# 11차 추가문제 답안

## 파이썬 기본: 변수입력

| 수강생 공지 사항   * 제출은 프로그래머스를 통해 해주시기 바랍니다([링크](https://campus.programmers.co.kr/app/courses/24550/curriculum)). * 파일명은 아래와 같은 형식으로 제출해주세요   + 교육생번호\_이름\_교과목\_문항\_N차시\_강의명.doc   ex) DR-11111\_홍길동\_파이썬\_연습문항\_1차시\_환경및기본.doc  ex) DR-11111\_홍길동\_파이썬\_추가문항\_1차시\_환경및기본.doc   * 마감 기한은 문제가 나간 당일 자정까지입니다. |
| --- |

### 1~20 사이의 숫자 중에서 무작위로 중복되지 않는 숫자 10개를 뽑아서 출력하는 프로그램을 작성하시오.

* 새로운 무작위 숫자를 생성했을 때 기존에 저장되어 있던 숫자들과 비교해서 중복되지 않는 숫자이면 새로 저장하고, 중복된다면 버리고 새로운 숫자를 생성.
* 10개의 중복되지 않는 숫자들이 생성될 때까지 반복. 즉,무한 반복하면서 10개가 채워지면 반복을 종료해야 함

**답**

| Example Code |
| --- |
| import random  numSet = set() # 빈 집합 생성    # 무작위로 10개의 중복되지 않는 숫자 생성  while True:        n = random.randint(1, 20)        numSet.add(n)        if len(numSet) == 10:              break    # 리스트로 변환 후 출력  numList = list(numSet)  for n in numList:        print(n, end=' ') |

### 각 학생이 듣는 과목을 집합으로 저장하고, 모든 학생이 듣는 과목, 한 명이라도 듣는 과목, 한 명씩만 듣는 과목을 구하는 프로그램을 작성하시오.

student\_courses = {

"학생1": {"수학", "과학", "영어"},

"학생2": {"수학", "음악", "미술"},

"학생3": {"수학", "과학", "음악"}}

**답**

| Example Code |
| --- |
| # 학생들이 듣는 과목 집합을 딕셔너리로 표현  student\_courses = {        "학생1": {"수학", "과학", "영어"},        "학생2": {"수학", "음악", "미술"},        "학생3": {"수학", "과학", "음악"}  }    # 모든 학생이 듣는 과목을 구하기 위해 첫 번째 학생의 과목 집합을 기준으로 초기화  all\_students\_courses = student\_courses["학생1"].copy()    # 모든 학생이 듣는 과목을 구하기  for student, courses in student\_courses.items():        all\_students\_courses = all\_students\_courses.intersection(courses)    # 한 명이라도 듣는 과목을 구하기 위해 모든 학생의 과목 집합을 합집합으로 구함  any\_student\_courses = set()  for courses in student\_courses.values():        any\_student\_courses = any\_student\_courses.union(courses)  # 한 명씩만 듣는 과목을 구하기 위해 과목별로 카운트  course\_count = {}  for courses in student\_courses.values():        for course in courses:              if course in course\_count:                    course\_count[course] += 1              else:                    course\_count[course] = 1  # count가 1인것만 출력  unique\_courses = {course for course, count in course\_count.items() if count == 1}    # 결과 출력  print(f"모든 학생이 듣는 과목: {all\_students\_courses}")  print(f"한 명이라도 듣는 과목: {any\_student\_courses}")  print(f"한 명씩만 듣는 과목: {unique\_courses}") |

### 다음 문장은 Robert Zaks라는 사람이 쓴 "From Outer Space"라는 SF소설에서 일부 발췌한 내용이다. 해당 내용은 저작권이 만료된 자료로 전문은 구텐베르크 프로젝트 사이트 ([링크](https://www.gutenberg.org/ebooks/68860))에서 구할 수 있다. 해당 내용을 단어 단위로 잘라서 각 단어들이 몇회 나타나는지 기록하고, 가장 많이 반복된 단어와 두 번째로 많이 반복된 단어를 화면에 출력하는 프로그램을 작성한다.

The first they had heard of the strangers from outer space was when the new ultra short-wave frequencies were used. Professor Kennicot of Palmira University was the first to find how to generate and control them. He tried to transform the wavelengths upward to a range either auditory or visual but for some reason power was lost in the process.

