CLPS Matlab Programming Workshop

Session 3 of 3
Written by Jae-Young Son
December 12, 2018

Roadmap

- **Session 1: A programmer's essential toolkit**
  - What is information processing?
  - Variables
  - Functions
  - Arrays and matrices
  - Control logic
- **Session 2: Computer graphics**
  - Shapes
  - Animation
  - Placement of objects in space
  - Layering
  - Images
- **Session 3: Putting it all together**
  - Monitoring and recording user input (from keyboards and mice, and getting RTs)
  - Nesting functions
  - Simple image-rating experiment

1.1 – Basic keyboard input

1.1 – Basic keyboard input

```
while 1
```

1.1 – Basic keyboard input

```
while 1

    % Continuously check for keypress

    [keyDown, keyTime, whichKey] = KbCheck(-1);
```

1.1 – Basic keyboard input

```
while 1
    % Continuously check for keypress
    [keyDown, keyTime, whichKey] = KbCheck(-1);
```

Was ANY key pressed?
0 (false) or
1 (true)

1.1 – Basic keyboard input

```
while 1

    % Continuously check for keypress

    [keyDown, keyTime, whichKey] = KbCheck(-1);
```

Was ANY key pressed?
0 (false) or
1 (true)

Time when function was called (regardless whether any key was pressed)

1.1 – Basic keyboard input

```
while 1
    % Continuously check for keypress
    [keyDown, keyTime, whichKey] = KbCheck(-1);
```

Was ANY key pressed?
0 (false) or
1 (true)

Time when function was called (regardless whether any key was pressed)

256-element array (representing each character on an English keyboard):
0 (this key was not pressed) or
1 (this key was pressed)

1.1 – Basic keyboard input

```
while 1

    % Continuously check for keypress

    [keyDown, keyTime, whichKey] = KbCheck(-1);

    % Runs if keypress is made
```

Was ANY key pressed?
0 (false) or
1 (true)

Time when function was called (regardless whether any key was pressed)

256-element array (representing each character on an English keyboard):
0 (this key was not pressed) or
1 (this key was pressed)

1.1 – Basic keyboard input

```
while 1
    % Continuously check for keypress
    [keyDown, keyTime, whichKey] = KbCheck(-1);
    % Runs if keypress is made
    if keyDown == 1
        % Identifies which key was pressed
        userInput = KbName(whichKey);
```

Was ANY key pressed? 0 (false) or 1 (true)

Time when function was called (regardless whether any key was pressed)

256-element array (representing each character on an English keyboard): 0 (this key was not pressed) or 1 (this key was pressed)

1.1 – Basic keyboard input

```
while 1
    % Continuously check for keypress
    [keyDown, keyTime, whichKey] = KbCheck(-1);
```

Was ANY key pressed? 0 (false) or 1 (true)

Time when function was called (regardless whether any key was pressed)

256-element array (representing each character on an English keyboard): 0 (this key was not pressed) or 1 (this key was pressed)

```
    % Runs if keypress is made
    if keyDown == 1
        % Identifies which key was pressed
        userInput = KbName(whichKey);
        return
    end
end
```

1.1 – Basic keyboard input

```
while 1

    % Continuously check for keypress

    [keyDown, keyTime, whichKey] = KbCheck(-1);

    % Runs if keypress is made

    if keyDown == 1

        % Identifies which key was pressed

        userInput = KbName(whichKey);

        return

    end

end
```

Was ANY key pressed?
0 (false) or
1 (true)

Time when function was called (regardless whether any key was pressed)

256-element array (representing each character on an English keyboard):
0 (this key was not pressed) or
1 (this key was pressed)

Identifies the name of the key that corresponds to each element of **whichKey** that returns TRUE

This might lead to unexpected program behavior if your user accidentally presses down multiple keys at the same time!

