Week 09 Studio Environment Model, Arrays and Loops

CS1101S AY21/22 Semester 1

Studio 05E

11 Oct 2021

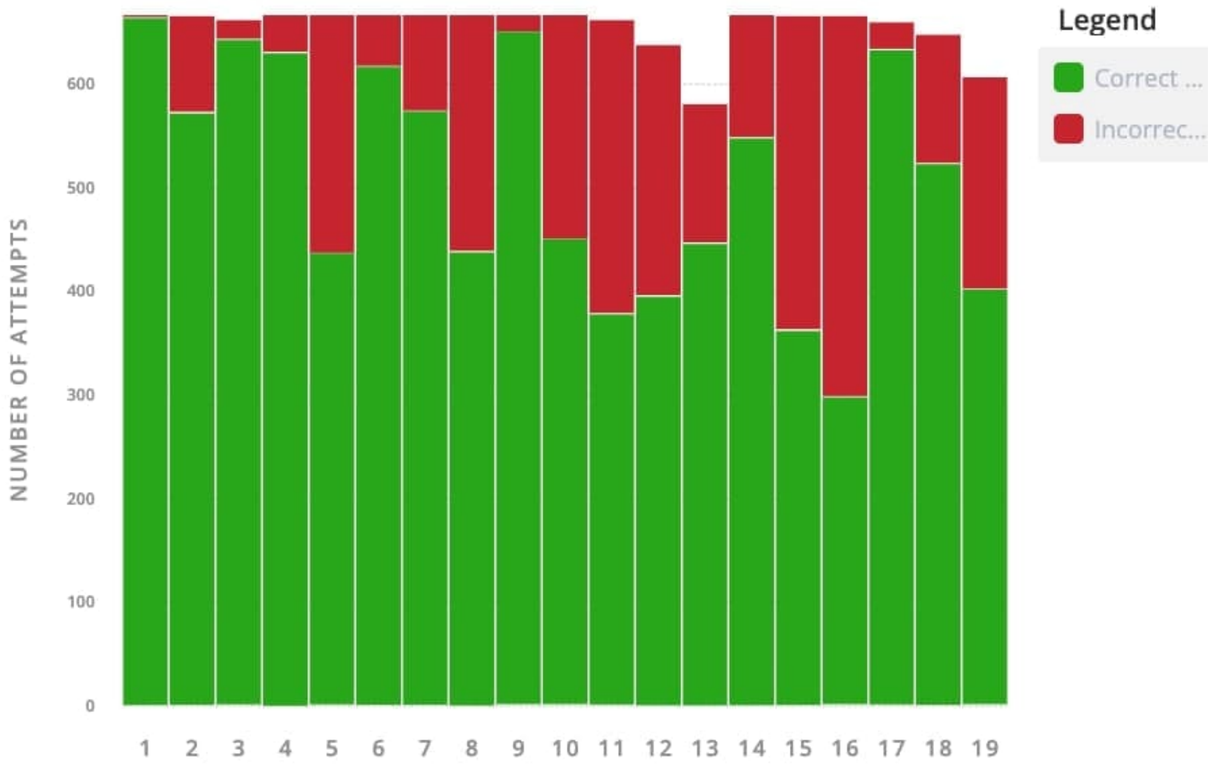
Yan Xiaozhi (David)
@david_eom
yan_xiaozhi@u.nus.edu

Admin

- Contact tracing (QR code + class photo)
- Mission on environment model
- Mastery check II
- Consultations
  - Am open to it! Prepare your questions to ask
- Midterm results
  - 12%, PE and finals are still very important
  - As much as possible try NOT to SU this mod!

45.24	Average	46	Median	13.775	Standard Deviation
4	Lowest	70	Highest		

QUESTION STATISTICS



Recap

Environment Model

What Is It?

- Substitution model breaks down with states
- Environment:
  - A sequence of frames that store name bindings
    - Frame: essentially key-value pairs
  - Each frame (except the global frame) has a pointer to its enclosing environment
  - 2 special frames:
    - Global: contains all pre-declared declarations (values and functions)
    - Programme: top level declarations

Environment Model

When to Create New Frames?