Apparently he gave them a sufficient jolt with extra voltage, however, because they were picked up by the strangers in outer space as a signal. The heaviside layer did not stop these wavelengths.

Professor Kennicot was startled one day when he heard, or thought he heard, a soundless voice in his mind.

이 내용을 단어로 잘라서 각 단어들이 몇 회 나타나는지 기록하고, 가장 많이 반복된 단어와 두 번째로 많이 반복된 단어를 화면에 출력하는 프로그램을 작성한다.

- 특수 문자는 무시하기로 한다. 문자열의 split() 함수를 사용하면 마침표는 무시하 지만, 콤마(',') 같은 특수 문자는 단어의 일부로 포함될 수 있다. 따라서 단어를 자를 때 콤마를 제거해야 한다.

- 많이 나타난 빈도수가 중복되는 단어들이 있을 수 있다. 이 경우에는 그 중 한 개를 출력한다.

**답**

| Example Code |
| --- |
| words = s.split()  wordsCount = {}  for word in words:        newWord = word.strip(' \t\n,') # 콤마도 제거        if newWord in wordsCount:              wordsCount[newWord] += 1        else:              wordsCount[newWord] = 1  # 출력  #for key in wordsCount.keys():  # print(key, ":", wordsCount[key])    # 가장 빈도수가 높은 것과 두 번째로 높은 것 찾기  firstCount = 0 # 가장 많이 나타난 단어의 빈도수  secondCount = 0 # 두 번째로 많이 나타난 단어의 빈도수  firstWord = ""  secondWord = ""  for key in wordsCount.keys():        if wordsCount[key] >= firstCount: # 더 많이 나타난 단어가 있다면              firstCount = wordsCount[key]              firstWord = key        if wordsCount[key] < firstCount and wordsCount[key] > secondCount:              secondCount = wordsCount[key]              secondWord = key  print(f"가장 많이 나타난 단어: {firstWord}, 횟수: {firstCount}")  print(f"두 번째로 많이 나타난 단어: {secondWord}, 횟수: {secondCount}") |

### 13번 문제의 문자열에서 가장 많이 나타난 알파벳 문자와 나타난 횟수를 화면에 출력하는 프로그램을 작성한다.

- 특수 문자는 무시한다.

- 공백을 포함해서 특수 문자는 무시하기로 한다.

- 많이 나타난 빈도수가 중복되는 단어들이 있을 수 있다. 이 경우에는 그 중 한 개를 출력한다.

**답**

| Example Code |
| --- |
| s = """  """  d = {}  for ch in s:        if ch.isalpha():              if ch in d:                    d[ch] += 1              else:                    d[ch] = 1  count = 0  character = ''  for key in d.keys():        if d[key] >= count: # 더 많이 나타난 단어가 있다면              count = d[key]              character = key  print(f"가장 많이 나타난 문자: {character}, 횟수: {count}") |

### [13](#_heading=h.3znysh7)번 문제의 문자열에서 특수 문자까지 포함해서 두 번째로 많이 나타난 글자와 횟수를 화면에 출력하는 프로그램을 작성한다.

- 문자열에서 보이는 특수 문자는 마침표('.')와 콤마(',')가 있다. 여기서는 두 가지만 따로 세기로 한다.

**답**

| Example Code |
| --- |
| s = """  # 정답13 과 동일한 지문  """  d = {}  for ch in s:        if ch.isalpha() or ch == '.' or ch == ',':              if ch in d:                    d[ch] += 1              else:                    d[ch] = 1    firstCount = 0  firstCharacter = ''  secondCount = 0  secondCharacter = ''  for key in d.keys():        if d[key] >= firstCount: # 더 많이 나타난 글자가 있다면              firstCount = d[key]        if d[key] < firstCount and d[key] > secondCount:              secondCount = d[key]              secondCharacter = key  print(f"두 번째로 많이 나타난 문자: {secondCharacter}, 횟수: {secondCount}") |

### 알파벳 중에서 13번 문제의 문자열에 나타나지 않는 글자가 어떤 것이 있는지 화면에 출력하는 프로그램을 작성한다(대문자와 소문자를 따로 구별하지 않는다). 단 집합을 이용해서 문자가 사용되었는지 확인한다. 만약 모든 알파벳이 나타난다 면 "나타나지 않은 알파벳은 없습니다"를 출력한다.

