Selection Sort
Lesson 00

CSC140 Foundations of Computer Science

10 February 2020

Here is the first and most important lesson of our course:

Programmers always write for two audiences.

1. Programmers write instructions for a machine.

2. Programmers write for teammates, clients, and other people. They write not just for their current teammates but also for future teammates and for those programmers who will replace them when they leave a project.

This program is incomplete. It contains code written in the Python programming language but no explanation of its purpose or methods. We will write an explanation of the purpose and methods in English. We will add our explanation to the program today.

Here is a second important lesson:

Programmers solve big problems by dividing them into smaller problems.

Imagine if Leo Tolstoy had not divided *War and Peace* into chapters. Imagine if the one chapter contained only a single paragraph and the paragraph contained only one sentence. Who could read such a book?

Dividing a big book into parts not only makes reading the book easier, it also makes writing the book easier.

Software engineers similarly make their programs easier to read, write, and understand by dividing them into parts. The parts in this program are functions. Each function begins with the word **def**.

By copying this code, you will "learn through your fingers." You will make mistakes, but, because you will have the correct code in front of yourself and because we will be working together, you will be able to fix your errors without

too much trouble. Typing the code will help you begin to get a feel for programming and for the programmer's habit of building big things by assembling small parts.

Learning to program computers is like learning to play a sport or learning to play a musical instrument. Practice is essential and mistakes are unavoidable. You will learn by doing. You will learn from your mistakes.

In today's exercise, you will test each function. In this way, you will gain an understanding of how the program works. Along the way, we will ask some quesions:

- Can we be sure that a program does what we want it to do?
- How sure?
- How can we gain the greatest confidence?

Today's lesson is an **exit ticket**. You must understand all parts of this lesson thoroughly by the end of the term. You must be able to reproduce this program.

An Example

```
import numpy as np

def find_min( data ):
    best_guess_so_far = data[0]
    for i in range(1, len(data) ):
        if data[i] < best_guess_so_far:
            best_guess_so_far = data[i]
    return best_guess_so_far

def find_pos_min( data ):
    best_guess_so_far = 0
    for i in range(1, len(data) ):
        if data[i] < data[best_guess_so_far]:
            best_guess_so_far = i
    return best_guess_so_far

def find_pos_min_in_suffix( data, start ):
    best_guess_so_far = start
    for i in range( start + 1, len(data) ):
        if data[i] < data[best_guess_so_far]:
            best_guess_so_far = i
    return best_guess_so_far
```

```
def swap( data, i, j ):
    temp = data[i]
    data[i] = data[j]
    data[j] = temp

def select_sort( data ):
    for i in range( len(data) ):
        j = find_pos_min_in_suffix( data, i )
        swap( data, i, j )

if __name__ == '__main__':
    values = [np.random.randint(10, 100) for i in range(12)]

    print( values )

    print( "minimum value = ", find_min(values) )

    print( "index of minimum value = ", find_pos_min( values ) )

    select_sort( values )

    print( values )
```