

CompSci 101 - Assignment 02

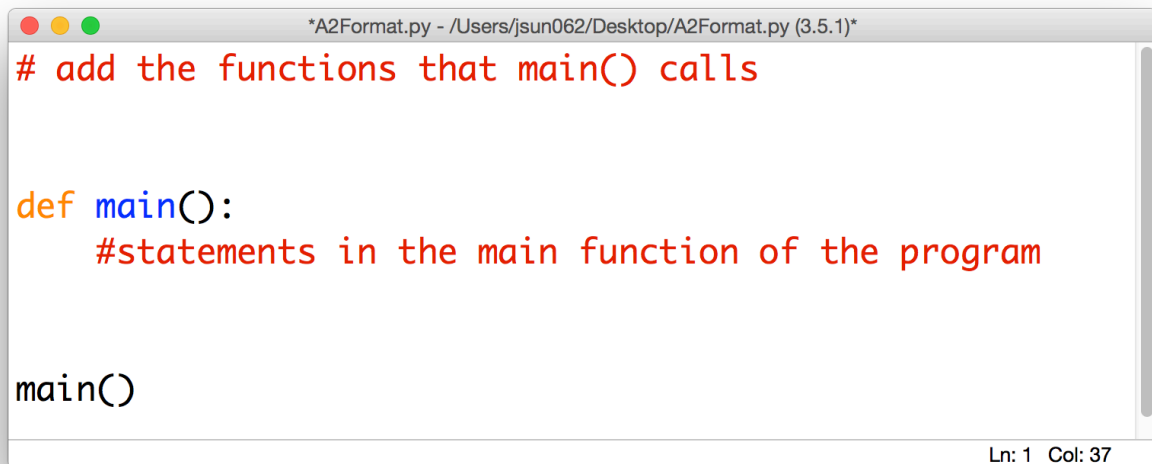
Due: 4:30pm, Tuesday 17 April 2018.

Worth: This assignment is marked out of 25 and is worth 2.5% of your final mark.

Topics covered:

- Functions
- If statements
- Loops (While or For)

NOTE (VERY IMPORTANT): Your programs for this assignment must all have the following format:



```
*A2Format.py - /Users/jsun062/Desktop/A2Format.py (3.5.1)*  
# add the functions that main() calls  
  
def main():  
    #statements in the main function of the program  
  
main()  
  
Ln: 1 Col: 37
```

You will be adding your own functions to the program. **There must not be any variables used outside any of the functions.**

NOTE (VERY IMPORTANT): Your file must include a docstring at the top of the file containing your name, UPI, ID number and a description of the program.

Submit the file containing your exercises using the Assignment Dropbox:

<https://adb.auckland.ac.nz/Home/>

QUESTION 1 (25 MARKS)

Write a program to print out any sized multiplication table between 1 to 20 (inclusive). A sample output (e.g., 10x10) is shown below.

```
Enter the size of the multiplication table (1-20): 10
-----
                        10x10 Times Table
          1      2      3      4      5      6      7      8      9      10
-----
1 |      1      2      3      4      5      6      7      8      9      10
2 |      2      4      6      8     10     12     14     16     18     20
3 |      3      6      9     12     15     18     21     24     27     30
4 |      4      8     12     16     20     24     28     32     36     40
5 |      5     10     15     20     25     30     35     40     45     50
6 |      6     12     18     24     30     36     42     48     54     60
7 |      7     14     21     28     35     42     49     56     63     70
8 |      8     16     24     32     40     48     56     64     72     80
9 |      9     18     27     36     45     54     63     72     81     90
10 |     10     20     30     40     50     60     70     80     90    100
-----
```

Note that the size of the multiplication table is obtained from the user input, i.e., an integer value between 1 and 20 (both inclusive). This means that the layout of the table varies according to the given size. For example, a 4x4 table and a 7x7 table will have the following different output.

```
Enter the size of the multiplication table (1-20): 4
-----
                        4x4 Times Table
          1      2      3      4
-----
1 |      1      2      3      4
2 |      2      4      6      8
3 |      3      6      9     12
4 |      4      8     12     16
-----
```

```
Enter the size of the multiplication table (1-20): 7
-----
                        7x7 Times Table
          1      2      3      4      5      6      7
-----
1 |      1      2      3      4      5      6      7
2 |      2      4      6      8     10     12     14
3 |      3      6      9     12     15     18     21
4 |      4      8     12     16     20     24     28
5 |      5     10     15     20     25     30     35
6 |      6     12     18     24     30     36     42
7 |      7     14     21     28     35     42     49
-----
```

Your program must give the correct output in the same format as the outputs in the examples.

