Designing and Using Functions

1.

a. `2`
a. `7`
a. `4`

2. a. `max(3, 4)`, then `abs(-5)`, then `min(4, 5)`.
b. `max(2, 8)`, then `min(4, 6, 8)`, then `abs(4)`.
c. `max(5.572, 3.258)`, then `abs(-2)`, then `round(5.572, 2)`.

3.

```
def triple(num):
    """ (number) -> number

    Return num tripled.

    >>> triple(3)
    9
    """

    return num * 3
```

4.

```
def absolute_difference(number1, number2):
    """ (number, number) -> number

    Return the absolute value of the difference between number1
    and number2.

    >>> absolute_difference(3, 7)
    4
    """
    return abs(number1 - number2)
```

5.

```
def km_to_miles(km):
    """ (number) -> float

    Return the distance km in miles.

    >>> km_to_miles(5)
    3.125
    """

    return km / 1.6
```

6.

```
def average_grade(grade1, grade2, grade3):
    """ (number, number, number) -> number

    Return the average of the grade1, grade2, and grade3, where
    each grade ranges from 0 to 100, inclusive.

    >>> average_grade(80, 95, 90)
    88.33333333333333
    """
    return (grade1 + grade2 + grade3) / 3
```

7.

```
def top_three_avg(grade1, grade2, grade3, grade4):
    """ (number, number, number, number) -> number

    Return the average of the top three of grades grade1, grade2,
    grade3, and grade4.

    >>> top_three_avg(50, 60, 70, 80)
    70
    """

    # Here is one solution that does not use average_grade from Q6.
    total = grade1 + grade2 + grade3 + grade4
    top_three = total - min(grade1, grade2, grade3, grade4)
    return top_three / 3

    # Here is a different solution that does use the function from Q6.
    return max(average_grade(grade1, grade2, grade3),
               average_grade(grade1, grade2, grade4),
               average_grade(grade1, grade3, grade4),
               average_grade(grade2, grade3, grade4))
```

8.

```
def weeks_elapsed(day1, day2):
    """ (int, int) -> int
    day1 and day2 are days in the same year. Return the number of full weeks
    that have elapsed between the two days.

    >>> weeks_elapsed(3, 20)
    2
    >>> weeks_elapsed(20, 3)
    2
    >>> weeks_elapsed(8, 5)
    0
    >>> weeks_elapsed(40, 61)
    3
    """
    return int(abs(day1 - day2) / 7)
```

9.

```
Description                Example
Parameter                  num
Argument                   3
Function name              square
Function call              square(3)
```

10.

```
def square(num):
    """ (number) -> number

    Return the square of num.

    >>> square(3)
    9
    """
    return num ** 2
```

...or:

```
def square(num):
    """ (number) -> number

    Return the square of num.

    >>> square(3)
    9
    """
    return num * num
```