**프로그래밍 실습 #7**

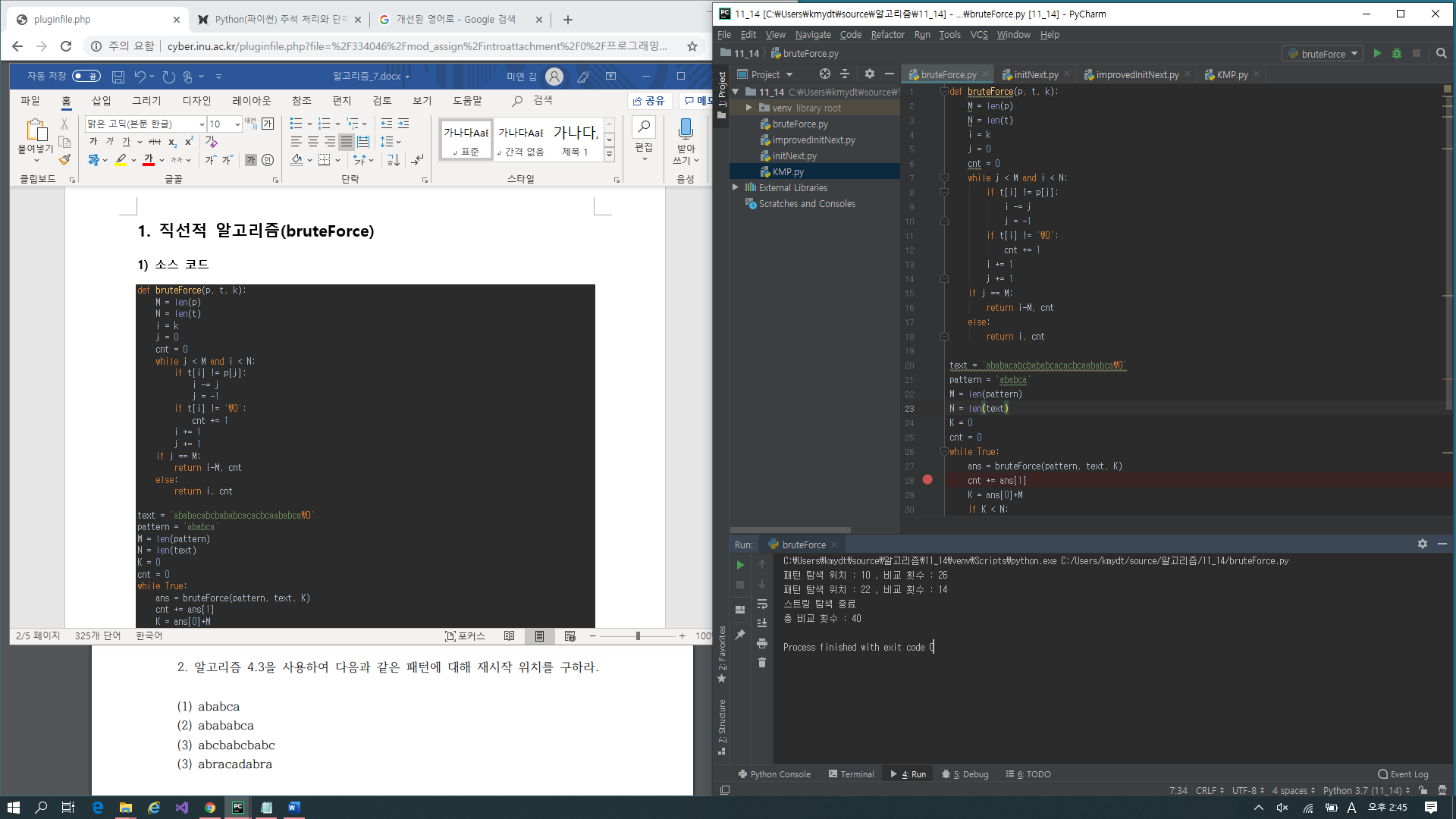
201501489 최영진

**1. 직선적 알고리즘(bruteForce)**

**1) 소스 코드**

def bruteForce(p, t, k):  
 M = len(p)  
 N = len(t)  
 i = k  
 j = 0  
 cnt = 0  
 while j < M and i < N:  
 if t[i] != p[j]:  
 i -= j  
 j = -1  
 if t[i] != '\0':  
 cnt += 1  
 i += 1  
 j += 1  
 if j == M:  
 return i-M, cnt  
 else:  
 return i, cnt  
  
text = 'ababacabcbababcacacbcaababca\0'  
pattern = 'ababca'  
M = len(pattern)  
N = len(text)  
K = 0  
cnt = 0  
while True:  
 ans = bruteForce(pattern, text, K)  
 cnt += ans[1]  
 K = ans[0]+M  
 if K < N:  
 print('패턴 탐색 위치 :', ans[0], ', 비교 횟수 :', ans[1])  
 else:  
 break  
print('스트링 탐색 종료')  
print('총 비교 횟수 :', cnt)

