Ch 16-A Advanced Practice - Function

1. Searching

In the file first_perfect_square.py, define a Python function first_perfect_square(numbers) that takes a list of integers (which might be empty) as a parameter. It should return the index of the first number in the list that is a perfect square. Caution: return the index, not the value in the list at that index!

If no element of the list is a perfect square, return -1. We don't care what your function does if its input is not a list of integers (that is, it's allowed to crash if the input is not a list of integers.)

Example of first_perfect_square:

```
>>> first_perfect_square(list(range(5)))
0
>>> first_perfect_square([2, 4, 6, 8, 10, 12])
1
>>> first_perfect_square([6, 8, 10, 12, 9])
4
>>> first_perfect_square([1,1])
0
>>> first_perfect_square([-6, 6, -2, 2, -3, 3])
-1
>>> first_perfect_square([42])
-1
>>> first_perfect_square([])
-1
>>> first_perfect_square([12345678912345678**2])
0
```

2. Counting

In the file num_perfect_squares.py, define a Python function num_perfect_squares(numbers) that takes a list of integers (which might be empty) as a parameter.

It should return the number of elements of the input list that are perfect squares.

We don't care what your function does if its input is not a list of integers (that is, it's allowed to crash.)

Example:

```
>>> num_perfect_squares([])
0
>>> num_perfect_squares([0])
1
>>> num_perfect_squares([0,1])
2
>>> num_perfect_squares(list(range(10)))
4
>>> num_perfect_squares([3]*10)
0
>>> num_perfect_squares([4]*10)
10
>>> num_perfect_squares([-4, -2, 0, 2, 4])
2
>>>
```

3. Second Largest Element

In second_largest.py, define a function second_largest(values) that takes a list containing at least two items as input and returns the second largest item in the list. In the case that the largest item occurs more than once, it would then be considered both the largest and second largest item. For example, the list [1, 2, 3, 3] has 3 as the largest and second largest item. You may assume that there are at least two items in the list and that the items in the list are all the same type, that is, all numbers or all strings or all Booleans; if not, we don't care what your function does. Strings are compared lexicographically. Specifically, in "alpha", "beta", "delta", and "gamma"; since "a" < "b" and "b" < "d" and "d" < "g", the string "delta" is second largest.

Example:

```
>>> second_largest([3, -2, 10, -1, 5])
5
>>> second_largest([-2, 1, 1, -3, 5])
1
>>> second_largest([1,2,3,3])
3
>>> second_largest(["alpha", "gamma", "beta", "delta"])
'delta'
>>> second_largest([3.1, 3.1])
3.1
>>> second_largest([True, False, False, True])
True
>>> second_largest([False, False, True])
False
```

4 Print French Numbers

For this assignment, use the Python source file french_numbers.py below, which contains some starter code for this assignment.

```
#####################################################################
# print french for the numbers between lo and hi (inclusive)
def print_french(lo, hi):
    return None

def digit(num, pos):
    return (num // 10**(pos-1)) % 10

def num_in_french(num): # assumes 0 <= num <= 100

    ones_list = ["zero", "un", "deux", "trois", "quatre", "cinq", "six", "sept", "huit", "neuf", "dix","onze", "douze", "treize",
"quatorze", "quinze", "seize", "dix-sept", "dix-huit", "dix-neuf"]

    tens_list = ["", "dix", "vingt", "trente", "quarante", "cinquante",  "soixante", "soixante", "quatre-vingt", "quatre-vingt"]

    # Part 1: get the ones and tens digits of num

    # Part 2: fill in code below for numbers 1, 2, 3, ..., 19 and 100

    # Part 4: case when the numbers are 70, 71, 72,...79, and 90, 91, 92,...99

    # Part 5: otherwise the case when the numbers are 20, 30, 40, ...

    # Part 6: otherwise the case when the numbers are 21, 31, 41, ...

    # Part 7: everything else, the most general case for 22, 23, ... 29, 32, 33, ..., 39, 42, ...
#####################################################################
```

The French counting system uses a mixture of decimal (ten-based) and vigesimal (twenty-based) counting (see the table below). In the file french_numbers.py given you, you will complete the function num_in_french(num) that takes a integer number between 0 and 100 and returns a string that is that number written in French. You will also write a tester that will let you visually check that you get the same output as the French translations shown below.

Number	French	Literally
0	zero	0
1	un	1
2	deux	2
3	trois	3
4	quatre	4
5	cinq	5
6	six	6
7	sept	7
8	huit	8
9	neuf	9
10	dix	10
11	onze	11
12	douze	12
13	treize	13
14	quatorze	14
15	quinze	15
16	seize	16
17	dix-sept	10-7
18	dix-huit	10-8
19	dix-neuf	10-9

Number	French	Literally
20	vingt	20
21	vingt et un	20 and 1
22	vingt-deux	20-2
23	vingt-trois	20-3
24	vingt-quatre	20-4
25	vingt-cinq	20-5
26	vingt-six	20-6
27	vingt-sept	20-7
28	vingt-huit	20-8
29	vingt-neuf	20-9
30	trente	30
31	trente et un	30 and 1
32	trente-deux	30-2
33	trente-trois	30-3
34	trente-quatre	30-4
35	trente-cinq	30-5
36	trente-six	30-6
37	trente-sept	30-7
38	trente-huit	30-8
39	trente-neuf	30-9

