As you enter: Choose a better name for each of the four functions below. Each function is run with a single list as its input, e.g. M1([3, 9, 0, 14]) [L1]

```
def M1(lst):
  smallest = lst[0]
  for elem in lst:
    if elem < smallest:
      smallest = elem
  return smallest

def M2(lst):
  running_sum = 0
  for elem in lst:
    running_sum += 1
  return running_sum
```

```
def M3(lst):
  saved = lst[0]
  for elem in lst:
    if elem > saved:
      saved = elem
  return saved

def M4(lst):
  running_product = 1
  for elem in lst:
    running_product *= elem
  return product
```

FUNCTIONS II (WRITING)

ADMIN STUFF

- Reminder that HW2: Personality Quiz Project due 9/28
  - Presentations after Fall Break (but sign up by 9/28!)
- It is now time to think about scheduling accomodations through Weingarten for Midterm 1. I cannot do this for you or help you if you forget.
- Midterm 1 is two weeks from today

What are functions?

Functions are named groups of statements/code

- Functions have names, which we can use to call them
- Functions have inputs, which we can use to...
  - control the specific way that they behave on different calls
  - process different information with them on different calls
- Functions can (but don't have to!) return values

```
result = max(17, 34)
```

`max` is the name of a function. We can call it like `max(17, 34)`, which passes in `17` and `34` as its two inputs. The value returned by this function call is `34`, which is saved in `result`

ANATOMY OF A FUNCTION

- **Function definitions** consist of the function's signature as well as a block of statements called its **body**
  - A **function signature** consists of:
    - the function's name
    - the list of parameters that it takes as input.

Anatomy of a Function

```
def multiply_two_numbers(a, b):
    print(f"Multiplying {a} x {b}!")
    product = a * b
    return product
```

The **signature**:

```
def multiply_two_numbers(a, b):
```

The **body**:

```
    print(f"Multiplying {a} x {b}!")
    product = a * b
    return product
```

PRACTICE [C1]

What's wrong with the following code? Re-write the code and fix it so it works.

```
def multiply(x, y, z):
  total = x * y * z
  print(str(total))



multiply(5, 2.5, 3)
print("5 * 2.5 * 3 = " + str(total))
```

HOW DO I WRITE MY OWN FUNCTIONS?

Step 1: Identify what you want it to do

Step 2: Write some code that does what you want it to do

Step 3: Write a signature that describes what you want to do

Step 4: Combine!

ACTIVITY: WORKING TOWARDS WRITING A FUNCTION

Assuming you have a list `lst` containing a bunch of numbers, write a couple of loops that print out all of the *negative* numbers and then all of the *non-negative numbers*. **[C2] (but leave a little space at the top)**

e.g.

```
lst = [9, -19, 31, -13, 1, 2] # <-- don't write this line though
# TODO: Your loop(s) here
```

```
-19 -13 9 31 1 2
```

You're not writing a whole function yet! Just write some lines & loops like you've been doing before.

ACTIVITY: WORKING TOWARDS WRITING A FUNCTION

Write the **signature** for a function that prints out all of the *negative* numbers and then all of the *non-negative numbers*. **[L2] (but leave some space on the right side)**

Remember: a signature consists of a `def`, a function name, and a list of parameters the function should be called with.

ACTIVITY: WORKING TOWARDS WRITING A FUNCTION

Add a signature to the code you wrote for **[C2]** in order to turn it into a function that can be called.

Then, in the right part of **[L2]**, write an example of a **function call** that would print out the following output:

```
-30 -14 3 19 8
```

PRACTICE: WRITING A FUNCTION

[C3] (but keep it to the left)

Write a function that accepts an integer and then prints out the text `"It's even!"` if the integer is even or `"It's not even!"` if the integer is odd

Reminder:

Step 1: Identify what you want it to do

Step 2: Write some code that does what you want it to do

Step 3: Write a signature that describes what you want to do

Step 4: Combine!

PRACTICE: WRITING A FUNCTION

[C3] (but to the right)

Rewrite your function so that it **returns** `True` if the integer is even and `False` if the integer is odd

PRACTICE: WRITING A FUNCTION [ON THE BACK]

Write a function called `is_valid` that checks if a string *starts* with a `T` or an `H`.

If it does, your function should return `True`, otherwise it should return `False`

PRACTICE: CALLING A FUNCTION [L3]

Complete the following code snippet that asks a user for a string, `message`, and a number, `num`. If `message` starts with a `T` or an `H` then it should print `message` `num` times, otherwise it should print `"whoops! not a valid string"`

Use your `is_valid` function :)

```
message = input("what is your message? ")
num = int(input('how many times should I print it? '))

# TODO: Finish this code
```