개변수들):
    def 메소드이름(self, 매개변수들):
```

- 멤버 변수 : 객체의 속성, 데이터
- 초기화(생성자) : 객체가 생성될 때 실행되는 special method. 주로 멤버 변수를 초기화.
- 소멸자 : 객체가 소멸될 때 실행되는 special method.
- 메소드 : 객체의 기능, 행동을 정의한 함수.

➢ 인스턴스(Intance) : 클래스의 형태를 기초로 실제로 생성된 객체

5) 추상 클래스

- 메소드의 목록만 가진 클래스
- 추상 클래스를 상속한 클래스에서는 추상 메소드를 필수적으로 재정의해야 함.
- 추상 클래스는 인스턴스를 생성할 수 없음.
- 상속을 위해 사용.
- 예)

```
from abc import *

class Animal(metaclass=ABCMeta):
    @abstractmethod
    def move(self):
        pass

class Human(Animal):
    def move(self):
        print("두 발로 움직입니다.")

class Dog(Animal):
    def move(self):
        print("네 발로 움직입니다.")

h=Human()
h.move()

d=Dog()
d.move()
```

- 추상 클래스는 metaclass 를 ABCMeta 라고 지정해야 함.
- 추상 클래스는 하나 이상의 추상 메소드를 가짐.

```
두 발로 움직입니다.
네 발로 움직입니다.
```

6) 추상 클래스와 인터페이스

- 파이썬에는 공식적으로 interface 가 존재하지 않음.(파이썬의 추상 클래스는 단일 상속, 다중 상속 모두 가능)
- 추상 클래스 중 추상 메소드만 존재하는 것을 인터페이스로 간주.

자바	
추상 클래스(Abstract Class)	**인터페이스(Interface)**
추상 메소드가 하나 이상 포함. 단일 상속.	모든 메소드가 추상 메소드. 다중 상속 가능.

7) 정답

```
from abc import *

①class DeliveryStore(metaclass=ABCMeta):
    @abstractmethod
    def set_order_list(self, order_list):
        pass

    @abstractmethod
    def get_total_price(self):
        pass

②class Food:
    def __init__(self, name, price):
        self.name = name
        self.price = price

③class PizzaStore(DeliveryStore):
    def __init__(self):
        menu_names = ["Cheese", "Potato", "Shrimp", "Pineapple", "Meatball"]
        menu_prices = [11100, 12600, 13300, 21000, 19500];
        self.menu_list = []
        for i in range(5):
            self.menu_list.append(Food(menu_names[i], menu_prices[i]))

        self.order_list = []

④    def set_order_list(self, order_list):
        for order in order_list:
            self.order_list.append(order)

⑤    def get_total_price(self):
        total_price = 0
        for order in self.order_list:
            for menu in self.menu_list:
                if order == menu.name:
                    total_price += menu.price
        return total_price

def solution(order_list):
    delivery_store = PizzaStore()

    delivery_store.set_order_list(order_list)
    total_price = delivery_store.get_total_price()
    return total_price

#The following is code to output testcase.
order_list = ["Cheese", "Pineapple", "Meatball"]
ret = solution(order_list)

#Press Run button to receive output.
print("Solution: return value of the function is", ret, ".")
```

① metacalss 를 ABCMeta 로 지정하여 DeliveryStore 클래스를 추상 클래스 형태로 정의.

- 추상 메소드 set_order_list( ) 와 get_total_price( )를 생성.
- 두 개의 추상 메소드는 DeliveryStore 를 상속받는 자식 클래스에서 재정의되야 함.

② Food 클래스를 정의

- 초기화 (생성자) 메소드 __init__ 를 이용하여 멤버변수 name, price 를 초기화.

③ 추상 클래스 DeliveryStore 를 상속하는 PizzaStore 클래스 정의 : 빈 칸으로 되어 있는 부모 클래스로 DeliveryStore 로 입력.

- menu_list 를 멤버 변수로 생성하고, Food 클래스를 기초로 객체를 생성하여 menu_list 의 항목으로 추가.
- order_list 를 멤버 변수로 생성하여 빈 리스트를 할당.

④. 부모 클래스 DeliveryStore 클래스 안에 있는 추상 메소드인 set_order_list( )를 재정의하도록 빈 칸에 메소드명과 매개변수를 부모 클래스에서 지정한 대로 입력.

- set_order_list( ) : 주문 메뉴를 받아 order_list 에 저장.

⑤. 부모 클래스 DeliveryStore 클래스 안에 있는 추상 메소드인 get_total_price( ) 메소드를 재정의하도록 빈 칸에 메소드명과 매개변수를 부모 클래스에서 지정한 대로 입력.

- get_total_price( ) : order_list 에 있는 음식 가격의 총합을 return.

2. 문제 2

1) 문제 코드

```
def func_a(string, length):
    padZero = ""
    padSize = @@@
    for i in range(padSize):
        padZero += "0"
    return padZero + string

def solution(binaryA, binaryB):
    max_length = max(len(binaryA), len(binaryB))
    binaryA = func_a(binaryA, max_length)
    binaryB = func_a(binaryB, max_length)

    hamming_distance = 0
    for i in range(max_length):
        if @@@:
            hamming_distance += 1
    return hamming_distance

#The following is code to output testcase.
binaryA = "10010"
binaryB = "110"
ret = solution(binaryA, binaryB)

#Press Run button to receive output.
print("Solution: return value of the function is", ret, ".")
```

2) 문제 개요

- 프로그램의 알고리즘에 맞는 구문을 완성하는 문제.
- 같은 길이의 문자열에서 같은 위치에 있지만, 서로 다른 문자의 개수를 구하는 프로그램에서 빈 칸을 채우는 문제.

3) 정답

```
①def func_a(string, length):
    padZero = ""
    padSize = length - len(string)
    for i in range(padSize):
        padZero += "0"
    return padZero + string

②def solution(binaryA, binaryB):
    max_length = max(len(binaryA), len(binaryB))
    binaryA = func_a(binaryA, max_length)
    binaryB = func_a(binaryB, max_length)

    hamming_distance = 0
    for i in range(max_length):
        if binaryA[i] != binaryB[i]:
            hamming_distance += 1
    return hamming_distance

#The following is code to output testcase.
binaryA = "10010"
binaryB = "110"
ret = solution(binaryA, binaryB)

