Professional Coding Specialist

COS Pro 파이썬 1 급

16 강-19 강. 모의고사 3 차

1. 모의고사 3 차(1-10 번)

과정 소개

Cos Pro 1 급 파이썬 3 차 문제를 풀어보며 문제 유형을 익히고, 파이썬을 이용하여 알고리즘을 구현하기 위해 필요한 관련 지식을 익혀보도록 한다.

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.

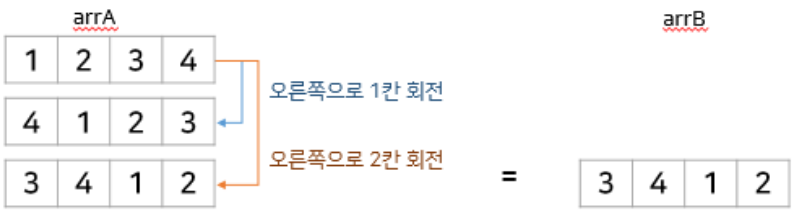
1. 문제 1

1) 문제 코드

```python
def func_a(arr):
    ret = arr + arr
    return ret

def func_b(first, second):
    MAX_NUMBER = 1001
    counter = [0 for _ in range(MAX_NUMBER)]
    for f, s in zip(first, second):
        counter[f] += 1
        counter[s] -= 1
    for c in counter:
        if c != 0:
            return False
    return True

def func_c(first, second):
    length = len(second)
    for i in range(length):
        if first[i : i + length] == second:
            return True
    return False

def solution(arrA, arrB):
    if len(arrA) != len(arrB):
        return False
    if func_@@@(@@@):
        arrA_temp = func_@@@(@@@)
        if func_@@@(@@@):
            return True
    return False

#아래는 테스트케이스 출력을 해보기 위한 코드입니다.
arrA1 = [1, 2, 3, 4]
arrB1 = [3, 4, 1, 2]
ret1 = solution(arrA1, arrB1)
#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret1, "입니다.")

arrA2 = [1, 2, 3, 4]
arrB2 = [1, 4, 2, 3]
ret2 = solution(arrA2, arrB2)
#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret2, "입니다.")
```

2) 문제 개요

- 문제 코드 안에 작성된 함수를 파악한 후 제시된 과제를 해결하기 위한 알고리즘대로 알맞은 함수를 호출하도록 코드를 완성하는 문제.
- 아래 그림과 같이 arrA 리스트를 회전하여 arrB 리스트를 만들 수 있는지 확인하는 프로그램.



3) 정답

```
①def func_a(arr):
    ret = arr + arr
    return ret

def func_b(first, second):
    MAX_NUMBER = 1001
②   counter = [0 for _ in range(MAX_NUMBER)]
③   for f, s in zip(first, second):
        counter[f] += 1
        counter[s] -= 1
④   for c in counter:
        if c != 0:
            return False
    return True

⑤def func_c(first, second):
    length = len(second)
⑥   for i in range(length):
        if first[i : i + length] == second:
            return True
    return False

def solution(arrA, arrB):
    if len(arrA) != len(arrB):
        return False
⑦   if func_b(arrA, arrB):
⑧       arrA_temp = func_a(arrA)
⑨       if func_c(arrA_temp, arrB):
            return True
    return False

#아래는 테스트케이스 출력을 해보기 위한 코드입니다.
arrA1 = [1, 2, 3, 4]
arrB1 = [3, 4, 1, 2]
ret1 = solution(arrA1, arrB1)
#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret1, "입니다.")

arrA2 = [1, 2, 3, 4]
arrB2 = [1, 4, 2, 3]
ret2 = solution(arrA2, arrB2)
#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret2, "입니다.")
```

① . func_a( ) 함수는 매개변수 arr 로 받은 리스트를 두 번 이어 붙여서 길이가 2 배인 리스트로 생성하여 return.

② . func_b( ) 함수 안에서 1000 개의 0 을 항목으로 갖는 리스트 counter 를 생성.

③ . zip( ) 함수를 이용하여 매개변수 first, second 로 받은 두 리스트의 항목을 동시에 가져옴.

- first 에서 가져온 항목과 동일한 인덱스를 갖는 counter 의 항목 값은 1 만큼 증가.

- second 에서 가져온 항목과 동일한 인덱스를 갖는 counter 의 항목값은 1 만큼 감소.
  ➔ first 의 항목과 second 의 항목이 동일하면 항목과 같은 값의 인덱스를 갖는 counter 의 항목은 값을 0 으로 유지하게 됨.

④. counter 리스트에 있는 항목값이 0 이 아닌 값이 존재하면 False 를 return.

⑤. func_c( ) 함수는 두 개의 매개변수 first, second 를 받아서 second 가 first 의 일부분인지 확인하여 그 결과를 return.

⑥. for 문을 이용하여 first 의 0 번 항목부터 second 길이만큼 자른 것이 second 와 같은지를 확인하여 같으면 True 를 return.

⑦. func_b( ) 를 이용하여 arrA 와 arrB 의 항목이 동일한 지 확인.

⑧. func_a( ) 를 이용하여 arrA 를 2 번 이어 붙인 arrA_temp 를 생성.

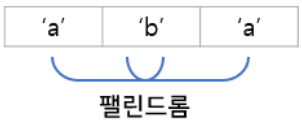
⑨. func_c( ) 를 이용하여 arrB 가 arrA_temp 의 일부분인지 확인하도록 함수를 호출.

2. 문제 2

1) 문제 코드

※ 팰린드롬(palindrome) : 문자열에 있는 앞뒤 문자의 위치를 바꾸어도 똑같은 문자열.

