EMAT10007 – Introduction to Computer Programming

Exercises – Week 3. Loops and Data Structures

Part 1. Loops

To complete this week's exercises you will also need to download the `HowManySquares.py` file from Blackboard.

Exercise 1 - For Loops (Essential)

"For" loops are typically used when you know how many times you need to repeat something, before ending the loop. You specify a limit to the number of loops you wish to run the code for, and then the loop will automatically stop.

1. Create a variable and assign it a string value. Write a loop which now prints both each letter and its position in the string. **Hint:** The `len()` function returns the length of a string.
2. Find the mistake(s) in the following program, which is meant to sum the first 10 multiples of 5:

   ```
   total = 0
   for i in range(1,10)
   total = total + 5 * i
   ```

   Fix the program so that the final value of `total` is 275, and print the value of `total`.
3. Compute the factorial of 10.
   Recall that the factorial of an integer n is defined as $n! = n \times (n-1) \times (n-2) \times \ldots 2 \times 1$.
4. Using a `for` loop and the `break` keyword, determine how many positive cubic numbers are less than 2,000. Recall that cubic number is a number of the form n^3 where n is an integer.
5. In the game FizzBuzz, we count from 1 to n, replacing any multiple of 3 with the word "Fizz" and any multiple of 5 with the word "Buzz" As follows:

   "1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14, FizzBuzz, ...".

   - Create two variables `mult3` and `mult5`, setting their values to be the strings `"Fizz"` and `"Buzz"`, respectively.
   - Create an additional variable `limit`, which will be the number we count up to.
   - At the beginning of your program, after you have assigned `mult3` and `mult5` their values, you will need to ask the user to **input** a value for `limit`. This can be done using the `input()` function, which waits for the user to input some *string* when you run your program, before continuing.

     Hint: You will need to **convert** the input string created by `input()` into an integer, using `int()`.
   - The computer should say each number from 1 to `limit`, replacing each multiple of 3 with the word "Fizz" and each multiple of 5 with "Buzz". What kind of **loop** will you need for this?

     Hints:

     – Start with the basic loop, printing out each number, and then work on replacing it with "Fizz", "Buzz", or "FizzBuzz", in stages.

– You will also need to use the `%` operator, which returns the remainder of a division e.g. $4\%3 = 1$ and $15\%3 = 0$, indicating that 15 is a multiple of 3. You will need to check whether each number is either a multiple of 3, a multiple of 5, or *both*.

Exercise 2 - While Loops (Essential)

"While" loops are used when the number of times the program needs to loop is not known beforehand. The `while` loop checks a condition statement at the beginning of each loop and will carry out the loop as long as this condition is true.

1. Can you finish the `while` loop by replacing the `<?>` in the code below?
   Note: `<?>` is used as a placeholder to represent something missing - this is an operation with an outcome of `True` or `False`. Multiple `<?>` in the same question are not necessarily representing the same thing.

   ```
   word = "Hello World"
   target_letter = <?>
   i = 0
   while <?>:
       i += <?>
   print("Target letter is at position", i)
   ```

   Try changing `word` and `target_letter`. Why do we have to increment the `i`?
2. How would you change the code to print all the occurrences of the letter?
   Hint: What type of loop would be best?
3. Finding characters or substrings in a string is very useful and so Python has a built-in function `str.find()`. Test your program is right by comparing with the `str.find()` method for some different target letters and words.
   (**Hint:** You may find the `help()` function useful for looking up methods that can be used for different object types. Running `help(str)` will return information on methods that can be used with strings.)
4. Open and run the `HowManySquares.py` program. Can you change the code to use the `+=` operator?
5. In the `HowManySquares.py` example we do not know when the cumulative sum will exceed $1,000,000$. Therefore, we use a `while` loop until the sum exceeds this limit, and therefore the condition no longer evaluates to `True`. Can you add a line that shows the results for every step of the `while` loop?
6. Can you rewrite Q1 using a `for` loop, an `if` statement and `enumerate()`?
   Hint: research the `enumerate` function. To get a feel for how this works, run the following code

   ```
   s = 'hello'
   enum = enumerate(s)
   print(list(enum))
   ```

7. Write a program that contains a list of numbers assigned to a variable. Your program should output the mean of the list of numbers for a list of any length.

Exercise 3 - Modelling using Loops (Essential)

1. Write a program that calculates how many years it would take for the value of a savings account to exceed £400, if the initial (and only) deposit made is £100 and the annual interest is 5%.
2. A ball is dropped (initial velocity $u = 0$ ms^{-1}) and falls towards the ground with an acceleration due to gravity of 9.81 ms^{-2}. It is assumed that no other forces act on the ball so the distance travelled by the ball, d (m), at time t (s), can be found by:

   $$d = ut + \frac{1}{2}at^2$$

   Print the distance from the start position the ball has fallen at 0.2 s intervals for 2 s, assuming the ball does not reach the ground within this time.
3. The Fibonacci sequence is a sequence of numbers where each number is the sum of the two preceding ones:

   $$f_n = f_{n-1} + f_{n-2}$$

   The first two numbers in the Fibonacci sequence are 0 and 1:

   $$f_0 = 0, f_1 = 1$$

   The sequence therefore starts as:

   0, 1, 1, 2, 3, 5, 8, ...