#Press Run button to receive output.
print("Solution: return value of the function is", ret, ".")
```

① func_a( ) 함수에서는 두 문자열의 길이 차이만큼 문자열 앞에 '0' 을 붙여 길이를 맞춘 문자열을 return.

- func_a( ) 함수의 인수로 받은 length - len(string) 값 만큼 '0' 을 문자열에 붙여 넣도록 padSize 의 비어 있는 수식을 완성.

② solution( ) 함수에서 두 문자열 간의 차이를 구하여 return.

- 두 개의 문자열을 저장한 binaryA 와 binaryB 중 길이가 더 긴 문자열 길이를 max_length 에 저장.
- func_a( ) 함수를 이용하여 두 문자열의 차이만큼 '0' 을 붙여서 binaryA 와 binaryB 에 저장되어 있는 문자열의 길이를 동일하게 맞춤.
- if 문에서는 두 개 문자열의 각 문자를 순서대로 하나씩 가져와 그 문자들이 서로 같지 않으면 hamming_distance 변수의 값을 1 만큼 증가시키도록 if 문의 조건식을 완성.

4) 보충 학습 : 파이썬 리스트를 활용하는 함수

① enumerate( ) : 리스트의 각 항목들을 (인덱스, 항목 값) 튜플 형태로 가져오는 함수

예 1)

```
x=['a','b','c']

e=list(enumerate(x))
print(e)
```

- enumerate( ) 함수를 사용하여 리스트 변수 x 의 인덱스와 항목 값을 튜플로 가져와 리스트로 변환

< 결과 >

```
[(0, 'a'), (1, 'b'), (2, 'c')]
```

예 2)

```
for i,v in enumerate(x):
  print(i,v)
```

- enumerate( ) 함수로 가져온 리스트 변수 x 의 인덱스와 항목 값을 for 문을 이용하여 변수 I, v 로 받아서 출력.

< 결과 >

```
0 a
1 b
2 c
```

② zip( ) : 두 개 이상의 리스트를 같은 인덱스 항목끼리 묶어 튜플 형태로 구성하여 한꺼번에 가져오는 함수.

예 1)

```
x1=[1,2,3]
x2=[4,5,6]

z=list(zip(x1,x2))
print(z)
```

- zip( ) 함수를 사용하여 리스트 변수 x1 과 x2 를 같은 인덱스의 항목끼리 묶어 튜플로 가져온 객체를 리스트로 변환

< 결과 >

```
[(1, 4), (2, 5), (3, 6)]
```

예 2)

```
for a,b in zip(x1,x2):
  print(a,b)
```

- zip( ) 함수로 가져온 리스트 변수 x1 과 x2 의 항목값을 for 문을 이용하여 변수 a, b 로 받아서 출력.

< 결과 >

```
1 4
2 5
3 6
```

5) 다른 코드 제안

① 문자열을 2 진수로 변환 후 해밍 거리 구하기-1 : zip( ) 함수 사용.

```
x='10010'
y='110'

cnt=0

max_length=max(len(x),len(y))

x='0'* (max_length-len(x) ) + x
y='0'* (max_length-len(y) ) + y

for a, b in zip(x,y):
    if a != b:
        cnt+=1

print(cnt)
```

- 두 문자열 중 긴 문자열의 길이를 max_length 로 지정.
- 차이 나는 길이만큼 '0' 을 채움.
- zip( ) 함수를 이용하여 x, y 에 있는 문자를 하나씩 가져와 값을 비교하여 같지 않으면 cnt 변수값 증가.

② 십진수를 2 진수로 변환 후 해밍 거리 구하기

```
s1=18;s2=6
print(bin(s1^s2).count('1'))
```

- s1 과 s2 에 대해 XOR(^) 연산을 한 결과값을 2 진수로 변환하여 나오는 수에서 '1' 의 개수를 구하여 출력.

③ 문자열을 2 진수로 변환 후 해밍 거리 구하기-2 : 비트 연산자 XOR(^) 을 사용.

```
x='10010'
y='110'

x=int(x,2)
y=int(y,2)

print(bin(x^y).count('1'))
```

- int( ) 함수를 이용하여 문자열 x, y 를 2 진수로 변환.
- x, y 에 대해 XOR(^) 연산을 한 결과값을 2 진수로 변환하여 나오는 결과에서 '1'의 개수를 구하여 출력.

3. 문제 3

1) 문제 코드

```
def func_a(numA, numB, exp):
    if exp == '+':
        return numA + numB
    elif exp == '-':
        return numA - numB
    elif exp == '*':
        return numA * numB

def func_b(exp):
    for index, value in enumerate(exp):
        if value == '+' or value == '-' or value == '*':
            return index

def func_c(exp, idx):
    numA = int(exp[:idx])
    numB = int(exp[idx + 1:])
    return numA, numB

def solution(expression):
    exp_index = func_@@@(@@@)
    first_num, second_num = func_@@@(@@@)
    result = func_@@@(@@@)
    return result

#The following is code to output testcase.
expression = "123+12"
ret = solution(expression)

#Press Run button to receive output.
print("Solution: return value of the function is", ret, ".")
```

2) 문제 개요

- 문제 코드 안에 작성된 함수의 알고리즘을 파악하여 알맞은 함수를 호출하도록 코드를 완성하는 문제.
- 문자열로 전달한 수식을 계산하여 결과를 받아 오는 프로그램으로, 프로그램 안에서 정의된 func_a( ), func_b( ), func_c( ) 함수들의 역할을 파악한 후 적절한 인수를 사용하여 필요한 함수를 호출하도록 빈 칸에 적어야 함.

