A Foundation for Programming

any program you might want to write

objects

functions and modules

build bigger programs and reuse code

graphics, sound, and image I/O

arrays

conditionals and loops

Math

text I/O

primitive data types

assignment statements

Functions (Static Methods)

Java function.
- Takes zero or more input arguments.
- Returns one output value.
- Side effects (e.g., output to standard draw).

Applications.
- Scientists use mathematical functions to calculate formulas.
- Programmers use functions to build modular programs.
- You use functions for both.

Examples.
- Built-in functions: Math.random(), Math.abs(), Integer.parseInt().
- Our I/O libraries: StdIn.readInt(), StdDraw.line(), StdAudio.play().
- User-defined functions: main().
Java functions. Easy to write your own.

$$f(x) = \sqrt{x}$$

Input: 2.0  Output: 1.414213…

Key point. Functions provide a new way to control the flow of execution.

What happens when a function is called:
- Control transfers to the function code.
- Argument variables are assigned the values given in the call.
- Function code is executed.
- Return value is assigned in place of the function name in calling code.
- Control transfers back to the calling code.

Note. This is known as “pass by value.”

Scope (of a name). The code that can refer to that name. Ex. A variable’s scope is code following the declaration in the block.

Best practice: declare variables to limit their scope.
Function Challenge 1a

Q. What happens when you compile and run the following code?

```java
public class Cubes1 {
    public static int cube(int i) {
        int j = i * i * i;
        return j;
    }
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = 1; i <= N; i++)
            StdOut.println(i + " " + cube(i));
    }
}
```

% javac Cubes1.java
% java Cubes1 6
1 1
2 8
3 27
4 64
5 125
6 216

Function Challenge 1b

Q. What happens when you compile and run the following code?

```java
public class Cubes2 {
    public static int cube(int i) {
        int j = i * i * i;
        return j;
    }
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = 1; i <= N; i++)
            StdOut.println(i + " " + cube(i));
    }
}
```

Function Challenge 1c

Q. What happens when you compile and run the following code?

```java
public class Cubes3 {
    public static int cube(int i) {
        int i = i * i * i;
        return i;
    }
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = 1; i <= N; i++)
            StdOut.println(i + " " + cube(i));
    }
}
```

Function Challenge 1d

Q. What happens when you compile and run the following code?

```java
public class Cubes4 {
    public static int cube(int i) {
        int i = i * i * i;
        return i;
    }
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = 1; i <= N; i++)
            StdOut.println(i + " " + cube(i));
    }
}
```
Function Challenge 1e

Q. What happens when you compile and run the following code?

```java
public class Cubes5 {
    public static int cube(int i) {
        return i * i * i;
    }

    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = 1; i <= N; i++)
            StdOut.println(i + " " + cube(i));
    }
}
```

Gaussian Distribution

- Standard Gaussian distribution.
  - "Bell curve."
  - Basis of most statistical analysis in social and physical sciences.

Ex. 2000 SAT scores follow a Gaussian distribution with mean $\mu = 1019$, stddev $\sigma = 209$.

Mathematical functions. Use built-in functions when possible; build your own when not available.

```java
public class Gaussian {
    public static double phi(double x) {
        return Math.exp(-x*x / 2) / Math.sqrt(2 * Math.PI);
    }

    public static double phi(double x, double mu, double sigma) {
        return phi((x - mu) / sigma) / sigma;
    }
}
```

Overloading. Functions with different signatures are different.

Multiple arguments. Functions can take any number of arguments.

Calling other functions. Functions can call other functions.
Gaussian Cumulative Distribution Function

**Goal.** Compute Gaussian cdf Φ(z).

**Challenge.** No "closed form" expression and not in Java library.

![Gaussian Cumulative Distribution Function](image)

Bottom line. 1,000 years of mathematical formulas at your fingertips.

---

**SAT Scores**

**Q.** NCAA requires at least 820 for Division I athletes. What fraction of test takers in 2000 do not qualify?

**A.** Φ(820, 1019, 209) ≈ 0.17051. [approximately 17%]

![SAT Scores Distribution](image)

\[
\Phi(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \, dx
\]

---

Java function for Φ(z)

```java
public class Gaussian {
    public static double phi(double x) {
        // as before
    }
}
```

accurate with absolute error less than 8 * 10⁻¹⁶

\[
\Phi(z, \mu, \sigma) = \frac{1}{\sqrt{2\pi} \sigma} \int_{-\infty}^{(z-\mu)/\sigma} e^{-t^2/2} \, dt
\]

---

**Gaussian Distribution**

**Q.** Why relevant in mathematics?

**A.** Central limit theorem: under very general conditions, average of a set of random variables tends to the Gaussian distribution.

