Q1. **파이썬에서는 리스트 형태의 데이터를 자주 사용합니다. 그래서 리스트 를 잘 다루는 것이 중요한데, 다음으로 주어진 리스트 데이터를 다뤄봅시다.**

풀이1) 파라미터로 받은 num_list에서 **반복문**과 **조건문**을 활용해, 홀수인 원소들만을 뽑아 새로운 리스트인 answer에 추가한다.

- **반복문** : for idx in num_list
- **조건문** : idx%2==0 (홀수는 2로 나누었을때 나머지가 0인 원소)
- **소스코드**

```
>>> num_list = [1, 5, 7, 15, 16, 22, 28, 29]
>>> def get_odd_num(num_list):
...     answer =  list()
...     for idx in num_list:
...             if idx%2!=0:
...                     answer.append(idx)
...     return answer
...
>>> print(get_odd_num(num_list))
[1, 5, 7, 15, 29]
>>>
```

풀이2) 삼항 연산자를 응용해서 앞의 1번 풀이의 반복문과 조건문을 한줄로 표현한 것이다.

- [**num** for **num** in **num_list** if **num % 2 == 1**] : 파라미터로 받은 num_list의 원소들 중에서 홀수인 num만 반환한다.
- **소스코드**

```
>>> num_list = [1, 5, 7, 15, 16, 22, 28, 29]
>>>
>>> def get_odd_num(num_list):
...     return [num for num in num_list if num % 2 == 1]
...
>>> print(get_odd_num(num_list))
[1, 5, 7, 15, 29]
>>>
```

Q2. **데이터 처리를 위해서 문자열을 입력받았습니다. 그런데, 문자열을 받았 더니 단어 단위로 거꾸로 입력되었습니다. 이를 다시 원래대로 출력하는 함수를 작성해보세요.**

풀이1) sentence를 **'(띄어쓰기)' 단위로 나누어** reversed_sentence_list를 만들고 **reversed_sentence_list를 뒤집는다**. 또한, 뒤집은 **reversed_sentence_list의 각 요소들을 '(띄어쓰기)' 단위로 합쳐 문자열**을 만들어 반환한다.

- **'(띄어쓰기)' 단위 나누기** : sentence.split()
- **reversed_sentence_list 뒤집기 :** reversed_sentence_list.reverse()
- **reversed_sentence_list '(띄어쓰기)' 단위로 합치기** : ' '.join(reversed_sentence_list)
- **소스코드** :

```
>>> sentence = "way a is there will a is there Where"
>>>
>>> def reverse_sentence(sentence):
...     reversed_sentence_list = sentence.split(' ')
...     reversed_sentence_list.reverse()
...     return ' '.join(reversed_sentence_list)
...
>>> print(reverse_sentence(sentence))
Where there is a will there is a way
>>>
```

Q3. 이번 학기의 중간고사, 기말고사 점수가 발표되었습니다. 각 학생들의 점 수가 튜플 형태로 저장되어 있고, 이를 포함한 리스트가 있습니다. 이를 이용 해 각 학생들의 평균 점수를 출력하는 함수를 제작하세요.

풀이1) score 안에 있는 튜플의 수는 곧 **학생의 수이므로 이를 student_number로 저장**한다. 학생의 수만큼 for문을 돌면서 **순서대로 학생의 평균**을 구하고 출력한다.

- **학생의 수** : len(score), len()은 주어진 리스트의 길이를 return해준다.
- **순서대로 학생의 번호를 메기기 (n번 학생)** : {idx+1}번, for문은 0번 인덱스부터 접근하므로 1을 더해준다.
- **학생의 평균 구하기** : sum(score[idx]/2), 해당 학생의 중간고사 점수와 기말고사 점수가 들어있는 튜플의 합(sum)을 구하고 2로 나눈다.
- **소스코드**

```
>>> score = [(100, 100), (95, 90), (55, 60), (75, 80), (70, 70)]
>>> student_number = len(score)
>>>
>>> def get_avg(score):
...     for idx in range(student_number):
...         print(f'{idx+1} 번 , 평 균  : {sum(score[idx])/2}')
...
>>> get_avg(score)
1 번 , 평 균  : 100.0
2 번 , 평 균  : 92.5
3 번 , 평 균  : 57.5
4 번 , 평 균  : 77.5
5 번 , 평 균  : 70.0
>>>
```

풀이2) enumerate를 활용해 score로 부터 학생들의 중간고사/기말고사 점수(student)와 순서(i)를 함께 받아와서 순서대로 평균을 구한다.

- **학생들의 중간고사/기말고사 점수(student)와 순서(i)를 함께 받아오기** : enumerate는 주어진 리스트에서 인덱스와 원소를 동시에 접근하게 해주는 메소드입니다.
- **소스코드**

```
>>> score = [(100, 100), (95, 90), (55, 60), (75, 90), (70, 70)]
>>> def get_avg(score):
...     for i, student in enumerate(score):
...             print(f'{i + 1} 번 , 평 균  : {sum(student) / len(student)}')
...
>>> get_avg(score)
1 번 , 평 균  : 100.0
2 번 , 평 균  : 92.5
3 번 , 평 균  : 57.5
4 번 , 평 균  : 82.5
5 번 , 평 균  : 70.0
>>>
```

Q4. 두개의 납품처에서 각각 과일과 야채들이 납품되었습니다. 이를 각각 물 품의 갯수를 나타내는 2개의 딕셔너리로 정리했습니다. 물품을 정리하기 위해서 2 개의 딕셔너리 객체를 합쳐 출력하는 함수를 제작하세요.