```
① def func_a(numA, numB, exp):
    if exp == '+':
        return numA + numB
    elif exp == '-':
        return numA - numB
    elif exp == '*':
        return numA * numB

② def func_b(exp):
    for index, value in enumerate(exp):
        if value == '+' or value == '-' or value == '*':
            return index

③ def func_c(exp, idx):
    numA = int(exp[:idx])
    numB = int(exp[idx + 1:])
    return numA, numB

④ def solution(expression):
    exp_index = func_b(expression)
    first_num, second_num = func_c(expression, exp_index)
    result = func_a(first_num, second_num, expression[exp_index])
    return result

#The following is code to output testcase.
expression = "123+12"
ret = solution(expression)

#Press Run button to receive output.
print("Solution: return value of the function is", ret, ".")
```

3) 정답

①. func_a( ) 함수는 전달받은 연산자 문자에 해당하는 계산을 실행하여 return.

- 세 번째 매개변수 exp 의 값이 '+' 이면 numA + numB 를 실행.
- 세 번째 매개변수 exp 의 값이 '-' 이면 numA - numB 를 실행.
- 세 번째 매개변수 exp 의 값이 '*' 이면 numA * numB 를 실행.

②. func_b( ) 함수는 문자열에서 연산자가 위치한 인덱스를 찾아 return .

- enumerate( ) 함수를 이용하여 exp 에 저장되어 있는 문자열에서 문자 하나와 해당 문자의 인덱스를 value 와 index 로 받아옴.
- value 에 받아온 문자가 '+', '-', '*' 에 해당되면 인덱스를 return.

③. func_c( ) 함수는 문자열에서 연산자의 인덱스를 기준으로 문자열을 슬라이싱하여 두 수를 추출하고, 추출한 수를 정수로 변환하여 return.

- 문자열이 저장된 exp 에서 첫 번째 문자부터 idx 앞에 있는 문자까지 슬라이싱하여 numA 에 저장.
- 문자열이 저장된 exp 에서 idx 가 가리키는 문자부터 마지막 문자까지 슬라이싱하여 numB 에 저장.
- numA, numB 를 return.

④. solution( ) 함수에서 func_b( ) 를 이용하여 연산자의 인덱스를 찾은 뒤, func_c( )를 이용하여 연산자를 기준으로 두 수를 찾고, func_a( )를 이용하여 계산을 실행하도록 순서대로 알맞은 인수를 전달하며 함수를 호출하는 구문을 완성.

4) 다른 코드 제안

eval( ) : 문자열로 전달된 수식의 계산을 실행한 결과를 가져오는 함수.

```
expression = "123+12"

ret = eval(expression)
print(ret)
```

문자열로 작성된 수식을 eval( ) 함수를 사용하여 계산을 실행하고 결과를 ret 변수에 저장.

4. 문제 4

1) 문제 코드

```
#You may use import as below.
#import math

def solution(num):
    # Write code here.
    answer = 0
    return answer


#The following is code to output testcase.
num = 9949999;
ret = solution(num)

#Press Run button to receive output.
print("Solution: return value of the function is", ret, ".")
```

2) 문제 개요

- 제시된 과제를 해결하기 위해 solution( )에 프로그램 코드를 작성하는 문제.
- 매개변수 num 으로 전달된 값에 1 을 더해서 나온 결과값에서 0 으로 나타나는 숫자를 1 로 바꿔줘야 함.

3) 십진수 9950000 에서 각 자리에 사용된 숫자 추출하는 방법

9950000//10^0 을 10 으로 나눈 나머지	0	1 의 자리수
9950000//10^1 을 10 으로 나눈 나머지	0	10 의 자리수
9950000//10^2 을 10 으로 나눈 나머지	0	100 의 자리수
9950000//10^3 을 10 으로 나눈 나머지	0	1000 의 자리수
9950000//10^4 을 10 으로 나눈 나머지	5	10000 의 자리수
9950000//10^5 을 10 으로 나눈 나머지	9	100000 의 자리수
9950000//10^6 을 10 으로 나눈 나머지	9	1000000 의 자리수

4) 십진수 9950000 의 각 자리에 있는 모든 0 을 1 로 변경하는 절차

- 일의 자리에 있는 0 을 1 로 변경 → 10 의 0 제곱을 더함. $9950000 + 10^0 = 9950001$
- 십의 자리에 있는 0 을 1 로 변경 → 10 의 1 제곱을 더함. $9950001 + 10^1 = 9950011$
- 백의 자리에 있는 0 을 1 로 변경 → 10 의 2 제곱을 더함. $9950011 + 10^2 = 9950111$
- 천의 자리에 있는 0 을 1 로 변경 → 10 의 3 제곱을 더함. $9950111 + 10^3 = 9951111$

5) 정답

```
def solution(num):
①   num += 1
②   digit = 1
③   while num // digit % 10 == 0:
        num += digit
        digit *= 10
    return num


#The following is code to output testcase.
num = 9949999;
ret = solution(num)

#Press Run button to receive output.
print("Solution: return value of the function is", ret, ".")
```

①. 제시된 과제와 같이 인수로 전달받은 num 값을 1 만큼 증가.

②. digit 는 1,10, 100 순으로 증가하며 num 변수가 갖는 값의 자리수가 갖는 숫자를 추출하기 위해 사용하는 변수.

③. while 문을 num 값의 1 의 자리 숫자, 10 의 자리 숫자, 100 의 자리 숫자가 0 인 동안 실행하여 num 에 저장되어 있는 수에서 숫자가 0 인 것을 1 로 변경.

- num 값의 1 의 자리수가 0 이면 1 을 더하고, num 값의 10 의 자리수가 0 이면 10 을 더하고, num 값의 100 자리수가 0 이면 100 을 더하며 각 자리에 있는 0 을 1 로 교체.

6) 다른 코드 제안

```
def solution(num):
    # Write code here.
    answer = int(str(num + 1).replace('0','1'))
    return answer


#The following is code to output testcase.
num = 9949999;
ret = solution(num)

#Press Run button to receive output.
print("Solution: return value of the function is", ret, ".")
```

- 제시된 과제 대로 인수로 전달받은 num 값을 1 만큼 증가.
- 증가시킨 num 값을 문자열로 변환한 뒤 replace( ) 메소드를 이용하여 '0' 을 '1' 로 바꾸고 그것을 다시 정수형으로 변환.
- time 모듈의 time( ) 함수를 사용하여 프로그램 실행 시간을 측정하면, 변환해야 하는 수가 제시된 문제처럼 하나인 경우에는 프로그램 실행 시간에 차이가 없지만 만일 10 만건 이상의 수를 변환하는 경우에는 정답으로 제시된 코드가 실행 속도가 빠른 것을 확인할 수 있음.

5. 문제 5

1) 문제 코드

```
#You may use import as below.
#import math

def solution(n):
    # Write code here.
    answer = 0
    return answer


#The following is code to output testcase.
n1 = 3
ret1 = solution(n1)

#Press Run button to receive output.
print("Solution: return value of the function is", ret1, ".")

n2 = 2
ret2 = solution(n2)

