Studio 9
Arrays, Loops and the Env Model

CS1101S AY20/21 SEM 1
Studio 03A

Chen Xihao
Year 2 Computer Science

chenxihao@u.nus.edu
@BooleanValue

Studio 9
Agenda

- Admin

- Recap:

  - Arrays

  - Loops

  - Arrays AND loops :D

  - Environment model

- Studio sheet

- In-class studio sheet

Recap

Recap: Arrays

Recap
Arrays

- What is an array?

  - A data structure that stores a sequence of elements

    - Elements can be of different types

  - Similar to lists, but do not have idea of head or tail

  - Accessed using integer index, in O(1) time (random access)

    - 0 <= index <= length - 1

Recap
Arrays

- `const arr = [1, 2, 3];  // declare a new array of 3 elements`

- Accessing arrays:

  - `arr[0];     // 1`

  - `arr[100];  // undefined`

- Finding the size of array:

  - `array_length(arr);  // 3`

Recap
Arrays

- `const arr = [1, 2, 3];  // declare a new array of 3 elements`

- Mutating array elements:

  - `arr[1] = 4;  // [1, 4, 3];`

  - `arr[1];       // 4`

Recap
Arrays

- `const arr = [1, 2, 3];  // declare a new array of 3 elements`

- Appending to end of array:

  - `arr[3] = 999;`

  - `arr;  // [1, 4, 3, 999]`

  - `trick: ` `arr[array_length(arr)] = newElement;` `// how does this work?`

- What if we want to append to position 50?

  - `arr[50] = 999;  // [1, 2, 3, , , , …, 999]`

Recap
Arrays

- `const arr = [1, 2, 3];  // declare a new array of 3 elements`

- Note: index starts at 0 !!!!!!!!!!

  - if we have an array `[1, 2, 3]`

    - number 1 is at index 0

    - number 2 is at index 1

    - number 3 is at index 2

  - be careful of "off-by-one" errors!

Recap
Arrays - Matrices

- What is a matrix?

  - an array of arrays (extension: array of matrices, matrices of matrices)

  - recall: list of lists!

- `const matrix = [ [1, 2, 3], [4, 5, 6], [7, 8, 9] ];`

- `matrix[0];      // [1, 2, 3]`

- `matrix[0][1];  // 2`

  - recall: applicative order reduction
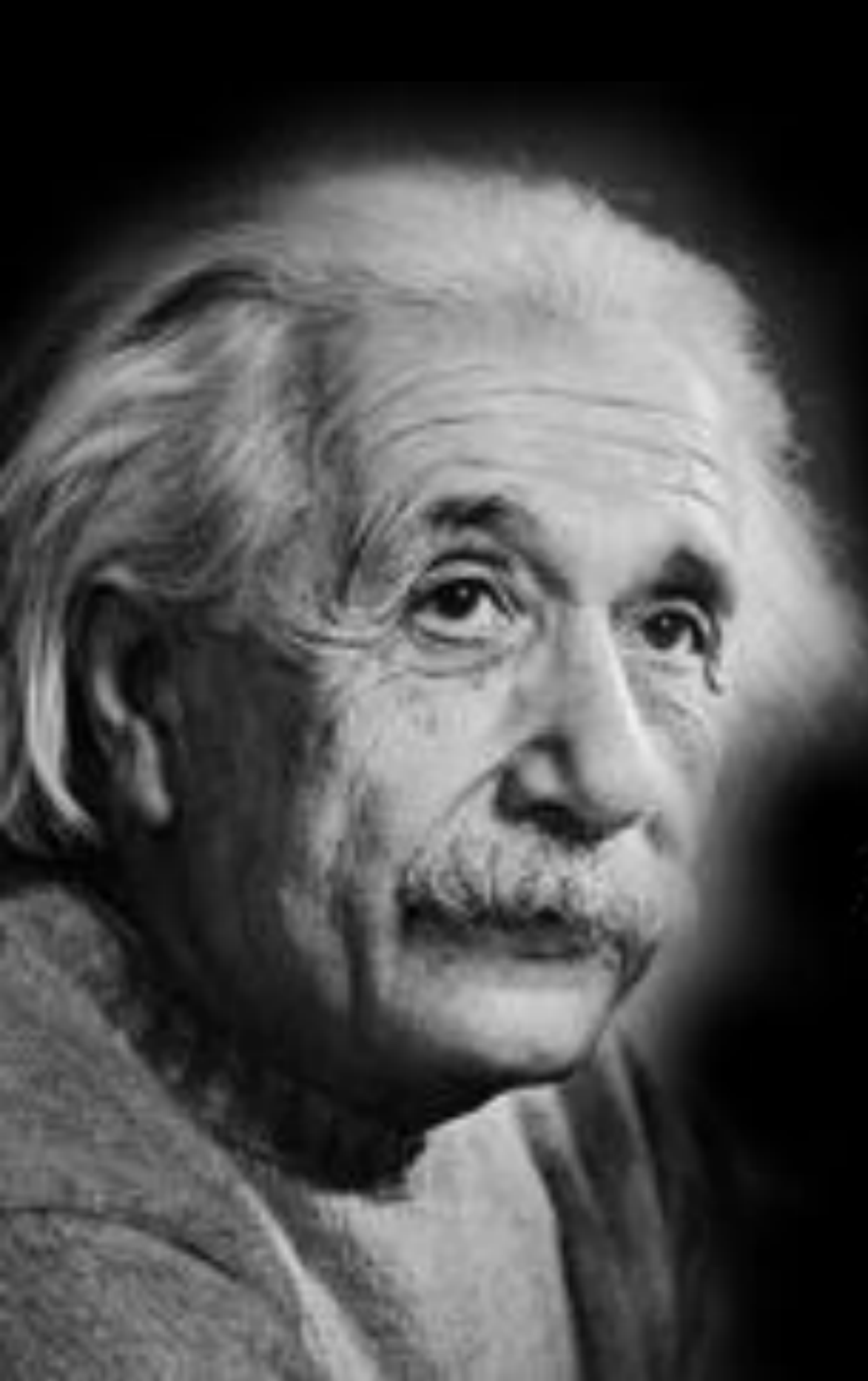
Any questions?

Recap: Loops

Recap
Loops

- What is a loop?

  - An expression (more accurately its an instruction)

  - Repeats itself for a certain number of iterations

  - Ends when some condition is reached

    - Checks for this condition at every iteration

"Insanity is doing the same thing over & over again & expecting different results."

Albert Einstein

Recap
Loops - While

- While loops:

  - Loops that repeat itself as long as the conditional expression is true

- Structure:

```
while (<condition>) {

    // do stuff

}
```

Recap
Loops - While

- An example:

```
let i = 0;

while (i < 10) {

    display(i);

    i = i + 1;  // recall: re-assignment

}


// displays 0, 1, 2, … 9
```

Recap
Loops - While

- Another example:

```
while (true) {

    display("hello");

}



// runs forever (at least theoretically)
```

Recap
Loops - for

- For loops:

  - A loop that repeats for a preset number of times

- Structure:

```
for (<initialisation>; <condition>; <re-assignment>) {

    // do stuff

}
```

Recap
Loops - for

- An example:

```
for (let i = 0; i < 10; i = i + 1) {

    display(i);

}



// displays 0, 1, 2, … 9
```

Recap
Loops - for

```
for (let i = 0; i < 10; i = i + 1) {

    display(i);

}
```

- How this works:

  1. declare i = 0 (loop control variable)

  2. check if i is less than 10

  3. execute body (displays value of i)

  4. re-assign i (update to new value)

  5. repeat steps 2 to 4

Recap
Loops - for

- Discussion:

  - Write this in terms of a while loop.

```
let x = 0;

for (let i = 10; i > 0; i = i - 1) {

    x = x * x;

}
```

Recap
Loops - for

```
let x = 0;

for (let i = 10; i > 0; i = i - 1) {

    x = x * x;

}
```

- Answer:

```
let x = 0;

let i = 10;

while (i > 0) {

    x = x * x;

    i = i - 1;

}
```

for (let i = 0; i < 5; i = i + 1)

credits: r/ProgrammerHumor

Recap
Loops

- Cautions:

  - The loop will NOT run when the condition fails

  - Be careful of "off-by-one" errors… (again)

  - Do NOT try to assign the control variable in the loop body

    - `for (let i = 0; i < 10; i = i + 1) { i = i - 1; }`

Recap
Loops

- What are the differences?

