QUEENSBOROUGH COMMUNITY COLLEGE
The City University of New York
Department of Engineering Technology

Programming Exercises – For loops, Lists and Tuples

1. A – D, **determine the output** displayed by the lines of code. E – G, **write the codes** by using the output. Save your code as *PE5_1.py*.

<table>
<tr><td>A</td><td><pre>a = list(range(5))
print(a)</pre></td><td>B</td><td><pre>b = []
for i in range (5):
    b.append(i)
print(b)</pre></td></tr>
<tr><td>Output</td><td></td><td>Output</td><td></td></tr>
<tr><td>C</td><td colspan="3"><pre>x = list(range(-10, 10))
print(x)
print(min(x), max(x), sum(x))</pre></td></tr>
<tr><td>Output</td><td colspan="3"></td></tr>
<tr><td>D</td><td colspan="3"><pre>even_num = list(range(2, 11, 2))
print(even_num[0], even_num[-1])</pre></td></tr>
<tr><td>Output</td><td colspan="3"></td></tr>
<tr><td>E</td><td colspan="3"><pre>#Print all the odd numbers from 1 to 9 inclusive in a list, odd_num.</pre></td></tr>
<tr><td>Output</td><td colspan="3">[1, 3, 5, 7, 9]</td></tr>
<tr><td>F</td><td colspan="3"><pre># Make a list of the first 10 cubes and use a for loop to print out
the value of each cube in a new line (see output below).</pre></td></tr>
<tr><td>Output</td><td colspan="3"><pre>1
8
27
64
125
216
343
512
729
1000</pre></td></tr>
<tr><td>G</td><td colspan="3"><pre>#Use a list comprehension to generate a list of the first 10 cubes.
Use a for loop to print out the value of each cube in a row
separated by a '|' (see output below).</pre></td></tr>
<tr><td>Output</td><td colspan="3">1|8|27|64|125|216|343|512|729|1000|</td></tr>
</table>

2. List slicing. Save it as *PE5_2.py*.
a) Use a **list comprehension** to generate a list of all even numbers from 0 to 100 **inclusive**.
b) Use slicing to print the first five even numbers in the list.
c) Use slicing to print the last five even numbers in the list.
d) Use slicing to print all list numbers between 20 and 30 inclusive.

Example Output

```
[0, 2, 4, 6, 8]
[92, 94, 96, 98, 100]
[20, 22, 24, 26, 28, 30]
```

3. Lists, comprehensions, loops and slicing. Save it as *PE5_3.py*.
a) Create a **list comprehension** of multiples of 4 from 0 to 10 **inclusive**.
b) Print this list as displayed in the example output.
c) Create a second **empty** list.
d) Use a loop to insert all elements from the first list to the second list.
Before storing into the new list, divide each copied element by 2.
This results in a new list of all multiples of 2 from 0 to 10 inclusive.
e) Print the second list as displayed in the example output.
f) Use slicing to **copy** the second list to a new third list.
g) Use a loop to divide and store each element of the third list by 2.
This will result in a list of the numbers 0 to 10 inclusive.
h) Print the third list as displayed in the example output.

Example Output

```
[0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40]

[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

4. Implement the code that replaces the name of each month with its three-letter abbreviation. Save it as *PE5_4.py*.
a) Create a list below and print it out.
months = ['january', 'february', 'march', 'april', 'may', 'june', 'july', 'august', 'september', 'october', 'november', 'december']
b) Use a ***for*** loop to store each month with its first three-letter abbreviation into a **new** list.
c) Print the value of each month in uppercase separated by a '|' in a row as displayed in the example output.
d) Print the new list.

Example Output

```
Original list:
['january', 'february', 'march', 'april', 'may', 'june', 'july', 'august', 'september', 'october', 'november', 'december']

Three-letter abbreviation 1 - 12:
JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC|

New list:
['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
```

5. Implement the following to create a list and produce the multiplication table. Save it as *PE5_5.py*.
a) Request an integer input *range*.
b) Implement a list of the numbers 1 to *range* inclusive.
c) Request an integer input *number*.
d) Use a loop upon this list to compute and print the multiplication table of the input *number*.
Input text can be any content. Just make sure to precisely match the output format below.

Example Output 1

```
Enter a range: 6

Enter an integer number: 6

Multiplication Table of 6
1    *    6    =    6
2    *    6    =    12
3    *    6    =    18
4    *    6    =    24
5    *    6    =    30
6    *    6    =    36
```

Example Output 2

```
Enter a range: 10

Enter an integer number: 10

Multiplication Table of 10
1    *    10   =    10
2    *    10   =    20
3    *    10   =    30
4    *    10   =    40
5    *    10   =    50
6    *    10   =    60
7    *    10   =    70
8    *    10   =    80
9    *    10   =    90
10   *    10   =    100
```

6. A – E, identify the errors and **rewrite** the statement in the correct syntax. Save your code as *PE5_6.py*.

A	`fruits = ["apple", "banana", "cherry"]`
	`for item in fruits`
	`    print(item)`

Debug	
B	`for i in range (1, 4):` `    print(i + '\t' + 2**i)`
Debug	

C	`#Why is there no output when you run the code?`
	`for j in range (1, 6, -1):`
	`    print(j)`

Debug	

D	`#How to display all the elements in uppercase?`
	`letters = ['a', 'b', 'c']`
	`for letter in letters:`
	`    letter = letter.upper()`
	`print(letters)`

Debug	

E	`fruits = ('apple', 'banana', 'cherry')`
	`print(fruits)`
	`fruits[0] = 'orange'`
	`fruits.append('pineapple')`
	`print(fruits)`

Debug	

7. A – F, determine and **explain** the output displayed by the lines of code. Save your code as *PE5_7.py*.

A	`fruits = ["apple", "pear", 'python',]`
	`for item in fruits:`
	`  print(f"{item.title()} is my favorite!")`
	`  print(f"I want to have more {item}.\n")`

Output	

B	`numbers = [1,2,3,4,5]`
	`for n in numbers:`
	`  print(n)`
	`print("That's all the numbers in the list.")`
	`print("numbers = ", numbers)`

Output	

C	`n = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`
	`count = 0`
	`for i in n:`
	`  print(i, end = '\t')`
	`  count += 1`
	`print(f"\nThere are {count} numbers in the list.")`
	`print("n = ", n)`

Output	

D	`languages = ["c++", "java", "python"]`
	`for code in languages:`
	`  print(code.upper(), end = " | ")`
	`else:`
	`  print("Enjoy coding!")`

Output	

E	`n = -6, 7, 3, -2, 6, 3, 9`
	`print(len(n), max(n), min(n), sum(n), sep = '\n')`
	`print(n.count(3), n.index(3), n[-6:6], sep = '\n')`
	`print(n, sorted(n), sep = '\n')`

Output	

F	`a = 2`
	`b = 3`
	`print(type(a+b))`
	`print(type((a+b,)))`
	`print(type(()))`

Output	