CSC1015F Assignment 4

Control (if, for, while)

Learning objectives

By the end of this assignment, you should be able to:

- Understand the need for iteration (otherwise known as looping) and the difference between the two major forms:
    - counter-controlled (for) and condition-controlled (while) loops.
- Be able to identify when and where to use the break, continue, else, and pass statements, which can be used to modify the behaviour of a loop structure.
- Be able to use for and while loops effectively in your programs.

Assignment Instructions

This assignment involves constructing Python programs that use input and output statements, '`if`' and '`if-else`' control flow statements, '`for`' statements, '`while`' statements and statements that perform numerical manipulation.

NOTE Your solutions to this assignment will be evaluated for correctness. Furthermore, your solutions will also be evaluated for the following qualities:

- Documentation
    - Use of comments at the top of your code to identify program purpose, author and date.
    - Use of comments within your code to explain each non-obvious functional unit of code.
- General style/readability
    - The use of meaningful names for variables and functions.
- Algorithmic qualities
    - Efficiency, simplicity

These criteria will be manually assessed by a tutor and commented upon. In this assignment, up to 10 marks will be deducted for deficiencies.

Question 1 [30 marks]

An Armstrong number (also known as narcissistic number) or digitally balanced number) is a number that is equal to the sum of its own digits each raised to the power of the number of digits.

Mathematical Definition:

A number *n* with *d* digits is an Armstrong number if:

$$n = \sum_{i=1}^{d} (digit_i)^d$$

where $digit_i$ represents each digit in the number.

Examples:

- 153 (3-digit number)

  $$1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$$

  Therefore, 153 is an Armstrong number.
- 9474 (4-digit number)

  $$9^4 + 4^4 + 7^4 + 4^4 = 6561 + 256 + 2401 + 256 = 9474$$

  Therefore, 9474 is an Armstrong number.

- 9475 (4-digit number)

  $$9^4 + 4^4 + 7^4 + 5^4 = 6561 + 256 + 2401 + 625 = 9843$$

  Therefore, 9475 is not an Armstrong number.

Write a program called '`armstrong.py`' that checks if a given number is an Armstrong number or not. Note that all the single-digit numbers are Armstrong numbers.

Sample IO *(The input from the user is shown in* ***bold*** *font – do not program this*):

```
Enter a number:
9474
9474 is an Armstrong number.
```

Sample IO *(The input from the user is shown in* ***bold*** *font – do not program this*):

```
Enter a number:
154
154 is not an Armstrong number.
```

Sample IO *(The input from the user is shown in* ***bold*** *font – do not program this*):

```
Enter a number:
ab
Invalid input. Please enter a valid integer.
```

Question 2 [30 marks]

Collatz Conjecture, also known as the `3x + 1 problem`, is a mathematical conjecture that applies to any positive integer *n*. The conjecture follows these rules:

1. If n is even, divide it by 2. That is, `n = n / 2`.
2. If n is odd, multiply it by 3 and add 1. That is, `n = 3n + 1`.
3. Repeat the process indefinitely.

The conjecture states that no matter what positive integer you start with, you will always eventually reach 1.

Example:
Starting with `n = 6`:

`6, 3, 10, 5, 16, 8, 4, 2, 1`

Write a program called '`collatz.py`' that accepts a positive integer and applies the rules above to eventually reach 1. Your program must give an appropriate message when a negative integer or zero is entered or when any other input other than an integer is supplied. (See the sample I/O below).

Sample IO *(The input from the user is shown in* ***bold*** *font – do not program this*):

```
Enter a positive integer:
15
15 46 23 70 35 106 53 160 80 40 20 10 5 16 8 4 2 1
```

Sample IO *(The input from the user is shown in* ***bold*** *font – do not program this*):

```
Enter a positive integer:
0
Please enter a positive integer.
```

Sample IO *(The input from the user is shown in* ***bold*** *font – do not program this*):

```
Enter a positive integer:
A
Invalid input. Please enter a valid integer.
```

Question 3 [40 marks]

Write a program called `rectanglegrid.py` that accepts a starting number, `n`, and two additional inputs: the number of rows (`r`) and columns (`c`). The program should print a grid of numbers, starting from `n`, with `r` rows and `c` columns.

Output will take the following form:

```
[d][d]d [d][d]d [d][d]d [d][d]d [d][d]d [d][d]d [d][d]d
[d][d]d [d][d]d [d][d]d [d][d]d [d][d]d [d][d]d [d][d]d
[d][d]d [d][d]d [d][d]d [d][d]d [d][d]d [d][d]d [d][d]d
[d][d]d [d][d]d [d][d]d [d][d]d [d][d]d [d][d]d [d][d]d
[d][d]d [d][d]d [d][d]d [d][d]d [d][d]d [d][d]d [d][d]d
[d][d]d [d][d]d [d][d]d [d][d]d [d][d]d [d][d]d [d][d]d
```

That is, numbers are printed using a field width of 3 and are right justified. ('`[d]`' represents an optional digit.) Fields are separated by a single space. There are no spaces after the final field.

Sample IO *(The input from the user is shown in* ***bold*** *font – do not program this*):

```
Enter the starting number (n):
2
Enter the number of rows (r):
6
Enter the number of columns (c):
9
  2   3   4   5   6   7   8   9  10
 11  12  13  14  15  16  17  18  19
 20  21  22  23  24  25  26  27  28
 29  30  31  32  33  34  35  36  37
 38  39  40  41  42  43  44  45  46
 47  48  49  50  51  52  53  54  55
```

Sample IO *(The input from the user is shown in* ***bold*** *font – do not program this*):

```
Enter the starting number (n):
88
Enter the number of rows (r):
6
Enter the number of columns (c):
8
 88  89  90  91  92  93  94  95
 96  97  98  99 100 101 102 103
104 105 106 107 108 109 110 111
112 113 114 115 116 117 118 119
120 121 122 123 124 125 126 127
128 129 130 131 132 133 134 135
```

Sample IO *(The input from the user is shown in* ***bold*** *font – do not program this*):

```
Enter the starting number (n):
78
Enter the number of rows (r):
-8
Enter the number of columns (c):
5
Rows and columns must be positive integers.
Enter the starting number (n):
56
Enter the number of rows (r):
6
Enter the number of columns (c):
-4
Rows and columns must be positive integers.
Enter the starting number (n):
78
Enter the number of rows (r):
5
Enter the number of columns (c):
8
 78  79  80  81  82  83  84  85
 86  87  88  89  90  91  92  93
 94  95  96  97  98  99 100 101
102 103 104 105 106 107 108 109
110 111 112 113 114 115 116 117
```

Sample IO *(The input from the user is shown in* ***bold*** *font – do not program this*):

```
Enter the starting number (n):
q
Invalid input. Please enter integers.
Enter the starting number (n):
2
Enter the number of rows (r):
w
Invalid input. Please enter integers.
Enter the starting number (n):
4
Enter the number of rows (r):
4
Enter the number of columns (c):
3
  4   5   6
  7   8   9
 10  11  12
 13  14  15
```

HINT: Use a '`for`' loop within a '`for`' loop for rows and columns. Use a while loop to accept input.

Submission

Create and submit a Zip file called '`ABCXYZ123.zip`' (where ABCXYZ123 is YOUR student number) containing `armstrong.py`, `collatz.py`, and `rectanglegrid.py`.

NOTES:

1. FOLDERS ARE NOT ALLOWED IN THE ZIP FILE.
2. As you will submit your assignment to the Automarker, the Assignment tab may say something like "Not Complete". THIS IS COMPLETELY NORMAL. IGNORE IT.