**Q.** Why relevant in the sciences?

**A.** Models a wide range of natural phenomena and random processes.

- Weights of humans, heights of trees in a forest.
- SAT scores, investment returns.

Caveat.

---

"Everybody believes in the exponential law of errors: the experimenters, because they think it can be proved by mathematics; and the mathematicians, because they believe it has been established by observation."

— M. Lippman in a letter to H. Poincaré
Building Functions

Functions enable you to build a new layer of abstraction.
- Takes you beyond pre-packaged libraries.
- You build the tools you need: \texttt{Gaussian.phi()}, ...

Process.
- Step 1: identify a useful feature.
- Step 2: implement it.
- Step 3: use it.
- Step 3': re-use it in any of your programs.

Crash Course in Sound

Sound. Perception of the vibration of molecules in our eardrums.

Concert A. Sine wave, scaled to oscillate at 440Hz.
Other notes. 12 notes on chromatic scale, divided logarithmically.

<table>
<thead>
<tr>
<th>note</th>
<th>frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>440.00</td>
</tr>
<tr>
<td>A#</td>
<td>466.16</td>
</tr>
<tr>
<td>B</td>
<td>493.88</td>
</tr>
<tr>
<td>C</td>
<td>523.25</td>
</tr>
<tr>
<td>C#</td>
<td>554.37</td>
</tr>
<tr>
<td>D</td>
<td>587.33</td>
</tr>
<tr>
<td>D#</td>
<td>622.25</td>
</tr>
<tr>
<td>E</td>
<td>659.26</td>
</tr>
<tr>
<td>F</td>
<td>698.46</td>
</tr>
<tr>
<td>F#</td>
<td>739.99</td>
</tr>
<tr>
<td>G</td>
<td>783.99</td>
</tr>
<tr>
<td>G#</td>
<td>830.61</td>
</tr>
<tr>
<td>A</td>
<td>880.00</td>
</tr>
</tbody>
</table>

Digital Audio

Sampling. Represent curve by sampling it at regular intervals.

\[ y(t) = \sin\left(\frac{2\pi \cdot t \cdot 440}{44,100}\right) \]

Notes, numbers, and waves
Musical Tone Function

Musical tone. Create a music tone of a given frequency and duration.

```java
public static double[] tone(double hz, double seconds) {   
    int SAMPLE_RATE = 44100;   
    int N = (int) (seconds * SAMPLE_RATE);   
    double[] a = new double[N+1];   
    for (int i = 0; i < N; i++) {   
        a[i] = Math.sin(2 * Math.PI * i * hz / SAMPLE_RATE);   
    }   
    return a;   
}
```

Remark. Can use arrays as function return value and/or argument.

Harmonics

Concert A. Play concert A for 1.5 seconds using StdAudio.

```java
double[] a = tone(440, 1.5); StdAudio.play(a);
```

Concert A with harmonics. Obtain richer sound by adding tones one octave above and below concert A.

Standard audio. Library for playing digital audio.

```java
public class StdAudio {
    void play(String file)  
        play the given .wav file
    void play(double[] a)  
        play sample for 1/44100 second
    void save(String file, double[] a) 
        save to a .wav file
    double[] read(String file) 
        read from a .wav file
}
```

Concert A. Play concert A for 1.5 seconds using stdAudio.
Harmonics

Play that tune. Read in pitches and durations from standard input, and play using standard audio.

```java
public static void main(String[] args) {
    while (!StdIn.isEmpty()) {
        int pitch = StdIn.readInt();
        double duration = StdIn.readDouble();
        double[] a = note(pitch, duration);
        StdAudio.play(a);
    }
}
```

2.2 Libraries and Clients

Library. A module whose methods are primarily intended for use by many other programs.

Client. Program that calls a library.

API. Contract between client and implementation.

Implementation. Program that implements the methods in an API.
The generation of random numbers is far too important to leave to chance. Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.

