1. (10.1) In Harry Potter, the currency consists of knuts, sickle, and galleon. There are 29 knuts in one sickle and 17 sickles in one galleon. Write a **function** that will return a converted amount of knuts into the fewest amount of coins possible. Only return a string with the non-zero values, meaning don't return something similar to "0 sickles". The argument for the function will be *knuts* (how many knuts to convert), if no argument is provided then the **default** should be 900 knuts.

   **Debug this Solution:**

```
1  def convert_knuts(knuts=450):
2    KNUTS_PER_SICKLE = 29
3    SICKLES_PER_GALLEON = 17
4    KNUTS_PER_GALLEON = KNUTS_PER_SICKLE * SICKLES_PER_GALLEON
5
6    galleons = knuts // KNUTS_PER_GALLEON
7    remaining_knuts = knuts // KNUTS_PER_GALLEON
8
9    sickles = remaining_knuts // KNUTS_PER_SICKLE
10   remaining_knuts = remaining_knuts % KNUTS_PER_SICKLE
11
12   output = ""
13
14   if galleons >= 0:
15     if galleons > 1:
16       output = output + str(galleons) + " galleons"
17     else:
18       output = output + str(galleons) + " galleon"
19
20   if sickles > 0:
21     if output:
22       output = output + " "
23     if sickles > 1:
24       output = output + str(sickles) + " sickles"
25     else:
26       output = output + str(sickles) + " sickle"
27
28   if remaining_knuts > 0:
29     if output:
30       output = output + " "
31     if remaining_knuts > 1:
32       output = output + str(remaining_knuts) + " knuts"
33     else:
34       output = output + str(remaining_knuts) + " knut"
35
36   return output
37
38
39 # Test the function with a sample input
40
41 print(convert_knuts(32)) # Expected output: "1 sickle 3 knuts"
42
43 print(convert_knuts()) # Expected output: "1 galleon 14 sickles 1 knuts"
44
45 print(convert_knuts(544)) # Expected output: "1 galleon 4 sickles 18 knuts"
46
47 print(convert_knuts(993)) # Expected output: "2 galleons 7 knuts"
48 # Note: convert_knuts(993) will not output 2 galleons 0 sickle 7 knuts
```

2. (10.2) In each input list, every number repeats at least once, except for two. Write a **function** that takes an array *numbers* and returns the two unique numbers.

**Debug this Solution:**

```
def return_unique(numbers):

  number_dicitonary = {}
  #load dictionary
  for number in range(len(numbers)):
    if number in number_dicitonary:
      number_dicitonary[number] = 1
    else:
      number_dicitonary[number] += 1

  unique_numbers = []
  #find unique numbers in dictionary
  for number in number_dicitonary.values():
    if number_dicitonary[number] == 1:
      unique_numbers.append(number)

  return unique_numbers


# Test the function with a sample input
print(return_unique([1, 9, 8, 8, 7, 6, 1, 6])) # Expected output: [9, 7]
print(return_unique([5, 5, 2, 4, 4, 4, 9, 9, 9, 1])) # Expected output: [2, 1]
print(return_unique([9, 5, 6, 8, 7, 7, 1, 1, 1, 1, 1, 9, 8])) # Expected output: [5, 6]
```

1. (10.1) Write a function called *flip_flop* that takes a string as an argument and returns a new word made up of the second half of the word first combined with the first half of the word second.

**Debug this Solution:**

```
1  def flip_flop(word):
2    length = len(word)
3    middle = length // 2
4
5    if length // 2 == 0:
6      first_half = word[middle:]
7      second_half = word[middle:]
8      return second_half + first_half
9    else:
10     first_part = word[:middle]
11     middle_char = word[middle]
12     last_part = word[middle+1:]
13     return last_part + middle_char + first_part
14
15 # Test the function with a sample input
16 print(flip_flop("abcd")) # Expected output: "cdab" (that is, cd then ab ... even length
       )
17 print(flip_flop("grapes")) # Expected output: "pesgra" (that is, pes then gra ... even
       length)
18 print(flip_flop("abcde")) # Expected output: "decab" (that is, de then c then ab ...
       odd length)
19 print(flip_flop("cranberries")) # Expected output: "rriesecranb" (that is, rries then e
        then cranb ... odd length)
```

2. (10.2) Write a **function** that returns a list with the factors of a given integer. The argument of the function will be *num* (integer to find factors for).

**Debug this Solution:**

```
1  def find_factors(num):
2    factors = []
3
4    for i in range(1, num):
5      if num % i != 0:
6        factors.add(i)
7
8    return factors
9
10 # Test the function with a sample input
11 print(find_factors(12)) # Expected output: [1, 2, 3, 4, 6, 12]
12 print(find_factors(17)) # Expected output: [1, 17]
13 print(find_factors(36)) # Expected output: [1, 2, 3, 4, 6, 9, 12, 18, 36]
```

Dark Helmet Debugger quiz
section 4

1. (10.1) In Harry Potter, the currency consists of knuts, sickle, and galleon. There are 29 knuts in one sickle and 17 sickles in one galleon. Write a **function** that will return a converted amount of knuts into the fewest amount of coins possible. Only return a string with the non-zero values, meaning don't return something similar to "0 sickles". The argument for the function will be *knuts* (how many knuts to convert), if no argument is provided then the **default** should be 900 knuts.