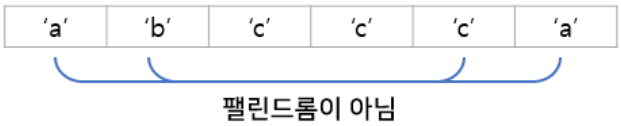
예시 1)



- 원본 문자열: 'aba'
- 앞뒤 바꾼 문자열 : 'aba'

→ 원본과 앞뒤를 바꾼 문자열이 동일하기 때문에 팰린드롬.

예시 2)



- 원본 문자열: 'abccca'
- 앞뒤 바꾼 문자열 : 'acccba'

→ 원본과 앞뒤를 바꾼 문자열이 다르기 때문에 팰린드롬이 아님.

```python
def func_a(arr, s):
    return s in arr

def func_b(s):
    length = len(s)
    for i in range(length // 2):
        if s[i] != s[length - i - 1]:
            return False
    return True

def func_c(palindromes, k):
    palindromes = sorted(palindromes)
    if len(palindromes) < k:
        return "NULL"
    else:
        return palindromes[k - 1]

def solution(s, k):
    palindromes = []
    length = len(s)
    for start_idx in range(length):
        for cnt in range(1, length - start_idx + 1):
            sub_s = s[start_idx : start_idx + cnt]
            if func_@@@(@@@) == True:
                if func_@@@(@@@) == False:
                    palindromes.append(sub_s)

    answer = func_@@@(@@@)
    return answer


#아래는 테스트케이스 출력을 해보기 위한 코드입니다.
s1 = "abcba"
k1 = 4
ret1 = solution(s1, k1)
#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret1, "입니다.")

s2 = "ccddcc"
k2 = 7
ret2 = solution(s2, k2)
#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret2, "입니다.")
```

2) 문제 개요

- 문제 코드 안에 작성된 함수를 파악한 후, 제시된 과제를 해결하기 위한 알고리즘대로 알맞은 함수를 호출하도록 코드를 완성하는 문제.
- 인수로 받은 문자열의 부분 문자열에서 '팰린드롬' 문자열들을 찾아서 리스트에 저장하고 K 번째로 큰 '팰린드롬' 문자열을 리턴하는 프로그램.

3) 정답

```python
def func_a(arr, s):
    return s in arr

def func_b(s):
    length = len(s)
    for i in range(length // 2):
        if s[i] != s[length - i - 1]:
            return False
    return True

def func_c(palindromes, k):
    palindromes = sorted(palindromes)
    if len(palindromes) < k:
        return "NULL"
    else:
        return palindromes[k - 1]

def solution(s, k):
    palindromes = []
    length = len(s)
    for start_idx in range(length):
        for cnt in range(1, length - start_idx + 1):
            sub_s = s[start_idx : start_idx + cnt]
            if func_b(sub_s) == True:
                if func_a(palindromes, sub_s) == False:
                    palindromes.append(sub_s)

    answer = func_c(palindromes, k)
    return answer


#아래는 테스트케이스 출력을 해보기 위한 코드입니다.
s1 = "abcba"
k1 = 4
ret1 = solution(s1, k1)
#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret1, "입니다.")

s2 = "ccddcc"
k2 = 7
ret2 = solution(s2, k2)
#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret2, "입니다.")
```

① func_a( ) 함수는 s 가 arr 안에 존재하는지 여부를 return.

② func_b( ) 는 s 가 팰린드롬 문자열인지 확인하여 return.

③ for 문을 이용하여 문자열 s 의 절반 길이만큼 반복하며 문자를 비교.

- s 의 첫 번째 문자와 s 의 마지막 문자를 비교하여 만일 다르면 false 를 return 하고, 그렇지 않으면 그 다음 반복 구문으로 넘어가서 s 의 두 번째 문자와 s 의 마지막에서 두 번째 문자를 비교하는 작업을 s 의 절반 항목까지 반복.

④ func_c( ) 함수의 매개변수 palindrome 에 저장된 리스트를 sorted( ) 함수를 이용하여 정렬.

⑤ func_c( ) 함수의 매개변수 k 보다 palindrome 에 저장된 리스트 항목 개수가 적으면 "NULL" 을 return 하고, 그렇지 않으면 k 번째 항목을 return.

⑥. solution( ) 함수에서는 중첩 for 문을 이용하여 인수로 받은 s 의 인덱스 0 번째 문자부터 시작하여 문자열 길이가 1,2,3..인 부분 문자열을 생성하여 sub_s 에 저장.

⑦. func_b( ) 를 이용하여 sub_s 가 '팰린드롬' 인지 확인.

⑧. func_a( ) 를 이용하여 palindromes 리스트에 sub_s 가 없으면 palindromes 의 항목으로 추가.

⑨. func_c( ) 를 이용하여 palindromes 리스트에서 K 번째 문자열을 가져와 answer 에 저장.

4) 다른 코딩 제안 : solution( ) 함수에서 부분 문자열 생성

```
def solution(s, k):
    palindromes =[]
    length = len(s)
    for start_idx in range(length):
        for end_idx in range(start_idx+1, length+1):
            sub_s = s[start_idx:end_idx]
            if func_b(sub_s) == True:
                if func_a(palindromes, sub_s) == False:
                    palindromes.append(sub_s)

    answer = func_c(palindromes, k)
    return answer
```

- 중첩 for 문에서 문자열 슬라이싱을 이용하여 부분 문자열을 생성할 때 부분 문자열 안의 문자 개수를 늘려가는 것이 아니라, 문자열을 슬라이싱 하는 끝 인덱스의 값을 늘리면서 부분 문자열을 생성하는 방법을 사용할 수 있음.

