



# LE/EECS 1015 (Section A: LAB 04) Week 4: Lab #4

Shogo Toyonaga  
York University  
Lassonde School of Engineering

# Goals of Lab 3

1. Practice writing **functions** (function design recipe)
2. Learn how to debug functions
3. Learn how to write **test cases** (doctest)



# Concept Review

## 1. Function Scoping

- Global Variables
- Local Variables

## 2. Function Memory Model

- Stacks

## 3. Testing

- Doc Tests
- Unit Tests

# Function Scoping

- **Global Variables** are accessible everywhere
- **Local Variables** are only accessible within a certain instance and disappear after their container is terminated.
- In general, you are encouraged to avoid using global variables as they can make troubleshooting unnecessarily difficult.

# PAUSECHAMP



While **global variables** may have a somewhat **negative connotation** and should be avoided in trivial cases, an argument can be made for their use:

1. Global variables are required in threaded programs to communicate with each other.
2. Shared memory for memory-parallel programming (multi-processor systems) requires the use of global variables.

Source: [http://heather.cs.ucdavis.edu/matloff/public\\_html/globals.html](http://heather.cs.ucdavis.edu/matloff/public_html/globals.html)

# Good Styles for Global Variables

1. If you need to access a **global variable** in a function, pass it as an **argument** instead (unless it is a constant).
2. Rather than declaring a **global variable** in a **function**, return the value and assign it to a variable in `__main__`.
3. If you need to modify the value of **global variables**, let the caller assign the value in `__main__`.

```
search_list ▾  
= def search_list(input: list, target: int) -> int:  
  """Given a list, return the index of target. If target is not in the list, return -1."""  
  for item in input:  
    if item == target:  
      return item  
  return -1  
  
test_case_1 = [1, 2, 3, 4, 5]  
target = 25  
  
print(search_list(test_case_1, target))
```

# Function Memory Model

Search in Files | Search | Stack Data ▾

search\_list(): sample-program.py, line 4 ▾ Options ▾

Variable	Value
▼ <b>locals</b>	<locals dict; len=3>
▶ input	[1, 2, 3, 4, 5]
item	2
target	25
▼ <b>globals</b>	<globals dict; len=10>
__doc__	None
__file__	'C:\Users\Shogo\Documents\GitHub\Teaching Assistant\Sandbox'
__loader__	<from importlib.external.SourceFileLoader object at 0x000001...

Debug I/O | Exceptions | Python Shell | Messages | OS Commands ▾

Debug I/O (stdin, stdout, stderr) appears below Options ▾

# Putting it all together

(◡ ◡ ◡)



# Challenge

Write a Python function *get\_time\_format(...)* such that it takes in the number of seconds as an argument.

Given the number of seconds, return a string output in the *24-hour digital clock* representation; *hh:mm:ss*.

Ensure the zero padding constraint.

# Challenge

```
def get_time_format(seconds):  
    pass
```

# Challenge

```
def get_time_format(seconds: int) -> str:  
    pass
```

# Challenge

```
def get_time_format(seconds: int) -> str:  
    clock_seconds = seconds % 60  
    clock_minutes = (seconds // 60) % 60  
    clock_hours = ((seconds // 60) // 60) % 24  
    return f'{clock_hours:02}:{clock_minutes:02}:{clock_seconds:02}'
```

# Challenge

```
import doctest

def get_time_format(seconds: int) -> str:
    """
    Convert a number of seconds into a string formatted as hh:mm:ss.

    >>> get_time_format(0)
    '00:00:00'

    >>> get_time_format(59)
    '00:00:59'

    >>> get_time_format(60)
    '00:01:00'

    >>> get_time_format(90061)
    '01:01:01'
    """

    clock_seconds = seconds % 60
    clock_minutes = (seconds // 60) % 60
    clock_hours = ((seconds // 60) // 60) % 24
    return f'{clock_hours:02}:{clock_minutes:02}:{clock_seconds:02}'

doctest.testmod()
```



## Lab 3 – Objectives

1. Task 1: Follow the Steps (/30)
2. Task 2: Debugging (/30)
3. Task 3: Implementation: Wheels (/10)
4. Task 4: Implementation: Tickets (/10)
5. Task 5: Implementation: XOR (/10)
6. Task 6: Implementation: Expressions (/10)



# Thank You!

Shogo Toyonaga

Lassonde School of Engineering

