CSX3001/ITX3001
CS1201 COMPUTER PROGRAMMING 1

CLASS 05 NESTED CONTROL STRUCTURE AND FOR-LOOP

NESTED LOOP, BREAK VS. CONTINUE, AND FOR-LOOP

PYTHON

REVIEW

```
1   countS = 0
2   nStar = 10
3   while countS < nStar :
4      print("*", end = "")
5      countS += 1
```

This program will print out 10 stars ("*"s) over a line. This is an example of definite loop because the number of times this loop will repeat is known in advance.

Consider a statement at line #4, `end=""` is an option for `print`() function to end the output string with empty string. Normally, when `print`() is called the output string will end with **newline (\n)** that is the reason why the next output will start over the next line.

You can use this option, `end=""`, if you want to continue the same line after the previous output string.

TODO: Remove `end=""` from line #4 and rerun the code again to see the result.

NESTED LOOP

Nested control structure is a concept of placing a control structure or more in another control structure. An example of nested control structure that you have studied so far is the ***nested if...else*** statement.

Loop control structure can be nested as well, which will become the ***nested loop***.

You can simply call nested loop as a loop or many loops inside a loop. A simple example can be found down here:

```
1   countL = 0
2   nLine = 5
3   while countL < nLine :
4      countS = 0
5      nStar = 10
6      while countS < nStar :
7         print("*", end="")
8         countS += 1
9      countL += 1
10     print()
```

This example will print 10 stars over 5 lines. The code will consist of the outer loop that start from line #3 to #10 and the inner loop that starts from line#6 up to line#8. The body of the outer loop contains the whole inner loop.

TODO: What is the purpose of `print()` at line #10 ? What will happen if you remove it?

All control structures can be nested. For example, nested if...else, nested loop, while loop is placed in the body of if statement, and etc.

BREAK AND CONTINUE STATEMENT

Syntax of break and continue statement:

```
break
```

```
continue
```

They are the reserved words in Python and used for early exit or skip statement(s) from loop control structure. The difference between **break** and **continue** statement are as follow:

- `break` is used for exit from the loop.
- `continue` is used for skipping the statements that come after, but the control still be in the loop.

Normally they are used together with conditional statement.
Let's consider the following example:

```
countL = 0
nLine = 10
print("There is an example")
while countL < 10 :
    countL += 1
    if (countL % 3) == 0 :
        continue
    print("statement #" + str(countL))
print("The last statement")
```

This program will skip printing `statement#3, statement#6`, and `statement#9` because of **continue** statement will skip the left part in the body of the loop and continue with the next round by testing with the condition of this loop (`countL < 10`) again.

TODO: Replace **continue** in the previous example with **break** and let's see what will be printed out.

TODO: What will be occurred if the **break** statement is executed in the body of inner loop? Will it break only the inner loop? or both inner and outer loop?

The break statement can make while loop and while…else loop run differently. More information and example will be given in class.

FOR-LOOP

```
for variable in sequence :
   code block
```

The for-loop in Python is used to iterate over a sequence (list, tuple, string) or other iterable objects. It takes the value of the item inside the sequence on each iteration. For-loop continues until it reaches the last item in the sequence. The body of for loop is separated from the rest of the code using indentation.

Example#1 of a for-loop:

```
myList = [1, 3, 5, 7, 9, 11]
for myVar in myList :
    print(myVar)
```

Example#2 of a for-loop:

```
myStr = "Hello Class"
for myChar in myStr :
    print(myChar)
```

RANGE() FUNCTION

```
range(start, stop, stepSize)
```

A sequence of numbers can be generated by using range() function. The range(10) will generate numbers from 0 to 9 (10 numbers).

```
print(list(range(10)))
```

list() function will converse a sequence from **range()** to a list.

