```
# 7311 Coding Assignment 1

# write code for the following functions.
# note: these functions use the non-binding type hints feature new in Python 3.5
# It's good to get in the habit of writing your code this way

# All these functions return None so that the code compiles.
# Your job is to write code that returns a correct implementation of the function.

def count_vowels(in_string: str) -> int:
    '''
    do: return the number of vowels (a,e,i,o,u)  in the string
    example: count_vowels('xyzaeiou') -> 5
    :param in_string:  a text string that may contain some number of vowels (a,e,i,o,u)
    :return: the number of vowels in the string
    '''
    vowels = ('a', 'e', 'i', 'o', 'u')
    vowel_count = 0
    for char in in_string:
        if char in vowels:
            vowel_count += 1
    return vowel_count


def get_max_value (x:int, y:int, z:int) -> int:
    '''
    do: return the largest of values passed as parameters
    example:  get_max_value(3,4,7) -> 7
    :param x:   some integer
    :param y:   some integer
    :param z:   some integer
    :return:     the largest of the integers
    '''
    return max([x, y, z])

def get_list_multiples(max_val: int, divisor: int) -> list:
    '''
    do: return a list of all the numbers from 0..max_val that are evenly divisible by divisor
    example:  get_list_multiples(20, 3) -> [0,3,6,9,12,15,18]
    :param max_val:  largest number to test
    :param divisor:  divisor
    :return: list of numbers evenly divisble by divisor
    '''
    if divisor < 1:
```

```
        return None
    multiples = []
    for num in range(max_val + 1):
        if num % divisor == 0:
            multiples.append(num)
    return multiples

def is_prime(n: int) -> bool:
    '''
    do: return True or False depending whether n is prime
    example: is_prime(23) -> True
    example: is_prime(20) -> False
    :param n:  number to test for primality
    :return:   boolean
    '''
    if n < 2:
        return False
    for num in range(2, n):
        if n % num == 0:
            return False
    return True

def get_sum (s : str) -> int:
    '''
    do: extract the characters from the string, convert to int and add
    example: get_sum("2345") -> 14
    :param s:   string with all integers
    :return:    sum of the characters when converted to integers
    '''
    summation = 0
    for char in s:
        summation += int(char)
    return summation

def is_sorted (mylist: list) -> bool:
    '''
    do: test whether the values in the list are sorted in ascending order
    example: is_sorted( [2,2,3,4,4,9,12] ) -> True
    example: is_sorted( [2,3,4,44, 9,12] ) -> False
    note: it is OK if two numbers repeat right after one another
    :param mylist:  list of integers
    :return:  boolean
    '''
    last_num = mylist[0]
```

```
    for num in mylist:
        if num < last_num:
            return False
        last_num = num
    return True

def is_sorted_and_unique(mylist : list) -> bool:
    '''
    do: test whether the values in the list are unique and sorted in ascending order
    example: is_sorted( [2,2,3,4,4,9,12] ) -> False
    example: is_sorted( [2,3,4,44, 9,12] ) -> False
    example: is_sorted( [2,3,4,9,12] ) -> True
    note: it is OK if two numbers repeat right after one another
    :param mylist:  list of integers
    :return:  boolean
    '''
    last_num = mylist[0]
    idx = 1
    while idx < len(mylist):
        num = mylist[idx]
        if num <= last_num:
            return False
        last_num = num
        idx += 1
    return True


def remove_dups_and_sort(mylist : list) -> list:
    '''
    do: remove duplicates and sort the result
    example:  remove_dups_and_sort([9,2,9,12,5] ) -> [2,5,9,12]
    :param mylist: list of integers
    :return: a list with dups removed and sorted
    '''
    return sorted(set(mylist))

def intersection (mylist1: list, mylist2: list) -> list:
    '''
    do: return a sorted list of the values common to both lit1 and list2
    example: intersection( [2,3,4,5,6], [5,6,9,2] ) -> [2,5,6]
    :param mylist1:  list of integers
    :param mylist2:   list of integers
    :return:  sorted list of integers shared by both lists
    '''
```

```
    set1 = set(mylist1)
    set2 = set(mylist2)
    common_values = set1.intersection(set2)
    return sorted(common_values)

def main():
    # these print statements use handy format strings. the code inside {..} is executed

    print ("Name: Charles Bryan")

    print ( f"count_vowels('ae9qaniou4'): {count_vowels('ae9qaniou4')}" )
    print ( f"get_max_value (2,8,4): {get_max_value (2,8,4)}" )

    print (f"get_list_multiples(30,4) : {get_list_multiples(30,4)}  " )
    print (f"is_prime(44) :  {is_prime(44)}" )
    print (f"get_sum('456') : {get_sum('456')} " )

    print (f"is_sorted([3,2,5,7]) : {is_sorted([3,2,5,7])}" )
    print (f"is_sorted([3,5,5,7,9]) : {is_sorted([3,5,5,7,9])}")
    print (f"is_sorted_and_unique([3,5,5,7,9]) : {is_sorted_and_unique([3, 5, 5, 7, 9])}")
    print (f"remove_dups_and_sort([3,4,5,4,7]) : {remove_dups_and_sort([3,4,5,4,7])} " )
    print (f"intersection([2,3,12,6,11], [12,3,11]) : {intersection([2,3,12,6,11], [12,3,11])}")


# The following statement means execute the main function when this code is executed directly
# If this code is imported as a module in some other python code, main will not be executed
if __name__ == '__main__':
    main()

[Running] python -u "/Users/charlesbryan/Desktop/CS_7311/7311.a2.coding1.py"
Name: Charles Bryan
count_vowels('ae9qaniou4'): 6
get_max_value (2,8,4): 8
get_list_multiples(30,4) : [0, 4, 8, 12, 16, 20, 24, 28]
is_prime(44) :  False
get_sum('456') : 15
is_sorted([3,2,5,7]) : False
is_sorted([3,5,5,7,9]) : True
is_sorted_and_unique([3,5,5,7,9]) : False
remove_dups_and_sort([3,4,5,4,7]) : [3, 4, 5, 7]
intersection([2,3,12,6,11], [12,3,11]) : [3, 11, 12]

[Done] exited with code=0 in 0.063 seconds
```