3. 문제 3

1) 문제 코드

```
#다음과 같이 import를 사용할 수 있습니다.
#import math

def solution(bishops):
    #여기에 코드를 작성해주세요.
    answer = 0
    return answer

#아래는 테스트케이스 출력을 해보기 위한 코드입니다.
bishops1 = ["D5"]
ret1 = solution(bishops1)

#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret1, "입니다.")

bishops2 = ["D5", "E8", "G2"]
ret2 = solution(bishops2)

#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret2, "입니다.")
```

2) 문제 개요

- 제시된 과제를 해결하기 위해 solution( )에 프로그램 코드를 작성하는 문제.
- 리스트로 전달된 비숍에게 잡히지 않고 기물을 놓을 수 있는 체스판의 위치의 수를 집계.
- 비숍은 자신의 현재 위치에서 대각선 방향에 있는 모든 기물들을 잡을 수 있음.

3) 정답

```
def solution(bishops):
① chess_map = [[1 for _ in range(8)] for _ in range(8)]
    for b in bishops:
        x = ord(b[0]) - ord('A')
②       y = int(b[1]) - 1

        chess_map[y][x] = 0

        dx = x - 1
③       y_up = y
        y_dn = y
        while dx >= 0:
            y_up = y_up + 1
            y_dn = y_dn - 1
            if y_up < 8:
④               chess_map[y_up][dx] = 0
            if y_dn >= 0:
                chess_map[y_dn][dx] = 0
            dx -= 1

        dx = x + 1
⑤       y_up = y
        y_dn = y
        while dx < 8:
            y_up = y_up + 1
            y_dn = y_dn - 1
            if y_up < 8:
⑥               chess_map[y_up][dx] = 0
            if y_dn >= 0:
                chess_map[y_dn][dx] = 0
            dx += 1

    answer = 0
⑦   for a in chess_map:
        answer += a.count(1)
    return answer

#아래는 테스트케이스 출력을 해보기 위한 코드입니다.
bishops1 = ["D5"]
ret1 = solution(bishops1)
#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret1, "입니다.")

bishops2 = ["D5", "E8", "G2"]
ret2 = solution(bishops2)
#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret2, "입니다.")
```

①. 체스판 크기에 맞춰 각 항목 값이 1 인 8 X 8 크기의 2 차원 리스트 chess_map 을 생성.

- 항목값 = 1 : 비숍에 잡히지 않고 기물을 배치할 수 있는 위치를 의미.

  chess_map = [[1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1],
[1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1]]

②. for 문을 이용하여 bishops 리스트에 있는 비숍의 위치값을 가져와 가로 위치에 해당하는 인덱스와 세로 위치에 해당하는 인덱스 x, y 값을 구하고, chess_map 에서 현 비숍의 위치에 해당하는 인덱스의 항목값을 0 으로 변경(항목값 = 0 은 비숍에 잡히는 위치를 의미)

❖ for 문 종료 후 chess_map =

```
[1, 1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 0, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1]
```

③. 현재 비숍의 왼쪽 방향으로 비숍에 의해 잡히는 위치를 찾기 위해 dx 변수에 x-1 을 할당하고, y_up 변수와 y_dn 변수에는 현 비숍의 y 값을 할당.

④. dx 값이 0 이상인 동안(체스판 왼쪽 끝을 벗어나지 않는 동안)

- y_up 은 1 만큼 증가시키고, y_dn 은 1 만큼 감소시킨 후 두 변수의 값이 체스판의 영역을 벗어나지 않으면 chess_map[y_up][dx] 항목과 chess_map[y_dn][dx] 항목값을 0 으로 변경.
- dx 의 값을 1 만큼 감소시킴.

❖ while 문 종료 후 chess_map =

```
[1, 1, 1, 1, 1, 1, 1, 1]
[0, 1, 1, 1, 1, 1, 1, 1]
[1, 0, 1, 1, 1, 1, 1, 1]
[1, 1, 0, 1, 1, 1, 1, 1]
[1, 1, 1, 0, 1, 1, 1, 1]
[1, 1, 0, 1, 1, 1, 1, 1]
[1, 0, 1, 1, 1, 1, 1, 1]
[0, 1, 1, 1, 1, 1, 1, 1]
```

⑤. 현재 비숍의 오른쪽 방향으로 비숍에 의해 잡히는 위치를 찾기 위해 dx 변수에 x+1 을 할당하고, y_up 변수와 y_dn 변수에는 현 비숍의 y 값을 할당.

⑥. dx 값이 8 미만인 동안(체스판 오른쪽 끝을 벗어나지 않는 동안)

- y_up 은 1 만큼 증가시키고, y_dn 은 1 만큼 감소시킨 후 두 변수의 값이 체스판의 영역을 벗어나지 않으면 chess_map[y_up][dx] 항목과 chess_map[y_dn][dx] 항목값을 0 으로 변경.
- dx 의 값을 1 만큼 증가시킴.

❖ while 문 종료 후 chess_map =

```
[1, 1, 1, 1, 1, 1, 1, 0]
[0, 1, 1, 1, 1, 1, 0, 1]
[1, 0, 1, 1, 1, 0, 1, 1]
[1, 1, 0, 1, 0, 1, 1, 1]
[1, 1, 1, 0, 1, 1, 1, 1]
[1, 1, 0, 1, 0, 1, 1, 1]
[1, 0, 1, 1, 1, 0, 1, 1]
[0, 1, 1, 1, 1, 1, 0, 1]
```

⑦. for 문을 이용하여 2 차원 리스트인 chess_map 리스트에서 한 항목씩 가져와 a 에 할당하면 a 에는 1 차원 리스트가 할당되고, a 에 대한 count( ) 메소드를 이용하여 항목 값이 1 인 항목의 개수를 answer 변수에 누적 집계.