1.1 – Basic keyboard input

```
while 1
    % Continuously check for keypress
    [keyDown, keyTime, whichKey] = KbCheck(-1);
    % Runs if keypress is made
    if keyDown == 1
        % Identifies which key was pressed
        userInput = KbName(whichKey);
        return
    end
end
```

Was ANY key pressed?
0 (false) or
1 (true)

Time when function was called (regardless whether any key was pressed)

256-element array (representing each character on an English keyboard):
0 (this key was not pressed) or
1 (this key was pressed)

```
userInput = userInput(1);
```

Identifies the name of the key that corresponds to each element of **whichKey** that returns TRUE

This might lead to unexpected program behavior if your user accidentally presses down multiple keys at the same time!

1.1 – Basic keyboard input: time limits + RTs

```
% Initialize time/response variables
timeLimit = 5;          % Limit trials to 5 seconds
startTime = GetSecs;    % Gets start time of trial
userInput = 'NaN';      % Default value of user's keypress
RT = NaN;               % Default value of reaction time
flag = 0;               % flag = 1 indicates that the user has responded

% Get keyboard responses
while GetSecs - startTime < timeLimit

    % Continuously check for keypress
    [keyDown, ~, whichKey] = KbCheck(-1);

    % Runs if keypress is made
    if keyDown == 1

        % Identifies which key was pressed
        userInput = KbName(whichKey);

        % Runs if keypress matches one of the keys allowed
        if any(strmatch(userInput, lower(keysAllowed)))
            endTime = GetSecs - startTime;
            RT = endTime;
            flag = 1;
            return
        end
    end
end
```

1.1 – Basic keyboard input: continuous keypresses

```
% Initialize time/response variables
timeLimit = 5;         % Limit trials to 5 seconds
startTime = GetSecs;   % Gets start time of trial
userInput = 'NaN';     % Default value of user's keypress
RT = NaN;              % Default value of reaction time
flag = 0;              % flag = 1 indicates that the user has responded
continuousPress = 0;   % 0 indicates that each keypress should be treated as a single event


% Get keyboard responses
while GetSecs - startTime < timeLimit

    % Continuously check for keypress
    [keyDown, ~, whichKey] = KbCheck(-1);

    % Runs if keypress is made
    if keyDown == 1

        % Identifies which key was pressed
        userInput = KbName(whichKey);

        % Runs if keypress matches one of the keys allowed
        if any(strmatch(userInput, lower(keysAllowed)))
            endTime = GetSecs - startTime;
            RT = endTime;
            flag = 1;

            % Treats extended key depression as a single event
            if continuousPress == 0
                KbReleaseWait;
            end

            return
        end
    end
end
```