#Press Run button to receive output.
print("Solution: return value of the function is", ret2, ".")
```

2) 문제 개요

- 제시된 과제를 해결하기 위해 solution( )에 프로그램 코드를 작성하는 문제.
- N x N 형태의 2 차원 리스트에 소용돌이 형태로 수를 할당한 후 (1,1) 부터 (N, N) 까지 대각선 위치에 있는 수의 합을 구해야 함.

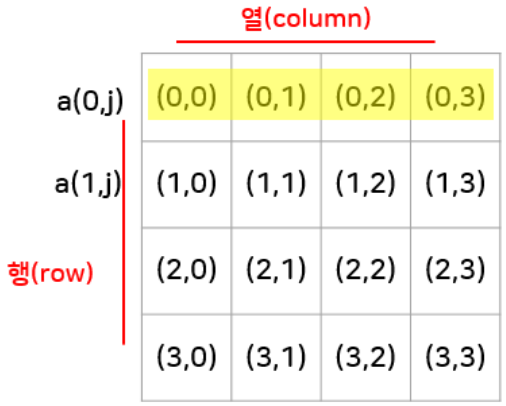
3) 보충 학습

① 그리디(Greedy) 알고리즘

- 각 단계에서 최적의 값을 찾는 알고리즘.
- 대부분 뛰어난 결과를 도출하지 못하지만 때때로 최적 해를 제시하는 경우도 있음.
- 대표적 사례 : 거스름돈 문제, 배낭에 짐 넣기 문제.

② 행렬(Matrix)

❖ 가로 줄은 행, 세로 줄은 열로 구성한 데이터 구조로 2 차원 리스트로 구현함.



❖ 파이썬으로 구현한 행렬 예

```
a=[[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]]


for i in range(4):
    for j in range(4):
        a[i][j]= '(' + str(i) + ',' + str(j)+ ')'
    print(a[i], end="₩n")
```

<결과>

```
['(0,0)', '(0,1)', '(0,2)', '(0,3)']
['(1,0)', '(1,1)', '(1,2)', '(1,3)']
['(2,0)', '(2,1)', '(2,2)', '(2,3)']
['(3,0)', '(3,1)', '(3,2)', '(3,3)']
```

③ 리스트 컴프리헨션

a = [0]* 2 ➡ **[0, 0]**

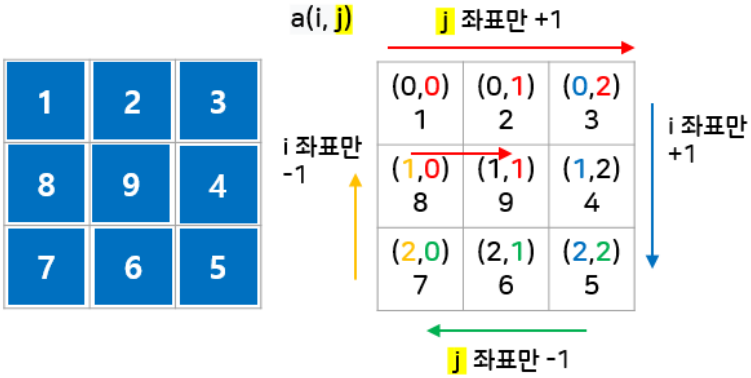
a = [I * 2 for i in range(5)] ➡ **[0, 2, 4, 6, 8]**

a = [[0] * 3 for _ in range(4)] ➡ **[[0,0,0],[0,0,0],[0,0,0],[0,0,0]]**

a = [[0 for _ in range(3)] for _ in range(4)] ➡ **[[0,0,0],[0,0,0],[0,0,0],[0,0,0]]**

4) 풀이

① N*N 행렬에 1 부터 N*N 까지의 수를 채우는 절차 이해



❖ 소용돌이 좌표 변화의 규칙: di=[0,1,0,-1] #i 좌표가 변하는 규칙
dj=[1,0,-1,0] #j 좌표가 변하는 규칙

- 수를 저장할 2 차원 리스트를 0 으로 초기화
- 소용돌이 방향으로 좌표를 움직이기 위한 좌표 변화 규칙을 설정
- 시작 위치 좌표를 (0,0) 으로 설정.
- 리스트에 처음 저장하는 숫자를 1 로 지정하여 입력.
- 규칙 di[0], dj[0]에 맞게 위치를 바꾸면서 숫자(+1) 입력.
- 다음에 입력할 j 의 위치가 없는 경우(j >= N), 이동 방향을 바꿔야 함.
- 규칙 di[1], dj[1] 에 맞게 위치를 바꾸면서 숫자(+1) 입력.
- 다음에 입력할 i 의 위치가 없는 경우(i >=N), 이동 방향을 바꿔야 함.
- 규칙 di[2], dj[2] 에 맞게 위치를 바꾸면서 숫자(+1) 입력
- 다음에 입력할 j 의 위치가 없는 경우(j < 0), 방향을 바꿔야 함.
- 규칙 di[3], dj[2] 에 맞게 위치를 바꾸면서 숫자(+1) 입력.
- 다음에 입력할 j 위치에 0 이 아닌 값이 입력되어 있으면, 방향을 바꿔야 함.
- 규칙 di[0], dj[0]에 맞게 위치를 바꾸면서 숫자(+1) 입력.

② 풀이한 코드

```
def solution(n):
    #Write code here.
    answer = 0
    pane = [[0]*n for _ in range(n)]

    di=[0,1,0,-1]
    dj=[1,0,-1,0]

    ci, cj=0,0
    num = 0
    k=0     #좌표 변화 규칙 리스트의 인덱스

    while 0 <= ci < n and 0 <= cj < n and pane[ci][cj]==0:
        num += 1
        pane[ci][cj]=num

        next_i = ci + di[k]
        next_j = cj + dj[k]

        if next_i == -1 or next_i == n or next_j == -1 or next_j == n or pane[next_i][next_j] != 0:
            k=(k+1)%4

        ci = ci + di[k]
        cj = cj + dj[k]

    for p1 in pane:
        print(p1)

    for t in range(n):
        answer += pane[t][t]

    return answer

#The following is code to output testcase.
n1 = 3
ret1 = solution(n1)

#Press Run button to receice
print("Solutin:return value of the function is", ret1,"."  )

#The following is code to output testcase.
n2 = 4
ret2 = solution(n2)