4. 문제 4

1) 문제 코드

```python
# 다음과 같이 import를 사용할 수 있습니다.
# import math

def solution(s1, s2):
    # 여기에 코드를 작성해주세요.
    answer = 0
    return answer

# 아래는 테스트케이스 출력을 해보기 위한 코드입니다.
s1 = "ababc"
s2 = "abcdab"
ret = solution(s1, s2)

#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret, "입니다.")
```

2) 문제 개요

- 제시된 과제를 해결하기 위해 solution( )에 프로그램 코드를 작성하는 문제.
- 문자열 슬라이싱을 이용하여 두 개의 문자열의 앞뒤 부분에서 공통된 문자 패턴을 찾고, 공통된 부분은 아래 그림과 같이 한 번만 보이도록 두 문자열을 붙여서 그 문자열의 길이를 구하는 문제.

3) 정답

```
def solution(s1, s2):
    # 여기에 코드를 작성해주세요.
    answer = 0
    comman_len = 0
①   min_len = min(len(s1), len(s2))
    #s1 의 앞 부분과 s2 의 뒷 부분의 공통 문자열 찾기
    for i in range(1,min_len+1):
②       if s1[:i] == s2[i * -1:]:
            comman_len = i
    #s2 의 앞 부분과 s1 의 뒷 부분의 공통 문자열 찾기
    for i in range(1,min_len+1):
③       if s2[:i] == s1[i * -1:]:
            comman_len = max(comman_len, i)
④   answer = len(s1) + len(s2) - comman_len
    return answer

# 아래는 테스트케이스 출력을 해보기 위한 코드입니다.
s1 = "ababc"
s2 = "abcdab"
ret = solution(s1, s2)

#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret, "입니다.")
```

① s1 문자열 길이와 s2 문자열 길이 중 작은 값은 min_len 으로 저장.

② for 문을 이용하여 문자열 s1 앞 부분과 s2 뒷 부분에서 공통 패턴의 길이를 구하여 comman_len 에 저장.

③ for 문을 이용하여 문자열 s1 의 뒷 부분과 s2 의 앞 부분에서 공통 패턴의 길이를 찾은 후, 앞에서 구한 comman_len 와 현재 구한 공통 패턴의 길이 중 큰 값을 다시 comman_len 에 저장 ← 문자열 s1, s2 의 최대 공통 패턴의 길이가 comman_len 에 저장됨.

④ 문자열 s1, s2 의 길이를 더한 값에 comman_len(최대 공통 패턴의 길이)를 뺀 값을 answer 에 할당.

5. 문제 5

1) 문제 코드

```python
# 다음과 같이 import를 사용할 수 있습니다.
# import math

def solution(phrases, second):
    # 여기에 코드를 작성해주세요.
    answer = ''
    return answer

#아래는 테스트케이스 출력을 해보기 위한 코드입니다.
phrases = "happy-birthday"
second = 3
ret = solution(phrases, second)

#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret, "입니다.")
```

2) 문제 개요

- 제시된 과제를 해결하기 위해 solution( )에 프로그램 코드를 작성하는 문제.
- 14 자까지 표시하는 화면에 '_____' 으로 시작하여 'happy-birthday'의 문자를 1 글자씩 왼쪽으로 이동하며 표시하고, 모든 문자를 표시한 뒤에는 다시 '_____' 부터 시작하여 한 글자씩 나타내도록 프로그램을 구성해야 함.

3) 정답

```python
def solution(phrases, second):
    # 여기에 코드를 작성해주세요.
    display = '_'*14 + phrases                  ①
    for i in range(second):                     ②
        display = display[1:] + display[0]
    answer = display[:14]                       ③
    return answer

#아래는 테스트케이스 출력을 해보기 위한 코드입니다.
phrases = "happy-birthday"
second = 14
ret = solution(phrases, second)

#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret, "입니다.")
```

①. 시작할 때 보여줘야 하는 문자열인 '_' 문자 14 개와 시간이 경과하면서 한 개씩 보여줘야 하는 문자열을 담은 phrases 를 붙인 것을 display 변수의 값으로 초기화.

②. 매개변수 second 의 값만큼 반복하며 현재 display 변수가 갖는 두 번째 문자부터 마지막 문자까지 슬라이싱 한 것에 현재 display 변수가 갖는 첫 번째 문자를 붙인 것을 다시 display 에 할당..

→display 변수가 갖는 문자열의 가장 왼쪽 문자는 오른쪽 끝으로 이동하고 나머지 문자들은 왼쪽으로 한 자리씩 이동.

③. 마지막 결과로 산출된 display 문자열에서 14 개의 문자만 잘라서 return.

※ 28 개의 문자가 회전하며 나타나기 때문에 for 문의 반복횟수를 second → secode%28 로 바꿔 사용하는 것이 더 효과적.

6. 문제 6

4) 문제 코드

```python
def solution(n):
    answer = 0
    primes = [2]
    for i in range (3, n + 1, 2) :
        is_prime = True
        for j in range(2, i) :
            if i % j == 0 :
                is_prime = False
                break
        if @@@ :
            primes.append(i)

    prime_len = len(primes)
    for i in range(0, prime_len - 2) :
        for j in range(i + 1, prime_len - 1) :
            for k in range(j + 1, prime_len) :
                if @@@ :
                    answer += 1
    return answer

#아래는 테스트케이스 출력을 해보기 위한 코드입니다.
n1 = 33
ret1 = solution(n1)
#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret1, "입니다.")

n2 = 9
ret2 = solution(n2)
#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret2, "입니다.")
```

5) 문제 개요

- 제시된 과제를 해결하기 위해 작성한 solution( ) 함수에서 비어 있는 구문을 채워 완성하는 문제.
- n 보다 작은 소수 3 개를 더하여 인수로 받은 n 을 만들 수 있는 개수를 찾는 프로그램에서 비어 있는 부분을 채워야 함.