- Evaluating a block `{ statement }`
  - Only if it has declarations inside
  - If the programme has no declarations at all, the PE will not be created e.g. `{ display(“hello world”); }`
- Function applications
  - Contains bindings of parameter variables to actual arguments
  - Again, no frame created if function is nullary

Environment Model

Functions

- Do not draw boxes for declarations
  - Recall: functions are essentially constant declarations
  - What about lambda expressions?
- Differentiate function declaration and applications
  - Declarations: googly eyes, one to body one to frame
  - Applications: create new frames

Environment Model

Functions

1. Evaluate argument expressions (applicative order reduction)
2. Identify environment where function should be evaluated from
3. Extend environment, create new frame
4. Bind parameter name to argument value in new frame
5. Extend new frame if there are declarations
6. Evaluate and update parent frames if necessary

Environment Model

Try It Out!

- ```
  const x = 0;
  function f(y) {
      let z = 7;
      return y + z;
  }
  if (x < 10) {
      const y = f(x);
  } else { }
  ```

- ```
  function curry(f) {
      return x => y => f(x, y);
  }

  curry(math_pow)(3)(4);
  ```

Environment Model

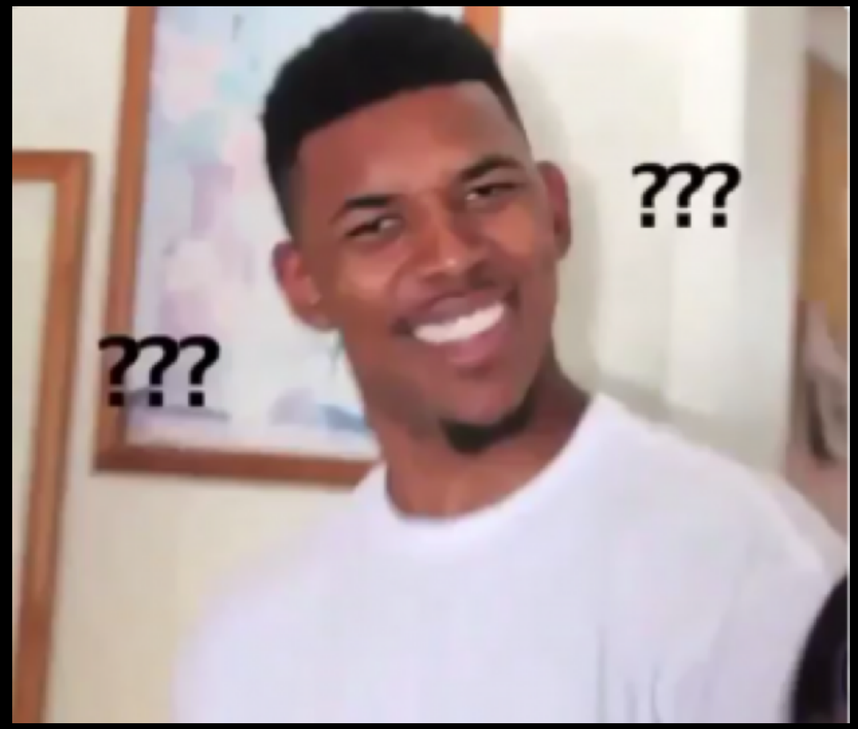
Any Drawbacks?

- Only tells us where the function should be evaluated in, not where the function is called from
- We need to keep track of the argument value and return value of the function
- Cannot tell us whether the a name is redeclared

Environment Model

Honestly, What's the Point?

- Seems extremely pointless
  - Reading assessment II
  - Useful for understanding especially when there are lambdas here and there
- Already have environment visualiser what
  - Used to be super buggy during my sem (huge improvements thanks to CP3108 mates)
  - Takes away the learning opportunity
  - Monstrosity for complex programmes



Arrays

What Is It?