**2) 실행 결과**



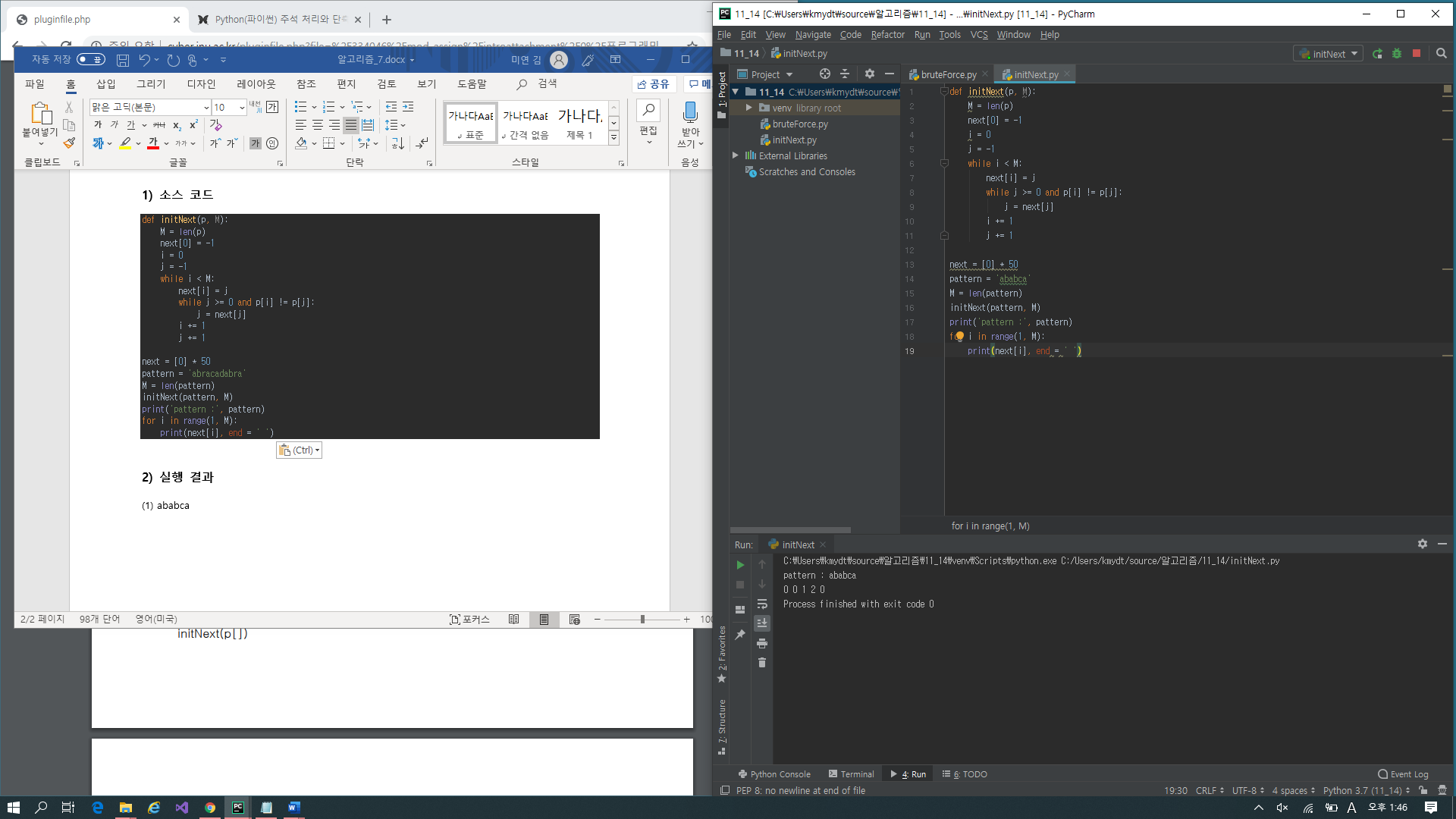
**2. 재시작 위치 알고리즘(initNext)**

**1) 소스 코드**

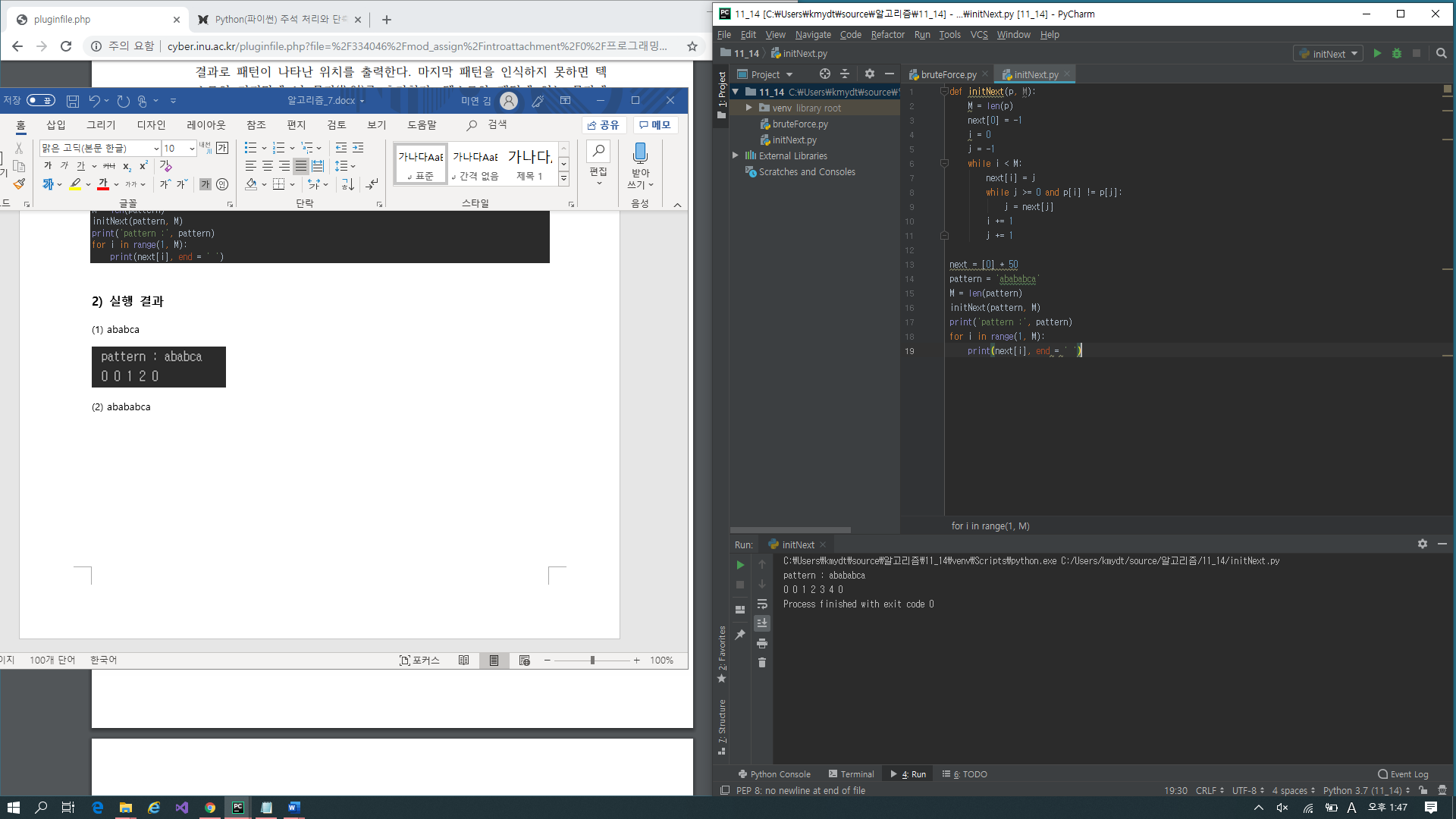
def initNext(p, M):  
 M = len(p)  
 next[0] = -1  
 i = 0  
 j = -1  
 while i < M:  
 next[i] = j  
 while j >= 0 and p[i] != p[j]:  
 j = next[j]  
 i += 1  
 j += 1  
  
