Review

- Expressions and operators
- Iteration
  - while-loop
  - for-loop

Coding styles and assignment hand-ins

- Headers
- Comments
- Indentation
- Parentheses
- Spacing
- Processing’s “Auto Format” command
- Ctrl-R/Ctrl-click

Examples

- text (demo text alignment)
- concentric
- forText
- forCircle
- flowers

for Loop

- Pattern

```
for (init; condition; update) {
  body
}
```

- Each section can be blank.
- Sequence: ①②③④ ②③④① (condition fails)

Examples

```java
for (int i = 0; i < 10; i++) {
  print(i);
} println();
```

```java
for (int i = 0; i < 10; i++) {
  if (i % 2 == 1) continue;
  print(i);
} println();
```

```java
void setup() {
  size(500, 500);
  float diameter = 500;
  while (diameter > 1) {
    ellipse(250, 250, diameter, diameter);
    diameter = diameter - 10;
  }
}
```

```java
void setup() {
  size(500, 500);
  for (float diameter = 500; diameter > 1; diameter -= 10) {
    ellipse(250, 250, diameter, diameter);
  }
}
```
break Statements

• Exit from a loop
• Typically used with an if statement

```java
while (cond) {
  break;
}
```

Example

```java
for(int i=1; i<=100; i++) {
  if (i > 50)
    break;
  println(i);
}
```

continue Statements

• Continue to the beginning of a loop
  – i.e., the condition will be checked
• Typically used with an if statement

```java
while (cond) {
  continue;
}
```

Example

```java
for(int i=1; i<=100; i++) {
  if (i >= 20 && i <= 30)
    continue;
  println(i);
}
```

More on Loops

• Loop index
  – for (int i=0; i<10; i++) {...}
  – start at 0 or 1?
  – stop at <n or <=n?
  – the value of i changes every iteration
• You can run it the other way around too!
  – for (int i=10; i>0; i--) {...}
Functions Informally

- The basic idea – we write a sequence of statements and then give that sequence a name. We can then execute this sequence at any time by referring to the name.
- Function definition: this is where you create a function and define exactly what it does
- Function call: when a function is used in a program, we say call it with its name and parameters.
- A function can only be defined once, but can be called many times.

Examples

```c
void setup() { ... }
void draw() { ... }
```

- Return value, function name, parameter list and function body
- A `void` function doesn’t return anything

```c
void circleAndLine() {
    ellipse(random(width), random(height), 10, 10);
    line(random(width), random(height),
         random(width), random(height));
}
```

Functions

- Modularity
  - Allow the programmer to break down larger programs into smaller parts.
  - Promotes organization and manageability.

- Reuse
  - Enables the reuse of code blocks from arbitrary locations in a program.

Function Example

- `manyShapesFunction`

Mathematical Functions

\[
y = f(x)
\]

\[
y = \text{twice}(x) = 2x
\]

\[
a = \text{area}(r) = \pi r^2
\]

\[
y = f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}
\]

Functions: Terminology

```
y = \text{twice}(x) = 2x
```

Function application:

\[
y = \text{twice}(5) \Rightarrow y = 10
\]
Functions: Defining Functions

\[ y = \text{twice}(x) = 2x \]

```c
float twice(float x) {
    return 2*x;
} // twice()
```

Function Parameters

- Parameters (arguments) can be “passed in” to a function and used in body.
- Parameters are a comma-delimited set of variable declarations.
- Parameters act as input to a function.
- Passing parameters provides a mechanism to execute a function with many different sets of input.
- We can call a function many times and get different results by changing its parameters.

What happens when we call a function?

- Execution of the main (calling) program is suspended.
- The argument expressions are evaluated.
- The resulting values are copied into the corresponding parameters.
- The statements in the function’s body are executed in order.
- Execution of the main program is resumed when a function exits (finishes).

Parameterizing a shape

- Have code that draws something with a bunch of coordinates
- Want to draw the same thing anywhere, in any size and repeat any number of times
- How is a shape defined?
  - a reference point (center, corner)
  - a base size
- To move, scale and repeat
  - put code in a function
  - x and y increments
  - scaling factor