  - While loops state only the conditions for the loop to run

    - You can use compound conditions using `||` and `&&`

  - For loops usually specify the number of iterations

    - Updates of control variable is stated more explicitly

Recap
Loops

- When to use which?

  - Intuitively:

    - For - when we know how many iterations we need to do

    - While - when we are unsure of how many iterations we need to do

  - If you are advanced:

```
function list_length_loop(xs) {
    let count = 0;
    for (let p = xs; !is_null(p); p = tail(p)) {
        count = count + 1;
    }
    return count;
}
```

Recap
Loops - Keywords

- We can use certain keywords to control the logic flow of loops:

  - `break` and `continue`

  - Can be used for both for-loops and while-loops

Recap
Loops - Keywords - Continue

- Continue - what does it do?

  - Continue… to the next iteration

  - Skips whatever's below this statement

Recap
Loops - Keywords - Continue

- An example:

```
for (let i = 0; i < 10; i = i + 1) {

    if (i === 7) {

        continue;

    } else {

        display(i);

    }

}
```

Result:
0
1
2
3
4
5
6
8
9

Recap
Loops - Keywords - Break

- Break - what does it do?

  - Break… out of the loop

  - Terminates the loop entirely, no matter which iteration it is

Recap
Loops - Keywords - Break

- An example:

```
for (let i = 0; i < 10; i = i + 1) {

    if (i === 7) {

        break;

    } else {

        display(i);

    }

}
```

Result:
0
1
2
3
4
5
6

Recap
Loops - Keywords

- A personal note from me:

  - I don't like to use breaks since you can add conditions into the while loop header

  - I also don't like to use continue since you can always add an if-else with an empty else block

  - Also they kind of disrupt the flow of logic :')

Recap
Loops - Keywords

- Should you use?

  - If you know how they work then go ahead!

  - But make sure you can trace and debug your programme

Recap
Loops

- Question: How long should a loop last?

- Answer: for a while! heheXD

Any questions?

Recap: Arrays and Loops

Recap
Arrays and Loops

- With loops, we can traverse an array

  - Wait… what's traverse?

  - Basically: visiting a specific sub-set of the array

    - Entire array, or

    - From index 0 to 2, or

    - Just index 1

Recap
Arrays and Loops

- An example:

```
const arr = [0, 1, 2, 3];

for (let i = 0; i < array_length(arr); i = i + 1) {

    display( arr[i] );

}


// prints 0, 1, 2, 3 each on a new line
```

Recap
Arrays and Loops

- Another example:

```
const arr = [[0, 1, 2, 3], [4, 5]];

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] );

    }

}

// visiting nested arrays using nested loops

// prints 0, 1, 2, 3, 4, 5 each on a new line
```

Recap
Arrays and Loops

- Key ideas:

  - We can visit every element in an array of some size with loops

  - We can nest arrays in arrays

  - We can also nest loops in loops

Recap
Arrays and Loops

- Put on your thinking cap:

  - Recall binary search on lists and quick sort on lists

  - Now, try to implement:

    - Binary search for arrays

    - Quick sort on arrays

- Hand-write your programme and NO GOOGLING!

Therapist: loop loops aren't real, they can't hurt you.

Loop loops:

```
1                loop { loop  { loop {
2         loop {                    loop {
3      loop {                          loop {
4    loop {                              loop {
5  loop {                                  loop {
6  loop {                                  loop {
7  loop {                                  loop {
8    loop {                              loop {
9      loop {                          loop {
10       loop {                      loop {
11          }}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
```

Any questions?

Recap: Environment Model

Recap
Environment Model

- Why environment model?

  - With states, the substitution model breaks down!

  - Can't keep track of states, since we substitute the value at declaration

Recap
Environment Model

- What is an environment?

  - A sequence of frames that store name bindings

Recap
Environment Model - Frames

- What is a frame?

  - A table of bindings

    - Associate variable names and the corresponding values

    - aka: shows key-value pairs

Recap
Environment Model - Frames

- An example:

```
{

    let a_number = 0;

    const a_bool = true;

    let a_str = "hello world!";

}
```
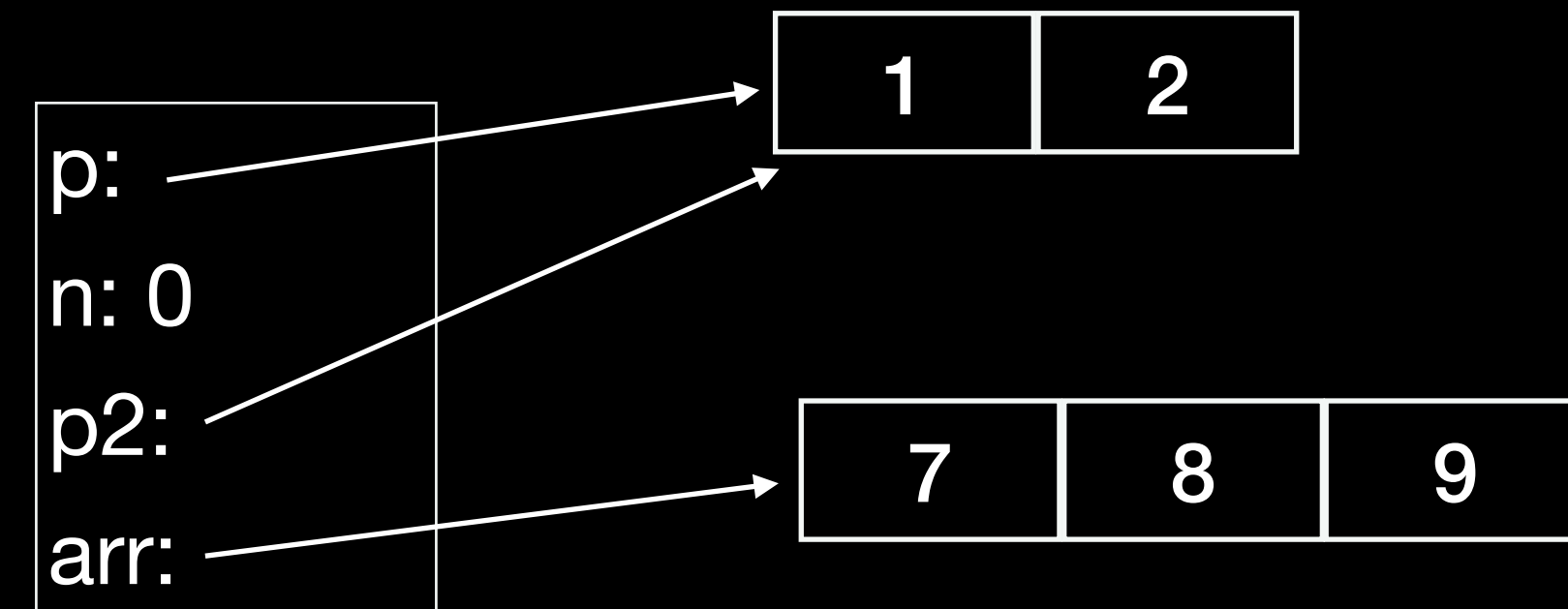
```
a_number: 0
a_bool := true
a_str: "hello world!"
```