We can also define the start, stop and step size as **range(start, stop, step size)**. The default step size value is 1 if not provided.

```python
print(list(range(2, 10, 2))
```

range() function can be used with for loop which you can see from the following example,

```python
for number in list(range(3, 16, 3)) :
    print(number)
    print(f"{number} * 5 = {number * 5}")
```

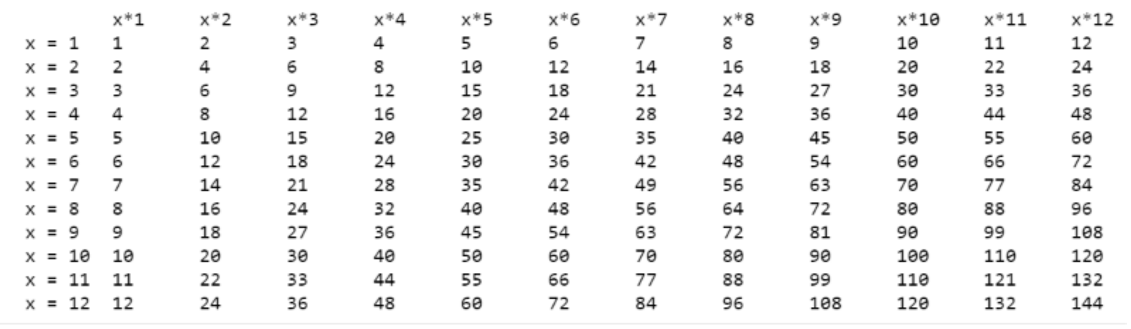
or by omitting **list()** function you can get the same result.

```python
for number in range(3, 16, 3) :
    print(number)
    print(f"{number} * 5 = {number * 5}")
```

EXERCISES

1) Write a Python code using nested while-loop to print out the following multiplication table of 12 x 12 (integer number). You have to strictly follow the format of the given sample run.

Sample Run:



```
         x*1     x*2     x*3     x*4     x*5     x*6     x*7     x*8     x*9     x*10    x*11    x*12
x = 1    1       2       3       4       5       6       7       8       9       10      11      12
x = 2    2       4       6       8       10      12      14      16      18      20      22      24
x = 3    3       6       9       12      15      18      21      24      27      30      33      36
x = 4    4       8       12      16      20      24      28      32      36      40      44      48
x = 5    5       10      15      20      25      30      35      40      45      50      55      60
x = 6    6       12      18      24      30      36      42      48      54      60      66      72
x = 7    7       14      21      28      35      42      49      56      63      70      77      84
x = 8    8       16      24      32      40      48      56      64      72      80      88      96
x = 9    9       18      27      36      45      54      63      72      81      90      99      108
x = 10   10      20      30      40      50      60      70      80      90      100     110     120
x = 11   11      22      33      44      55      66      77      88      99      110     121     132
x = 12   12      24      36      48      60      72      84      96      108     120     132     144
```

2) Repeat exercise 1) with for-loop.

3) Write a Python code that takes an integer, namely myInt, and then print the outputs ranging from myInt to zero. For odd iteration, the value of myInt is decreased by two and for even iteration the value of myInt is increased by 1. Assume that the first loop is an odd iteration. Also draw a flowchart of your code.

4) Write the Python code that asks for your year of birth and print the outputs as shown in the sample runs below.

```
Assume that your date of birth is 1 January.
Enter your year of birth (A.D.):2014
By December 2019, you are 6 years old.
By December 2018, you are 5 years old.
By December 2017, you are 4 years old.
By December 2016, you are 3 years old.
By December 2015, you are 2 years old.
By December 2014, you are 1 years old.
```

```
Assume that your date of birth is 1 January.
Enter your year of birth (A.D.):2016
By December 2019, you are 4 years old.
By December 2018, you are 3 years old.
By December 2017, you are 2 years old.
By December 2016, you are 1 years old.
```

Hint: use range(start, end, -1) to start the range from a higher value down to a lower value with a step -1. For example, range(5, 1, -1) is equivalent to 5, 4, 3, 2.

5) The following Python code prints all name started with capital 'G'. modify the following Python code in to print all name from the nameList that the second character of the name is alphabet 'e'.

```python
name_list = ['Peter', 'Tommy', 'George', 'Yuri', 'Sanchez', 'Geo']
for name in name_list:
    if name[0] == 'G':
        print(name)
```

The outputs should be 'Peter', 'George' and 'Geo'.

6) Write a Python code that takes two integers, a and b. Then print all integer between these two numbers (including a and b). Note that any integer ended with 3 and 7 will be replaced by *.