#Press Run button to receice
print("Solutin:return value of the function is", ret2,"."  )
```

- 2 차원 리스트를 초기화
- '소용돌이 방향으로 움직이기 위한 좌표 변화 규칙'을 리스트로 설정.
- 현재 위치를 (0, 0)으로 설정.
- 숫자를 1 부터 입력.
- k = 0 으로 초기화(소용돌이 좌표 인덱스)
- 아래를 반복(좌표가 범위 안에 있고 현재 위치의 값이 0 인 동안)
  - 소용돌이 변화 규칙[k]에 맞게 위치를 바꾸면서 숫자 값을 1 씩 증가하며 입력.
  - (다음 입력 좌표가 범위를 벗어났거나) or (다음 위치의 값이 0 이 아닌 경우) : 방향을 바꾼다. → 규칙[(k+1)%4]

5) 제공되는 정답

```
def in_range(i, j, n):
    return 0 <= i and i < n and 0 <= j and j < n

def solution(n):
    pane = [[0 for j in range(n)] for i in range(n)]
    dy = [0, 1, 0, -1]
    dx = [1, 0, -1, 0]
    ci, cj = 0, 0
    num = 1
    while in_range(ci, cj, n) and pane[ci][cj] == 0:
        for k in range(4):
            if not in_range(ci, cj, n) or pane[ci][cj] != 0:
                break
            while True:
                pane[ci][cj] = num
                num += 1
                ni = ci + dy[k]
                nj = cj + dx[k]
                if not in_range(ni, nj, n) or pane[ni][nj] != 0:
                    ci += dy[(k + 1) % 4]
                    cj += dx[(k + 1) % 4]
                    break
                ci = ni
                cj = nj

    ans = 0
    for i in range(n):
        ans += pane[i][i]
    return ans

#The following is code to output testcase.
n1 = 3
ret1 = solution(n1)

#Press Run button to receive output.
print("Solution: return value of the function is", ret1, ".")

n2 = 2
ret2 = solution(n2)

#Press Run button to receive output.
print("Solution: return value of the function is", ret2, ".")
```

- pane 을 n x n 형태의 2 차원 리스트로 생성하여 배열 항목들의 값을 0 으로 초기화.
- ci, cj 는 리스트의 현재 항목을 가리키는 인덱스를 나타내고 num 은 항목에 할당하는 값을 나타냄 → ci = 0, cj = 0, num = 1 로 초기화.
- ci, cj 가 리스트 인덱스 범위 내에 들어가고 ci, cj 가 가리키는 리스트의 항목 값이 0 인 동안 while 문을 이용하여 리스트의 항목값을 할당하는 작업을 반복 실행.
- 리스트에 소용돌이 순서대로 값을 할당하기 위해서
  ① 처음에는 가로 방향을 나타내는 인덱스가 n 미만인 동안, 가로 방향을 나타내는 인덱스만 1 만큼씩 증가하며 인덱스가 가리키는 리스트 항목에 num 값을 할당.
  ② 두 번째로는 세로 방향을 나타내는 인덱스가 n 미만인 동안, 세로 방향을 나타내는 인덱스만 1 만큼 증가하며 인덱스가 가리키는 리스트 항목에 num 값을 할당.
  ③ 세 번째로는 가로 방향을 나타내는 인덱스가 0 이상인 동안, 가로 방향을 나타내는 인덱스만 1 만큼씩 감소하며 인덱스가 가리키는 리스트 항목에 num 값을 할당.
  ④ 네 번째로는 세로 방향을 나타내는 인덱스가 0 이상인 동안, 세로 방향을 나타내는 인덱스만 1 만큼 감소하며 인덱스가 가리키는 리스트 항목에 num 값을 할당.

➔ 4 가지 인덱스 변화 형태가 순서대로 반복되며 리스트 항목에 값을 할당.

- 가로 방향 인덱스가 변화하는 값은 순서대로 [1, 0, -1, 0], 세로 방향 인덱스가 변화하는 값은 순서대로 [0, 1, 0, -1] → 이것을 dx = [1, 0, -1, 0], dy = [0, 1, 0, -1] 로 구현.
- 다음 번 값을 할당하기 위해 새로 계산한 인덱스 변수인 ni, nj 가 리스트의 범위를 벗어나거나 ni, nj 가 가리키는 항목의 값이 0 이 아니면, 인덱스를 변화시키는 규칙을 다음 번 순서로 바꾸어 ci, cj 를 지정하고, 그렇지 않으면 현재 인덱스를 변화하는 규칙을 그대로 사용하여 ci, cj 값을 지정해서 리스트에 값을 할당.
- 2 차원 리스트에 모든 값을 할당한 후 for 문을 이용하여 대각선 위치에 있는 항목의 합을 계산.

6. 문제 6

8) 문제 코드

```
def solution(pos):
    # Write code here.
    answer = 0
    return answer

#The following is code to output testcase.
pos = "A7"
ret = solution(pos)

#Press Run button to receive output.
print("Solution: return value of the function is", ret, ".")
```

9) 문제 개요

- 제시된 과제를 해결하기 위해 solution( )에 프로그램 코드를 작성하는 문제.
- 수평 1 칸 + 수직 2 칸 혹은 수평 2 칸 + 수직 1 칸으로 이동 가능한 기물인 나이트(knight)가 8 x 8 체스판에서 범위를 벗어나지 않고 이동할 수 있는 위치의 개수를 집계.

ex) 나이트의 현재 위치가 D4 인 경우 나이트가 이동할 수 있는 위치

: 현재 위치 D4 를 (x, y)좌표로 나타내면 (3, 3)

현재 위치에서 오른쪽 1 칸+위로 2 칸 이동	(x+1, y+2) → (4, 5)
현재 위치에서 오른쪽 1 칸+아래로 2 칸 이동	(x+1, y-2) → (4, 1)
현재 위치에서 왼쪽 1 칸+위로 2 칸 이동	(x-1, y+2) → (2, 5)
현재 위치에서 왼쪽 1 칸+아래로 2 칸 이동	(x-1, y-2) → (2, 1)
현재 위치에서 오른쪽 2 칸+위로 1 칸 이동	(x+2, y+1) → (5, 4)
현재 위치에서 오른쪽 2 칸+아래로 1 칸 이동	(x+2, y-1) → (5, 2)
현재 위치에서 왼쪽 2 칸+위로 1 칸 이동	(x-2, y+1) → (1, 4)
현재 위치에서 왼쪽 2 칸+아래로 1 칸 이동	(x-2, y-1) → (1, 2)

➔ 체스판을 벗어나지 않으면서 이동할 수 있는 위치의 개수 = 8 개

- 문자에 대한 정수 값을 가져오는 ord( ) 함수를 활용하여 현재의 좌표를 숫자로 표현.
(※ 참고: chr( ) - 아스키 값을 문자로 변환하는 함수)

10) 정답