- A data structure that stores a sequence of elements
  - Elements can be of different types
  - Present in all (at least the ones I know LOL) programming languages
- Similar to list but without the idea of head and tail
- Accessed using integer index in O(1) time (random access)
  - `0 <= idx <= length(arr) - 1`
  - “Off-by-one” error! Index start at 0!

Arrays

Example

- `const arr = [1, 2, 3];` | `let arr = [1, 2, 3];`
- Accessing: `arr[0];` | `arr[100];`
- Finding the size: `array_length(arr);`
- Mutating: `arr[1] = 4;`
- Appending:
  - `arr[3] = 999;` | `arr[100] = 99;`
  - `arr[array_length(arr)] = newElem; // how does this work?`

Nested Arrays

What Is It?

- Array of arrays
- Example: matrix
  - `const matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];`
  - `matrix[0];    // [1, 2, 3]`
  - `matrix[0][1]; // 2`

Loops

What Is It?

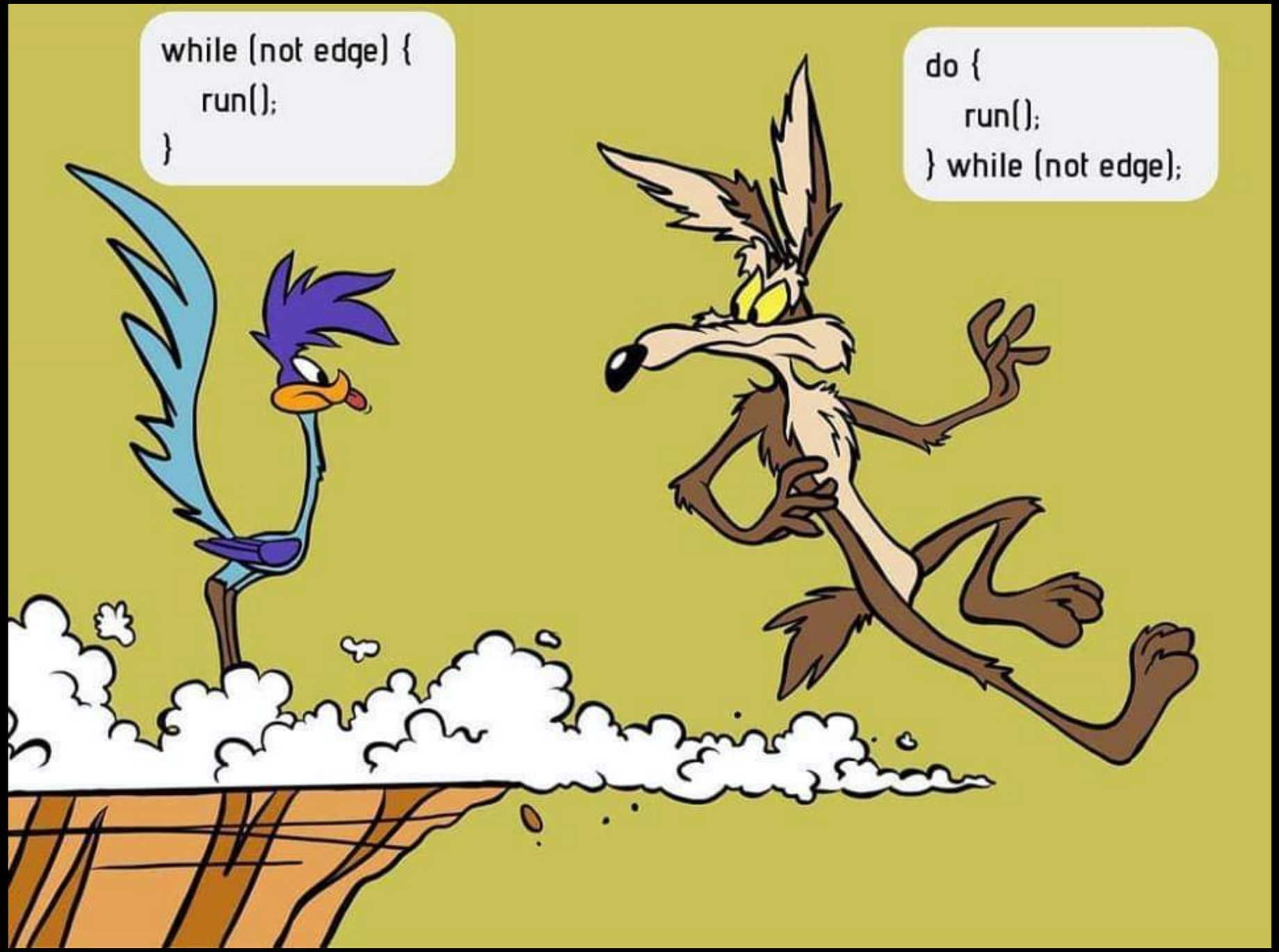
- Instruction that repeats itself for a certain number of iterations
- Ends when certain condition is reached
- Condition is checked at every iteration
- `for` loops and `while` loops

```
Loop:   beq  $9, $0, End
        add  $8, $8, $10
        addi $9, $9, -1
        beq  $0, $0, Loop
End:
```

