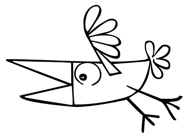
Exploring Sorting



First, download the file sorting.py from our website. The file contains a working version of Bubble Sort.

Here is how to test it.

```
>>> a = make_a_list(8)
>>> a
[64, 72, 94, 33, 84, 51, 32, 20]
>>> bubblesort(a)
>>> a
[20, 32, 33, 51, 64, 72, 84, 94]
>>>
```

So, in this example I assign the variable a to be equal to a list of length 8. I then call Bubble Sort on a. It actually changes the value of a. So when bubblesort is completed, list a is sorted.

Task 1. Answers to Questions
Your first task is to answer the four questions mentioned in the sorting.py file.

Task 2. Selection Sort
You are to implement Selection Sort.

Task 3. Merge -- merges 2 sorted lists
You are to implement merge which takes two sorted lists as arguments. For example if a and b are the following lists:

```
>>> a
[20, 32, 33, 51, 64, 72, 84, 94]
>>> b
[45, 56, 57, 61, 68, 68, 75, 80]
>>>
```

then merge(a, b) will return:

```
>>> merge(a, b)
[20, 32, 33, 45, 51, 56, 57, 61, 64, 68, 68, 72, 75, 80, 84, 94]
```

I have the start of that function in sorting.py. Simply delete the occurrences of print('TODO') and replace it with the correct code. Alternatively, you can write the function from scratch.

Task 4: Merge Sort
Implement the mergesort sort method described in class. For this sort method, it is easier to return a sorted list, rather than sort the list in place.

Task 5: Timing

Do some experiments and fill in the following table with the number of seconds it takes to execute the following:

	Numbers to sort		
	1024	**5120**	**10240**
Bubble Sort			
Selection Sort			
Merge Sort			

Task 6
Explain the results from Task 5. Do they agree with what we would expect given what we know about their big O values?

SUBMITTING
Submit this sheet with the answers for Task 1, Task 5, and Task 6 and the file sorting.py with the code for the other tasks to submit.o.bot_AT_gmail_DOT_com.