```
def solution(pos):
    dx = [1,1,-1,-1,2,2,-2,-2]
    dy = [2,-2,-2,2,1,-1,-1,1]
    cx = ord(pos[0]) - ord("A")
    cy = ord(pos[1]) - ord("0") - 1
    ans = 0
    for i in range(8):
        nx = cx + dx[i]
        ny = cy + dy[i]
        if nx >= 0 and nx < 8 and ny >= 0 and ny < 8:
            ans += 1
    return ans

#The following is code to output testcase.
pos = "A7"
ret = solution(pos)

#Press Run button to receive output.
print("Solution: return value of the function is", ret, ".")
```

① dx 에는 가로방향으로 바꿀 수 있는 변화값을 순서대로 나열해서 리스트로 생성.

② dy 에는 세로방향으로 바꿀 수 있는 변화값을 순서대로 나열해서 리스트로 생성.

③ cx = (pos 변수로 전달된 값의 첫 번째 문자를 숫자로 변환한 값 - 'A' 의 숫자값) 을 계산하여 현재 위치에 대한 x 좌표값을 구함.

④ cy = (pos 변수로 전달된 값의 두 번째 문자를 숫자로 변환한 값 - '0' 의 숫자 값) – 1 혹은 cy = int(pos 변수로 전달된 값의 두 번째 문자) – 1 을 계산하여 현재 위치에 대한 y 좌표 값을 구함.

⑤ for 문을 이용하여 현재 좌표인 (cx, cy) 에 dx 와 dy 를 이용하여 8 가지 변화 값을 적용한 좌표 (nx, ny)가 체스판의 범위인 0 이상 8 미만에 해당하는지 확인하며 ans 에 집계.

11) 다른 코드 제안

```
def solution(pos):
    d = [(1,2),(1,-2),(-1,-2),(-1,2),(2,1),(2,-1),(-2,-1),(-2,1)]

    cx = ord(pos[0]) - ord("A")
    cy = int(pos[1]) - 1

    answer = 0

    for cd in d:
        nx = cx + cd[0]
        ny = cy + cd[1]
        if nx >= 0 and nx < 8 and ny >=0 and ny < 8:
            answer+=1
    return answer

#The following is code to output testcase.
pos = "A1"
ret = solution(pos)

#Press Run button to receive output
print("Solution : return value of the function is",ret,".")
```

① x 좌표 변화값과 y 좌표 변화값을 한 쌍으로 갖는 튜플 8 개를 항목으로 갖는 리스트 d 를 구성.

② 현재 나이트의 위치에 대한 x 좌표와 y 좌표를 구함.

③ for 문을 이용하여 리스트 d 의 튜플 항목을 하나씩 가져와 (cx, cy) 에 있는 현재 나이트의 좌표값에 더했을 때, 체스 판의 범위인 0 이상 8 미만을 벗어나지 않으면 answer 값을 1 만큼 증가.

7. 문제 7

6) 문제 코드

```
def solution(arrA, arrB):
    arrA_idx = 0
    arrB_idx = 0
    arrA_len = len(arrA)
    arrB_len = len(arrB)
    answer = []
    while @@@:
        if arrA[arrA_idx] < arrB[arrB_idx]:
            answer.append(arrA[arrA_idx])
            arrA_idx += 1
        else:
            answer.append(arrB[arrB_idx])
            arrB_idx += 1
    while @@@:
        answer.append(arrA[arrA_idx])
        arrA_idx += 1
    while @@@:
        answer.append(arrB[arrB_idx])
        arrB_idx += 1
    return answer


#The following is code to output testcase.
arrA = [-2, 3, 5, 9]
arrB = [0, 1, 5]
ret = solution(arrA, arrB);

#Press Run button to receive output.
print("Solution: return value of the function is ", ret, " .")
```

7) 문제 개요

- 두 개의 정렬된 리스트를 오름차순으로 정렬된 상태로 합치도록 프로그램 빈 칸의 구문을 완성하는 문제(병합 정렬)

8) 정답

```
def solution(arrA, arrB):
① arrA_idx = 0
   arrB_idx = 0
② arrA_len = len(arrA)
   arrB_len = len(arrB)
   answer = []
③ while arrA_idx < arrA_len and arrB_idx < arrB_len:
       if arrA[arrA_idx] < arrB[arrB_idx]:
           answer.append(arrA[arrA_idx])
           arrA_idx += 1
       else:
           answer.append(arrB[arrB_idx])
           arrB_idx += 1
④ while arrA_idx < arrA_len:
       answer.append(arrA[arrA_idx])
       arrA_idx += 1
⑤ while arrB_idx < arrB_len:
       answer.append(arrB[arrB_idx])
       arrB_idx += 1
   return answer


#The following is code to output testcase.
arrA = [-2, 3, 5, 9]
arrB = [0, 1, 5]
ret = solution(arrA, arrB);

#Press Run button to receive output.
print("Solution: return value of the function is ", ret, " .")
```

① arrA, arrB 리스트의 항목을 가리키는 인덱스 변수 arrA_idx, arrB_idx 를 각각 0 으로 초기화.

② arrA, arrB 리스트의 길이를 구하여 arrA_len, arrB_len 에 저장.

③ 두 개의 리스트 항목을 가리키는 인덱스 변수 값이 모두 리스트의 길이 보다 작은 동안 반복 실행.

➔ arrA 리스트의 항목과 arrB 리스트의 항목의 크기를 비교해서 작은 값을 answer 에 추가하고 해당 리스트 항목을 가리키는 인덱스 값을 1 만큼 증가시킴.

④ arrA 를 가리키는 인덱스가 arrA 의 길이보다 작은 동안 남아 있는 나머지 항목을 answer 에 추가.

⑤ arrB 를 가리키는 인덱스가 arrB 의 길이보다 작은 동안 남아 있는 나머지 항목을 answer 에 추가.

9) 참고 코드

```
def solution(arrA, arrB):
    return sorted(arrA + arrB)