풀이1) 파리미터로 받은 두개의 딕셔너리를 첫번째 딕셔너리인 dict_first로 합치려고 한다. **dict_second의 key, value값을 for문을 통해 원소별로 받아온다**. **만일, 해당 원소의 key값이 dict_first에 존재한다면, value를 합친다**. **만일, dict_first에 존재하지 않다면, dict_first에 원소를 추가한다.**

- **dict_second의 key, value값을 for문을 통해 원소별로 받아오기 :** for key, value in dict_second.items()
- **key값이 dict_first에 존재한다면, value를 합친다 / 존재하지 않다면, dict_first에 원소를 추가 :**

```
if key in dict_first:                    # 존재한다면!
        dict_first[key] += value         # 두 값을 합친다!
else:                                    # 존재하지 않다면!
        dict_first[key] = value          # 원소를 추가한다!
```

- **소스코드**

```
>>> dict_first = {'사과': 30, '배': 15, '감': 10, '포도': 10}
>>> dict_second = {'사과': 5, '감': 25, '배': 15, '귤': 25}
>>>
>>> def merge_dict(dict_first, dict_second):
...     for key, value in dict_second.items():
...         if key in dict_first:
...             dict_first[key] += value
...         else:
...             dict_first[key] = value
...     print(dict_first)
...
>>> merge_dict(dict_first, dict_second)
{'사과': 35, '배': 30, '감': 35, '포도': 10, '귤': 25}
>>>
```

풀이2) collections.defaultdict(int)를 선언해주면 key값이 존재하지 않을 경우 0으로 인식한다. 즉, 이를 선언해주고 해당 딕셔너리에 파라미터로 받은 두개의 딕셔너리의 중복을 확인하지 않고 단지 모두 추가만 해준다면, 중복되지 않으면 새로운 원소를 추가해주고, 중복이 된다면 value들을 합쳐준다.

- **소스코드**

```
>>> import collections
>>>
>>> dict_first = {'사과': 30, '배': 15, '감': 10, '포도': 10}
>>> dict_second = {'사과': 5, '감': 25, '배': 15, '귤': 25}
>>>
>>> def merge_dict(dict_first, dict_second):
...     new_dict = collections.defaultdict(int)
...
...     for key, value in dict_first.items():
...             new_dict[key] += value
...     for key, value in dict_second.items():
...             new_dict[key] += value
...     return new_dict
...
>>> print(dict(merge_dict(dict_first, dict_second)))
{'사과': 35, '배': 30, '감': 35, '포도': 10, '귤': 25}
>>>
```

Q5. 단어들을 입력받았는데, 자꾸만 숫자들이 섞여들어가는 문제가 있습니다. 이를 처리하기 위해서 함수에 string을 전달 받은 뒤, string 안에서의 숫자를 제거한 후 string만 남은 리스트를 출력하세요.

풀이1) 파라미터로 받은 input은 string데이터 타입으로 시퀀스형 특징을 갖고 있다. 그러므로 리스트처럼 각 알파벳 또는 숫자를 for문을 통해 인덱스로 접근할 수 있다. 새로운 리스트를 먼저 선언해주고, 조건문을 이용하여 **만일 해당 인덱스가 숫자일 경우** '(띄어쓰기)' 를 새로운 리스트에 추가하고 알파벳일 경우 해당 알파벳을 새로운 리스트에 추가주는 과정을 반복한다.

이 과정이 끝나면 새로운 리스트의 모양은 다음과 같다.

['c', 'a', 't', ' ', ' ', 'd', 'o', 'g', ' ', ' ', 'c', 'o', 'w', ' ']

해당 리스트를 원하는 모양으로 합치기 위해서는 **공백을 기준으로 리스트를 잘라서 각 덩어리 별로 알파벳을 하나의 문자열로 합쳐주면 된다**. 그 결과 다음과 같다.

['cat', 'dog', 'cow']

- **만일 해당 인덱스가 숫자일 경우 :** id idx.isdigit() == True, isdigit() 메소드는 주어진 변수값의 타입이 숫자인지 아닌지를 판단해 True/False를 반환해준다.
- **공백을 기준으로 리스트를 잘라서 각 덩어리 별로 알파벳을 하나의 문자열로 합쳐주기 :** ''.join(answer).split(), ''.join()메소드로 리스트를 합치되, split()으로 공백을 구분하고 리스트로 반환해준다.
- **소스코드**

```
>>> inputs = "cat32dog16cow5"
>>> answer = list()
>>>
>>> def find_string(inputs):
...     for idx in inputs:
...         if idx.isdigit() == True:
...             answer.append(' ')
...             continue
...         else:
...             answer.append(idx)
...     return ''.join(answer).split()
...
>>> string_list = find_string(inputs)
>>> print(string_list)
['cat', 'dog', 'cow']
>>> _
```