

Arrays

Creative Coding & Generative Art in Processing 2
Ira Greenberg, Dianna Xu, Deepak Kumar, SCK

Sequencing

Refers to sequential execution of a program's statements

do this;
then do this;
and then do this;
•etc.

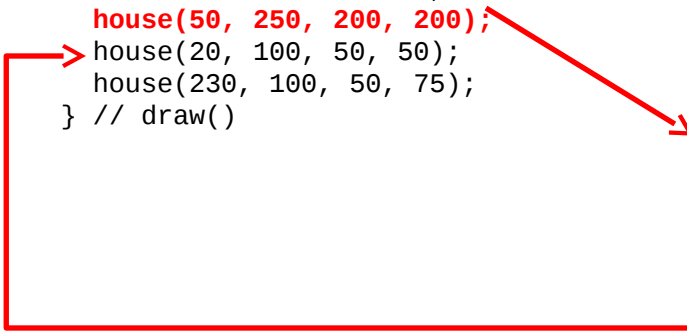
```
size(200,200);  
background(255);  
  
stroke(128);  
rect(20, 20, 40, 40);
```

Function Application

- Control transfers to the function when invoked
- Control returns to the statement following upon return
-

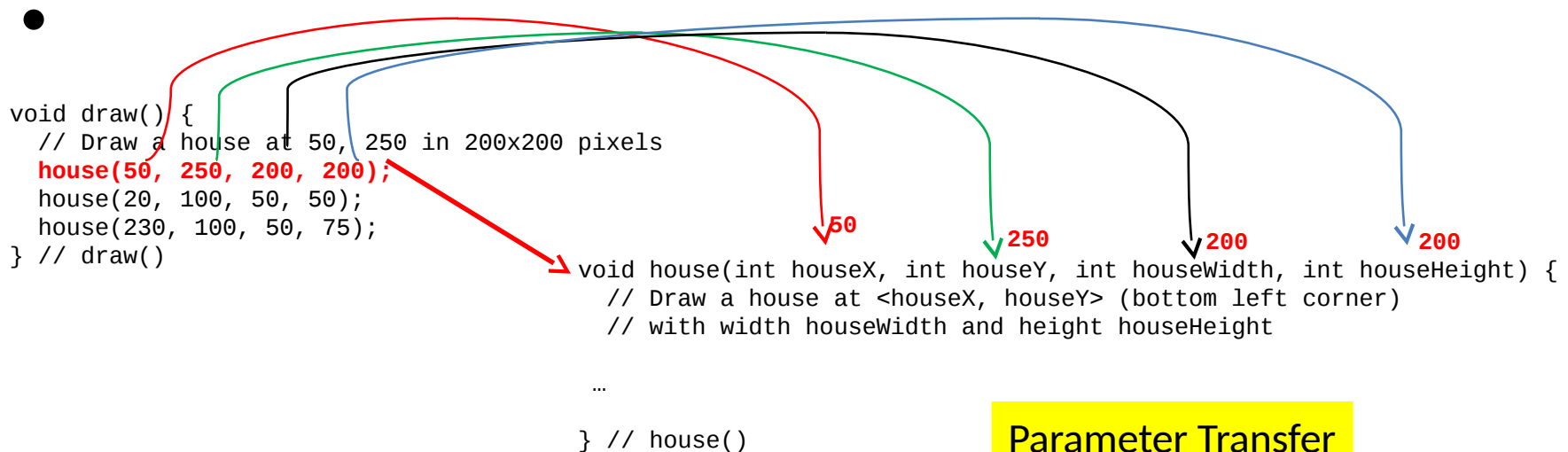
```
void draw() {  
    // Draw a house at 50, 250 in 200x200 pixels  
    house(50, 250, 200, 200);  
    house(20, 100, 50, 50);  
    house(230, 100, 50, 75);  
} // draw()
```

```
void house(int houseX, int houseY, int houseWidth, int houseHeight) {  
    // Draw a house at <houseX, houseY> (bottom left corner)  
    // with width houseWidth and height houseHeight  
    ...  
} // house()
```