Recap
Environment Model - Frames

- Another example:

```
{

  let p = pair(1, 2);

  let n = 0;

  let p2 = p;

  let arr = [7, 8, 9]

}
```

Recap
Environment Model - Frames

- Note:

  - For variables, use `:`

  - For constants: use `:=`
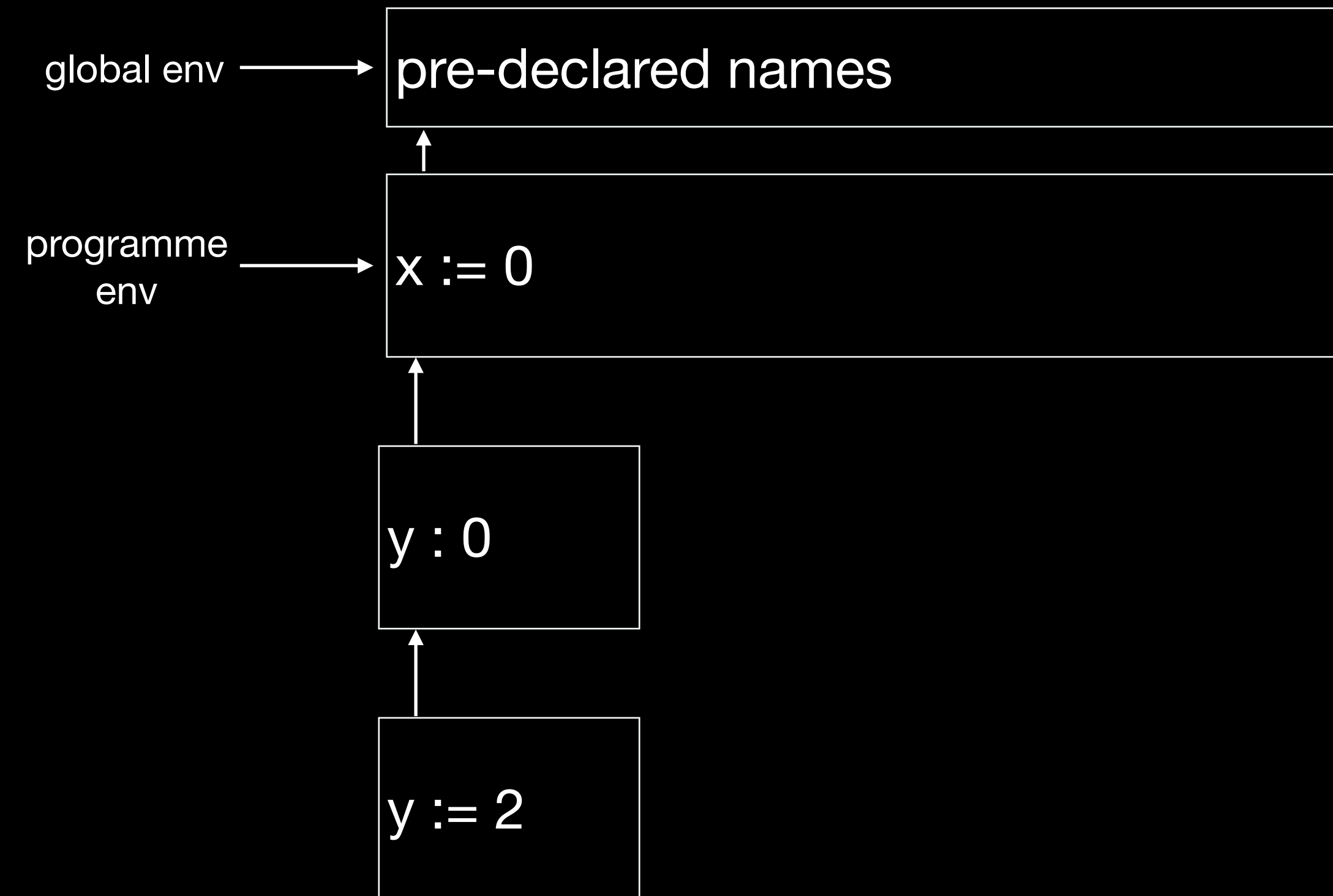
Recap
Environment Model - Environment

- What is an environment, again

  - A sequence of frames

  - Each frame has a pointer to its enclosing environment (except the global frame)

  - Two special frames:

    - global: contains all pre-declared stuff (primitive values and functions)

    - programme: top level declarations

Recap
Environment Model - Environment

- An example:

```
const x = 0;

{

    let y = 0;

    {

        const y = 2;

    }

}
```


global env ⟶ pre-declared names

programme env ⟶ x := 0

y : 0

y := 2

Recap
Environment Model - Environment

- Another example:

```
function f(x) {

    const y = 7;

    return x + y;

}
```

global env ⟶ | pre-declared names |

programme
env ⟶ | f := |

param: x
body:
    const y = 7;
    return x + y;
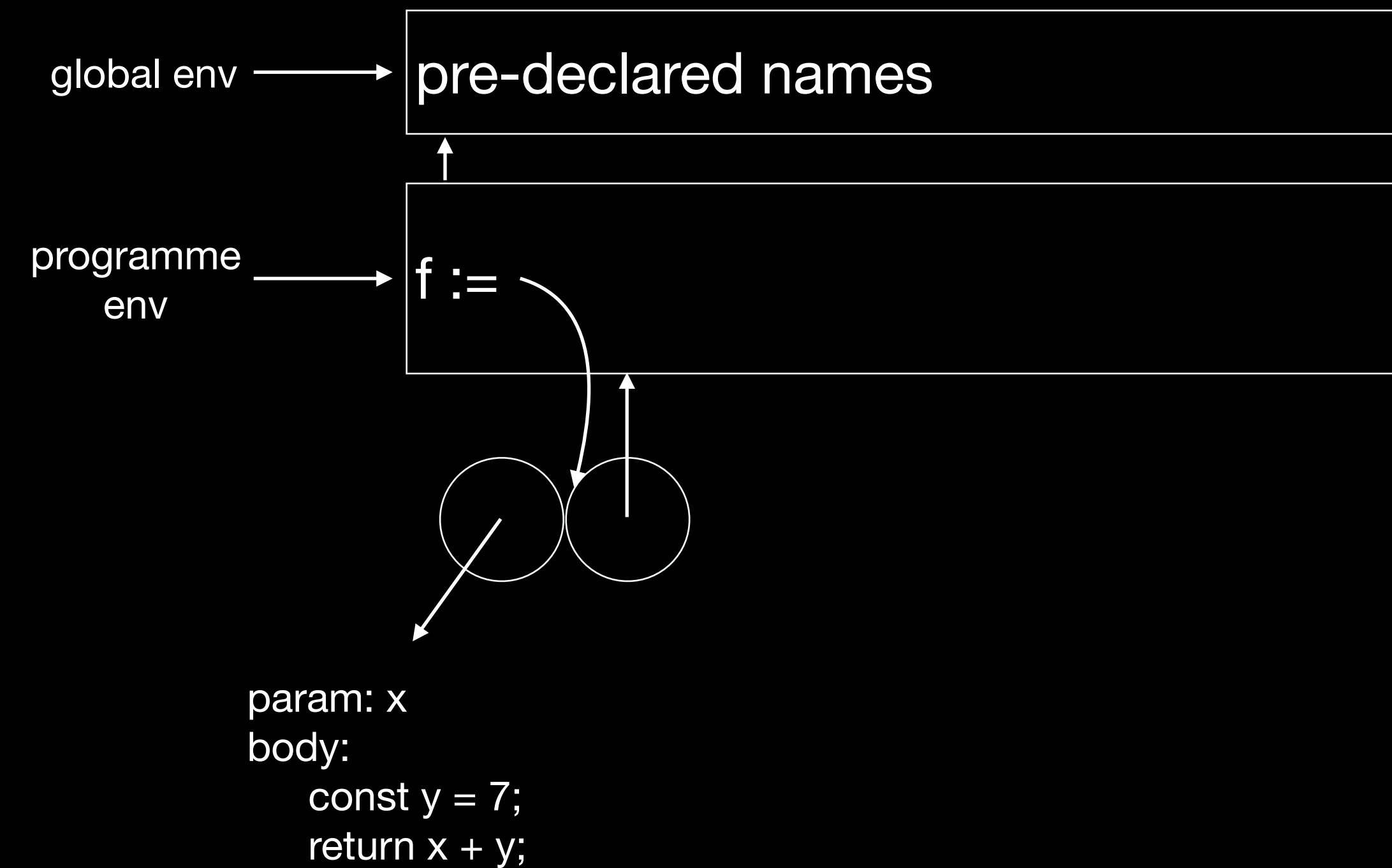
Recap
Environment Model - Functions