   **Debug this Solution:**

```
def convert_knuts(knuts=450):
  KNUTS_PER_SICKLE = 29
  SICKLES_PER_GALLEON = 17
  KNUTS_PER_GALLEON = KNUTS_PER_SICKLE * SICKLES_PER_GALLEON

  galleons = knuts // KNUTS_PER_GALLEON
  remaining_knuts = knuts // KNUTS_PER_GALLEON

  sickles = remaining_knuts // KNUTS_PER_SICKLE
  remaining_knuts = remaining_knuts % KNUTS_PER_SICKLE

  output = ""

  if galleons >= 0:
    if galleons > 1:
      output = output + str(galleons) + " galleons"
    else:
      output = output + str(galleons) + " galleon"

  if sickles > 0:
    if output:
      output = output + " "
    if sickles > 1:
      output = output + str(sickles) + " sickles"
    else:
      output = output + str(sickles) + " sickle"

  if remaining_knuts > 0:
    if output:
      output = output + " "
    if remaining_knuts > 1:
      output = output + str(remaining_knuts) + " knuts"
    else:
      output = output + str(remaining_knuts) + " knut"

  return output


# Test the function with a sample input

print(convert_knuts(32)) # Expected output: "1 sickle 3 knuts"

print(convert_knuts()) # Expected output: "1 galleon 14 sickles 1 knuts"

print(convert_knuts(544)) # Expected output: "1 galleon 4 sickles 18 knuts"

print(convert_knuts(993)) # Expected output: "2 galleons 7 knuts"
# Note: convert_knuts(993) will not output 2 galleons 0 sickle 7 knuts
```

2. (10.2) YouTube currently displays a like and a dislike button, allowing you to express your opinions about particular content. It's set up in such a way that you cannot like and dislike a video at the same time. There are two other interesting rules to be noted about the interface:

   (a) Pressing a button, which is already active, will undo your press.

   (b) If you press the like button after pressing the dislike button, the like button overwrites the previous "dislike" state. The same is true for the other way round.

   Write a **function** that takes in a list of button inputs *events* and returns the final state.

   **Debug this Solution:**

```
1  def like_or_dislike(events):
2    state = "like"
3
4    for event in range(events):
5      if event != state:
6        state = "nothing"
7      else:
8        state = event
9
10   return state
11
12 # Test the function with a sample input
13 print(like_or_dislike(["dislike"])) # Expected output: "dislike"
14 print(like_or_dislike(["like", "like"])) # Expected output: "nothing"
15 print(like_or_dislike(["dislike", "like"])) # Expected output: "like"
16 print(like_or_dislike(["like", "dislike", "dislike"])) # Expected output: "nothing"
```

1. (10.1) In Harry Potter, the currency consists of knuts, sickle, and galleon. There are 29 knuts in one sickle and 17 sickles in one galleon. Write a **function** that will return a converted amount of knuts into the fewest amount of coins possible. Only return a string with the non-zero values, meaning don't return something similar to "0 sickles". The argument for the function will be *knuts* (how many knuts to convert), if no argument is provided then the **default** should be 900 knuts.

   **Debug this Solution:**

```
1  def convert_knuts(knuts=450):
2    KNUTS_PER_SICKLE = 29
3    SICKLES_PER_GALLEON = 17
4    KNUTS_PER_GALLEON = KNUTS_PER_SICKLE * SICKLES_PER_GALLEON
5
6    galleons = knuts // KNUTS_PER_GALLEON
7    remaining_knuts = knuts // KNUTS_PER_GALLEON
8
9    sickles = remaining_knuts // KNUTS_PER_SICKLE
10   remaining_knuts = remaining_knuts % KNUTS_PER_SICKLE
11
12   output = ""
13
14   if galleons >= 0:
15     if galleons > 1:
16       output = output + str(galleons) + " galleons"
17     else:
18       output = output + str(galleons) + " galleon"
19
20   if sickles > 0:
21     if output:
22       output = output + " "
23     if sickles > 1:
24       output = output + str(sickles) + " sickles"
25     else:
26       output = output + str(sickles) + " sickle"
27
28   if remaining_knuts > 0:
29     if output:
30       output = output + " "
31     if remaining_knuts > 1:
32       output = output + str(remaining_knuts) + " knuts"
33     else:
34       output = output + str(remaining_knuts) + " knut"
35
36   return output
37
38
39 # Test the function with a sample input
40
41 print(convert_knuts(32)) # Expected output: "1 sickle 3 knuts"
42
43 print(convert_knuts()) # Expected output: "1 galleon 14 sickles 1 knuts"
44
45 print(convert_knuts(544)) # Expected output: "1 galleon 4 sickles 18 knuts"
46
47 print(convert_knuts(993)) # Expected output: "2 galleons 7 knuts"
48 # Note: convert_knuts(993) will not output 2 galleons 0 sickle 7 knuts
```

2. (10.2) In each input list, every number repeats at least once, except for two. Write a **function** that takes an array *numbers* and returns the two unique numbers.

**Debug this Solution:**

```
def return_unique(numbers):

  number_dicitonary = {}
  #load dictionary
  for number in range(len(numbers)):
    if number in number_dicitonary:
      number_dicitonary[number] = 1
    else:
      number_dicitonary[number] += 1

  unique_numbers = []
  #find unique numbers in dictionary
  for number in number_dicitonary.values():
    if number_dicitonary[number] == 1:
      unique_numbers.append(number)

  return unique_numbers


# Test the function with a sample input
print(return_unique([1, 9, 8, 8, 7, 6, 1, 6])) # Expected output: [9, 7]
print(return_unique([5, 5, 2, 4, 4, 4, 9, 9, 9, 1])) # Expected output: [2, 1]
print(return_unique([9, 5, 6, 8, 7, 7, 1, 1, 1, 1, 1, 9, 8])) # Expected output: [5, 6]
```