Complex problems: search, sort

Objectives

Using Python to solve complex problems

- Develop abstract data types
- Develop layered applications
- Implement search and sort methods
- Learn how to use lambda expressions in Python

Requirements

1. Given a list of numbers, search the list in order to:
    a. Determine all numbers from the list that are Armstrong numbers.
    Note: If sum of cubes of each digit of the number is equal to the number itself, then the number is called an Armstrong number.
    Example: 153 = 1^3 + 5^3 + 3^3
    b. Determine all numbers from the list that are even and Armstrong numbers.
    c. Determine all numbers from the list that are even/primes/perfect squares/ Armstrong numbers using a single algorithm with correct parameters.

2. Sort a list of numbers using Bubble Sort / Selection Sort / Insertion Sort / Quick Sort.

3. Develop an ADT *GeometricalShape* – information about name (e.g. square, triangle, hexagon), number of sides and length of each side.
    a. Create a repository to manage a list of shapes.
    b. Filter shapes with more than k sides.
    c. Filter shapes with perimeter higher than a given value and name of given length.
    d. Sort shapes based on the perimeter ascending / descending.
    e. Sort shapes having name starting with given letter according to perimeter.

    Use a single search/sort algorithm for all requirements.
    Use both own search/sort algorithm and Python functions filter and sorted.