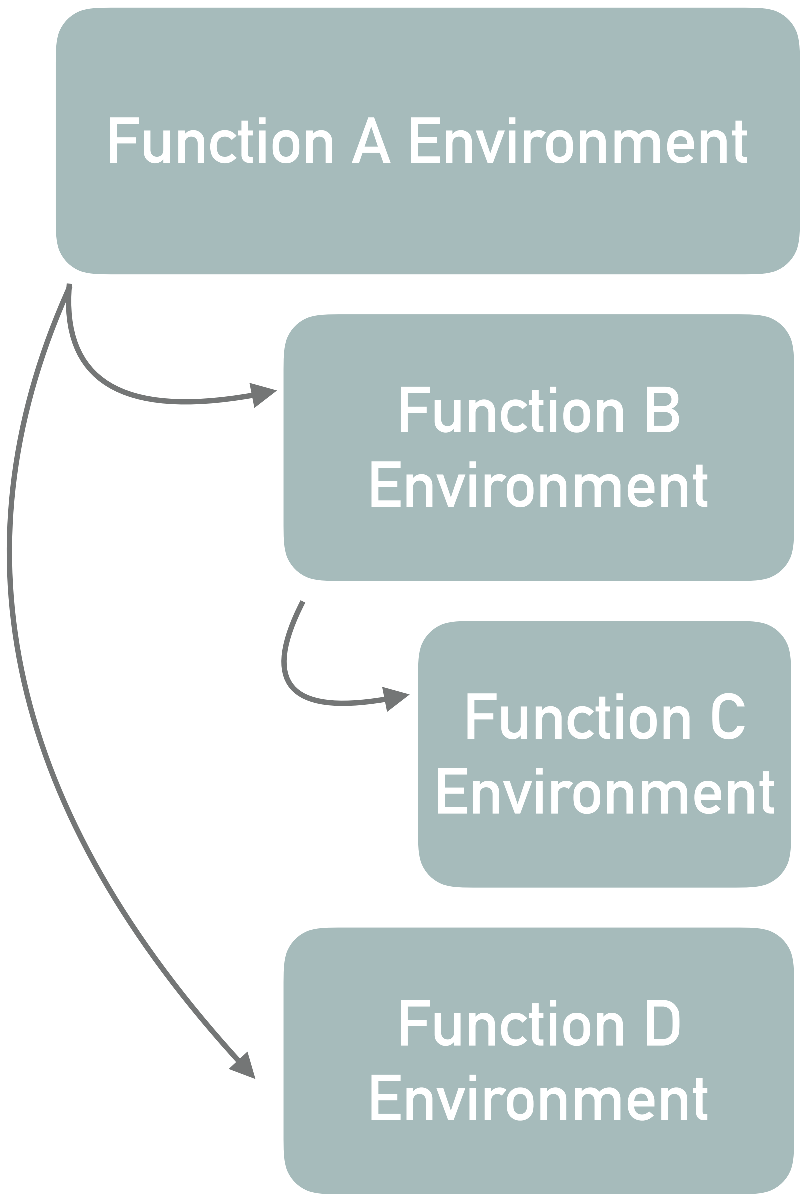
1.1 – Basic keyboard input: weird keys

```
% Deal with cases, spaces, and numbers
if strcmp(userInput, '1!')
    userInput = '1';
elseif strcmp(userInput, 'space')
    userInput = 'space';
elseif strcmp(userInput, '/?')
    userInput = '/';
elseif strcmp(userInput, '''"')
    userInput = '''';
else
    userInput = lower(userInput);
end
```