   (some sources omit the initial 0 and begin the sequence 1, 1 ...) Each number in this sequence is called a Fibonacci number.
   Fibonacci numbers are used in the analysis of financial markets (e.g. Fibonacci retracement) and computer algorithms (e.g. Fibonacci search technique, Fibonacci heap data structure) and is found in the patterning of many biological systems (e.g. Nautilus shells, sunflowers and pine cones).

   (A) Write a program that finds and prints the first 10 Fibonacci numbers.

   (B) How many Fibonacci numbers are less than (i) 100, (ii) 1,000, (iii) 10,000?
4. A prime number is a natural number greater than 1 that is not a product of two smaller natural numbers. In other words, a prime number cannot be written as a product of two natural numbers that are both smaller than it. Write a program that prints all prime numbers between 1 and 150.
   Hint Remember the modulo operator '%' gives the remainder when one number is divided by another.

Exercise 4 - More Loops (Advanced)

Use loops and conditionals to solve the following problems:

1. Use two `for` loops to compute the double sum

   $$S = \sum_{i=1}^{10} \sum_{j=0}^{5} j^2(i + j)$$

2. Write a program to print the following star pattern:

```
*
**
***
****
*****
****
***
**
*
```

3. The value of π can be approximated using the Leibniz formula:

$$\pi_N = \sum_{n=0}^{N} \frac{8}{(4n+1)(4n+3)}$$

where N is a large number. Taking the limit as $N \to \infty$ produces the exact value of π, but this requires evaluating an infinite number of terms, which is impossible on a computer. Therefore, we can only approximate the value of π by using a finite number of terms in the sum. Use this formula to compute approximations to π by taking $N = 100$, $N = 1,000$, and $N = 10,000$.

Part 2. Data structures

Exercise 5 - Lists (Essential)

1. Make two lists containing the values `[1,2]` and `[3,4]`.
2. Change the value 1 to the value 5.
3. Make a nested list that contains both lists.
4. Use two loops to print out all the values in the nested list (2x2 matrix) one by one.
5. Write a program that asks the user to input a list of 10 words (strings) and then creates a list containing the length of each word. Print out each word and word length, like so:

   `Word:  Algorithm - Word length:  9`

   Hint: You can read all 10 words at a time, as one large string, and use the `.split()` method on the string. The output will be a list of words as individual strings. Loop through the resulting list of words and print out the length of each word.
6. Using list comprehension, create lists of the following between 0 and 100:
   - odd numbers
   - multiples of 3
   - prime numbers (NB: this is quite tricky and can be considered an advanced question)

Exercise 6 - Tuples (Essential)

1. Make a tuple named `fondue_ingredients` containing the values "gruyere" and "vacherin".

2. Print all the items in the tuple.
3. Change the value "gruyere" to the value "cheddar". Does it work? Why?
   Note: Fondue recipes are sacred.
4. Is there a function to remove the last item of the tuple? How else could you do it?

Exercise 7 - Sets (Essential)

1. Make two set `s1 = {1,2,5,5,8}` and `s2 = {1,2,4,9,2}`. Print out the sets, are there any duplicates?
2. Can you access an element of the set based on index e.g. `s1[2]`?
3. Use the keyword in to check if 4 is in both sets.
4. Use the operators `&`, `|`, `-`, `^`. What do they do?
5. Remove the value 1 from the first set, and add the value 6.

Exercise 8 - Dictionaries (Essential)

1. Make a dictionary that contains {`"Xiaohan":21, "Christian":20, "Grace":20, "Sajid":21`}. Remember to give it a sensible name
2. Print out all the keys in the dictionary.
3. Add the item `"Nicola":19` to the dictionary.
4. Remove the item `Grace"`.
5. Add the item `"Sajid":22` to the dictionary. Are there two instances of Sajid now?
6. Check if `"Harry"` is in the dictionary.

Exercise 9 - Modelling using data structures (Essential)

Table 1 shows some data about the planets in our solar system. The mass of each planet is shown as a factor which when multiplied by the mass of Earth gives the actual mass of the planet in kg. The mass of Earth can be estimated as 5.9722×10^{24} kg. The rotation period is given in units of days (d) or hours (h).

1. Write a program that identifies and outputs the names and rotation periods of planets with a rotation period shorter than Earth's.

Exercise 10 - More data structures (Advanced)

1. Create a list of numbers and research how to sort the list so the numbers are in ascending order.
   Can you, instead output a list with the numbers sorted in descending order?

Planet	Diameter (km)	Mass	Rotation period (h)
Mercury	4,878	0.06	58.65
Venus	12,100	0.82	243
Earth	12,756	1.00	23.934
Mars	6,794	0.11	24.623
Jupiter	142,800	317.89	9.842
Saturn	120,000	95.17	10.233
Uranus	52,400	14.56	16
Neptune	48,400	17.24	18
Pluto	2,445	0.002	6.39

Table 1: Planet data taken from: https://www.rmg.co.uk/stories/topics/solar-system-data