next = [0] \* 50  
pattern = 'abracadabra'  
M = len(pattern)  
initNext(pattern, M)  
print('pattern :', pattern)  
for i in range(1, M):  
 print(next[i], end = ' ')

**2) 실행 결과**

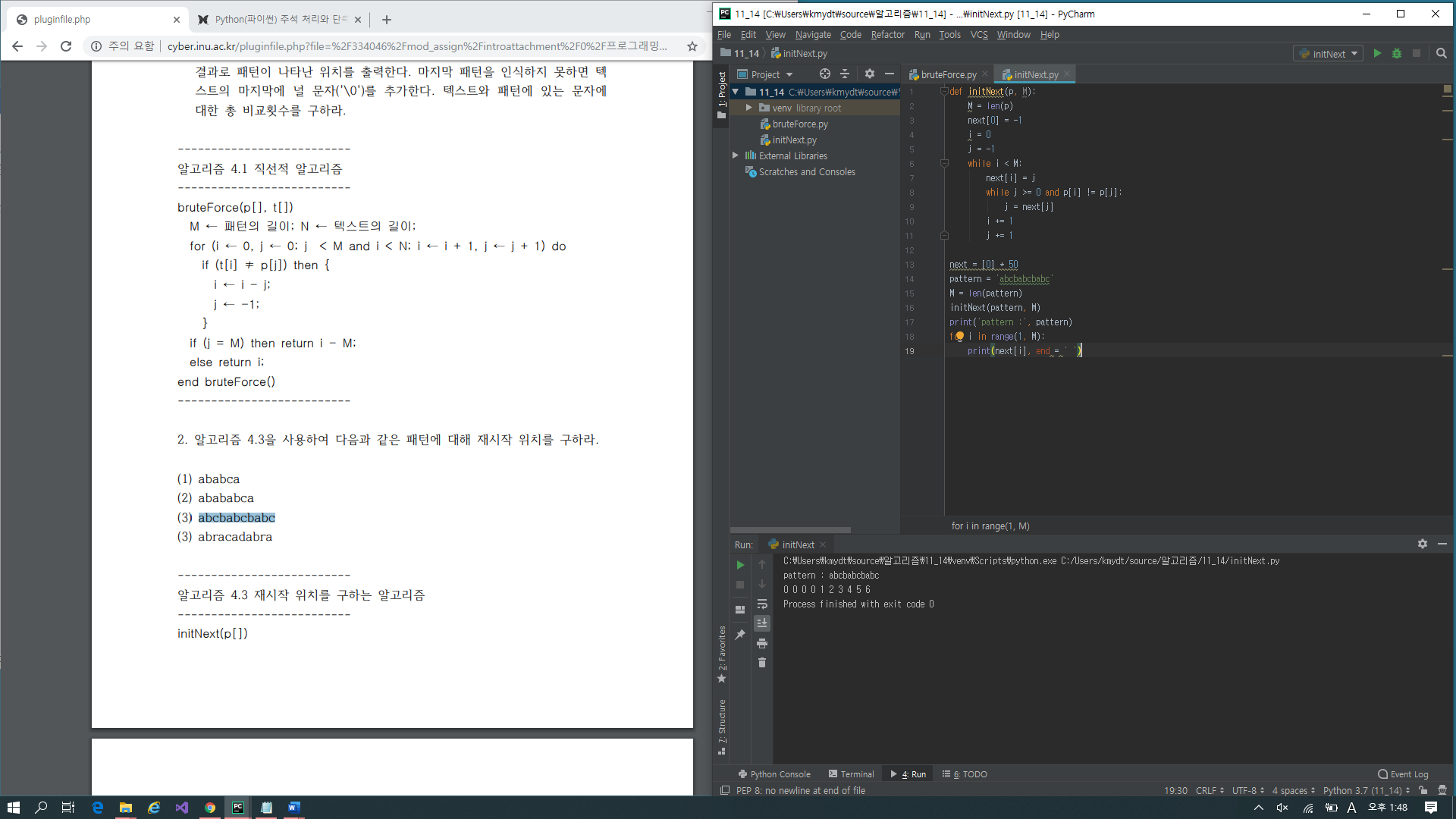