Function Application

- Control transfers to the function when invoked
- Control returns to the statement following upon return

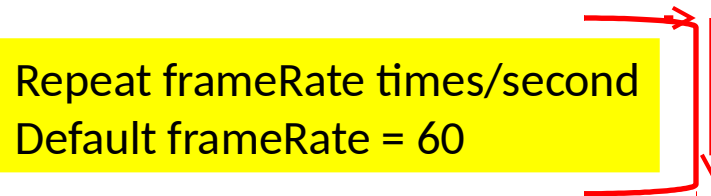


Repetition

Enables repetitive execution of statement blocks

lather
rinse
• repeat

Repeat frameRate times/second
Default frameRate = 60



```
void draw() {  
  do this;  
  then this;  
  and then this;  
  etc.  
} // draw()
```

The diagram illustrates the repetitive execution of a code block. A yellow rectangular box contains the text "Repeat frameRate times/second" and "Default frameRate = 60". Two red arrows originate from the box: one points to the opening curly brace of the `draw()` function, and the other points to the closing curly brace, indicating the scope of the repetition.

Loops: Controlled Repetition

While Loop

```
while (<condition>) {  
    stuff to repeat  
}
```

-

- **Do-While Loop**

-

- ```
do {
 stuff to repeat
} while (<condition>)
```

# Writing Conditions in Processing

Boolean expressions can be written using boolean operators.

Here are some simple expressions...

< less than  $5 < 3$

<= less than/equal to  $x <= y$

== equal to  $x == (y+j)$

!= not equal to  $x != y$

> greater than  $x > y$

• >= greater than/equal to  $x >= y$

# Logical Operations

Combine two or more simple boolean expressions using logical operators:

&&      and       $(x < y) \ \&\& \ (y < z)$

||      or       $(x < y) \ || \ (x < z)$

!      not       $! (x < y)$

| A     | B     | A && B | A    B | !A    |
|-------|-------|--------|--------|-------|
| false | false | false  | false  | true  |
| false | true  | false  | true   | true  |
| true  | false | false  | true   | false |
| true  | true  | true   | true   | false |



# Loops: Critical Components

## Loop initialization

Things to do to set up the repetition

- 
- **Loop Termination Condition**
- 
- When to terminate the loop
- 
- **Loop Body**
- 
- The stuff to be repeated

# Key Computing Ideas

The computer follows a program's instructions.

There are four modes:

- 

- **Sequencing**

- All statements are executed in sequence

- Function Application**

- Control transfers to the function when invoked

- Control returns to the statement following upon return

- Repetition**

- Enables repetitive execution of statement blocks

- Selection**

# Selection: If Statement

```
if (<condition>) {
 do this
}
```

```
if (<condition>) {
 do this
}
else {
 do that
}
```

```
if (<condition>) {
 do this
}
else if (<condition>) {
 do that
}
else if (...) {
 ...
}
else {
 whatever it is you wanna do
}
```

At most ONE block is selected and executed.

# Variables

- `int x = 0;`
- `float delta = 0.483;`
- `color darkOliveGreen = color(85, 107, 47);`
- `String colorName = "Dark Olive Green";`
- `PImage castle = loadImage("myCastle.jpg");`

# A Set of Sample Values

| Petroleum | Coal | Natural Gas | Nuclear | Renewable | Hydropower |
|-----------|------|-------------|---------|-----------|------------|
| 40.0      | 23.0 | 22.0        | 8.0     | 4.0       | 3.0        |

float petroleum = 40.0;

float coal = 23.0;

float naturalGas = 22.0;

float nuclear = 8.0;

float renewable = 4.0;

index →  
float hydropower  
consumption

Declaration



float[] consumption;

consumption = new float[6];



Creation

| 0    | 1    | 2    | 3   | 4   | 5   |
|------|------|------|-----|-----|-----|
| 40.0 | 23.0 | 22.0 | 8.0 | 4.0 | 3.0 |