- Function declarations: "googly eyes"

  - yellow arrow:

    - reference to function obj

  - cyan arrow:

    - param and body

  - magenta arrow:

    - environment in which the function is declared and should be evaluated

- Note: cyan and magenta arrows may not be from the same frame!

global env ⟶ pre-declared names

programme
env ⟶ f :=

param: x
body:
    const y = 7;
    return x + y;
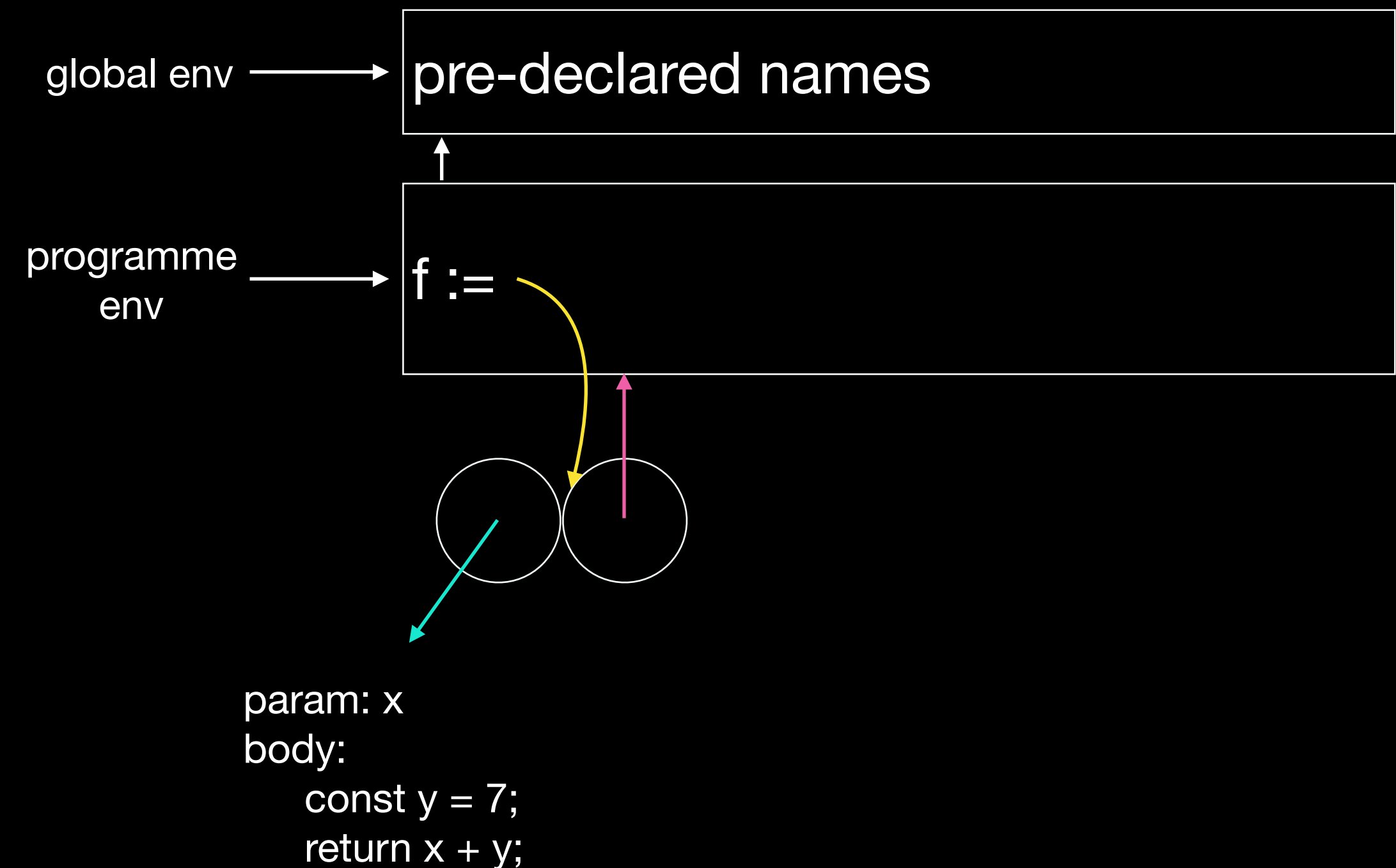
Recap
Environment Model - Functions

- Function declarations:

  - Do not draw boxes for declarations!

- Differentiate between function <u>declarations</u> and <u>applications</u>

  - Declarations: googly eyes

  - Applications: create new frames

- Tip:

  - Write "…" for param and body during exam as long as it's not ambiguous

Recap
Environment Model - Functions

- Function applications:

  1. Evaluate argument expressions (applicative order reduction)

  2. Identify environment where function should be evaluated from (the right eyeball arrow)

  3. Extend from that environment and create a new frame (A)

  4. In the new frame (A), bind the parameter name to argument value

  5. If there are declarations in the function body, extend from frame A and bind the declarations

  6. Evaluate the function and update parent frames if necessary

Recap
Environment Model - Functions

- An example:

```
const x = 0;

function f(y) {

    let z = 7;

    return y + z;

}

if (x < 10) {

    const y = f(x);

} else {}
```
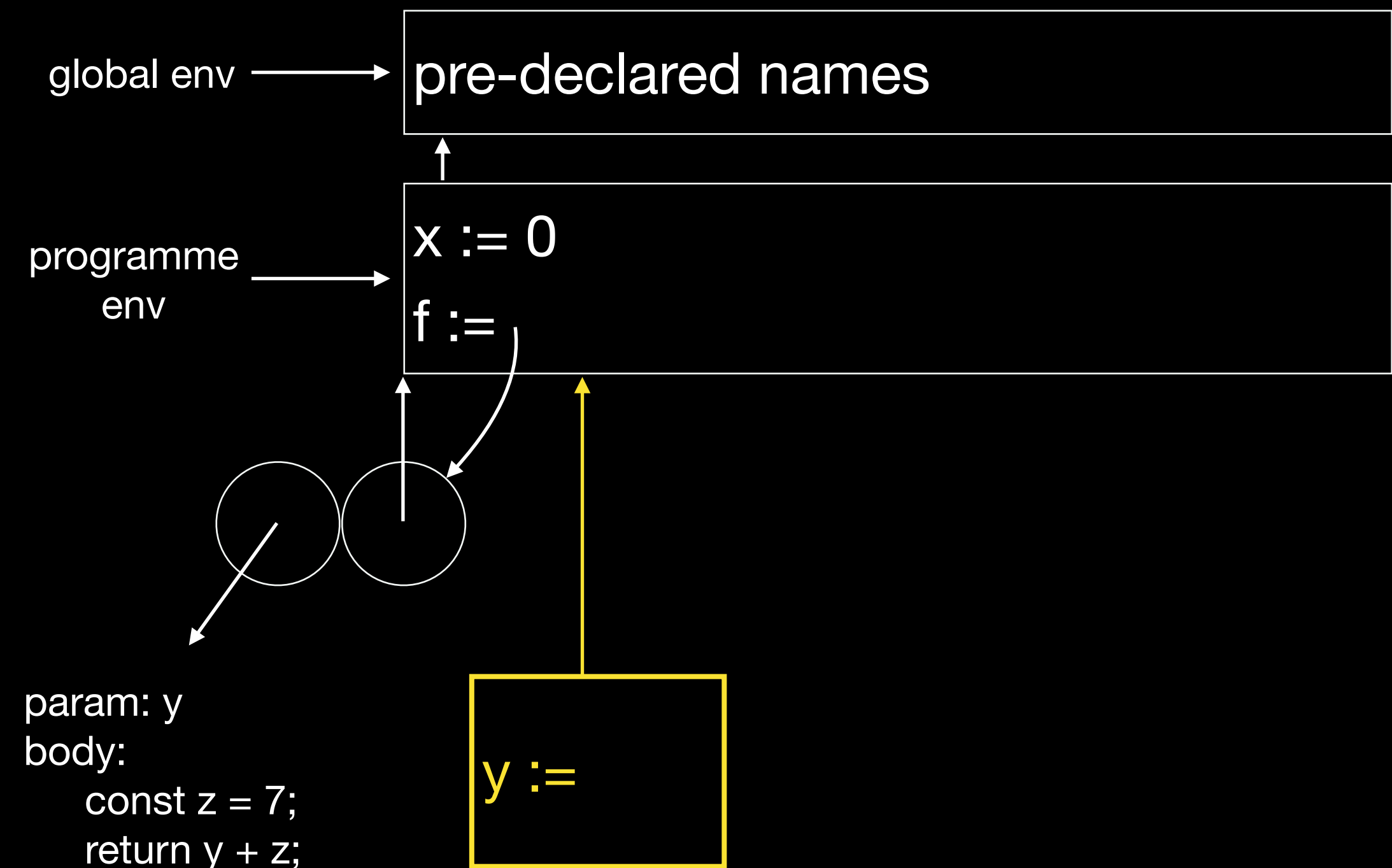
Recap
Environment Model - Functions

• An example:

```
const x = 0;

function f(y) {

    let z = 7;

    return y + z;

}

if (x < 10) {

    const y = f(x);

} else {}
```

global env ⟶ pre-declared names

programme env ⟶
x := 0
f :=

param: y
body:
    const z = 7;
    return y + z;

y :=

Step 0: create new frame for block with declaration of `y`
(we "prepare" the environment here before applying the function)

Recap
Environment Model - Functions

• An example:

```
const x = 0;

function f(y) {

    let z = 7;

    return y + z;

}

if (x < 10) {

    const y = f(x);

} else {}
```

global env → pre-declared names

programme env →
x := 0
f :=

param: y
body:
    const z = 7;
    return y + z;

y :=

Step 1: evaluate function arguments: x = 0

Recap
Environment Model - Functions

- An example:

```
const x = 0;

function f(y) {

    let z = 7;

    return y + z;

}

if (x < 10) {

    const y = f(x);

} else {}
```
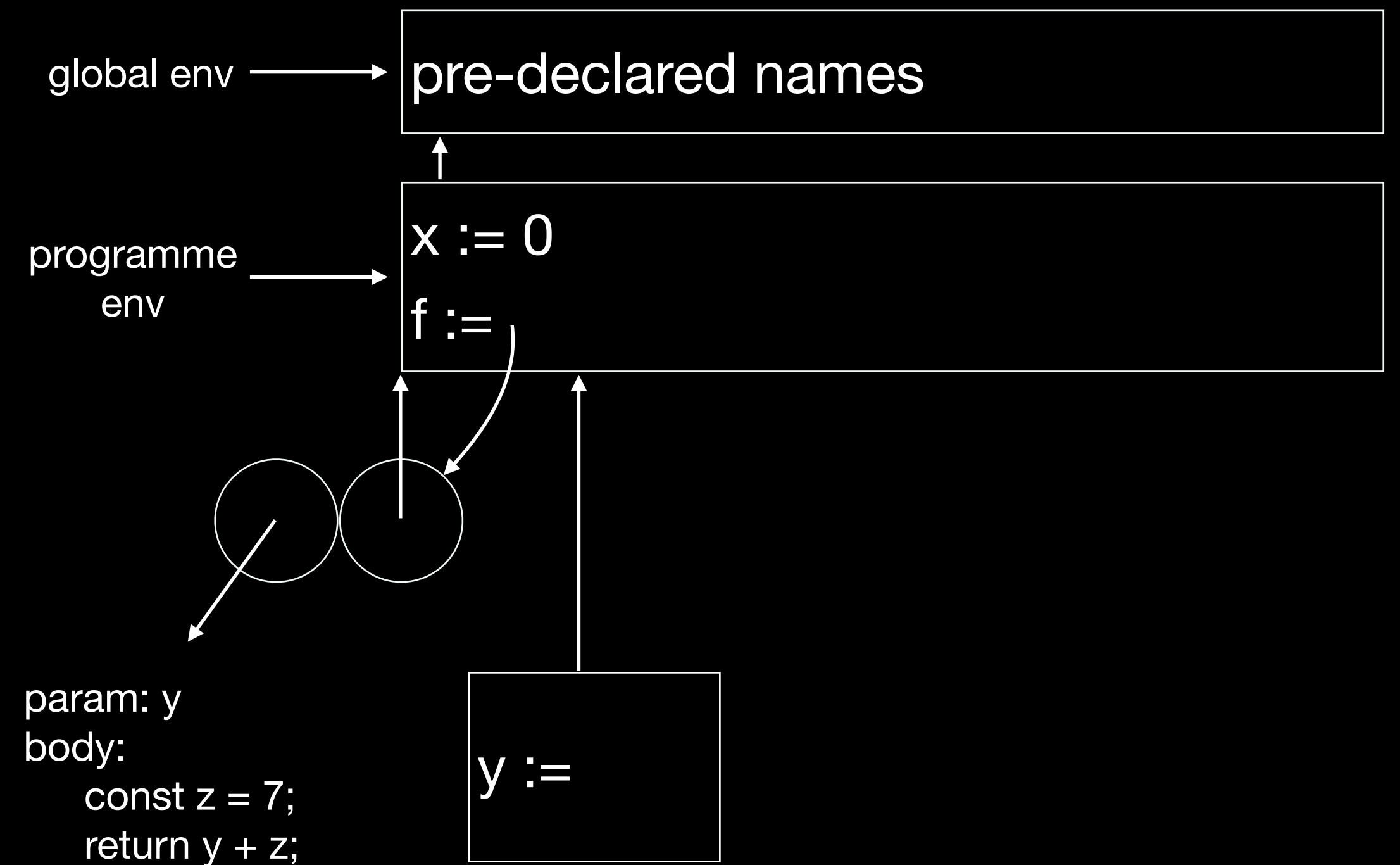
global env ——→ pre-declared names

programme env ——→

x := 0

f :=

param: y
body:
    const z = 7;
    return y + z;

y :=

Step 2: identify frame in which f is to be evaluated
(programme env)