6) 정답

```
def solution(n):
    answer = 0
①   primes = [2]
②   for i in range (3, n + 1, 2) :
        is_prime = True
③       for j in range(2, i) :
            if i % j == 0 :
④               is_prime = False
                break
        if is_prime == True :
⑤           primes.append(i)

    prime_len = len(primes)
    for i in range(0, prime_len - 2) :
        for j in range(i + 1, prime_len - 1) :
⑥           for k in range(j + 1, prime_len) :
                if (primes[i] + primes[j] + primes[k] ) == n :
                    answer += 1
    return answer

#아래는 테스트케이스 출력을 해보기 위한 코드입니다.
n1 = 33
ret1 = solution(n1)
#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret1, "입니다.")

n2 = 9
ret2 = solution(n2)
#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret2, "입니다.")
```

① primes 에 소수인 2 를 항목으로 갖는 리스트를 할당.

② for 문을 이용하여 3 이상 n 이하의 홀수를 i 로 가져옴.

③ 중첩 for 문을 이용하여 2 부터 i 보다 작은 정수를 j 로 가져옴.

④ i 를 j 로 나눈 나머지가 0 이면 j 는 i 의 약수이므로 i 는 소수가 아니기 때문에 is_prime = False 로 한 후 break 명령을 이용하여 중첩 for 문 종료.

⑤ is_prime 값이 True 이면 i 는 소수이므로 primes 에 추가.

- if 문을 이용하여 is_prime 의 값이 True 이면 primes 에 i 를 추가.

⑥ 중첩 for 문을 이용하여 primes 리스트에 있는 소수 3 개를 중복되지 않게 가져옴.

- 첫 번째 for 문으로 primes 의 항목을 0 번째부터 가져와 i 에 저장.
- 두 번째 for 문으로 primes 의 항목을 i 다음부터 가져와 j 에 저장.
- 세 번째 for 문으로 primes 의 항목을 j 다음부터 가져와 k 에 저장.
- i + j + k 의 결과가 n 과 같으면 answer 값을 1 만큼 증가.

7. 문제 7

5) 문제 코드

```python
def solution(k):
    answer = []
    for i in range(1, k + 1):
        square_num = i * i
        divisor = 1
        while square_num % divisor != 0:
            front = square_num // divisor
            back = square_num % divisor
            divisor *= 10
            if back != 0 and front != 0:
                if front + back == i:
                    answer.append(i)
    return answer

#아래는 테스트케이스 출력을 해보기 위한 코드입니다. 아래 코드는
k = 500
ret = solution(k)

#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은 ", ret, " 입니다.")
```

6) 문제 개요

- 제시된 과제가 바르게 수행되도록 문제 코드를 수정하는 문제.
- 어떤 수를 제곱한 결과를 두 부분으로 나눈 뒤, 나뉘어진 두 수를 더한 것이 원래의 수와 같은 것을 '카프리카 수'라고 하는데, 인수로 받은 k 보다 작거나 같은 '카프리카 수'를 찾아오는 프로그램에서 잘못된 부분을 수정해야 함.
- 제곱한 수를 두 부분으로 나누는 방법에 대한 이해가 필요함.
  예) 3125 를 두 부분으로 나누는 방법.
  ① 312 와 5 로 나누기 : 3125//10 = 312 ; 3125%10 = 5
  ② 31 와 25 로 나누기 : 3125//100 = 31 ; 3125%100 = 25
  ③ 3 와 125 로 나누기 : 3125//1000 = 3 ; 3125%1000 = 125

7) 정답

```
def solution(k):
    answer = []
①   for i in range(1, k + 1):
        square_num = i * i
②       divisor = 1
        while square_num // divisor != 0:
③           front = square_num // divisor
            back = square_num % divisor
            divisor *= 10
            if back != 0 and front != 0:
④               if front + back == i:
                    answer.append(i)
    return answer

#아래는 테스트케이스 출력을 해보기 위한 코드입니다. 아래 코드
k = 500
ret = solution(k)

#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은 ", ret, " 입니다.")
```

① for 문을 이용하여 1 부터 k 이하의 정수를 가져옴.

② 제곱수를 두 부분으로 나누기 위한 기준 수를 갖는 divisor 를 10^0= 1 로 초기화.

③ 제곱수를 divisor(기준 수)로 나눈 몫이 0 이 아닌 동안 반복하도록 조건식을 수정.

- front = 제곱수를 divisor(기준 수)로 나눈 몫.
- back = 제곱수를 divisor(기준 수)로 나눈 나머지.

  → 제곱수를 두 부분으로 나누어 front 와 back 에 저장.
- divisor 에 10 을 곱하여 다음 번 반복에 사용할 기준 수를 계산.

④ front 와 back 의 값이 모두 0 이 아니면서 두 변수를 더한 값이 본래의 수와 같으면 answer 에 추가.

8. 문제 8

4) 문제 코드

```python
def solution(k, student):
    answer = 0
    for s in student:
        s -= 4*k
        if s <= 0:
            break
        answer += (s + k - 1) // k
    return answer

#아래는 테스트케이스 출력을 해보기 위한 코드입니다. 아래 코
k1 = 1
student1 = [4, 4, 4, 4]
ret1 = solution(k1, student1)

#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret1, "입니다.")

k2 = 3
student2 = [15, 17, 19, 10, 23]
ret2 = solution(k2, student2)

#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret2, "입니다.")
```

5) 문제 개요

- 제시된 과제가 바르게 수행되도록 문제 코드를 수정하는 문제.
- 교실에 선풍기 4 대가 있다는 조건에서 한 대의 선풍기가 송풍할 수 있는 학생 수 k 와 각 반의 학생 수 리스트 student 를 매개변수로 받아서 바람을 받지 못하는 학생 수를 구한 다음 더 필요한 선풍기 수를 구해야 함.