# A Set of Sample Values

//Declare and create an array with size 6

```
float[] consumption = new float[6];
```

//store values

```
consumption[0] = 40.0;
```

```
consumption[1] = 23.0;
```

```
consumption[2] = 22.0;
```

```
consumption[3] = 8.0;
```

```
consumption[4] = 4.0;
```

```
consumption[5] = 3.0;
```



Fixed size

# A Set of Sample Values

```
//Define, create and initialize the data in an array
float[] consumption = {40.0, 23.0, 22.0, 8.0, 4.0, 3.0};
```

# Arrays

// An array to hold the names of all the days in a week

- `String[] weekDays = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"};`
- `// two arrays, each containing high and low temperature values`
- `float[] highTemps, lowTemps;`
- `int[] count; // an array of integers`
- `PImage[] photos; // an array of photos`
- `// An array to hold the names of months in a`



# Indexing, Size and Loops

```
int[] n = new int[1000];
for (int i=0; i < n.length; i++) {
 n[i] = i;
}
```

```
int[] n = new int[1000];
for (int i= n.length-1; i>=0; i--) {
 n[i] = i;
}
```

# for-each Loop

- Syntax
- `for (variable : arrayName) { // do something with the value of variable }`

—

- Example

```
String[] energySource = {"Petroleum", "Coal", "Natural
Gas", "Nuclear", "Renewable", "Hydropower"};
for(String str : energySource) {
 println(str);
}
```

# Example: A Simple Bar Graph

```
String[] energySource = {"Petroleum", "Coal", "Natural Gas", "Nuclear",
 "Renewable", "Hydropower"};
float[] consumption = {40.0, 23.0, 22.0, 8.0, 4.0, 3.0};
void setup() {
 size(400, 400); smooth();
} // setup()
void draw() { // set up plot dimensions relative to screen size
 float x = width*0.1;
 float y = height*0.9;
 float delta = width*0.8/consumption.length;
 float w = delta*0.8;
 background(255);
 for (float value : consumption) { // draw the bar for value
 // first compute the height of the bar relative to sketch window
 float h = map(value, 0, 100, 0, height);
 fill(0);
 rect(x, y-h, w, h);
 x = x + delta; }
} // draw()
```

# Array Operations

- `String[] energySource = {"Petroleum", "Coal", "Natural Gas", "Nuclear", "Renewable", "Hydropower"};`
- `float[] consumption = {40.0, 23.0, 22.0, 8.0, 4.0, 3.0};`

# Printing

```
println(consumption.length);
println(consumption);
```

```
6
[0 40.0
]
[1 23.0
]
[2 22.0
]
[3 8.0
]
[4 4.0
]
[5 3.0
]
```

```
println(energySource);
```

```
[0 Petroleum
]
[1 Coal
]
[2 Natural Gas
]
[3 Nuclear
]
[4 Renewable
]
[5 Hydropower
]
```

# Try it

Given the following arrays,

- `String[] energySource = {"Petroleum", "Coal", "Natural Gas", "Nuclear", "Renewable", "Hydropower"};`
- `float[] consumption = {40.0, 23.0, 22.0, 8.0, 4.0, 3.0};`

write commands to print the values from `energySource` and `consumption` in the format shown here:

Petroleum, 40.0

Coal, 23.0

Natural Gas, 22.0

Nuclear, 8.0

Renewable, 4.0

Hydropower, 3.0

# Min, Max and Sorting

- `float smallest = min(consumption);`
- `float largest = max(consumption);`
- `println(sort(consumption));`
- `println(sort(energySource));`

# Other Array Operation

- Reverse the ordering of elements in an array  
–reverse()
- Expand the size of the array  
–append(), expand()
- Shorten it  
–shorten()
- Concatenate or split arrays  
–concat(), subset(), splice()
- Copy the contents of an array  
–arrayCopy()