   For example, if you enter 2 and 12 for a and b, the code will print

```
Enter a value for a: 2
Enter a value for b: 12
2 * 4 5 6 * 8 9 10 11 12
```

   If you enter 2 and 28, the code will print

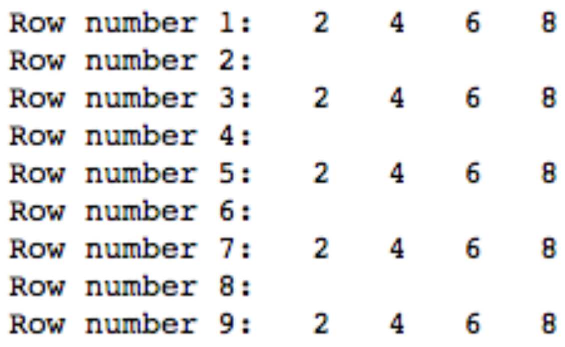
```
Enter a value for a: 2
Enter a value for b: 28
2 * 4 5 6 * 8 9 10 11 12 * 14 15 16 * 18 19 20 21 22 * 24 25 26 * 28
```

7) Based on the following Python code, modify the code to print 9 rows each with values 1 to 9.

```
for i in range(3):
    print('Row number', i + 1, end = ": ")
    for j in range(1, 6):
        print(j, end = " ")
    print() # to enter a new line
```

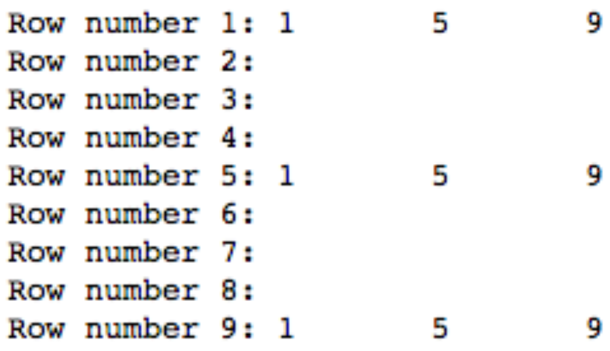
```
Row number 1: 1 2 3 4 5 6 7 8 9
Row number 2: 1 2 3 4 5 6 7 8 9
Row number 3: 1 2 3 4 5 6 7 8 9
Row number 4: 1 2 3 4 5 6 7 8 9
Row number 5: 1 2 3 4 5 6 7 8 9
Row number 6: 1 2 3 4 5 6 7 8 9
Row number 7: 1 2 3 4 5 6 7 8 9
Row number 8: 1 2 3 4 5 6 7 8 9
Row number 9: 1 2 3 4 5 6 7 8 9
```

8) Modify the code again to print odd rows (rows no. 1, 3, 5, 7 and 9) with even values (2, 4, 6, 8) for each row.



```
Row number 1:   2   4   6   8
Row number 2:
Row number 3:   2   4   6   8
Row number 4:
Row number 5:   2   4   6   8
Row number 6:
Row number 7:   2   4   6   8
Row number 8:
Row number 9:   2   4   6   8
```

9) Modify the code again to print rows (1, 5 and 9) with values (1, 5, 9) for each row.



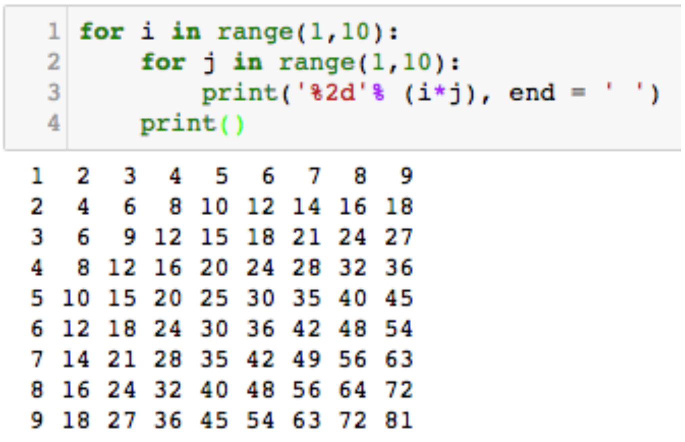
```
Row number 1: 1       5       9
Row number 2:
Row number 3:
Row number 4:
Row number 5: 1       5       9
Row number 6:
Row number 7:
Row number 8:
Row number 9: 1       5       9
```

ASSIGNMENTS

Complete the following exercises in Python IDLE. You must name the python file as, {your-id}_class0{number}_{course-code}_{section-number}_assignment{number}.py for example, for assignment 1 will be named,