(1) ababca



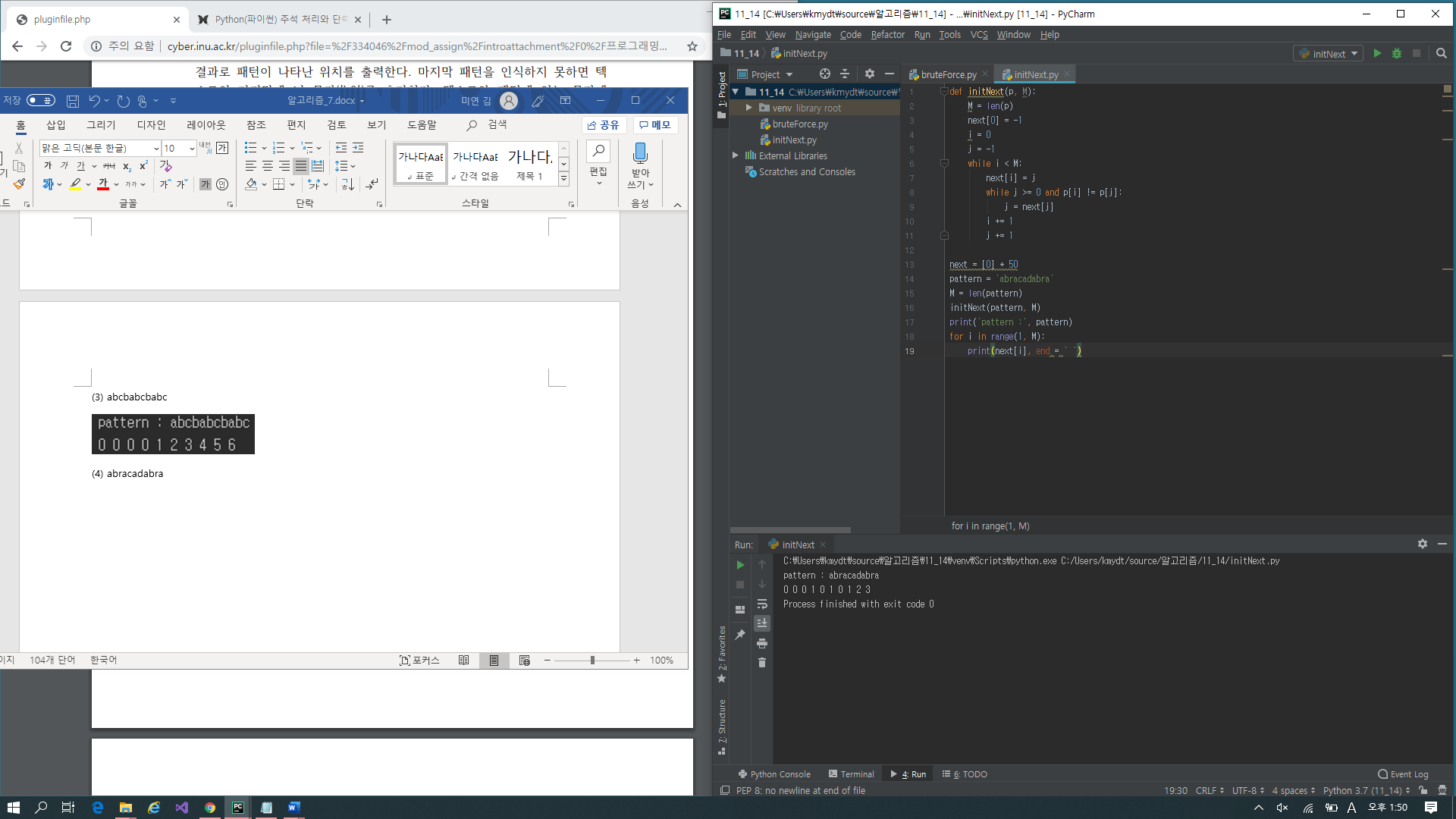
(2) abababca



(3) abcbabcbabc



(4) abracadabra



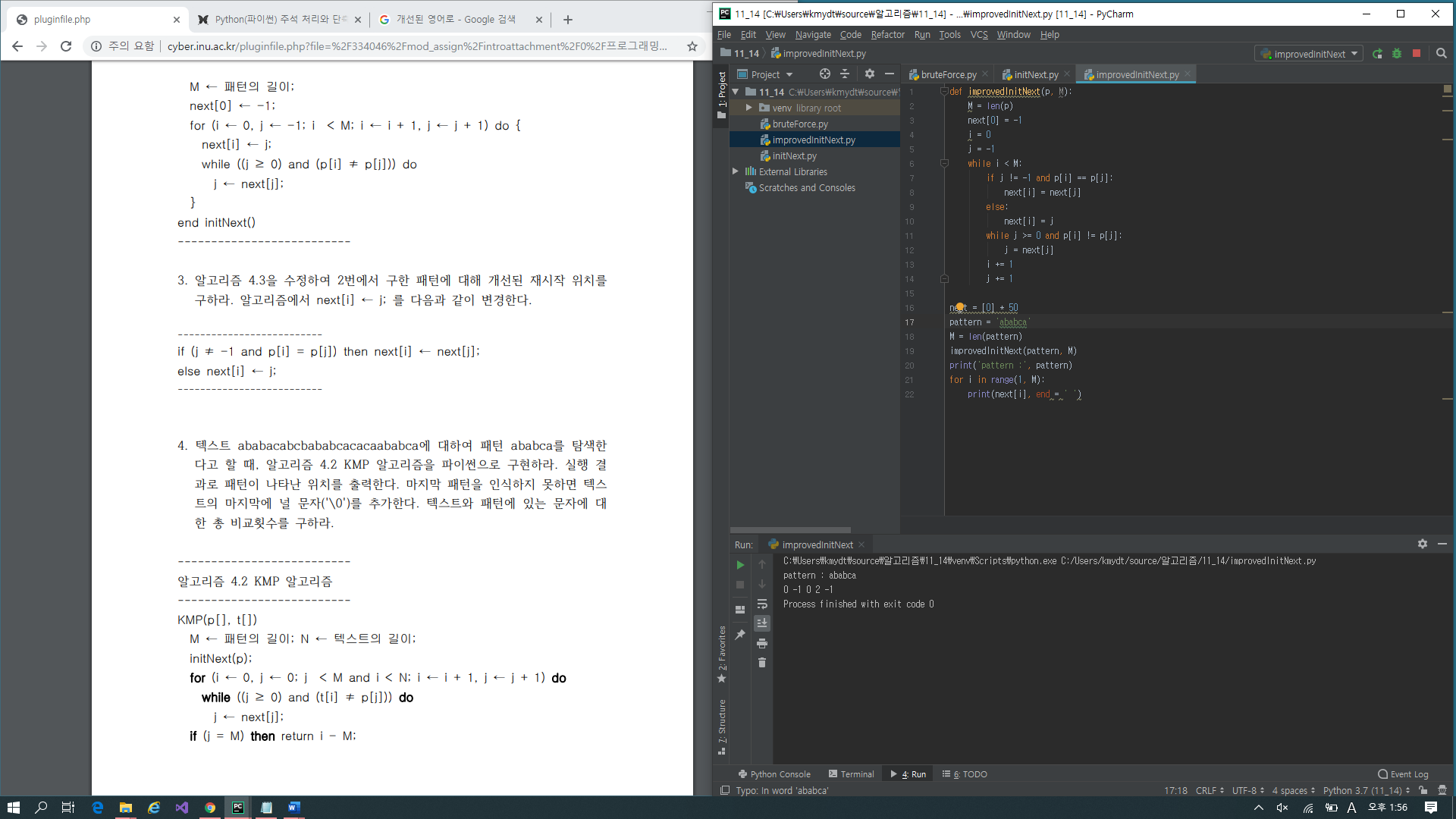
**3. 개선된 재시작 위치 알고리즘(improvedInitNext)**

