

## CSCA08H Winter 2022 Worksheet: Nested Lists and Loops

1. Consider this code:

```
data = [['a', 'b'], [3, 4], ['cat', 'mouse', 'elephant']]
sublist = data[2]
```

For each pair of expressions, circle the one that evaluates to 3:

(a)	(b)	(c)	(d)
data[2]	data[1]	sublist[0]	data[2][0]
len(data[2])	data[1][0]	len(sublist[0])	len(data[2][0])

2. Which of the following code fragments does *not* create a nested list (a list that contains at least one other list)?

(a) 

```
nums = []
for i in range(4):
    nums = nums + [i]
```

(b) 

```
nums = [0, 1, 2, 3]
nums[-1] = [3, 4, 5]
```

(c) 

```
nums = []
for i in range(4):
    nums.append([i])
```

(d) 

```
nums = [0, 1, 2, 3]
letters = ['a', 'b', 'c', nums]
```

3. Consider this code:

```
teams = [['Canadiens', 'Leafs', 'Senators'], ['Jets'], ['Oilers', 'Canucks']]
```

Which of the following expressions will *not* evaluate to 5?

(a) `len(teams[0]) + len(teams[-1])`    (b) `len(teams[0] + teams[2])`

(c) `len(teams) - 1`    (d) `len(teams[0][1])`

## CSCA08H Winter 2022 Worksheet: Nested Lists and Loops

4. Complete the examples in the docstring and then the function body.

```
from typing import List

def digital_sum(nums_list: List[str]) -> int:
    """Return the sum of all the digits in all strings in nums_list.

    Precondition: s.isdigit() holds for each string s in nums_list.

    >>> digital_sum(['64', '128', '256'])
    34
    >>> digital_sum(['12', '3'])
    """
    
    """
```

5. Complete the examples in the docstring and then the function body.

```
def can_pay_with_two_coins(denoms: List[int], amount: int) -> bool:
    """Return True if and only if it is possible to form amount, which is a
    number of cents, using exactly two coins, which can be of any of the
    denominations in denoms.

    >>> can_pay_with_two_coins([1, 5, 10, 25], 35)
    True
    >>> can_pay_with_two_coins([1, 5, 10, 25], 20)
    True
    >>> can_pay_with_two_coins([1, 5, 10, 25], 12)
    """
    
    """
```