# Machine Learning Workshop 1 – CSE 2793

## MINOR ASSIGNMENT-3: FUNCTIONS AND LOOPS

Q 01 Write function **`calc_primes_up_to(max_value)`** to compute all prime numbers up to a given value. As a reminder, a prime number is a natural number greater than 1 and exclusively divisible by itself and by 1.
Check your algorithm with the following values:

| Input | Result |
|---|---|
| 15 | [2, 3, 5, 7, 11, 13] |
| 25 | [2, 3, 5, 7, 11, 13, 17, 19, 23] |
| 50 | [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47] |

Q 02 Create function **`calc_checksum(digits)`** that performs the following position-based calculation for the checksum of a number of any length given as a string, with the $n$ digits modeled as $z_1$ to $z_n$:

$$z_1 z_2 z_3 \dots z_n \Rightarrow (1 * z_1 + 2 * z_2 + 3 * z_3 + \cdots + n * z_n)\%10$$

Check your algorithm with the following values:

| Input | Sum | Result |
|---|---|---|
| "11111" | 1 + 2 + 3 + 4 + 5 = 15 | 15 % 10 = 5 |
| "87654321" | 8 + 14 + 18 + 20 + 20 + 18 + 14 + 8 = 120 | 120 % 10 = 0 |

Q 03 Compute all combinations of the values a, b, and c (each starting from 1 and less than 100) for which the following formula holds:

$$a^2 + b^2 = c^2$$

**Bonus:**

Reduce the running time of $O(n^3)$ to $O(n^2)$.

Q 04 This exercise deals with three-digit Armstrong numbers. By definition, these are numbers for whose digits $x, y,$ and $z$ from 1 to 9 satisfy the following equation:

$$x * 100 + y * 10 + z = x^3 + y^3 + z^3$$

Write function **`calc_armstrong_numbers()`** to compute all Armstrong numbers for $x$, $y,$ and $z$ (each $< 10$).

Q 05 Two numbers $n_1$ and $n_2$ are called friends (or related) if the sum of their divisors is equal to the other number:

$$sum(divisors(n_1)) = n_2$$
$$sum(divisors(n_2)) = n_1$$

Write function **`calc_friends(max_exclusive)`** to compute all friends numbers up to a passed maximum value.

| Input | Divisors |
|---|---|
| ∑(*divisors*(220)) = 284 | div(220) = 1, 2, 4, 5, 10, 11, 20, 22, 44, 55, 110 |
| ∑(*divisors*(284)) = 220 | div(284) = 1, 2, 4, 71, 142 |
| ∑(*divisors*(1184)) = 1210 | div(1184) = 1, 2, 4, 8, 16, 32, 37, 74, 148, 296, 592 |
| ∑(*divisors*(1210)) = 1184 | div(1210) = 1, 2, 5, 10, 11, 22, 55, 110, 121, 242, 605 |

Q 06

a. Write function **`gcd(a, b)`** that computes the greatest common divisor (GCD) using iteration.
Implement the above using Euclid's algorithm, which is an iterative computation based on the following observation:
if x is greater than y, then if y divides x, the gcd of x and y is y;
otherwise,
the gcd of x and y is the same as the gcd of x % y and y.

b. Write function **`lcm(a, b)`** that computes the greatest common divisor (GCD) using iteration.

$$\text{lcm} = \frac{a * b}{\text{gcd}\ (a, b)}$$

**`GCD`**

| Input 1 | Input 2 | Result |
|---|---|---|
| 42 | 7 | 7 |
| 42 | 28 | 14 |
| 42 | 14 | 14 |

**`LCM`**

| Input 1 | Input 2 | Result |
|---|---|---|
| 2 | 7 | 14 |
| 7 | 14 | 14 |
| 42 | 14 | 42 |

Q 07 A perfect number is one whose divisors add up to the number.
Example: The first perfect number is 6, because 1, 2, and 3 are its proper divisors, and 1+2+3=6
Write a Python program that prints all perfect numbers in between 1 and 500.

Q 08 Write function **`to_binary(n)`** that iteratively converts the given positive integer into a textual binary representation. No call to **`int(x, base)`** may be used.

| Input | Result |
|---|---|
| 5 | "101" |
| 7 | "111" |
| 22 | "10110" |
| 42 | "101010" |
| 256 | "100000000" |

Q 09 Write conversions to octal and hexadecimal numbers by implementing the corresponding functions **`to_octal(n)`** and **`to_hex(n)`**. Again, no call to **`int(x, base)`** may be used.

| Input | Method | Result |
|---|---|---|
| 7 | octal | "7" |
| 8 | octal | "10" |
| 42 | octal | "52" |
| 15 | hexadecimal | "F" |
| 77 | hexadecimal | "4D" |

Q 10 A palindrome is a word that reads the same from the front and the back. You can extend this definition to the digits of a number. Write an iterative function `is_number_palindrome(number)` but without converting the number into a string and then using string functionalities like `[::-1]`.

Q 11 Write a Python program called `functionGrowth()` that prints a table of the values log N, N,N log N, $N^2$, $N^3$, and $2^N$ for N = 16, 32, 64, ..., 2048. Use tabs (`\t` characters) to line up columns.

Q 12 Write the Python programs to print the following four patterns using for loop using four different programs.

| (a) | (b) | (c) | (d) |
|---|---|---|---|
| * | 1 | 1 | 1 |
| * * | 1 2 | 2 2 | 2 3 |
| * * * | 1 2 3 | 3 3 3 | 4 5 6 |
| * * * * | 1 2 3 4 | 4 4 4 4 | 7 8 9 10 |
| * * * * * | 1 2 3 4 5 | 5 5 5 5 5 | 11 12 13 14 15 |

Q 13 Write a Python program that displays all the numbers from 100 to 1,000, ten per line, that are divisible by 5 and 6. Numbers are separated by exactly one space.

Q 14 Write a Python program to evaluate the function $sin(x)$ and $cos(x)$ as defined by the infinite series expansions.

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \ldots$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} \ldots$$

The acceptable error for computation is $10^{-6}$.

Q 15 Assume that $x$ is a positive variable of type double. Write a Python program that uses the Taylor series expansion to set the value of sum to

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots\ldots$$