Not an exhaustive list! Only a couple of representative examples shown here

1.2 – Custom (nested) functions: variable passing

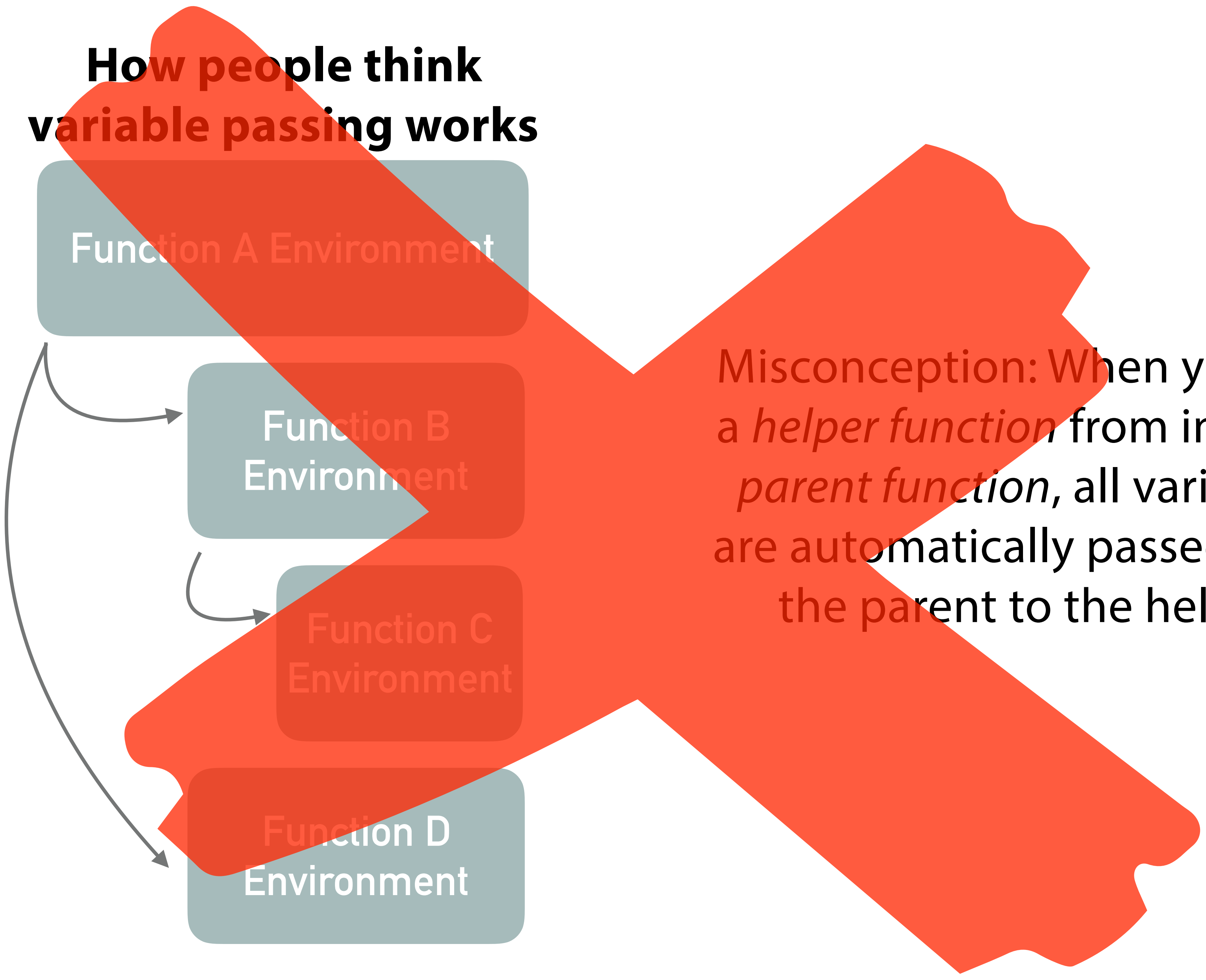
How people think variable passing works



Misconception: When you call a *helper function* from inside a *parent function*, all variables are automatically passed from the parent to the helper