**Standard Random**

Our library to generate pseudo-random numbers.

```java
public class StdRandom {
   public static double uniform(double a, double b) {
      return a + Math.random() * (b - a);
   }
   public static int uniform(int N) {
      return (int) (Math.random() * N);
   }
   public static boolean bernoulli(double p) {
      return Math.random() < p;
   }
   public static double gaussian() {
      // see Exercise 1.2.27
      return Math.random();
   }
   public static double gaussian(double m, double s) {
      return uniform(0, 1) * s + m;
   }
   public static double gaussian(double m, double s) {
      return uniform(-1, 1) * s * gaussian() + m;
   }
   public static boolean bernoulli(double p) {
      return Math.random() < p;
   }
   public static double uniform(double lo, double hi) {
      return uniform(lo, hi);
   }
   public static int uniform(int lo, int hi) {
      return uniform(lo, hi);
   }
   public static void shuffle(double[] a) {
      // randomly shuffle the array a[]
   }
   public static int getRandNumber() {
      return 4; // chosen by fair dice roll
   }
   // gaussian with mean = 0, stddev = 1
   public static double gaussian() {
      // see Exercise 1.2.27
      return Math.random();
   }
   // gaussian with given mean and stddev
   public static double gaussian(double mean, double stddev) {
      return mean + (stddev * gaussian());
   }
   ...}
```

**Unit Testing**

Unit test. Include `main()` to test each library.

```java
public class StdRandom {
   ... 
   public static void main(String[] args) {
      int N = Integer.parseInt(args[0]);
      for (int i = 0; i < N; i++) {
         StdOut.printf("%d\n", i);
         StdOut.printf("%d\n", bernoulli(.5));
         StdOut.printf("%d\n", bernoulli(.5));
         StdOut.println();
      }
   }
}
```

```bash
% java StdRandom 5
61 21.76541  true 9.30910
57 43.64327 false 9.42369
31 30.86201  true 9.06366
92 39.59314  true 9.00896
36 28.27256 false 8.66800
```
Using a Library

```java
public class RandomPoints {
    public static void main(String args[]) {
        int N = Integer.parseInt(args[0]);
        for (int i = 0; i < N; i++) {
            double x = StdRandom.gaussian(0.5, 0.2);
            double y = StdRandom.gaussian(0.5, 0.2);
            StdDraw.point(x, y);
        }
    }
}
```

% javac RandomPoints.java
% java RandomPoints 10000

Statistics

Ex. Library to compute statistics on an array of real numbers.

```java
public class StdStats {
    public static double max(double[] a) {
        double max = Double.NEGATIVE_INFINITY;
        for (int i = 0; i < a.length; i++) {
            if (a[i] > max) max = a[i];
        }
        return max;
    }

    public static double mean(double[] a) {
        double sum = 0.0;
        for (int i = 0; i < a.length; i++)
            sum = sum + a[i];
        return sum / a.length;
    }

    public static double stddev(double[] a) {
        // see text
    }
}
```

Ex. Library to compute statistics on an array of real numbers.

\[
\mu = \frac{a_0 + a_1 + \cdots + a_{n-1}}{n}, \quad \sigma^2 = \frac{(a_0 - \mu)^2 + (a_1 - \mu)^2 + \cdots + (a_{n-1} - \mu)^2}{n - 1}
\]
Modular Programming

- Divide program into self-contained pieces.
- Test each piece individually.
- Combine pieces to make program.

Ex. Flip N coins. How many heads?
- Read arguments from user.
- Flip one fair coin.
- Flip N fair coins and count number of heads.
- Repeat simulation, counting number of times each outcome occurs.
- Plot histogram of empirical results.
- Compare with theoretical predictions.

```java
public class Bernoulli {
    public static int binomial(int N) {
        int heads = 0;
        for (int j = 0; j < N; j++)
            if (StdRandom.bernoulli(0.5)) heads++;
        return heads;
    }

    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        int T = Integer.parseInt(args[1]);
        int[] freq = new int[N+1];
        for (int i = 0; i < T; i++)
            freq[binomial(N)]++; // perform T trials of N coin flips each
        double[] normalized = new double[N+1];
        for (int i = 0; i <= N; i++)
            normalized[i] = (double) freq[i] / T;
        StdStats.plotBars(normalized);
        double mean = N / 2.0, stddev = Math.sqrt(N) / 2.0;
        double[] phi = new double[N+1];
        for (int i = 0; i <= N; i++)
            phi[i] = Gaussian.phi(i, mean, stddev);
        StdStats.plotLines(phi, theoretical prediction);
    }
}
```

Bernoulli Trials

Dependency Graph

Modular programming. Build relatively complicated program by combining several small, independent, modules.
Libraries

Why use libraries?
- Makes code easier to understand.
- Makes code easier to debug.
- Makes code easier to maintain and improve.
- Makes code easier to reuse.