Number	French	Literally
40	quarante	40
41	quarante et un	40 and 1
42	quarante-deux	40-2
43	quarante-trois	40-3
44	quarante-quatre	40-4
45	quarante-cinq	40-5
46	quarante-six	40-6
47	quarante-sept	40-7
48	quarante-huit	40-8
49	quarante-neuf	40-9
50	cinquante	50
51	cinquante et un	50 and 1
52	cinquante-deux	50-2
53	cinquante-trois	50-3
54	cinquante-quatre	50-4
55	cinquante-cinq	50-5
56	cinquante-six	50-6
57	cinquante-sept	50-7
58	cinquante-huit	50-8
59	cinquante-neuf	50-9

Number	French	Literally
60	soixante	60
61	soixante et un	60 and 1
62	soixante-deux	60-2
63	soixante-trois	60-3
64	soixante-quatre	60-4
65	soixante-cinq	60-5
66	soixante-six	60-6
67	soixante-sept	60-7
68	soixante-huit	60-8
69	soixante-neuf	60-9
70	soixante-dix	60-10
71	soixante et onze	60 and 11
72	soixante-douze	60-12
73	soixante-treize	60-13
74	soixante-quatorze	60-14
75	soixante-quinze	60-15
76	soixante-seize	60-16
77	soixante-dix-sept	60-10-7
78	soixante-dix-huit	60-10-8
79	soixante-dix-neuf	60-10-9

Number	French	Literally
80	quatre-vingts	4-20s*
81	quatre-vingt-un	4-20-1
82	quatre-vingt-deux	4-20-2
83	quatre-vingt-trois	4-20-3
84	quatre-vingt-quatre	4-20-4
85	quatre-vingt-cinq	4-20-5
86	quatre-vingt-six	4-20-6
87	quatre-vingt-sept	4-20-7
88	quatre-vingt-huit	4-20-8
89	quatre-vingt-neuf	4-20-9
90	quatre-vingt-dix	4-20-10
91	quatre-vingt-onze	4-20-11
92	quatre-vingt-douze	4-20-12
93	quatre-vingt-treize	4-20-13
94	quatre-vingt-quatorze	4-20-14
95	quatre-vingt-quinze	4-20-15
96	quatre-vingt-seize	4-20-16
97	quatre-vingt-dix-sept	4-20-10-7
98	quatre-vingt-dix-huit	4-20-10-8
99	quatre-vingt-dix-neuf	4-20-10-9
100	cent	100

** Programming Steps **

As you follow the steps below, you will find a corresponding comment in the french_numbers.py file that indicates you where you should write your code for that step.

Step 1 In the num_in_french function use the digit function to assign the ones digit and tens digit of num to appropriate named variables.

Step 2 First, let's consider the simplest cases. In the Python function num_in_french(num) given to you, you will notice that there are two lists of strings already defined for you. The ones_list contains the French for numbers 0 through 19, and the tens_list contains the French for 10, 20, 30, ... 90.
Add code to num_in_french(num) so that it returns the French translation when num is between 0 and 19 inclusive or when num is 100.

Step 3 In the file french_numbers.py complete the Python function print_french(lo, hi) that prints the numbers from lo to hi inclusively, along with the French translation.
You may assume that lo is less than or equal to hi.

Example usage (at this point) :

```
>>>print_french(0, 3)
0 zero
1 un
2 deux
3 trois
>>>print_french(18, 19)
18 dix-huit
19 dix-neuf
>>>print_french(100, 100)
100 cent
```

Step 4 If you look carefully at the tens_list, you will notice that it repeats "soixante"
for 60 and 70 and repeats "quatre-vingt" for 80 and 90 because starting at 60 French switches to 20-based counting.
That is, the 70s and 90s will need to be handled differently than the 20s, 30s, ..., 60s, and 80s.
Add code to num_in_french function for numbers in the seventies and nineties.
Take special note that 71 and 91 are translated in slightly different ways.
Therefore, you will need to used an if-else statement nested inside this case to handle any exceptions.
Example usage (at this point) :

```
>>>print_french(12, 13)
12 douze
13 treize
>>>print_french(70, 73)
70 soixante-dix
71 soixante et onze
72 soixante-douze
73 soixante-treize
>>>print_french(90, 92)
90 quatre-vingt-dix
91 quatre-vingt-onze
92 quatre-vingt-douze
```

Step 5 Let's now consider all the multiples of 10. That is, write code in num_in_french so that
it returns the correct French for 20, 30, 40, ... Take note that 80 does not follow the general rule here.
Example usage (at this point) :

```
>>>print_french(19, 20)
19 dix-neuf
20 vingt
>>>print_french(40, 40)
40 quarante
>>>print_french(79, 80)
79 soixante-dix-neuf
80 quatre-vingts
```

Step 6 Next consider the case when the ones digit is 1, that is, for 21, 31, ...
Again, take note that 81 does not follow the general rule here.

Example usage (at this point):

```
>>>print_french(20, 22)
20 vingt
21 vingt et un
22 vingt-deux
>>>print_french(51, 51)
51 cinquante et un
>>>print_french(80, 82)
80 quatre-vingts
81 quatre-vingt-un
82 quatre-vingt-deux
```

Step 7 Finally, consider all the remaining numbers, that is the most common case for the 2-digit numbers,
22, 23, 24, ..., 29, 32, 33, ..., 39, 40, ... At this point num_in_french should return the correct french
for all numbers from 0 through 100. Visually compare the French numbers your solution returns with all the numbers in table above.