`while` Loops

What Is It?

- Loops that repeat itself as long as the condition is true
- Syntax:
  - ```
    while (<condition>) {
        // do something
    }
    ```
- Example:
  - ```
    while (true) {
        display(“hello”);
    }
    ```


while (not edge) {
run();
}
do {
run();
} while (not edge);

`for` Loops

What Is It?

- Loops that repeats for a preset number of times
- Syntax:
  - ```
    for (<initialisation>; <condition>; <re-assignment>) {
        // do something
    }
    ```
- Example:
  - ```
    for (let i = 0; i < 5; i = i + 1) { display(i); }
    ```
- All for loops can be written to while loops!


i = 0
i = 1
i = 2
i = 3
i = 4
i = 5

`for` Loops

Tips

- Be careful about the “off-by-one” errors (again)
- Do NOT try to assign the control variable in the loop body!
  - ```
    for (let i = 0; i < 5; i = i + 1) {
        i = i - 1;
        display(i);
    }
    ```
    ```

Loops

When to Use Which?

- For loops:
  - When we know how many iterations we need to do
- While loops:
  - When we are not sure about the number of iterations required
- Of course, you can do big brain stuff like:
  - `for (let p = lst; !is_null(p); p = tail(p)) { // loop body }`

Loops

Question of the Day

- How long should a loop last?
- `for a while :)`

Loops

Keywords

- Use keywords to control the logic flow of loops, can be used for both `for` loops and `while` loops
  - `break`: terminate the loop entirely, no matter which iteration it is
  - `continue`: carry on with the next iteration, skip whatever is below

Loops

Personal Note on Keywords

- I don’t like using keywords like break and continue unless necessary!
- `break`:
  - Can always add if-else with an empty else block
- `continue`:
  - Can always add conditions with && into the while loop header
- Disrupt the logic flow, but feel free to use if you are 100% sure about how they work!

Loops

Example on Keywords

```
for (let i = 0; i < 10; i = i + 1) {
    if (i == 7) {
        continue; OR break;
    } else {
        display(i);
    }
}
```

Array and Loops

Combine Them Together

- Traverse an array with loops (visit a specific subset of the array)
- Visit every single element in a doubly nested array:
  - ```
    for (let i = 0; i < array_length(arr); i = i + 1) {
      for (let j = 0; j < array_length(arr[i]), j = j + 1) {
        display(arr[i][j]);  // do something
      }
    }
    ```
- Idea: we can nest arrays in arrays, we can nest loops in loops
    ```

Array and Loops

Challenge

- Try to implement
  - Binary search for arrays
  - Sorting algorithms for arrays (selection sort, insertion sort, quick sort…)
  - Hand-write your programme WITHOUT googling!

Any Questions?