#The following is code to output testcase.
arrA = [-2, 3, 5, 9]
arrB = [0, 1, 5]
ret = solution(arrA, arrB);

#Press Run button to receive output.
print("Solution: return value of the function is ", ret, " .")
```

- 리스트에 대해 + 연산을 수행하면 두 리스트가 병합됨.
- 파이썬 함수 sorted( ) 를 이용하여 리스트 항목의 크기대로 정렬.

10) 보충 학습 : 정렬 함수, 메소드 정리

문법	설명
a.sort( )	오름차순 정렬, 리스트 순서 자체를 바꿈, 리스트 메소드
a.sort(reverse=True)	내림차순 정렬, 리스트 순서 자체를 바꿈. 리스트 메소드
a.sort(key=str.lower)	소문자로 변경해서 정렬(대문자:upper). 리스트 메소드
a.reverse( )	리스트 항목값들의 순서를 역순으로 바꿈. 리스트 메소드
sorted(a)	오름차순 정렬. 리스트 자체의 순서를 바꾸지 않음. 파이썬 함수
sorted(a, reverse=True)	내림차순 정렬. 리스트 자체의 순서를 바꾸지 않음. 파이썬 함수
reversed(a)	역순. 리스트 자체의 순서를 바꾸지 않음. 파이썬 함수. reversed( )의 결과는 제너레이터로 돌려주기 때문에 list(reversed(a)) 로 사용해야 바뀐 결과값을 확인할 수 있음.

8. 문제 8

5) 문제 코드

```
def solution(N, votes):
    vote_counter = [0 for i in range(N+1)]
    for i in votes:
        vote_counter[i] += 1

    max_val = max(vote_counter)

    answer = []
    for idx in range(1, N + 1):
        if vote_counter[idx] == max_val:
            answer.append(vote_counter[idx])
    return answer


#The following is code to output testcase. The code below is correct and
N1 = 5
votes1 = [1,5,4,3,2,5,2,5,5,4]
ret1 = solution(N1, votes1)

#Press Run button to receive output.
print("Solution: return value of the function is ", ret1, " .")


N2 = 4
votes2 = [1, 3, 2, 3, 2]
ret2 = solution(N2, votes2)

#Press Run button to receive output.
print("Solution: return value of the function is ", ret2, " .")
```

6) 문제 개요

- 제시된 과제가 바르게 수행되도록 문제 코드를 수정하는 문제.
- 투표 결과를 저장하고 있는 votes 리스트의 항목 값을 vote_counter 리스트의 인덱스로 활용하여 득표수를 집계.

7) 정답

```
def solution(N, votes):
① vote_counter = [0 for i in range(N+1)]
② for i in votes:
        vote_counter[i]+=1

③ max_val = max(vote_counter)

    answer = []
    for idx in range(1, N + 1):
④       if vote_counter[idx] == max_val:
            answer.append(idx)
    return answer

N1 = 5
votes1 = [1,5,4,3,2,5,2,5,5,4]
ret1 = solution(N1, votes1)

#Press Run button to receive output.
print("Solution: return value of the function is ", ret1, " .")


N2 = 4
votes2 = [1, 3, 2, 3, 2]
ret2 = solution(N2, votes2)

#Press Run button to receive output.
print("Solution: return value of the function is ", ret2, " .")
```

① N+1 개의 항목을 갖는 vote_counter 를 생성하고 모든 항목값을 0 으로 초기화.

② for 문을 이용하여 매개변수 votes 에 있는 투표값을 하나씩 가져와 votes 의 항목값과 동일한 인덱스로 갖는 vote_counter 의 항목을 1 만큼 증가 → 투표 번호 별 득표 수를 vote_counter 리스트의 항목값으로 집계하게 됨.

③ 최대 득표수를 찾기 위하여 vote_counter 에 저장되어 있는 항목값 중 최댓값을 찾아서 max_val 에 저장.

④ for 문을 사용하여 vote_counter 의 항목들을 하나씩 가져와 최대 득표수와 동일한 항목값을 찾아서 최대 득표수와 동일한 항목 값을 갖는 인덱스를 answer 리스트에 추가하는 것으로 수정.

9. 문제 9

7) 문제 코드

```
def func(record):
    if record == 0:
        return 1
    elif record == 1:
        return 2
    return 0

def solution(recordA, recordB):
    cnt = 0
    for i in range(len(recordA)):
        if recordA[i] == recordB[i]:
            continue
        elif recordA[i] == func(recordB[i]):
            cnt = cnt + 3
        else:
            cnt = cnt - 1
    return cnt

#The following is code to output testcase. The code below is correct and
recordA = [2,0,0,0,0,0,1,1,0,0]
recordB = [0,0,0,0,2,2,0,2,2,2]
ret = solution(recordA, recordB)


#Press Run button to receive output.
print("Solution: return value of the function is", ret, ".")
```

8) 문제 개요

- 제시된 과제가 바르게 수행되도록 문제 코드를 수정하는 문제.
- 가위, 바위, 보를 했을 때 이긴 경우는 계단 위치를 나타내는 cnt 변수를 3 만큼 증가시키고, 졌을 경우는 cnt 변수 값을 1 만큼 감소. 단, 현재 위치가 계단의 제일 아래인 경우에는 졌을 때도 cnt 변수값을 감소시키지 않는다는 것을 구현해야 함.

9) 정답

```
def func(record):
    if record == 0:
        return 1
① elif record == 1:
        return 2
    return 0

def solution(recordA, recordB):
    cnt = 0
    for i in range(len(recordA)):
        if recordA[i] == recordB[i]:
            continue
② elif recordA[i] == func(recordB[i]):
            cnt += 3
        else:
            cnt = max(0, cnt - 1)
    return cnt

#The following is code to output testcase. The code below is co
recordA = [2,0,0,0,0,0,1,1,0,0]
recordB = [0,0,0,0,2,2,0,2,2,2]
ret = solution(recordA, recordB)