**1) 소스 코드**

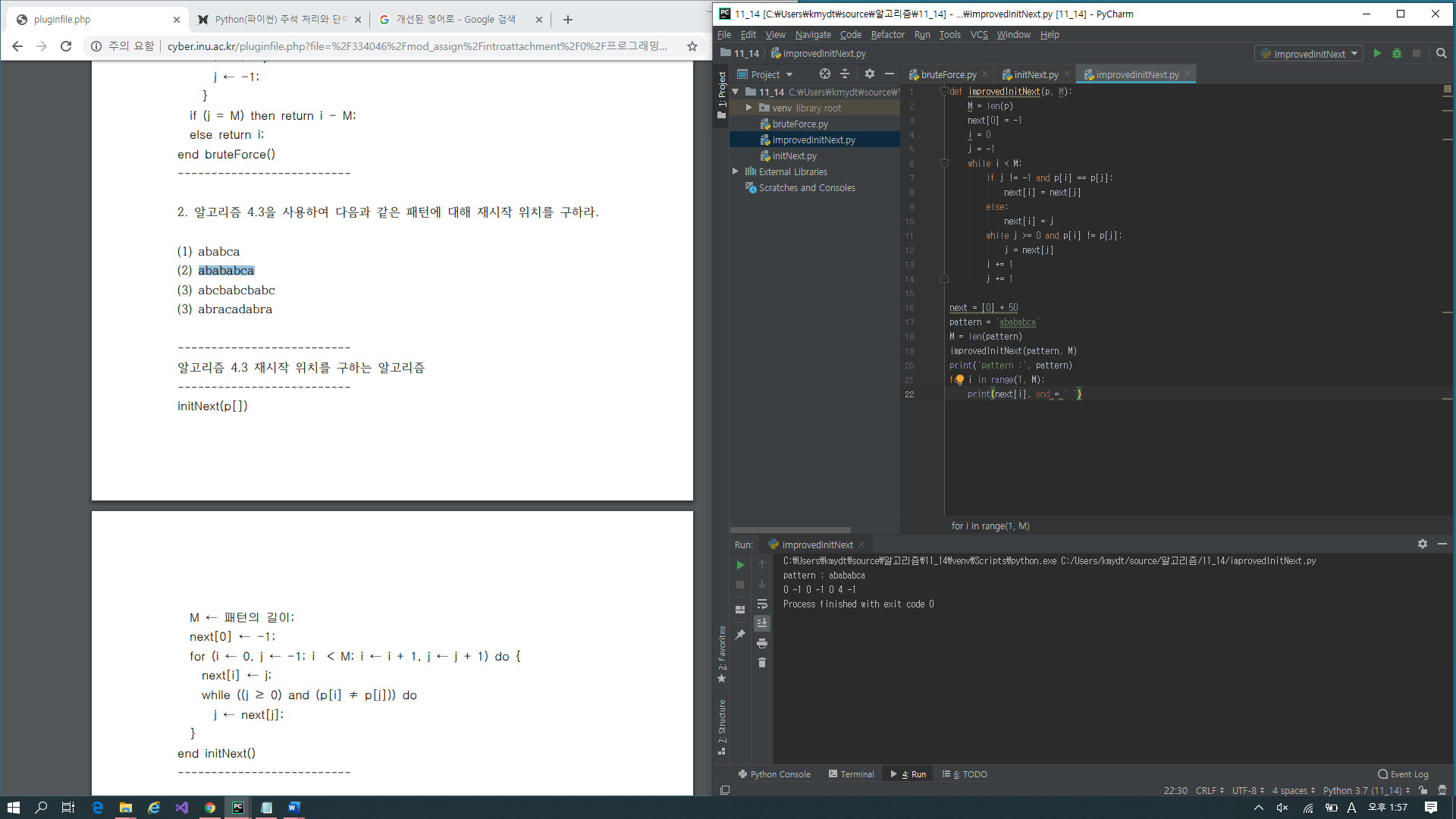
def improvedInitNext(p, M):  
 M = len(p)  
 next[0] = -1  
 i = 0  
 j = -1  
 while i < M:  
 if j != -1 and p[i] == p[j]:  
 next[i] = next[j]  
 else:  
 next[i] = j  
 while j >= 0 and p[i] != p[j]:  
 j = next[j]  
 i += 1  
 j += 1  
  
next = [0] \* 50  
pattern = 'abracadabra'  
M = len(pattern)  
improvedInitNext(pattern, M)  
print('pattern :', pattern)  
for i in range(1, M):  
 print(next[i], end = ' ')