6120001_class05_csx3001_541_assignment1.py

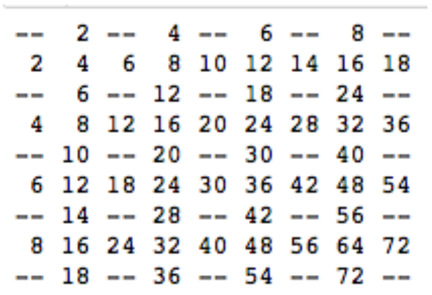
1) Based on the given Python code, modify the code to print the number where all digits are even number. Otherwise, "--" will be printed as shown in the expected output below.



```
for i in range(1,10):
    for j in range(1,10):
        print('%2d'% (i*j), end = ' ')
    print()
```

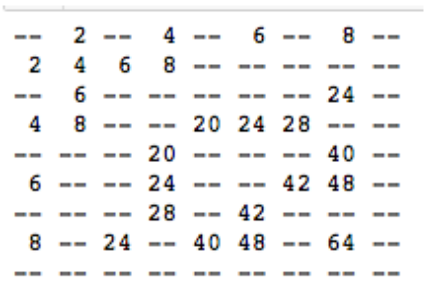
```
 1  2  3  4  5  6  7  8  9
 2  4  6  8 10 12 14 16 18
 3  6  9 12 15 18 21 24 27
 4  8 12 16 20 24 28 32 36
 5 10 15 20 25 30 35 40 45
 6 12 18 24 30 36 42 48 54
 7 14 21 28 35 42 49 56 63
 8 16 24 32 40 48 56 64 72
 9 18 27 36 45 54 63 72 81
```

a) 1st **expected output** (only even numbers):



```
--  2 --  4 --  6 --  8 --
 2  4  6  8 10 12 14 16 18
--  6 -- 12 -- 18 -- 24 --
 4  8 12 16 20 24 28 32 36
-- 10 -- 20 -- 30 -- 40 --
 6 12 18 24 30 36 42 48 54
-- 14 -- 28 -- 42 -- 56 --
 8 16 24 32 40 48 56 64 72
-- 18 -- 36 -- 54 -- 72 --
```

b) 2nd **expected output** (only even numbers and both digits must be even numbers):



```
--  2 --  4 --  6 --  8 --
 2  4  6  8 -- -- -- -- --
--  6 -- -- -- -- -- 24 --
 4  8 -- -- 20 24 28 -- --
-- -- -- 20 -- -- -- 40 --
 6 -- -- 24 -- -- 42 48 --
-- -- -- 28 -- 42 -- -- --
 8 -- 24 -- 40 48 -- 64 --
-- -- -- -- -- -- -- -- --
```

2) Write a Python code to print all integer numbers between 0 and 30 with the following conditions:
    a. All printed numbers must be divisible by 3 (otherwise, * symbol is printed), and
    b. the number between 10 – 15 and 20 – 25 will be replaced by an underscore symbol.

Expected output is given below.

```
0 * * 3 * * 6 * * 9 _ _ _ _ _ _ * * 18 * _ _ _ _ _ _ * 27 * * 30
```

3) Write a Python code that takes two integer inputs, namely numA and numB. Note that numA might be higher or lower than numB. The code will print integer numbers ranging from a lower integer value to a higher integer value, and shall not print integer number(s) that is(are) divisible by 3 and 7. However, if the sum of the printed integer numbers is greater than 5 times of the higher value, the code immediately stops printing the number. For example, if you enter 31 and 14 for numA and numB, respectively, the output should be

```
14 15 16 17 18 19 20 22 23
```

Note that 21 is not printed due to the fact that it is divisible by 3 and 7.

4) Write a python code that find out the lowest different between two numbers from the given list. Note that the numbers in the given list can be changed.

Sample #1

```
numbers = [4, 1, 2, 9, 7, 100, 5, 0, 99, 100]
```

```
The lowest different between two numbers is 0.
```

Sample #2

```
numbers = [100, 10, 200, 25, 7, 20]
```

```
The lowest different between two numbers is 3.
```

5) Based on the assignment #5, suppose there are two given list. Find the lowest different between two numbers from the two list.

Sample #1

```
numberA = [4, 1, 2, 9, 7, 100, 5, 0, 99, 100]
numberB = [100, 10, 200, 25, 7, 20]
```

```
The lowest different between two numbers is 0.
```

Sample #2

```
numberA = [1, 52, 36, 25, 8, -12]
numberB = [3, -54, 232, 4, 76, 340]
```

```
The lowest different between two numbers is 1.
```