1.2 – Custom (nested) functions: variable passing

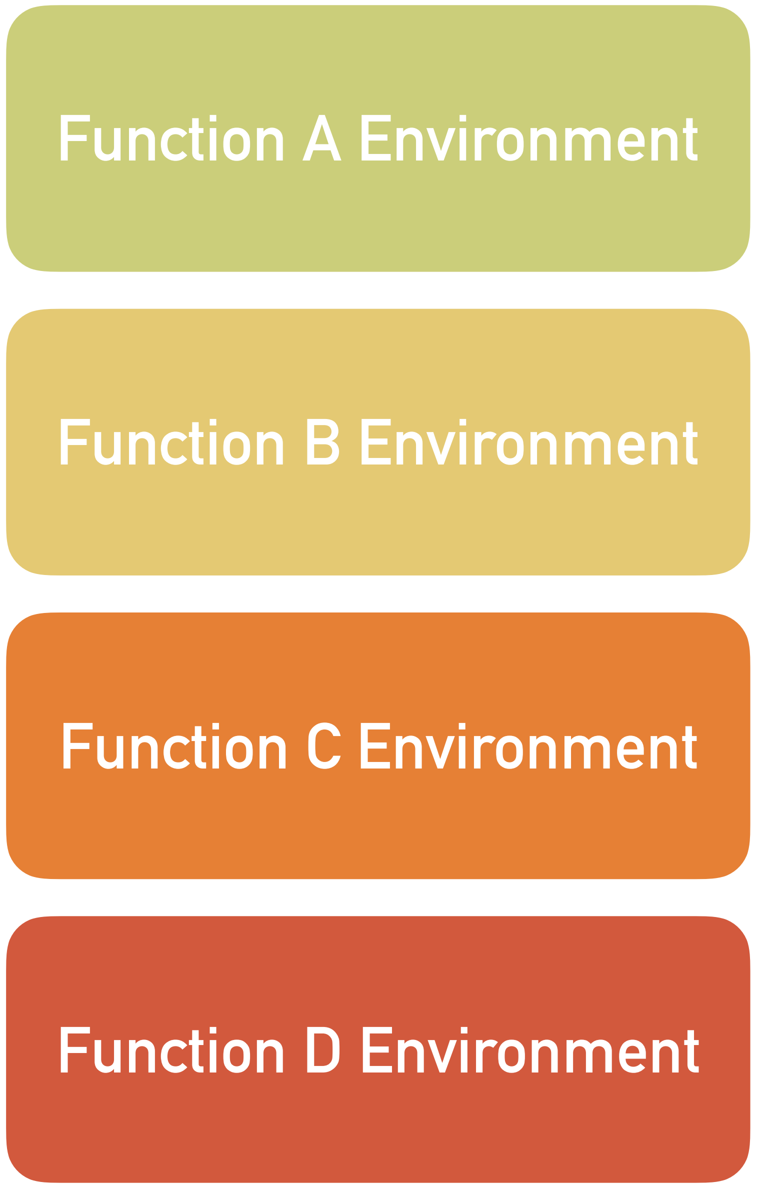
How people think variable passing works



Misconception: When you call a *helper function* from inside a *parent function*, all variables are automatically passed from the parent to the helper

1.2 – Custom (nested) functions: variable passing

How variable passing actually works



Important: When you call a *helper function* from inside a *parent function*, the only variables that are passed from the parent to the helper are the ones that you **explicitly** pass!

1.2 – Custom (nested) functions: variable passing

How variable passing actually works

Function A Environment

Function B Environment

Function C Environment

Function D Environment

Important: When you call a *helper function* from inside a *parent function*, the only variables that are passed from the parent to the helper are the ones that you **explicitly** pass!

What happens in Vegas stays in Vegas…

and what happens in a function environment stays there unless it's passed to a different function!

1.2 – Custom (nested) functions: defining a function

```
function [userInput, RT, flag] = keyrec(keysAllowed, timeLimit,
continuousPress)
```

1.2 – Custom (nested) functions: defining a function

Lets Matlab know you want to define a function

```
function [userInput, RT, flag] = keyrec(keysAllowed, timeLimit,
continuousPress)
```

1.2 – Custom (nested) functions: defining a function

Lets Matlab know you want to define a function

```
function [userInput, RT, flag] = keyrec(keysAllowed, timeLimit,
continuousPress)
```

Function arguments (i.e., what variables do you want to pass to the helper function?)

1.2 – Custom (nested) functions: defining a function

Lets Matlab know you want to define a function

```
function [userInput, RT, flag] = keyrec(keysAllowed, timeLimit,
continuousPress)
```

These variables return the *relevant* computations performed by the helper function

Function arguments (i.e., what variables do you want to pass to the helper function?)

1.2 – Custom (nested) functions: defining a function

Lets Matlab know you want to define a function

```
function [userInput, RT, flag] = keyrec(keysAllowed, timeLimit,
continuousPress)
```

These variables return the *relevant* computations performed by the helper function

Function arguments (i.e., what variables do you want to pass to the helper function?)

Example: calling the keyrec helper function

```
while flag ~= 1
    [userInput, RT, flag] = keyrec({'f', 'j'});
end
```

1.2 – Custom (nested) functions: defaults

```
function [userInput, RT, flag] = keyrec(keysAllowed, timeLimit,
continuousPress)


% Set time limit to infinity if not otherwise specified
if ~exist('timeLimit', 'var')
    timeLimit = inf;
end

% Turn continuous press OFF if not otherwise specified
if ~exist('continuousPress', 'var')
    continuousPress = 0;
end
```

Nice to set some “defaults” in case your user doesn’t specify every argument in their function call!