Practical 04 & 05
Sort

1. Selection Sort

The following code is an implementation of the Selection Sort algorithm that sorts a sequence of numbers in ascending order.

```
# Sort a sequence in ascending order using the selection sort
# algorithm
def selectionSort( theSeq ):
    n = len( theSeq )

    for i in range(n - 1):
        # Assume the ith element is the smallest.
        smallNdx = i
        # Determine if any other element contains a smaller value.
        for j in range(i+1, n):
            if theSeq[j] < theSeq[smallNdx]:
                smallNdx = j

        # Swap the ith value and smallNdx value only if the
        # smallest value is not already in its proper position.
        if smallNdx != i:
            tmp = theSeq[i]
            theSeq[i] = theSeq[smallNdx]
            theSeq[smallNdx] = tmp


# Test codes
list_of_numbers = [10, 51, 2, 18, 4, 31, 13, 5, 23, 64, 29]

print('Input List:', list_of_numbers)
selectionSort(list_of_numbers)
print('Sorted List:', list_of_numbers)
```

The output of the above code is as follows:

```
Input List: [10, 51, 2, 18, 4, 31, 13, 5, 23, 64, 29]
Sorted List: [2, 4, 5, 10, 13, 18, 23, 29, 31, 51, 64]
```

Modify the function – `selectionSort()`, to take in an additional parameter `sortOrder` (e.g. with a value 'A' to represent ascending, and 'D' for descending), that determines whether to sort the list of numbers in ascending or descending order. A sample output of the modified program can look like the following:

```
Sorting in ascending order:
Input List: [10, 51, 2, 18, 4, 31, 13, 5, 23, 64, 29]
Sorted List: [2, 4, 5, 10, 13, 18, 23, 29, 31, 51, 64]

Sorting in descending order:
Input List: [2, 4, 5, 10, 13, 18, 23, 29, 31, 51, 64]
Sorted List: [64, 51, 31, 29, 23, 18, 13, 10, 5, 4, 2]
```

2. Insertion Sort

The following code is an implementation of the Insertion Sort algorithm that sorts a sequence of numbers in ascending order.

```
# Sorts a sequence in ascending order using the insertion sort
# algorithm
def insertionSort( theSeq ):
    n = len( theSeq )

    # Starts with the first item as the only sorted entry.
    for i in range(1, n):
        # Save the value to be positioned
        value = theSeq[i]

        # Find the position where value fits in the
        # ordered part of the list.
        pos = i
        while pos > 0 and value < theSeq[pos - 1]:
            # Shift the items to the right during the search
            theSeq[pos] = theSeq[pos-1]
            pos -= 1

        # Put the saved value into the open slot.
        theSeq[pos] = value

# Test codes
list_of_numbers = [10, 51, 2, 18, 4, 31, 13, 5, 23, 64, 29]

print('Input List:', list_of_numbers)
insertionSort(list_of_numbers)
print('Sorted List:', list_of_numbers)
```

The output of the above code is as follows:

```
Input List: [10, 51, 2, 18, 4, 31, 13, 5, 23, 64, 29]
Sorted List: [2, 4, 5, 10, 13, 18, 23, 29, 31, 51, 64]
```

Modify the function – `insertionSort()`, to print out the values of the list during each and every pass of the algorithm. A sample output of the modified program can look like the following:

```
Insertion Sort
Pass:  0      [10, 51, 2, 18, 4, 31, 13, 5, 23, 64, 29]
Pass:  1      [10, 51, 2, 18, 4, 31, 13, 5, 23, 64, 29]
Pass:  2      [2, 10, 51, 18, 4, 31, 13, 5, 23, 64, 29]
Pass:  3      [2, 10, 18, 51, 4, 31, 13, 5, 23, 64, 29]
Pass:  4      [2, 4, 10, 18, 51, 31, 13, 5, 23, 64, 29]
Pass:  5      [2, 4, 10, 18, 31, 51, 13, 5, 23, 64, 29]
Pass:  6      [2, 4, 10, 13, 18, 31, 51, 5, 23, 64, 29]
Pass:  7      [2, 4, 5, 10, 13, 18, 31, 51, 23, 64, 29]
Pass:  8      [2, 4, 5, 10, 13, 18, 23, 31, 51, 64, 29]
Pass:  9      [2, 4, 5, 10, 13, 18, 23, 31, 51, 64, 29]
Pass:  10     [2, 4, 5, 10, 13, 18, 23, 29, 31, 51, 64]
```

3. Sorting with Python built-in function – `sorted()`

   a. Write a function that takes in a list of strings and return a list with the strings in sorted order, except group all the strings that begin with 'H' first.

      For example, providing the following input list:
      - ['Bougainvillea', 'Orchids', 'Hibiscus', 'Frangipani', 'Honeysuckle']

      will return the following output list:
      - ['Hibiscus', 'Honeysuckle', 'Bougainvillea', 'Frangipani', 'Orchids']

      (**HINT**: Try making 2 lists and sorting each of them before combining them.)

   b. Write a function that takes in a list of non-empty tuples and return a list sorted in increasing order by the last element in each tuple.

      For example, providing the following input list:
      - [(1, 7), (1, 3), (3, 4, 5), (2, 2)]

      will return the following output list:
      - [(2, 2), (1, 3), (3, 4, 5), (1, 7)]

      (**HINT**: Try using a custom `key=` function to extract the last element form each tuple.)

-- End of Practical --