6) 정답

```python
def solution(k, student):
    answer = 0
    for s in student:
①       s -= 4*k
        if s <= 0:
②           continue
③       answer += (s + k - 1) // k
    return answer

#아래는 테스트케이스 출력을 해보기 위한 코드입니다. 아래 코
k1 = 1
student1 = [4, 4, 4, 4]
ret1 = solution(k1, student1)

#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret1, "입니다.")

k2 = 3
student2 = [15, 17, 19, 10, 23]
ret2 = solution(k2, student2)

#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret2, "입니다.")
```

⑧. for 문을 이용해 학급 당 학생 수를 s 로 받은 후, 교실에 있는 4 대의 선풍기의 바람을 받지 못하는 학생 수를 구함.

- { 학생 수 s – (선풍기 4 대 * 한 대의 선풍기가 송풍하는 학생 수) } 를 계산하면 바람을 받지 못하는 학생 수가 계산됨.

⑨. 바람을 받지 못하는 학생 수가 0 이하이면 현재 학급에서는 더 필요한 선풍기 수를 구할 필요가 없으므로 다음 학급의 학생 수를 가져와서 작업을 계속하기 위한 명령인 continue 를 사용.

← 문제 코드에 있던 break 를 사용하면, 바람을 받지 못한 학생 수가 0 이하인 경우 반복문은 더 이상 실행되지 않고 강제 종료되어 다음 학급에서 필요한 선풍기 대수를 구할 수 없음.

⑩. 바람을 받지 못하는 학생들을 위해 추가로 구입해야 하는 선풍기 대수를 계산.

- ◆ { 바람을 받지 못한 학생 수를 k(선풍기 한 대가 송풍하는 학생 수)로 나눈 몫 + k 로 나눈 나머지가 1 이상 k-1 이하인 경우에도 제공해야 하는 선풍기 1 대 }의 계산이 이루어져야 함.
- ◆ { 바람을 받지 못하는 학생수 + (k 로 나누었을 때 나타날 수 있는 최대 나머지 = k-1) } 값을 k 로 나눈 몫을 구함.
- ◆ (바람을 받지 못한 학생 수/k) 를 계산한 값을 math 모듈의 ceil( ) 함수를 이용하여 계산해도 동일한 결과를 얻을 수 있음.

9. 문제 9

4) 문제 코드

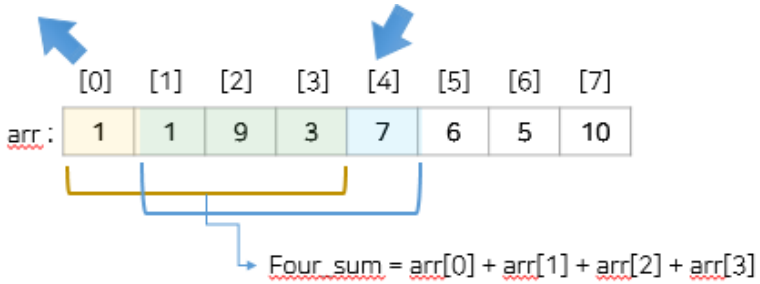
```python
def solution(revenue, k) :
    answer = 0
    rsum = sum(revenue[0:k])
    answer = rsum
    for i in range(len(revenue)) :
        rsum = rsum - revenue[i - k] + revenue[i]
        if answer < rsum :
            answer = rsum
    return answer

#아래는 테스트케이스 출력을 해보기 위한 코드입니다. 아래 코드는 절
revenue1 = [1, 1, 9, 3, 7, 6, 5, 10]
k1 = 4
ret1 = solution(revenue1, k1)

#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은 ", ret1, " 입니다.")

revenue2 = [1, 1, 5, 1, 1]
k2 = 1
ret2 = solution(revenue2, k2)

#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은 ", ret2, " 입니다.")
```

5) 문제 개요

- 제시된 과제가 바르게 수행되도록 문제 코드를 수정하는 문제.
- 슬라이딩윈도우 방식을 사용하여 매출액 리스트 revenue 에서 연속된 k 개의 매출액 합계가 최대인 값을 찾는 코드에서 잘못된 곳을 찾아 수정해야 함.
- 슬라이딩 윈도우 방식

  : 일정한 너비의 창문을 옆으로 미는 것처럼 연속된 일정 개수의 항목을 이용하여 문제를 풀 때 사용하는 방법.

  예) 연속된 4 개 항목의 합을 구하는 경우.



  ◆ 리스트 arr 에서 연속된 4 개의 항목 합계를 구하여 Four_sum 에 저장한 후, 다음 4 개의 항목 합을 구하기 위해서 Four_sum 에서 arr[0]을 빼고, arr[4] 를 더하면 원하는 값을 구할 수 있음.

6) 정답

```
def solution(revenue, k) :
    answer = 0
①   rsum = sum(revenue[0:k])
    answer = rsum
②   for i in range(k,len(revenue)) :
③       rsum = rsum - revenue[i - k] + revenue[i]
        if answer < rsum :
            answer = rsum
    return answer

#아래는 테스트케이스 출력을 해보기 위한 코드입니다. 아래 코드는 잘못
revenue1 = [1, 1, 9, 3, 7, 6, 5, 10]
k1 = 4
ret1 = solution(revenue1, k1)

#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은 ", ret1, " 입니다.")

revenue2 = [1, 1, 5, 1, 1]
k2 = 1
ret2 = solution(revenue2, k2)

#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은 ", ret2, " 입니다.")
```

⑤. 첫 번째 매출액부터 k 번째 매출액까지의 합을 구하여 rsum 변수와 answer 변수값으로 초기화.