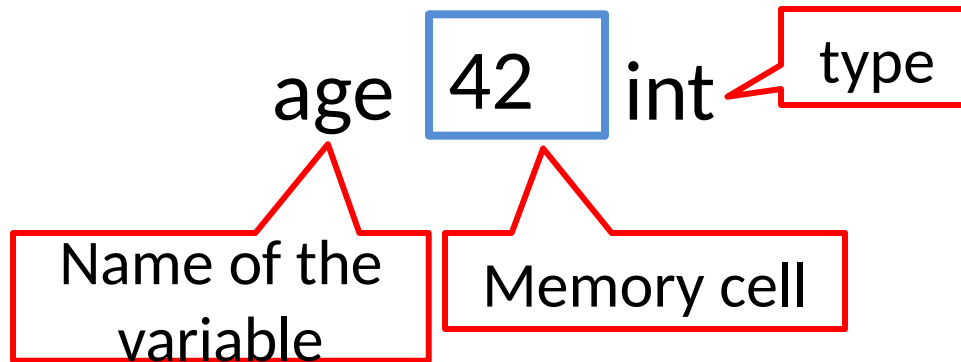
# Variables Types: Primitive Types

- Primitive types

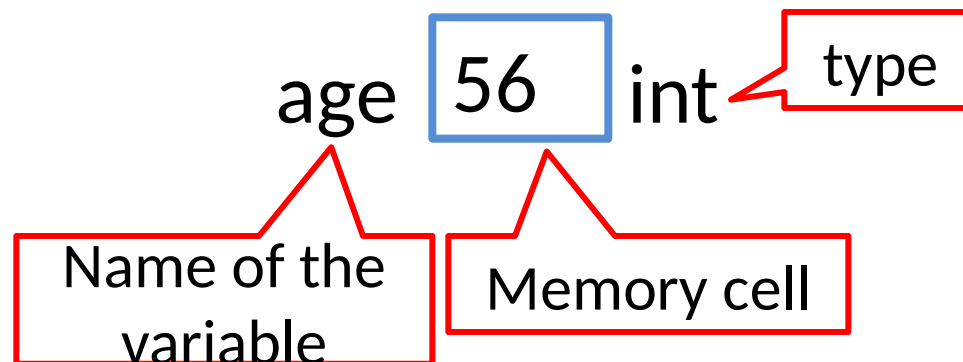
- int, long, short, byte, float, double, char, boolean

- E.g.

- int age = 42;

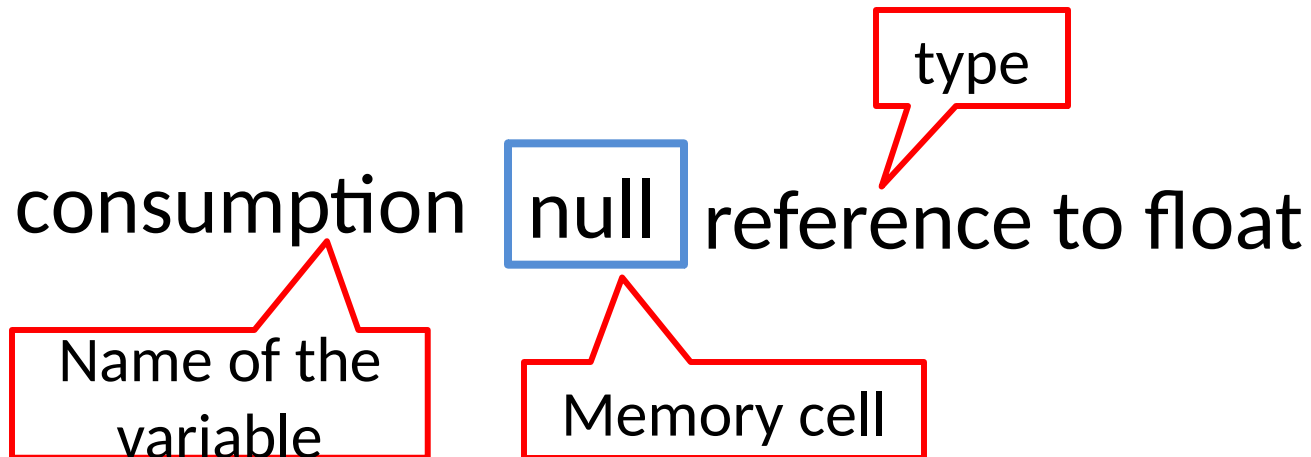


- age = 56;