Recap
Environment Model - Functions

- An example:

```
const x = 0;

function f(y) {

    let z = 7;

    return y + z;

}

if (x < 10) {

    const y = f(x);

} else {}
```
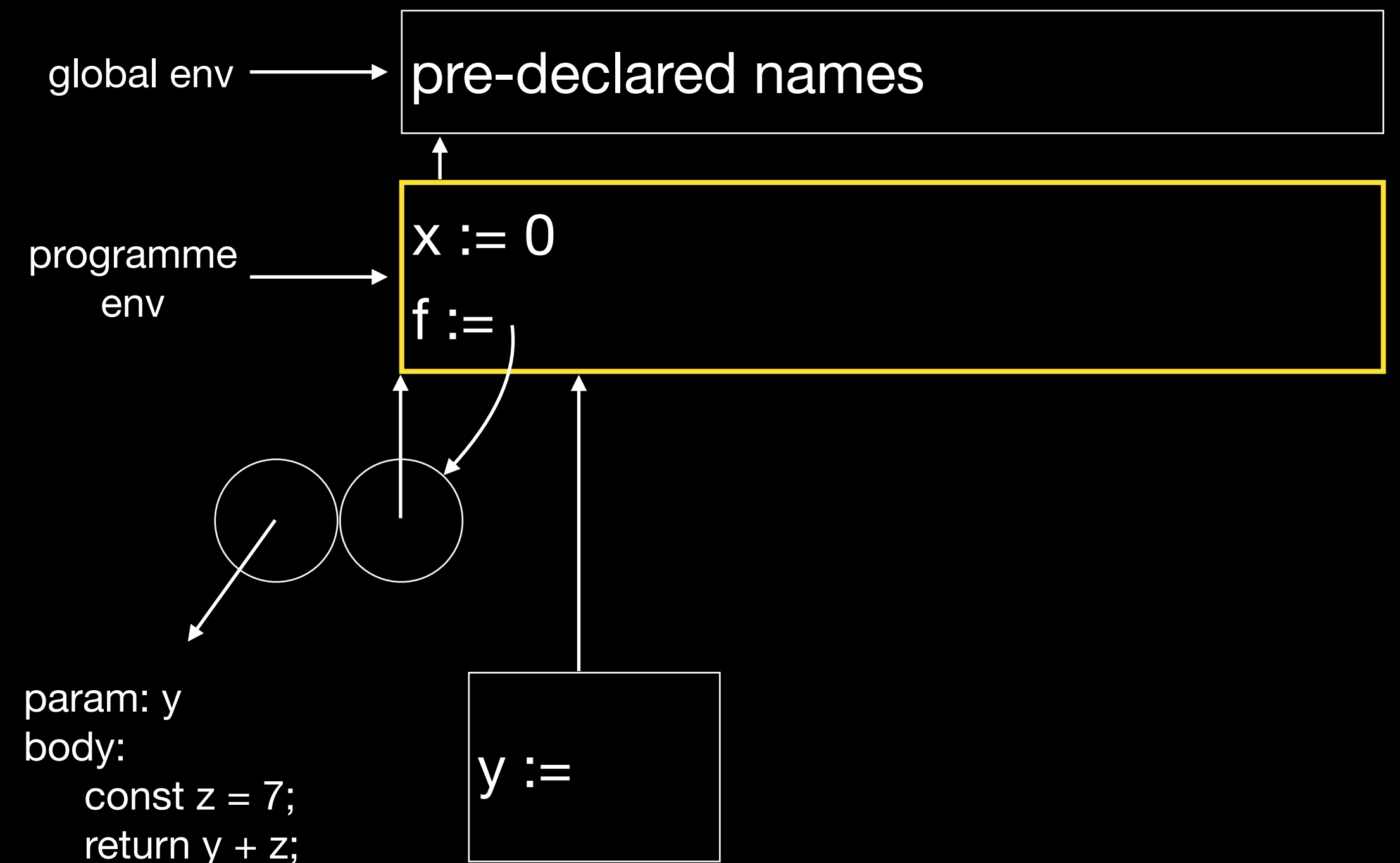
global env ⟶ pre-declared names

programme env ⟶ x := 0
f :=

param: y
body:
    const z = 7;
    return y + z;

y :=

Step 3: extend that frame and create a new frame for params

Recap
Environment Model - Functions

- An example:

```
const x = 0;

function f(y) {

    let z = 7;

    return y + z;

}

if (x < 10) {

    const y = f(x);

} else {}
```
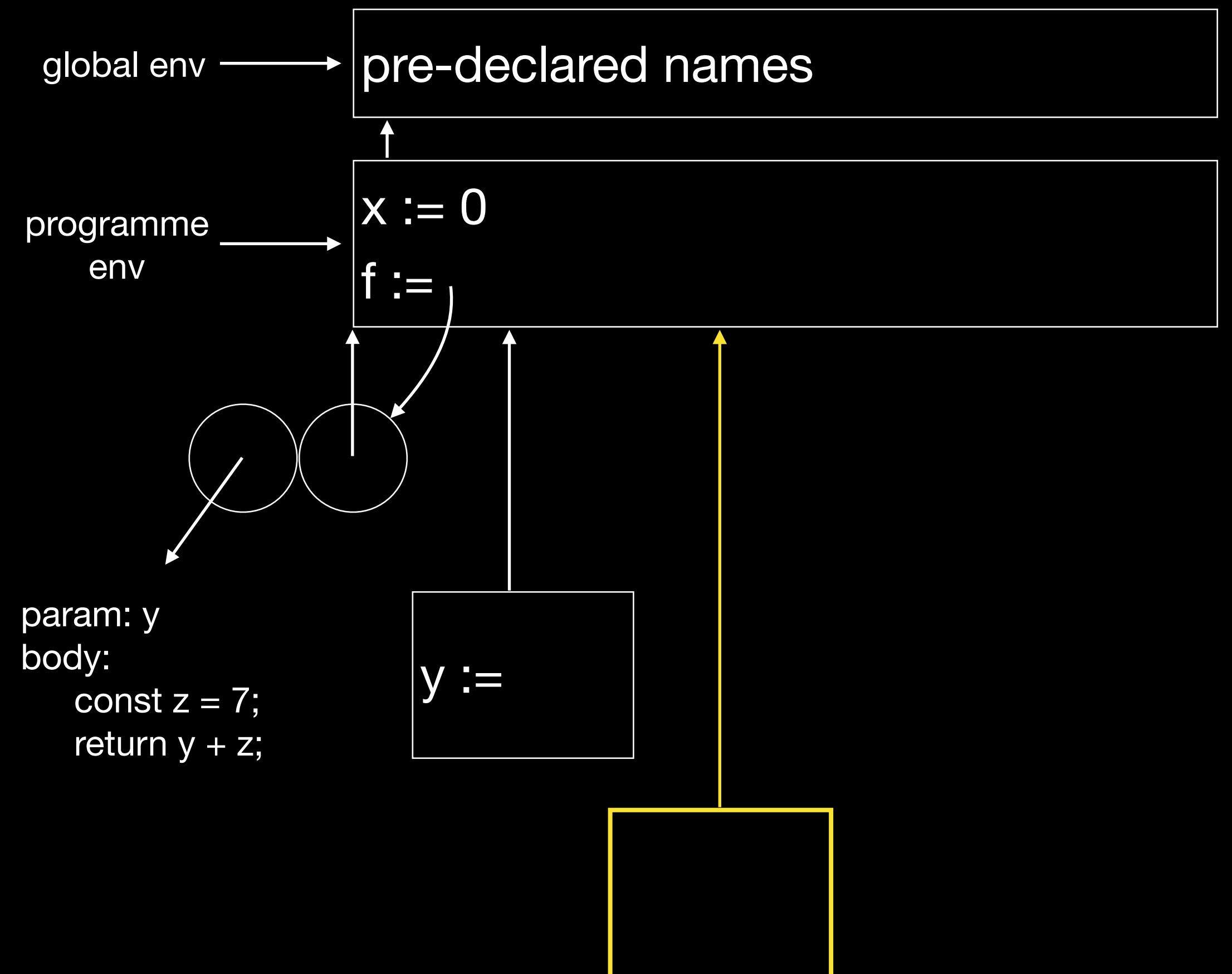
global env → pre-declared names