⑥. for 문을 이용하여 k 부터 (매출액 항목 수 – 1) 까지의 값을 i 로 받아 옴.

⑦. 새로운 k 개의 매출액 합계를 계산 : 현재 구한 매출액 합계에서 가장 앞에 위치한 항목 값을 빼고 현 매출액 합계에서 마지막 항목의 다음 값을 더해서 새로운 k 개의 매출액 합계를 계산.

7) 보충 학습 : 연속된 k 개의 합 중 최댓값 구하기 - 브루트 포스 방식 vs. 슬라이딩 윈도우 방식

① 브루트 포스 방식 :

기준 항목을 선택한 후 중첩 for 문을 이용하여 기준 항목부터 k 개의 항목을 합산하여 연속된 k 개의 합 중 최댓값을 찾음.

```python
def solution(revenue, k) :
    answer = 0

    for i in range(len(revenue)-k+1) :
        rsum = 0
        #i 번째부터 k개의 합 구하기
        for j in range(i, i+k):
            rsum += revenue[j]
        answer = max(answer, rsum)
    return answer
```

```python
def solution2(revenue, k):
    answer = 0

    for i in range(len(revenue)-k+1):
        rsum = 0
        #i번째부터 k개의 합 구하기
        for j in range(k):
            rsum += revenue[i+j]
        answer=max(answer, rsum)
    return answer
```

➔ 시간 복잡도 = O(k*n)

② 슬라이딩 윈도우 방식 :

처음 k 개의 합계를 구한 뒤 for 문을 이용해 반복하면서 이전에 구했던 합계 구간에서 이전 구간의 첫 번째 항목값은 빼고 이전 구간의 바로 다음에 있는 항목값을 더하면서 연속된 k 개의 합계 중 최댓값을 찾음.

➔ 이전에 구한 합계 값 일부를 사용함으로써 반복되는 계산 횟수를 줄일 수 있음.

```python
def solution(revenue, k) :
    answer = 0
    rsum = sum(revenue[0:k])
    answer = rsum
    for i in range(k,len(revenue)) :
        rsum = rsum - revenue[i - k] + revenue[i]
        if answer < rsum :
            answer = rsum
    return answer
```

➔ 시간 복잡도 = O(n)

8) 이론 정리 : 브루트 포스, 슬라이딩 윈도우, 투 포인터

① 특징

브루트 포스 (Brute Force)	슬라이딩 윈도우 (Sliding Window)	투 포인터 (Two Pointer)
모든 경우를 탐색해야 할 때 사용	연속된 구간의 데이터에 대한 문제일 때 사용	데이터들이 정렬되어 있고 중복된 값이 없을 때 사용
중첩 반복문을 사용하여 순회	일정 구간을 정하여 이동하며 답을 구함	두 개의 포인터를 활용

② 시간 복잡도

구분	브루트 포스	슬라이딩 윈도우	투 포인터
세 수의 합이 K 의 배수가 되는 경우 구하기	$O(n^3)$		
연속된 K 개 수의 합의 최댓값 구하기	O(k*n)	O(n)	
세 수의 합이 K 가 되는 수 구하기	$O(n^3)$		$O(n^2)$

10. 문제 10

4) 문제 코드

```
class Customer:
    def __init__(self, id, time, num_of_people):
        self.id = id
        self.time = time
        self.num_of_people = num_of_people

class Shop:
    def __init__(self):
        self.reserve_list = []

    def reserve(self, customer):
        self.reserve_list.append(customer)
        return True

class HairShop@@@:
    def __init__(self):
        super().__init__()

    @@@:
        if @@@ != 1:
            return False
        for r in self.reserve_list:
            if @@@:
                return False
        self.reserve_list.append(customer)
        return True

class Restaurant@@@:
    def __init__(self):
        super().__init__()

    @@@:
        if @@@:
            return False
        count = 0
        for r in self.reserve_list:
            if @@@:
                count += 1
        if count >= 2:
            return False
        self.reserve_list.append(customer)
        return True


def solution(customers, shops):
    hairshop = HairShop()
    restaurant = Restaurant()

    count = 0
    for customer, shop in zip(customers, shops):
        if shop == "hairshop":
            if hairshop.reserve(Customer(customer[0], customer[1], customer[2])):
                count += 1
        elif shop == "restaurant":
            if restaurant.reserve(Customer(customer[0], customer[1], customer[2])):
                count += 1

    return count

#아래는 테스트케이스 출력을 해보기 위한 코드입니다.
customers = [[1000, 2, 1],[2000, 2, 4],[1234, 5, 1],[4321, 2, 1],[1111, 3, 10]]
shops = ["hairshop", "restaurant", "hairshop", "hairshop", "restaurant"]
ret = solution(customers, shops)

#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret, "입니다.")
```

5) 문제 개요

- 일반 클래스인 Shop 클래스를 상속하는 HairShop 클래스와 Restaurant 클래스를 정의하는 코드를 완성하는 문제.