# Variables Types: References

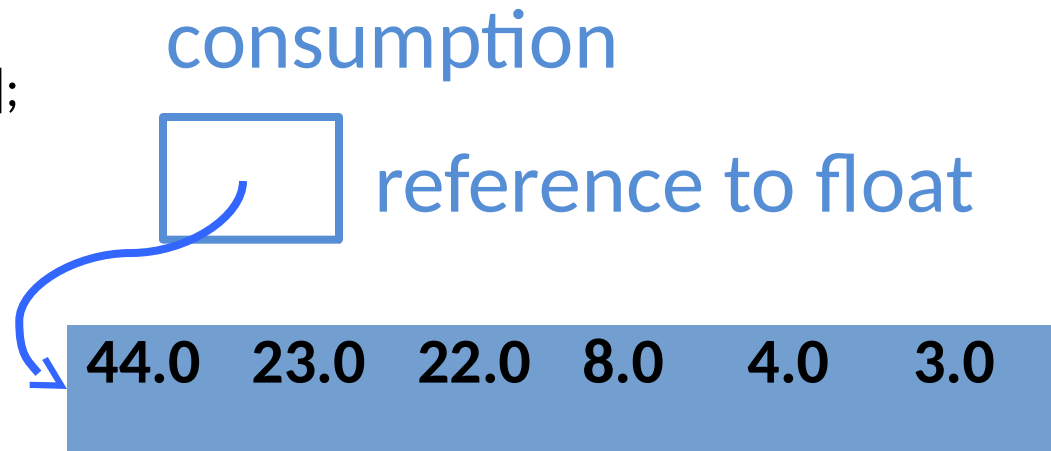
- Reference type
- float[] consumption;  
—



# Variables Types: References

- Reference type

- `consumption = new float[6];`
- `consumption[0] = 44.0;`
- `consumption[0] = 23.0;`
- `consumption[0] = 22.0;`
- `consumption[0] = 8.0;`
- `consumption[0] = 4.0;`
- `consumption[0] = 3.0;`



- The **starting address of the first cell** (that is, the one that becomes `consumption[0]`) is stored in the cell containing the reference to float.

# Reference Variables

- Variables that denote arrays and objects (discussed in Chapter 6 ) are called *reference variables* (or *reference types*).
  - E.g., String, color, and PImage.

# Binding for Primitive Types

- What is the binding for y?
  - `int x = 10;`
  - `int y;`
- `y = x;`

# Binding for Arrays

- What is the result?
- `int[] a = {10, 20, 30};`
- `int[] b;`

`b = a;`

`b[0] = 100;`

`println(a[0]);`

# Arrays as Parameters

```
// Bar Graph using a barGraph() function
String[] energySource = {"Petroleum", "Coal", "Natural Gas", "Nuclear", "Renewable",
 "Hydropower"};
float[] consumption = {40.0, 23.0, 22.0, 8.0, 4.0, 3.0};
void setup() { size(400, 400); smooth(); } // setup()
void draw() { background(255); barGraph(consumption); } // draw()

void barGraph(float[] data) { // set up dimensions relative to screen size
 float x = width*0.1; float y = height*0.9;
 float delta = width*0.8/data.length;
 float w = delta*0.8;
 for (float i : data) { // draw the bar for ith data value
 // first compute the height of the bar relative to sketch window
 float h = map(i, 0, 100, 0, height);
 fill(0); rect(x, y-h, w, h);
 x = x + delta;
 }
} // barGraph()
```