**2) 실행 결과**

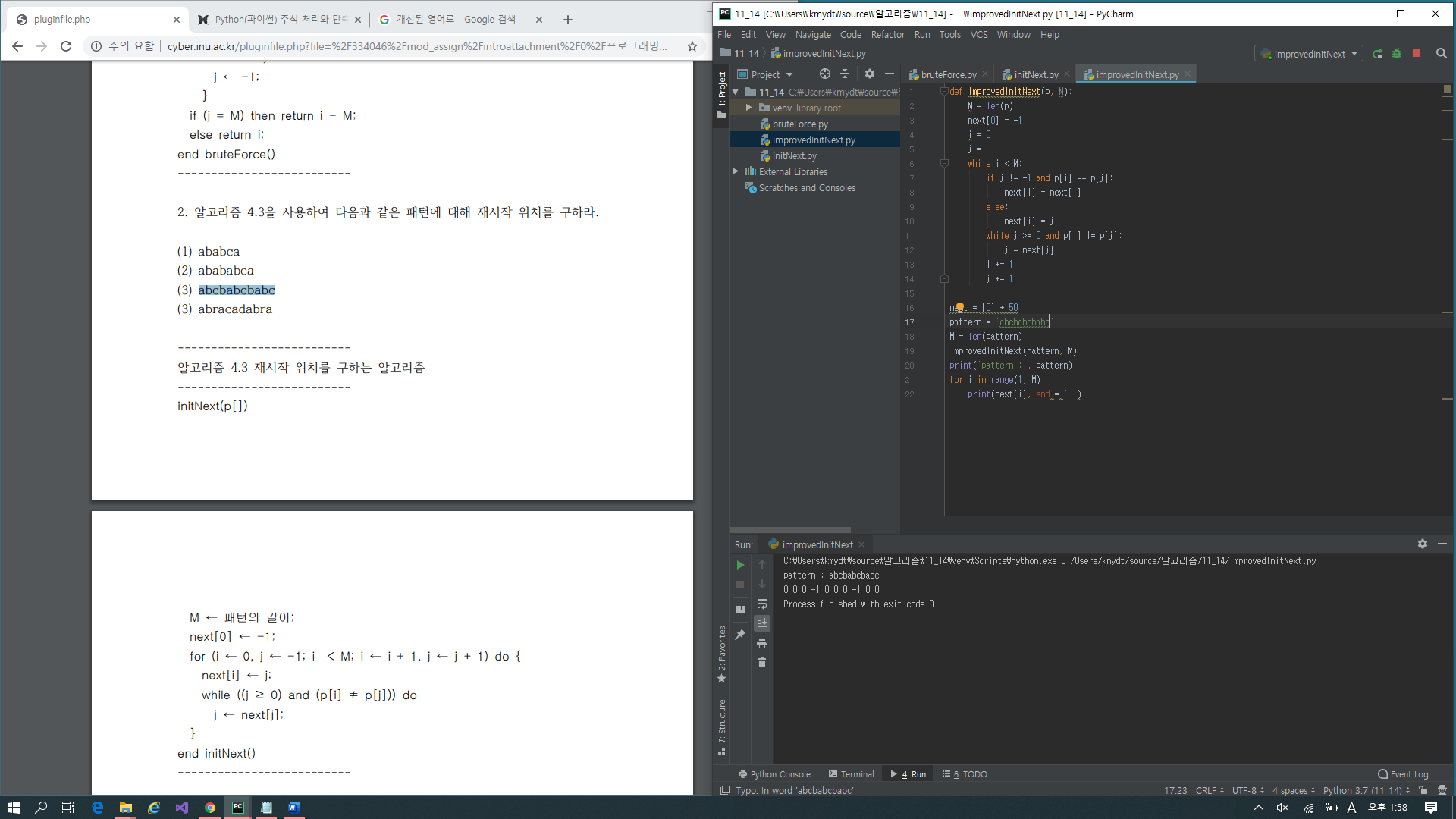
(1) ababca



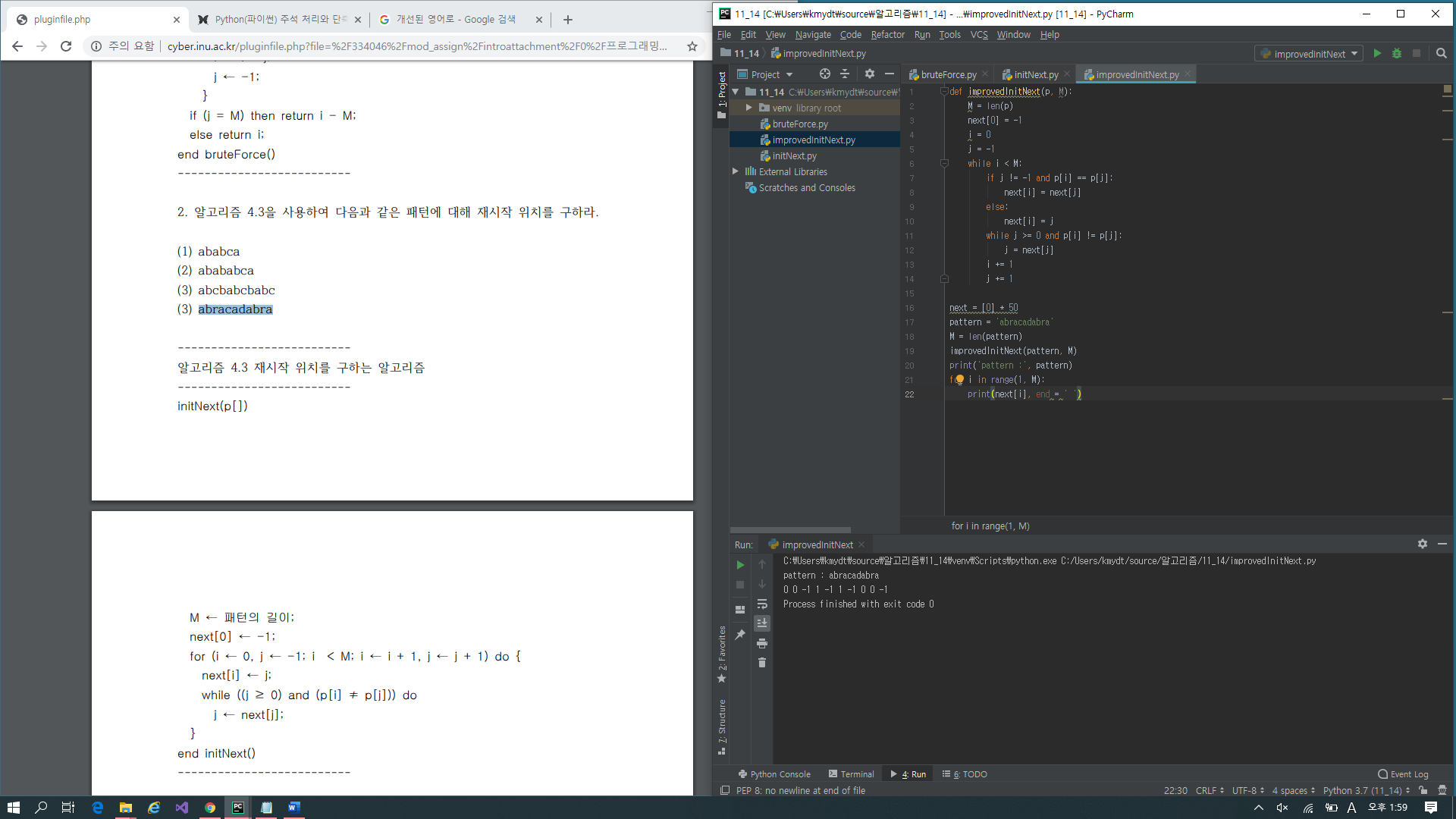
(2) abababca



(3) abcbabcbabc



(4) abracadabra

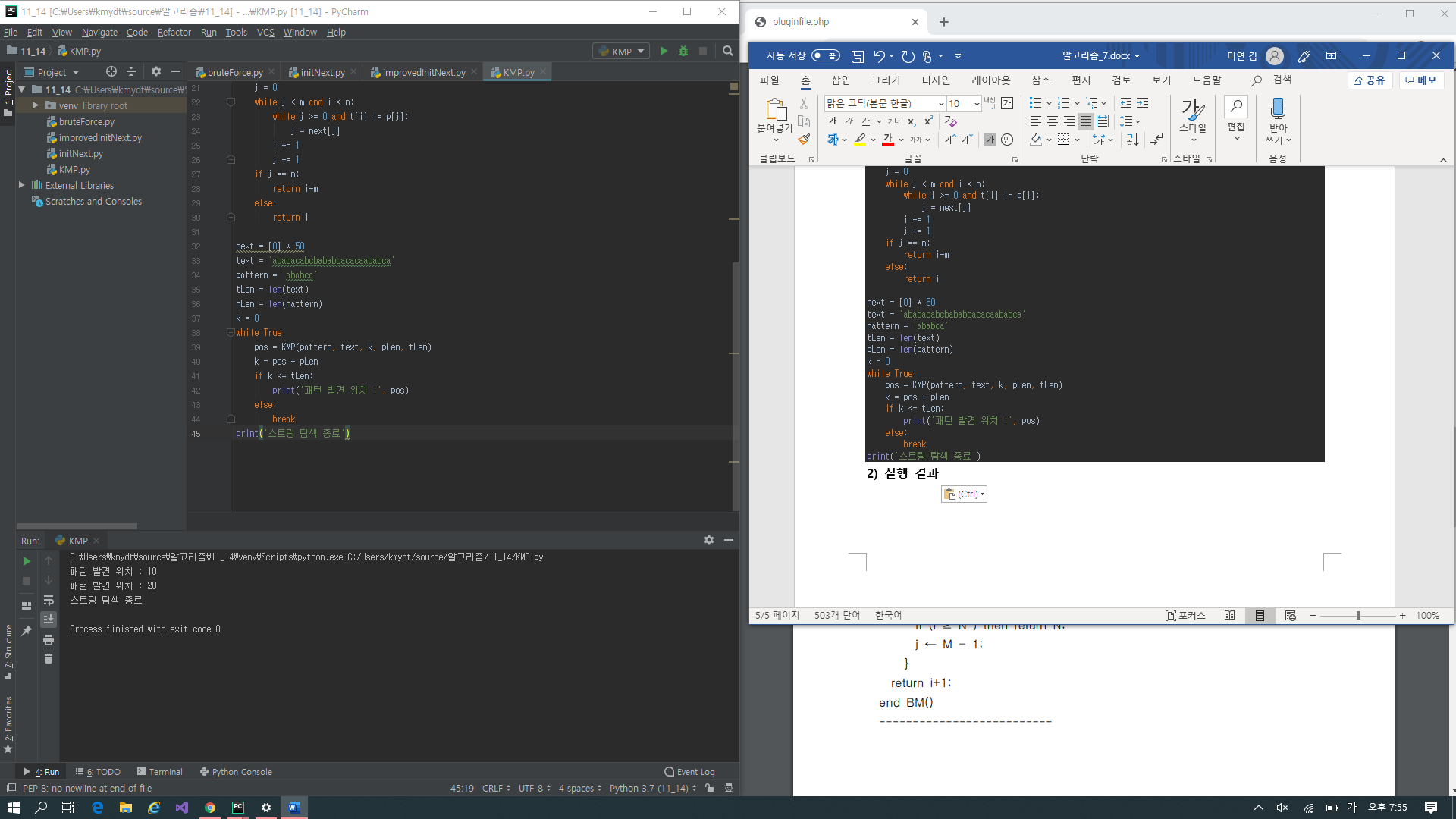


**4. KMP 알고리즘**

**1) 소스 코드**

def improvedInitNext(p, m):  
 m = len(p)  
 next[0] = -1  
 i = 0  
 j = -1  
 while i < m:  
 if j != -1 and p[i] == p[j]:  
 next[i] = next[j]  
 else:  
 next[i] = j  
 while j >= 0 and p[i] != p[j]:  
 j = next[j]  
 i += 1  
 j += 1  
  
def KMP(p, t, k, m, n):  
 m = len(p)  
 n = len(t)  
 improvedInitNext(p, m)  
 i = k  
 j = 0  
 while j < m and i < n:  
 while j >= 0 and t[i] != p[j]:  
 j = next[j]  
 i += 1  
 j += 1  
 if j == m:  
 return i-m  
 else:  
 return i  
  
next = [0] \* 50  
text = 'ababacabcbababcacacaababca'  
pattern = 'ababca'  
tLen = len(text)  
pLen = len(pattern)  
k = 0  
while True:  
 pos = KMP(pattern, text, k, pLen, tLen)  
 k = pos + pLen  
 if k <= tLen:  
 print('패턴 발견 위치 :', pos)  
 else:  
 break  
print('스트링 탐색 종료')