**답**

| Example Code |
| --- |
| # 사용된 알파벳을 담을 집합 생성  chSet = set()  # 문자열에 있는 모든 알파벳을 집합에 추가(중복된 글자들은 집합에서 걸러짐)    for ch in s:        if ch.isalpha():              chSet.add(ch.lower())  # 알파벳 중에 집합에 없는 글자만 출력  for ch in "abcdefghijklmnopqrstuvwxyz":        if not ch in chSet:              print(ch) |

### 13번 문제의 문자열에서 각 알파벳 별로 시작하는 단어들을 모두 출력하는 프로그램을 작성한다. 단. 대문자와 소문자는 구별하지 않고, 중복된 단어는 한 번만 출력한다. 예를 들어, t로 시작하는 단어들을 출력하는 것은 다음과 같다.

t : The, they, them, tried, to, transform, these, thought,

**답**

| Example Code |
| --- |
| # 알고리즘:  # 알파벳을 키로 딕셔너리를 생성하고 값을 빈 집합으로 채울 것  # 문자열을 단어로 분리해서 리스트로 구성  # 리스트에 있는 각 단어에서(특수 문자 제거 후) 첫 번째 글자를 확인하고, 해당 알파벳의 집합에 추가(소문자로 변환 후 추가. 중복되는 단어들은 집합에서 걸러냄)  # 딕셔너리에서 집합의 크기가 1이상인 것들만 화면에 출력    s = """  # 정답13 과 동일한 지문  """  d = {}  alphabet = "abcdefghijklmnopqrstuvwxyz"  for ch in alphabet:        d[ch] = set()  words = s.split()  wordsCount = {}  for word in words:        newWord = word.strip(' \t\n,') # 콤마도 제거        if newWord[0].isalpha():              d[newWord[0].lower()].add(newWord.lower())  for ch in d.keys():        if len(d[ch]) > 0:              print(ch + " : ", end = '')              lst = list(d[ch])              for word in lst:                    print(word + ", ", end = '')              print() |

### 서울시에 살고 있는 구별 인구 중 10개를 가나다순으로 뽑아서 문자열 (guPopulation)을 만들었다. 이 문자열을 이용해서 구:인구 형태의 딕셔너리를 구성하라.

<요구사항>

- 사용자는 정수를 입력할 것이라고 가정

- 만약 사용자가 입력한 정수값(num)보다 작은 구가 없으면 “인구가 num보다 작은 구가 없습니다＂를 출력

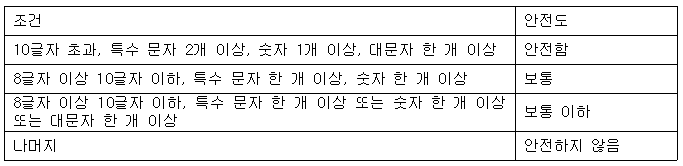
- 구:인구 형태로 출력

**답**

| Example Code |
| --- |
| guPopulation = "강남구,233363,강동구,199088,강북구,144410," + \        "강서구,267442,관악구,273736,광진구,166638,구로구,180027," + \        "금천구,114402,노원구,217272,도봉구,138120"    # 문자열에서 리스트 구성. 리스트의 요소는 구, 인구,  # 구 인구, ...  lst = guPopulation.split(',')  # 리스트로부터 딕셔너리 생성  d = {}  for i in range(0, len(lst), 2):        d[lst[i]] = int(lst[i + 1])    # 인구 입력 받기  num = int(input("비교할 인구를 입력하세요: "))    # 딕셔너리에 있는 구/인구 정보와 인구 비교  count = 0  for gu, population in d.items():        if population < num:              print(f"{gu}:{population}")              count += 1 # 출력된 구가 있음  # count == 0이면 출력된 구가 없음  if count == 0:        print(f"인구가 {num}보다 작은 구가 없습니다") |

### 영문 알파벳, 특수문자, 숫자로만 구성된 비밀 번호 문자열을 입력 받고, 아래 표를 기준으로 어느 정도안전한가를 출력하는 프로그램을 작성한다.

표) 비밀번호 안전도 점검 기준(기본적으로 조건은 and임)



특수 문자는 키보드로 입력할 수 있는 문자들을 나타내는 것으로,"~`!\@#$%^&:',./?><|“로 가정한다.

