**Numerals Systems**

Numeral systems are useful because they provide a way to represent numbers in a way that is easily understandable and computable by humans and machines. They allow us to represent numbers using a finite set of symbols or digits, which simplifies mathematical operations, comparisons, and other calculations.

**Project:**

The code in this project allows to convert numbers between different numeral systems - including binary, decimal and hexadecimal.

Overall, this code provides a useful set of tools for working with different numeral systems in Python, and could be a valuable resource for anyone working on related projects.

This code defines a class called NumeralsSystems, which can be used to convert between different numeral systems. The class has six options for conversion:

- from decimal to binary numeral system

- from hexadecimal to binary numeral system

- from binary to decimal numeral system

- from hexadecimal to decimal numeral system

- from binary to hexadecimal numeral system

- from decimal to hexadecimal numeral system

The class takes two arguments, an option and a number. The option is a string that represents which conversion should be performed, and the number is either an integer or a string, depending on the option.

The class has getter and setter methods for the option and number attributes. The setters validate the input and raise a ValueError if the input is invalid.

The class also has a method called set\_convert\_number, which calculates the converted number based on the option and number inputs. The conversion is performed using private methods that are specific to each conversion option.

Overall, this code provides a flexible and extensible solution for performing numeral system conversions in Python.

The code includes comments to help explain how it works. It is designed to be easy to use, with clear input and output formats for each function.

**Unit Testing**

The code defines several test cases for testing the methods of the NumeralsSystems class using Python's built-in unittest module. The tests include checking the initialization of the class, setting and getting the option and number variables, and handling value errors for invalid inputs.