#Press Run button to receive output.
print("Solution: return value of the function is", ret, ".")
```

① func( ) 함수는 전달받은 가위/바위/보 에 대해서 승리하는 가위/바위/보를 리턴.

- 매개변수 record 의 값이 0(가위)인 경우 0(가위)을 이기는 값인 1(바위)을 리턴.
- 매개변수 record 의 값이 1(바위)인 경우 1(바위)을 이기는 값인 2(보)를 리턴.
- 매개변수 record 의 값이 0(가위), 1(바위) 이 아닌 2(보)인 경우 2(보)를 이기는 값인 0(가위)을 리턴.

② for 문을 이용하여 recordA 와 recordB 에 있는 가위/바위/보 항목들을 순서대로 비교

- recordA 의 항목과 recordB 항목 값이 같으면 비김 : 다음 번 반복 수행.
- recordA 의 항목 값이 recordB 의 항목을 이기는 값과 같은 경우 cnt 에 3 을 더함.
- recordA 의 항목이 recordB 의 항목에 지는 값인 경우 cnt 를 1 만큼 감소. 단, 계단이 제일 아래인 경우에는 0 으로 유지해야 하므로 max( ) 함수를 이용하여 0 과 (cnt-1) 중 큰 값을 cnt 에 할당하도록 코드를 수정.

10) 다른 코딩 제안

① A 가 승리하기 위한 패턴 규칙

A	B	
가위 0	보 2	(2+1)%3 = 0
바위 1	가위 0	(0+1)%3 = 1
보 2	바위 1	(1+1)%3 = 2

- B 가 낸 값에 1 을 더한 것을 3 으로 나눈 나머지가 A 와 같다면 B 가 패배(A 가 승리)

② 작성한 코드

```
def solution(recordA, recordB):
    cnt = 0
    for i in range(len(recordA)):
        if recordA[i] == recordB[i]:
            continue
        elif recordA[i] == ((recordB[i]+1)%3):
            cnt += 3
        else:
            cnt = max(0, cnt-1)
    return cnt

#The following is code to output testcase. The code below is correct a
recordA = [2,0,0,0,0,0,1,1,0,0]
recordB = [0,0,0,0,2,2,0,2,2,2]
ret = solution(recordA, recordB)


#Press Run button to receive output.
print("Solution: return value of the function is", ret, ".")
```

11) 보충 학습 : continue, break, else

① continue

```
for i in range(11):
    if i % 2 == 1:
        continue
    print(i)
```

- continue 문을 실행하면 for 문 안에서 continue 다음으로 나오는 명령들은 실행하지 않고, for 문의 시작으로 돌아가서 다음 범위 값을 가지고 for 문의 명령을 반복 실행.
- 실행 결과 : 0 부터 10 까지의 짝수만 출력.

② break

```
for i in range(11):
    if i % 2 == 1:
        break
    print(i)
```

- break 문을 실행하면 for 문에 의한 반복이 강제 종료.
- 실행 결과 : 0 출력 후 for 문 실행 종료.

③ else : 조건문(if) 이 아닌 반복문에서 사용하는 else

```
for i in range(11):
    if i % 2 == 1:
        continue
    print(i)
else:
    print("for문 종료")
```

- 반복문이 break 문에 의해 강제 종료되지 않고 정상적으로 종료되었을 때 else 문 안에 있는 명령들을 실행.
- 실행 결과 : 제시된 반복문 모두 break 에 의한 강제 종료 없이 정상 종료되므로 else 문 안에 있는 print( )가 실행됨.

```
i = 1
while i <= 5:
    print(i)
    i += 1
else:
    print("완료")
```

10. 문제 10

6) 문제 코드

```
def solution(prices):
    INF = 1000000001;
    tmp = INF
    answer = -INF
    for price in prices:
        if tmp != INF:
            answer = max(answer, tmp - price)
        tmp = min(tmp, price)
    return answer

#The following is code to output testcase. The code below is correct and
prices1 = [1, 2, 3];
ret1 = solution(prices1);

#Press Run button to receive output.
print("Solution: return value of the function is", ret1, ".")

prices2 = [3, 1];
ret2 = solution(prices2);

#Press Run button to receive output.
print("Solution: return value of the function is", ret2, ".")
```

7) 문제 개요

- 제시된 과제가 바르게 수행되도록 문제 코드를 수정하는 문제.
- 주식 가격 리스트에서 매도 값(뒤에 있는 항목) – 매수 값(앞에 있는 항목)을 계산한 것 중 가장 큰 값 찾기.
- 주식을 팔 때는 반드시 먼저 매수한 후에 매도를 해야 하므로 주식 가격 리스트의 최댓값 – 최솟값을 하게 되면 원하는 값을 구할 수 없음.

8) 정답

```
def solution(prices):
    INF = 1000000001;
① tmp = INF
    answer = -INF
    for price in prices:
②       if tmp != INF:
            answer = max(answer, price - tmp)
③       tmp = min(tmp, price)
    return answer

#The following is code to output testcase. The code below is correct and you sha
prices1 = [1, 2, 3];
ret1 = solution(prices1);

#Press Run button to receive output.
print("Solution: return value of the function is", ret1, ".")

prices2 = [3, 1];
ret2 = solution(prices2);

#Press Run button to receive output.
print("Solution: return value of the function is", ret2, ".")
```

① answer 변수에는 임의의 최솟값을, tmp 변수(이전에 가져온 주식 가격 중 최솟값을 저장하기 위해 사용)에는 임의의 최댓값을 할당.

② for 문을 이용하여 prices 리스트에 있는 주식 가격을 차례로 가져와서 tmp 변수가 초기값이 아닌 경우 (현재 가져온 주식 가격 – 이전에 가져온 주식 가격 중 최솟값)을 계산한 것과 answer 에 저장된 값 중 큰 값을 max( ) 함수를 이용하여 answer 변수에 할당.

③ for 문 마지막에는 tmp 값과 현재 가져온 주식 가격 중 작은 값을 min( ) 함수를 이용하여 tmp 변수에 저장.

9) 다른 코딩 제안

```
def solution(prices):
    answer = -10000000001  #answer 최대 수익

    for i in range(len(prices)):
        for j in range(i+1, len(prices)):
            tmp_profit = prices[j] - prices[i]
            answer = max(answer, tmp_profit)

    return answer

prices1 = [1,2,3]
ret1 = solution(prices1)
print(prices1, '최대 수익',ret1)
```

- 선택 정렬 방법처럼 첫 번째 for 문에 의해서 매수 금액을 선택하면 두 번째 for 문을 이용하여 매수 금액 뒤에 등장하는 매도 금액을 모두 가져와 (매도 금액 - 매수 금액) 을 계산한 것 중 큰 값을 answer 에 저장.
- 시간의 복잡도 = $O(n^2)$ 이므로 정답으로 제시된 코드보다 시간이 더 오래 걸림.