**답**

| Example Code |
| --- |
| specialChars = "~`!\@#$%^&\*-\_+=;:',./?><|"  pw = input("비밀 번호로 사용할 문자열을 입력하세요: ")  d = {"upper":0, "special":0, "num":0}  for ch in pw:        if ch.isupper():              d["upper"] += 1        if ch in specialChars:              d["special"] += 1        if ch.isdigit():              d["num"] += 1  if len(pw) > 10 and d["special"] >= 2 and d["num"] >= 1 and d["upper"] >= 1:        print("안전함")  elif len(pw) >= 8 and len(pw) <= 10 and d["special"] >= 1 and d["num"] >= 1:        print("보통")  elif len(pw) >= 8 and len(pw) <= 10 and (d["special"] >= 1 or d["num"] >= 1 or  d["upper"] > 1):        print("보통 이하")  else:        print("안전하지 않음") |

### 아래 표는 년도별로 고등학교 학급당 학생 수를 보인다(출처: 2021교육 기본 통계 결과 발표 자료) 표를 자료구조로 저장하고, 다음 문제에 답하는 프로그램을 작성한다.

1. 전해에 비해 가장 급격하게 학생 수가 줄어든 해는 언제인가?

2. 학급당 학생 수가 30명 미만으로 떨어진 해는 언제인가?

3. 2010년부터 2021년 사이에 평균적으로 학급당 학생 수는 어느 정도 감소했는가?

표) 년도별 고등학교 학급당 학생 수

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

**답**

| Example Code |
| --- |
| d = {2010:33.7, 2011:33.1, 2012:32.5, 2013:31.9, 2014:30.9, 2015:30.0, 2016:29.3,       2017:28.2, 2018:26.2, 2019:24.5, 2020:23.4, 2021:23}  # 1 전 해에 비해 가장 급격하게 학생 수가 줄어든 해 찾기  print("1)번 문제")  max = 0  year = 0  for i in range(2011, 2022):        if d[i - 1] - d[i] >= max:              year = i              max = d[i - 1] - d[i]  print(f"{year}에 {max:.1f}명 감소한 것이 최근 가장 급격하게 줄어든 경우입니다.")    # 2 학급당 학생 수가 30명 미만으로 떨어진 해는 언제인가  print("2)번 문제")  for i in range(2010, 2022):        if d[i] < 30:              print(f"학생 수가 30명 미만으로 떨어진 첫 해는 {i}년입니다.")              break    # 3 2010년부터 2021년까지 평균적으로 감소한 학생 수  print("3)번 문제")  sum = 0  for i in range(2011, 2022):        sum += (d[i - 1] - d[i])  print(f"2010년부터 2021년까지 평균적으로 감소한 학생 수는 {sum / 11:.2f}명이다") |

| write your answer |
| --- |

### 주어진 텍스트 파일에서 각 단어의 빈도수를 계산하고, 빈도수가 높은 단어 순으로 정렬하여 출력하는 프로그램을 작성하세요.

**<**제약 사항**>**

- 대소문자를 구분하지 않습니다. (예: "Python"과 "python"은 같은 단어로 간주)

- 구두점은 무시합니다. (예: "hello,"와 "hello"는 같은 단어로 간주)

- 결과는 빈도수가 높은 단어부터 내림차순으로 정렬하여 출력합니다.

<입력 파일 예시 (input.txt)>

Hello, world! This is a test. Hello again, world. Test the Python code.

<출력 예시>

hello: 2

world: 2

test: 2

...

<힌트>

- 파일 읽기: with open('input.txt', 'r') as file:

- 문자열 처리: str.lower(), str.translate()

- 단어 분리: str.split()

- collections.Counter를 사용하여 단어 빈도수 계산

- sorted()를 사용하여 정렬

**답**

| Example Code |
| --- |
| import string  from collections import Counter  # 파일에서 텍스트 읽기  with open('input.txt', 'r') as file:        text = file.read()  # 대소문자 무시, 구두점 제거  text = text.lower().translate(str.maketrans('', '', string.punctuation))  # 단어로 분리  words = text.split()  # 단어 빈도수 계산  word\_counts = Counter(words)  # 빈도수 기준으로 정렬하여 출력  for word, count in word\_counts.most\_common():        print(f"{word}: {count}") |

| write your answer |
| --- |