The main() and display_heading() functions are given below and they should not be changed in any way. Copy this code into your program.

```
def display_heading(size, spaces):
    display_separator(size, spaces)
    title = str(size) + "x" + str(size) + " Times Table"
    print(get_word(title, (size+1)*spaces))
    print(get_row(0, size, spaces))
    display_separator(size, spaces)

def main():
    size = get_user_input()
    spaces = len(str(size*size))+2
    display_heading(size, spaces)
    display_table(size, spaces)
    display_separator(size, spaces)

main()
```

Note that the variable 'spaces' represents the available spaces in a column for displaying a number, which is set to the length of the maximum number (i.e., size*size) plus 2 in the main() function. Complete the following six functions so that the completed program runs as described above:

- `def get_number(num, spaces):`

This function returns a string representation of the parameter 'num' that is the same length (size) of the given 'spaces' value with the number aligned to the right. For example, if the parameter variable 'num' is 12 and 'spaces' is 6, the function returns a string value of "12", which has 4 empty spaces (where $4 = (6 - \text{len}("12"))$) on the left side of the string "12".

- `def get_word(words, spaces):`

This function returns a string representation of the parameter 'words' that is the same length (size) of the given 'spaces' value with the words aligned in the middle. For example, if the parameter variable 'words' is the string "hello" and 'spaces' is 15, the function returns a string value "hello", which has 5 empty spaces (where $5 = (15 - \text{len}("hello")) / 2$) on each side of the word "hello". Note that if the total number of empty spaces after subtracting the length of the words (from the given 'spaces' value) is an odd number, the left hand side of the words will have one more empty space than that of the right hand side. If the length of the words is longer than the value in 'spaces', the function returns the 'words' itself.

- `def get_row(num, size, spaces):`

This function returns the string representation of the content of one row in the table. The content of a row consists of two parts, i.e., the number value for the row followed by the "|" symbol and the actual values (i.e., multiplication results) in that row up to the size of the table. Note that the parameter variable 'num' stores the row value, the parameter 'size' stores the table size and the parameter 'spaces' represents the length of the available space available for printing each number in a column. If the 'num' value is 0, it represents the row in the table heading. Otherwise, it is a row in the content of the table, where a loop structure will be needed to append the string values in each column (by calling the get_number function) one by one to the resulting string.

- `def display_separator(size, spaces):`

This function prints out the separation line which consists of a row of “-” symbols. The number of “-” symbol printed is given by $(size+1)*spaces$.

- `def display_table(size, spaces):`

This function displays the content of the table by printing out each row (by calling the `get_row` function with the desired parameter values) one by one within a loop structure.

- `def get_user_input():`

This function obtains the size of the multiplication table from the user. You can assume that the user is always going to enter a positive integer value. However, you do need to check whether the integer value entered by the user is a valid number between 1 and 20 (both inclusive). If the input is outside that range, the function will print an error message and ask for the correct input, and it will continue to do so until a valid number is obtained. Finally, the function returns the user input size. A sample output with the error handling on a 13x13 table is shown below.

```
Enter the size of the multiplication table (1-20): 25
Table size should be between 1 and 20.
Please try again: 13
```

13x13 Times Table													
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	2	3	4	5	6	7	8	9	10	11	12	13
2	2	4	6	8	10	12	14	16	18	20	22	24	26
3	3	6	9	12	15	18	21	24	27	30	33	36	39
4	4	8	12	16	20	24	28	32	36	40	44	48	52
5	5	10	15	20	25	30	35	40	45	50	55	60	65
6	6	12	18	24	30	36	42	48	54	60	66	72	78
7	7	14	21	28	35	42	49	56	63	70	77	84	91
8	8	16	24	32	40	48	56	64	72	80	88	96	104
9	9	18	27	36	45	54	63	72	81	90	99	108	117
10	10	20	30	40	50	60	70	80	90	100	110	120	130
11	11	22	33	44	55	66	77	88	99	110	121	132	143
12	12	24	36	48	60	72	84	96	108	120	132	144	156
13	13	26	39	52	65	78	91	104	117	130	143	156	169

Include this exercise in a module (file), named 'YourUPIA2Q1.py', e.g., abcd001A2Q1.py.

Mark Allocation:

The mark allocations for each of the six functions to be implemented are as follows, where the total is 25.

- `def get_number(num, spaces)` – 3 marks
- `def get_word(words, spaces)` – 5 marks
- `def get_row(num, size, spaces)` – 5 marks
- `def display_separator(size, spaces)` – 3 marks
- `def display_table(size, spaces)` – 5 marks
- `def get_user_input()` – 4 marks