programme env → x := 0

f :=

param: y
body:
    const z = 7;
    return y + z;

y :=

y : 0

Step 4: bind parameter name to argument value

Recap
Environment Model - Functions

- An example:

```
const x = 0;

function f(y) {

    let z = 7;

    return y + z;

}

if (x < 10) {

    const y = f(x);

} else {}
```
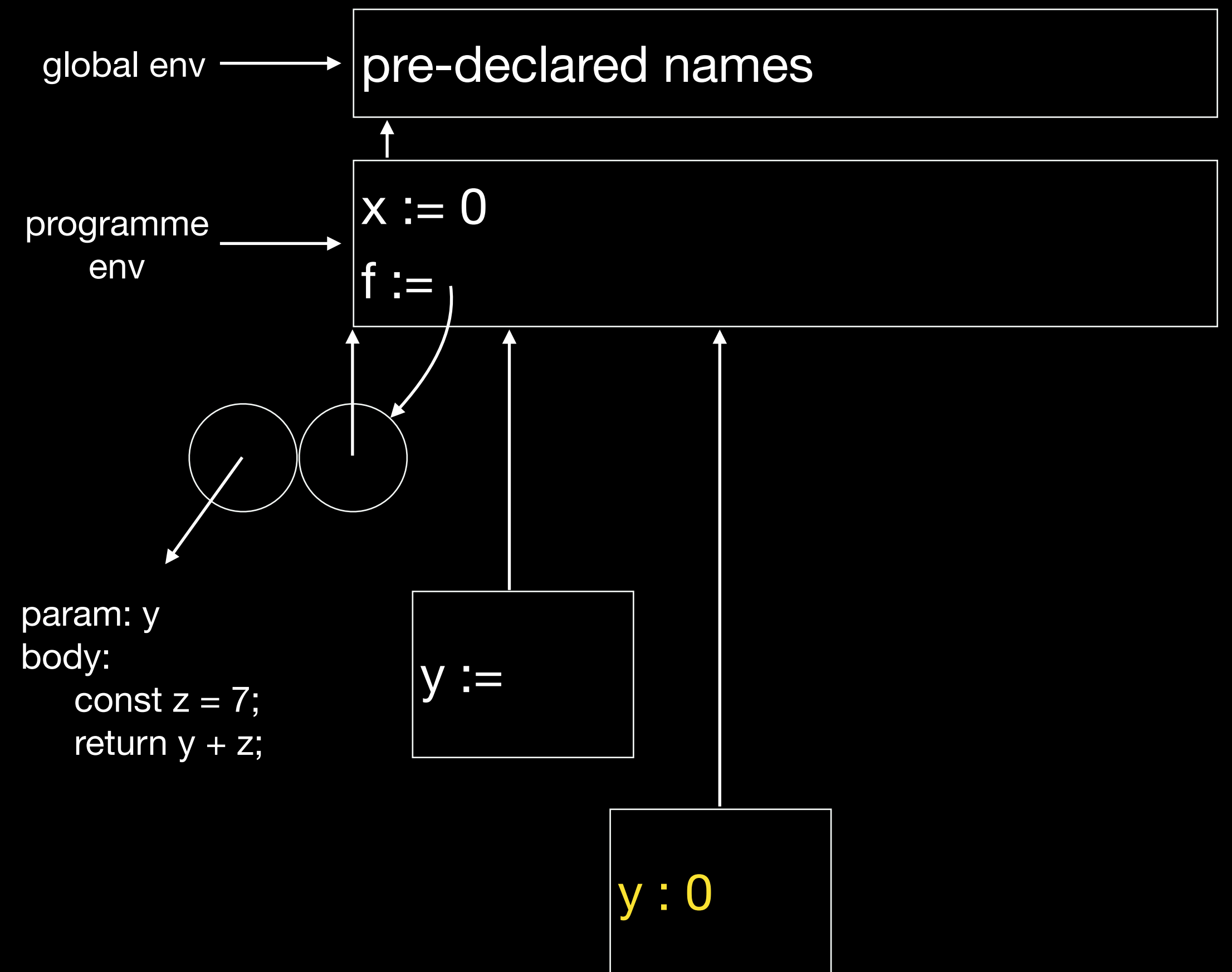
global env ⟶ pre-declared names

programme
env ⟶ x := 0
f :=

param: y
body:
    const z = 7;
    return y + z;

