FIT1045 Algorithmic Problem Solving – Workshop 5.

Objectives

The **objectives of this workshop** are:

- To understand selection sort.
- To implement algorithms for sorting a list.
- To construct a solution to a problem by decomposition into smaller problems.
- To implement functions in Python.

Useful Links

For this workshop, you may find it useful to review some of the following concepts:

- Using files (Lecture 4)
- Loops (Lecture 2)
- Lists (Lecture 3)
- Insertion Sort (Lecture 4)
- Selection Sort (Lecture 4)
- Function decomposition (Lecture 3)
- Some string operations (`https://docs.python.org/3/library/stdtypes.html#string-methods` or Perkovic book, page 97–98.)
- Functions (`https://docs.python.org/3/tutorial/controlflow.html#defining-functions`)

Task 0 (Prelab), to be completed before class:

Write a Python function named `printElements` that iterates over a list and uses the string `format` function (see the useful links above) to print each element alongside its index in the format shown below.

For example: if your input list is [7, 3, 5, 4] then the function should print:

```
At index 0 is the element 7.
At index 1 is the element 3.
At index 2 is the element 5.
At index 3 is the element 4.
```

Now create a `selectionSort` function based on the code for Selection Sort given in Lecture 8 (Greedy Approach) Slide 7. Use `printElements` and `selectionSort` in a new function named `printAscending` that accepts a list as input, makes a copy of it by slicing it (recall the x[a:b] syntax), then sorts and prints the copied list with its elements in ascending order. Make sure that within `printAscending` you are calling the printing and sorting functions you have just created: do NOT copy and paste code.

For example: with the same input list as above, `printAscending` should print:

```
At index 0 is the element 3.
At index 1 is the element 4.
At index 2 is the element 5.
At index 3 is the element 7.
```

Before you continue on to the rest of the workshop, make sure to grab the text files from Moodle that we will be using for the following tasks. They can be found in the "data files" folder under Week 7.

Task 1:

Write a Python program that asks the user for the name of a file containing information about the distance a vendor is from Monash and the cost of an item at that vendor. Each line in the file consists of a distance (as an integer) and an item price (as a float) separated by a comma.

For example: the file Tiny.txt contains:

```
130,266.07
46,174.14
169,187.01
179,488.69
53,401.53
128,106.88
97,398.33
152,493.87
20,205.43
94,248.14
```

Your program should create a list of distance/cost pairs based on the file contents. It should print the list, so that you can check that your program works correctly.

For example: your program may do the following:

```
Enter name of file:  Tiny.txt
[[130, 266.07], [46, 174.14], [169, 187.01], [179, 488.69], [53, 401.53], [128, 106.88],
[97, 398.33], [152, 493.87], [20, 205.43], [94, 248.14]]
```

Improving Readability

In order to improve the readability of the output, write a function `printList` that takes as input a list containing distance/cost pairs and prints one pair on each line in the format shown below. Use this function to print the list after it is read from the file.

For example: your program may output the following:

```
130 kms, $266.07
46 kms, $174.14
169 kms, $187.01
179 kms, $488.69
53 kms, $401.53
128 kms, $106.88
97 kms, $398.33
152 kms, $493.87
20 kms, $205.43
94 kms, $248.14
```

Task 2

You would like to find a vendor close to Monash. Write a function `selectionSortDistance` that takes as input a list of distance/cost pairs and sorts the list in increasing order of distance using **Selection Sort**. Use this function to modify your program from Task 1, so that the list is printed in increasing order of distance.

For example: your program may do the following:

```
Enter name of file:  Tiny.txt
20 kms, $205.43
46 kms, $174.14
53 kms, $401.53
94 kms, $248.14
97 kms, $398.33
128 kms, $106.88
130 kms, $266.07
152 kms, $493.87
169 kms, $187.01
179 kms, $488.69
```

Task 3

You would like to be able to sort by price so that you can easily compare the cheapest price of the item compared to the prices of vendors close to Monash. Write a function `selectionSortPrice` that takes as input a list of distance/cost pairs and sorts the list in increasing order of price using **Selection Sort**. Use this function to modify your program from Task 2, so that the list is printed in increasing order of price.

For example: your program may do the following:

```
Enter name of file: Tiny.txt
128 kms, $106.88
46 kms, $174.14
169 kms, $187.01
20 kms, $205.43
94 kms, $248.14
130 kms, $266.07
97 kms, $398.33
53 kms, $401.53
179 kms, $488.69
152 kms, $493.87
```

Task 4

Modify your program from Task 3, so that it repeatedly asks the user to select from the following options:

- Print - prints the list in the order it is currently sorted (or unsorted order if it has not yet been sorted).
- Sort1 - sorts the current list in ascending order of distance.
- Sort2 - sorts the current list in ascending order of price.
- Quit - program stops.

For example: your program may do the following:

```
Enter name of file:  Tiny.txt
Enter choice Print, Sort1 (distance), Sort2 (price) or Quit:  Print
130 kms, $266.07
46 kms, $174.14
169 kms, $187.01
179 kms, $488.69
53 kms, $401.53
128 kms, $106.88
97 kms, $398.33
152 kms, $493.87
20 kms, $205.43
94 kms, $248.14
Enter choice Print, Sort1 (distance), Sort2 (price) or Quit:  Sort1
Enter choice Print, Sort1 (distance), Sort2 (price) or Quit:  Print
20 kms, $205.43
46 kms, $174.14
53 kms, $401.53
94 kms, $248.14
97 kms, $398.33
128 kms, $106.88
130 kms, $266.07
152 kms, $493.87
169 kms, $187.01
179 kms, $488.69
Enter choice Print, Sort1 (distance), Sort2 (price) or Quit:  Sort2
Enter choice Print, Sort1 (distance), Sort2 (price) or Quit:  Print
128 kms, $106.88
46 kms, $174.14
169 kms, $187.01
```

```
20 kms, $205.43
94 kms, $248.14
130 kms, $266.07
97 kms, $398.33
53 kms, $401.53
179 kms, $488.69
152 kms, $493.87
Enter choice Print, Sort1 (distance), Sort2 (price) or Quit:  Quit
```