def show_output():
    number = 1
    count = 10
    value = 4

    while count > 4:
        count = count - 2
        print(str(number) + ":", count, value)
        value += count
        number += 1

    print()
    print(str(number) + ":", count, value)

def main():
    show_output()

main()
Complete the function

- For an integer, a divisor is a number which divides exactly into the integer (a factor of the integer), e.g., the divisors of 6 are 1, 2, 3, 6.

Complete the get_all_divisors() function. Note that 1 and the number itself are divisors (as they divide into the number exactly).

```python
def get_divisor_string(number):

def main():
    print(get_divisor_string(24))
    print(get_divisor_string(25))
    print(get_divisor_string(5628))

main()
```

1 2 3 4 6 8 12 24
1 5 25
1 2 3 4 6 7 12 14 21 28 42 67 84 134 201 268 402 469 804 938 1407 1876 2814 5628
Complete the function

- The `get_dice_throws_result()` function throws a number of dice (given by `num_dice_throws`) and counts how often the dice value, `dice_to_check` occurs. Complete the function.

```python
def get_dice_throws_result(num_dice_throws, dice_to_check):

def main():
    print(get_dice_throws_result(30000, 6), "sixes thrown (out of 30000 throws")
    print(get_dice_throws_result(6, 6), "sixes thrown (out of 6 throws")
    print(get_dice_throws_result(600000, 6), "sixes thrown (out of 600000 throws")
main()
```
A perfect number is an integer that is equal to the sum of its divisors (excluding the number itself), e.g., $28 = 1 + 2 + 4 + 7 + 14$. Complete the following two functions. The check_perfection() function checks for perfection and prints either ' # is a perfect number' or '# is NOT a perfect number'.

```python
def get_divisor_sum(number):
    pass

def check_perfection(number):
    pass

def main():
    check_perfection(28)
    check_perfection(54)
    check_perfection(496)

main()
```

28 is a perfect number
54 is NOT a perfect number
496 is a perfect number
The get_legal_number() function repeatedly prompts the user for a number until the user number is within (both inclusive) the two numbers passed as parameters. The function returns the user number. Complete the function:

```python
def get_legal_user_num(lower, upper):
    prompt = "Enter a number (" + str(lower) + "-" + str(upper) + ")": "

def main():
    print(get_legal_user_num(0, 6))
    print(get_legal_user_num(10, 20))
    print(get_legal_user_num(1, 2))

main()
```

Enter a number (0-6): -1
Enter a number (0-6): 8
Enter a number (0-6): 6
Enter a number (10-20): 13
Enter a number (1-2): 3
Enter a number (1-2): 0
Enter a number (1-2): 2

The following function keeps prompting the user to guess a hidden number until the user correctly guesses the number. At each guess the function lets the user know if the guess is too high or too low. The function also keeps track of (and prints) the number of guesses. Complete the user_number_guess() function:

```python
def user_number_guess(computer_num):
    prompt = "Enter your guess (1 - 99): "

    user_input = int(input(prompt))
    if user_input < computer_num:
        print("Too low")
        user_number_guess(computer_num)
    elif user_input > computer_num:
        print("Too high")
        user_number_guess(computer_num)
    else:
        print("Correct! Number of guesses: " + str(guess_count))

def main():
    user_number_guess(random.randint(1, 100))

main()
```

Example output:
```
Enter your guess (1 - 99): 50
Too high
Enter your guess (1 - 99): 25
Too high
Enter your guess (1 - 99): 13
Too low
Enter your guess (1 - 99): 20
Too low
Enter your guess (1 - 99): 23
Correct! Number of guesses: 5
```