**2) 실행 결과**

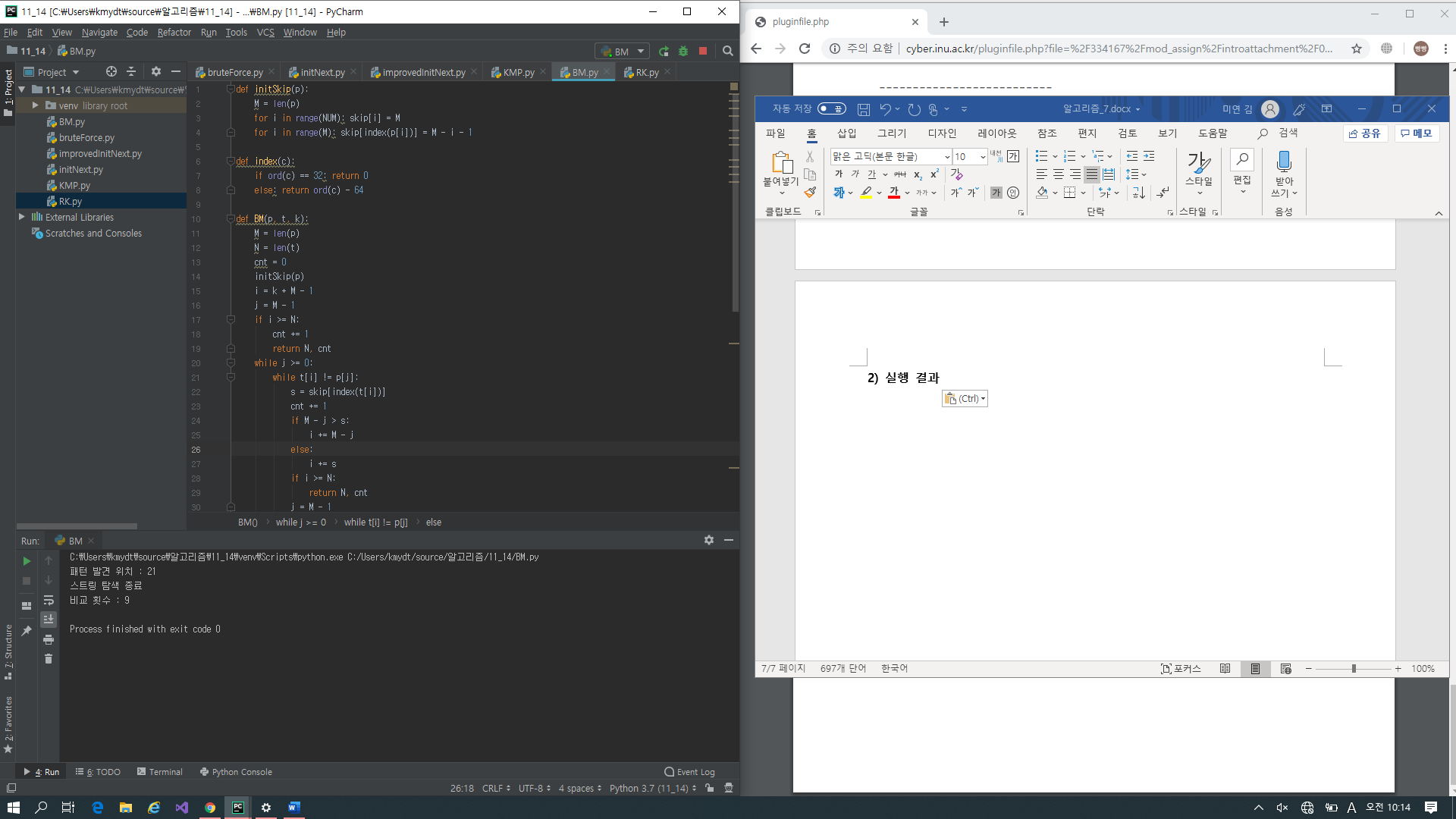


**5. 보이어-무어 알고리즘**

**1) 소스 코드**

def initSkip(p):  
 M = len(p)  
 for i in range(NUM): skip[i] = M  
 for i in range(M): skip[index(p[i])] = M - i - 1  
  
def index(c):  
 if ord(c) == 32: return 0  
 else: return ord(c) - 64  
  
def BM(p, t, k):  
 M = len(p)  
 N = len(t)  
 cnt = 0  
 initSkip(p)  
 i = k + M - 1  
 j = M - 1  
 if i >= N:  
 cnt += 1  
 return N, cnt  
 while j >= 0:  
 while t[i] != p[j]:  
 s = skip[index(t[i])]  
 cnt += 1  
 if M - j > s:  
 i += M - j  
 else:  
 i += s  
 if i >= N:  
 return N, cnt  
 j = M - 1  
 i -= 1  
 j -= 1  
 return i + 1, cnt  
  
NUM = 27  
skip = [0] \* NUM  
text = 'STRING STARTING CONSISTING'  
pattern = 'STING'  
M = len(pattern)  
N = len(text)  
K = 0  
cnt = 0  
while True:  
 ans = BM(pattern, text, K)  
 K = ans[0] + M  
 cnt += ans[1]  
 if K <= N:  
 print('패턴 발견 위치 :', ans[0])  
 else:  
 break  
print('스트링 탐색 종료')  
print('비교 횟수 :', cnt)

**2) 실행 결과**



**6. 라빈-카프 알고리즘**

**1) 소스 코드**

def index(c):  
 if ord(c) == 32:  
 return 0  
 else:  
 return ord(c) - 64  
  
def RK(p, t, k):  
 dM = 1  
 h1 = 0  
 h2 = 0  
 M = len(p)  
 N = len(t)  
 for i in range(1, M):  
 dM = (d \* dM) % q  
 for i in range(M):  
 h1 = (h1 \* d + index(p[i])) % q  
 if k == 0:  
 print('h1 :', h1)  
 i = k  
 j = 0  
 while i < N and j < M:  
 h2 = (h2 \* d + index(t[i])) % q  
 i += 1  
 j += 1  
 i = k  
 while h1 != h2:  
 if i + M >= N:  
 return N  
 h2 = (h2 + d \* q - index(t[i]) \* dM) % q  
 h2 = (h2 \* d + index(t[i + M])) % q  
 print('h2 :', h2)  
 if i > N - M:  
 return N  
 i += 1  
  
 return i  
  
q = 33554393  
d = 32  
text = 'STRING STARTING CONSISTING'  
pattern = 'STING'  
M = len(pattern)  
N = len(text)  
K = 0  
while True:  
 pos = RK(pattern, text, K)  
 K = pos + M  
 if K <= N:  
 print('패턴 발견 위치 :', pos)  
 else:  
 break  
print('스트링 탐색 종료')

**2) 실행 결과**