y :=

y : 0

z : 7

Step 5: bind declarations in body

Recap
Environment Model - Functions

- An example:

```
const x = 0;

function f(y) {

    let z = 7;

    return y + z;

}

if (x < 10) {

    const y = f(x);

} else {}
```
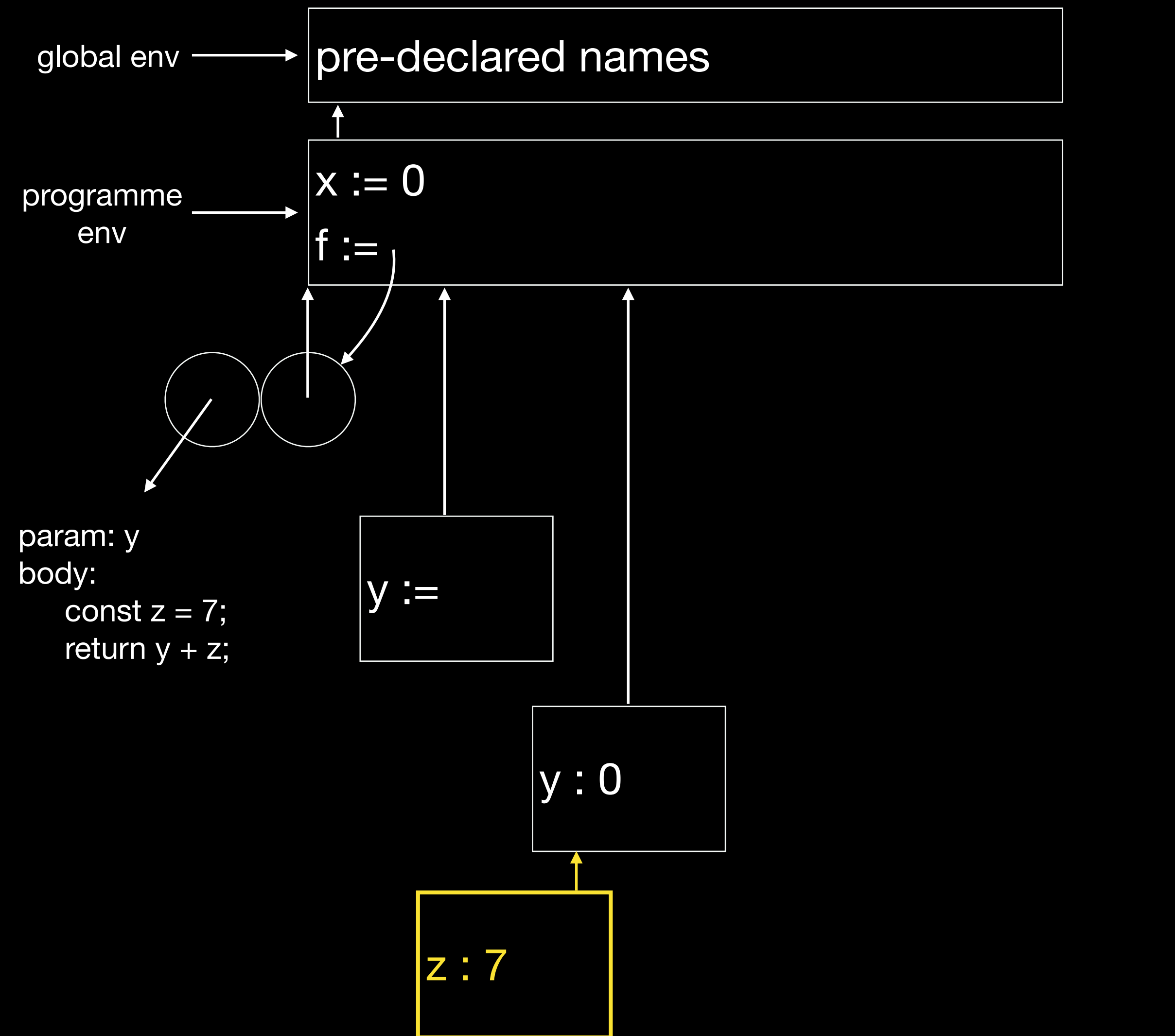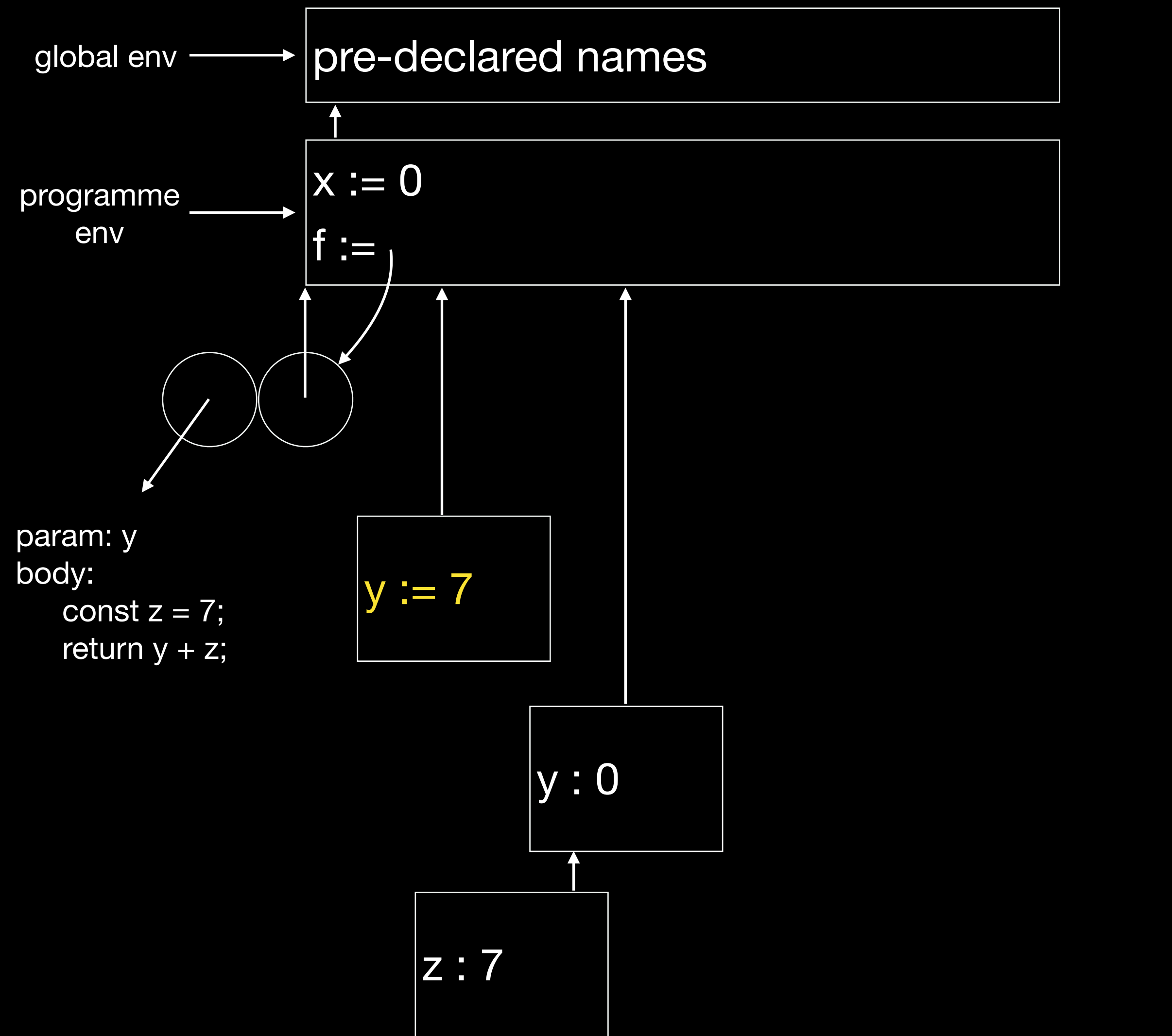
global env ⟶ pre-declared names

programme env ⟶ x := 0
f :=

param: y
body:
    const z = 7;
    return y + z;

y := 7

y : 0

z : 7

Step 6: evaluate and update

Recap
Environment Model - Functions

- When to create new frames?

  - Recall: frames are created to represent name-value bindings

  - Hence, we create frames when there are declarations!

    - If the block has a declaration, extend current environment and add a new frame containing those bindings

    - Else, don't create frames

Recap
Environment Model - Functions

- An example: if the programme has no declarations at all, the "programme env" will not be created at all!

```
{

    display("hello world!");

}



// does not create the programme env
```

Recap
Environment Model

- Shortcomings:

  - The environment model doesn't tell us where the function is called from

    - Only where it should be evaluated in

  - We need to keep track of what the argument value is, and the return value of the function

  - The environment model can't tell you whether a name is redeclared

Any questions?

End of Recap

End of File