Quiz 9

1. Examine each of the following snippets of code and list the output when it runs. Write 'error' if you think the code will crash (i.e. raise an error) during execution. Write '- nothing -' if you think the code will not produce any output.

```
# part 1
sentence = '-apple-orange-pear-'
print(sentence.strip('-'))
```

Answer:

["", "apple", "orange", "pear", ""]

```
# Part 2
my_dict = {}
my_dict['lucky'] = [1]
numbers = my_dict['lucky']
numbers.append(2)
print(my_dict)
```

{lucky: [1]}
{lucky:

Answer:

{'lucky': [1,2]}

```
# Part 3
inventory = {(1,2): 12}
print(inventory[(1,2)])
```

Answer:

12

2. Complete the following program making full use of the scaffolding code. The program keeps prompting the user for two numbers, num1 and num2 until the following conditions are fulfilled:
    a. num1 is larger than num2
    b. both numbers must be larger than 5

    After which, the sum of the two numbers is printed.

```
enter num1>1
enter num2>3
invalid

enter num1>20
enter num2>-1
invalid

enter num1>10
enter num2>6
valid. Sum is 16!
```

```
def is_conditions_fulfilled(num1, num2):
    # TODO: complete the return statement
    # Write your answer below:
    return ((num1 > num2) and (num1>5) and (num2>5))









num1 = int(input("Enter num 1> "))
num2 = int(input("Enter num 2> "))

while  not is_conditions_fulfilled(num1, num2):
    print('invalid')
    print()
    num1 = int(input("Enter num 1> "))
    num2 = int(input("Enter num 2> "))

print(f'valid. Sum is {num1 + num2}!')
```

3. Complete the following code so that when `q3_test.py` is executed, it will produce the following output:

```
Test 1:Checking return data type
Expected:True True
Actual  :True True

Test 2
Expected:{'odd': [5, 1, 13, 99], 'even': [6, 8, 10]}
Actual  :{'odd': [5, 1, 13, 99], 'even': [6, 8, 10]}

Test 3
Expected:{'odd': [], 'even': [6, 2, 4]}
Actual  :{'odd': [], 'even': [6, 2, 4]}

Test 4
Expected:{'odd': [], 'even': []}
Actual  :{'odd': [], 'even': []}
```

```
# content of q3_test.py
import q3

numbers = [5, 6, 1, 13, 8, 10, 99]
result = q3.sort_into_odd_even(numbers)
print('Test 1:Checking return data type')
print('Expected:True True')
print('Actual  :' + str(isinstance(result, dict)),
     isinstance(result['odd'], list))
print()

numbers = [5, 6, 1, 13, 8, 10, 99]
result = q3.sort_into_odd_even(numbers)
print('Test 2')
print("Expected:{'odd': [5, 1, 13, 99], 'even': [6, 8, 10]}")
print('Actual  :' + str(result))
print()

numbers = [6, 2, 4]
result = q3.sort_into_odd_even(numbers)
print('Test 3')
print("Expected:{'odd': [], 'even': [6, 2, 4]}")
print('Actual  :' + str(result))
print()

numbers = []
result = q3.sort_into_odd_even(numbers)
print('Test 4')
print("Expected:{'odd': [], 'even': []}")
print('Actual  :' + str(result))
print()
```

```
# Your Answer (q3.py)
def sort_into_odd_even(numbers):
    odd = []
    even = []
    for i in numbers:
        if i % 2 == 0:
            even.append(i)
        else:
            odd.append(i)

    final = {'odd': odd, 'even': even}
    return final
```

4. Draw the memory state diagram for the following program at the point of time when the program reaches line 4:

```
1  def do_magic(numbers):
2     numbers[0].append(7)
3     numbers[1] = numbers[1] + [5,6]
4     # How does the memory state diagram look here?
5
6
7  numbers = [[1], [2, 3]]
8  do_magic(numbers)
9  print(numbers)
```

Answer: