Lab-5 Python List Comprehension - Lab activity

50 marks - Deadline: 5.00 PM - you can submit latest by 5.10 PM. No submissions will be considered after this, so prepare accordingly.

Instructions:

- Wherever required, use the code snippets given in this document.
- Submit two `.py` files
  - `oneToFour.py` - contains code for Q1, Q2, Q3, Q4.
  - `five.py` - contains code only for Q5.
- Format of `oneToFour.py` - The code for each question should be separated using comments like this:

```
# Q1
......  code goes here
# Q2
......  code goes here
# Q3
......  code goes here
# Q4
......  code goes here
```

  you can download the started code from here oneToFour.py
- Output format of `oneToFour.py` (don't print the comments)

```
[('hyderabad', 9), ('mumbai', 6), ('bengal', 6)] #q1
[4, 5, 6, 8, 10, 12, 12, 15, 18] #q2
[3, 2, 5, 2 ...] #q3 # whatever the output is
[0, 0, 0] #q4
```

- Output format of `five.py`

```
map function:  1.205
comprehension:  1.635538
for loop:  2.18075
```

.

.

.

.

Moodle submission file structure:

```
<rollno>.zip
      oneToFour.py
      five.py
```

Rubrics:

- Format of `oneToFour.py` - **2.5 marks**
- Moodle submission:
    - naming and directory structure - **2.5 marks**
- Questions - 0 for wrong output, full for expected output - no step marking
    - **5 marks x 2**
    - **10 marks x 2**
    - **15 marks x 1**

---

5 marks

Q1. Write single line code to generate the following output using `s` as input:

```
# 9, 6, 6 are lengths of the corresponding strings
[('hyderabad', 9), ('mumbai', 6), ('bengal', 6)]
```

```
s=["hyderabad", "mumbai", "bengal"]
```

---

5 marks

Q2.Write single line code to generate the following output using `l1`, `l2` as input.

```
# multiplied each item of l1 with each item of l2
[4, 5, 6, 8, 10, 12, 12, 15, 18]
```

```
l1=[1,2,3]
l2=[4,5,6]
```

---

10 marks

Q3. Use a nested list comprehension to find all of the numbers from 1–50 that are divisible by any single digit in range(5–7).

---

10 marks

Q4. Write single line code to generate the following output using `l1`, `l2` as input:

```
# items of l1 and l2 at same index are added
[0, 0, 0]
```

```
l1=[5, 10, 15]
l2=[-5, -10, -15]
```

---

15 marks

Q5. Report the performance of list comprehension vs. loops vs map function.

Use `get_price()` function to calculate price of each element from `old_prices` list. Return a list of final prices.

Use the following starter code. You can download it from here five.py

```
import random
import timeit
old_prices = [random.randrange(100) for _ in range(100000)]
def get_price(old_price):
    return old_price * 1.8

def get_prices_with_map():
    # logic goes here

def get_prices_with_comprehension():
    # logic goes here

def get_prices_with_loop():
    # logic goes here

print("map function: ", timeit.timeit(get_prices_with_map, number=100))
print("comprehension: ",timeit.timeit(get_prices_with_comprehension, number=100))
print("for loop: ", timeit.timeit(get_prices_with_loop, number=100))
```