- HairShop 클래스의 reserve( ) 메소드에서는 신규 예약 고객 수가 1 명인 경우에 예약 가능하지만, 신규 예약 고객의 예약 시간이 기존 예약 손님들의 예약 시간과 중복되면 예약할 수 없도록 코딩해야 함.
- Restaurant 클래스의 reserve( ) 메소드에서는 신규 예약 고객 수가 2 명 이상 8 명 이하인 경우에 예약 가능하지만, 신규 예약 고객의 예약 시간에 기존 예약 건수가 2 건이 존재하면 예약할 수 없도록 코딩해야 함.
- solution( ) 함수는 [고객 식별번호, 고객이 신청한 예약 시간, 예약 인원 수]를 항목으로 갖는 리스트 customers 와 'hairshop' 혹은 'restaurant' 로 구성된 리스트 shops 를 인수로 받고, HairShop 클래스의 객체와 Restaurant 클래스의 객체를 생성하고 reserve( ) 메소드를 실행해서 고객 예약 횟수를 집계.

6) 정답

```
① class Customer:
    def __init__(self, id, time, num_of_people):
        self.id = id
        self.time = time
        self.num_of_people = num_of_people

② class Shop:
    def __init__(self):
        self.reserve_list = []

    def reserve(self, customer):
        self.reserve_list.append(customer)
        return True

③class HairShop(Shop):
    def __init__(self):
        super().__init__()

    def reserve(self, customer):
        if customer.num_of_people != 1:
            return False
        for r in self.reserve_list:
            if r.time == customer.time:
                return False
        self.reserve_list.append(customer)
        return True

class Restaurant(Shop):
    def __init__(self):
        super().__init__()

④   def reserve(self, customer):
        if customer.num_of_people < 2 or customer.num_of_people > 8:
            return False
        count = 0
        for r in self.reserve_list:
            if r.time == customer.time:
                count += 1
        if count >= 2:
            return False
        self.reserve_list.append(customer)
        return True


def solution(customers, shops):
    hairshop = HairShop()
    restaurant = Restaurant()

    count = 0
⑤   for customer, shop in zip(customers, shops):
⑥       if shop == "hairshop":
            if hairshop.reserve(Customer(customer[0], customer[1], customer[2])):
                count += 1
⑦       elif shop == "restaurant":
            if restaurant.reserve(Customer(customer[0], customer[1], customer[2])):
                count += 1

    return count

#아래는 테스트케이스 출력을 해보기 위한 코드입니다.
customers = [[1000, 2, 1],[2000, 2, 4],[1234, 5, 1],[4321, 2, 1],[1111, 3, 10]]
shops = ["hairshop", "restaurant", "hairshop", "hairshop", "restaurant"]
ret = solution(customers, shops)

#[실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret, "입니다.")
```

- Customer 클래스의 생성자 메소드에 의해서 세 개의 멤버 변수를 매개변수로 전달된 값으로 초기화.
  - id : 고객 id (1 이상 10,000 이하의 자연수)를 매개변수 id 의 값으로 초기화.

- time : 예약 시간 (0 이상 23 이하의 자연수)를 매개변수 time 의 값으로 초기화.
- num_of_people : 예약 인원 수 (1 이상 10 이하의 자연수)를 매개변수 num_of_people 로 초기화.

- Shop 클래스 정의
  - 생성자 메소드에서 멤버 변수 reserve_list 를 생성하여 빈 리스트를 할당.
  - reserve( ) 메소드에서 매개변수로 받은 customer 를 reserve_list 에 추가하고 True 를 return.
- Shop 클래스를 상속하도록 HairShop 클래스 정의.
  - 생성자 메소드는 부모 클래스의 생성자 메소드를 그대로 실행하므로 HairShop 클래스의 멤버 변수로 reseve_list 가 빈 리스트로 생성됨.
  - reserve( ) 메소드를 오버라이드
    - ◆ Customer 클래스 타입의 매개변수 customer 의 멤버 변수 num_of_people 값이 1 이 아니면 False 를 리턴.
    - ◆ reserve_list 에 있는 예약 고객 데이터에서 customer 의 멤버 변수 time 과 중복된 예약 시간이 존재하면 False 를 return.
    - ◆ 위의 두 가지 조건에 해당하지 않으면 reserve_list 에 매개변수 customer 를 추가하고 True 를 return.

- Shop 클래스를 상속하도록 Restaurant 클래스 정의.
  - 생성자 메소드는 부모 클래스의 생성자 메소드를 그대로 실행하므로 Restaurant 클래스의 멤버 변수로 reseve_list 가 빈 리스트로 생성됨.
  - reserve( ) 메소드를 오버라이드
    - ◆ Customer 클래스 타입의 매개변수 customer 의 멤버 변수 num_of_people 값이 2 보다 작거나 8 보다 크면 False 를 리턴.
    - ◆ reserve_list 에 있는 예약 고객 데이터에서 매개변수 customer 의 멤버 변수 time 값과 같은 예약 고객 수를 count 변수에 집계한 후, count 변수의 값이 2 이상이면 False 를 return.
    - ◆ 위의 두 가지 조건에 해당하지 않으면 reserve_list 에 매개변수 customer 를 추가하고 True 를 return.
- for 반복문에서 zip( ) 함수를 이용하여 customers, shops 리스트에서 같은 인덱스 항목을 순서대로 받아와 각각 customer, shop 에 저장.
- shop 의 값이 'hairshop' 이면 hairshop 객체의 reserve( ) 메소드를 실행하고 그 결과가 True 이면 예약 건수를 집계하는 count 값을 1 만큼 증가.
  - hairshop 의 reserve( ) 메소드의 매개변수에, customer 의 항목 3 개를 이용하여 생성한 Customer 클래스의 객체를 전달.

- shop 의 값이 'restaurant' 이면 restaurant 객체의 reserve( ) 메소드를 실행하고 그 결과가 True 이면 예약 건수를 집계하는 count 값을 1 만큼 증가.
  - restaurant 의 reserve( ) 메소드의 매개변수에 customer 의 항목 3 개를 이용하여 생성한 Customer 클래스의 객체를 전달.