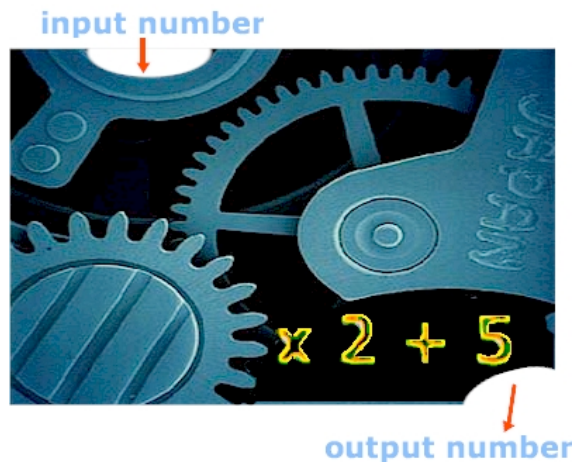


2.1 Functions



Functions (Static Methods)

Java function.

- Takes zero or more input arguments.
- Returns zero or one output value.
- May cause side effects (e.g., output to standard draw).

more general than mathematical functions

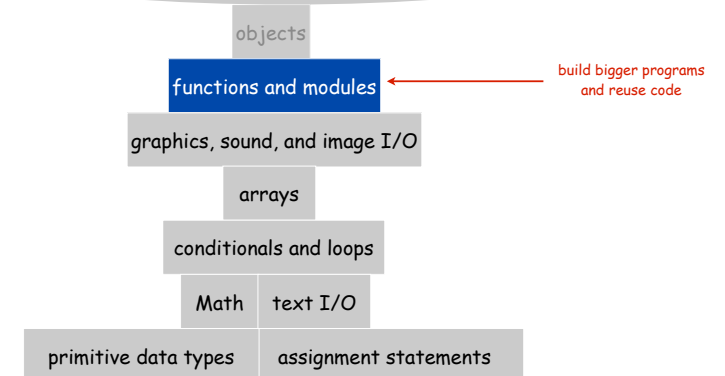
Applications.

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

Examples.

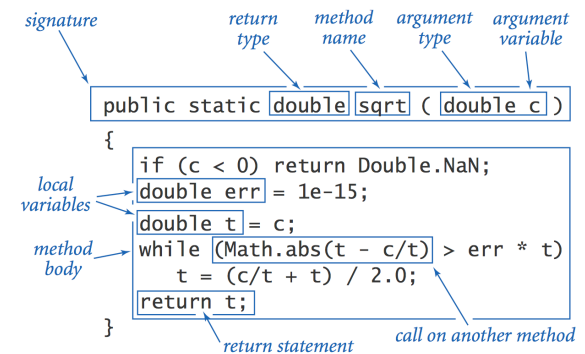
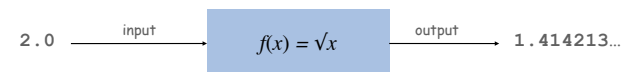
- Built-in functions: `Math.random()`, `Math.abs()`, `Integer.parseInt()`.
- Our I/O libraries: `StdIn.readInt()`, `StdDraw.line()`, `StdAudio.play()`.
- User-defined functions: `main()`.

any program you might want to write



Anatomy of a Java Function

Java functions. Easy to write your own.



```
public class Gambler {
    public static void main(String[] args) {
        int stake = Integer.parseInt(args[0]);
        int goal = Integer.parseInt(args[1]);
        int trials = Integer.parseInt(args[2]);
        . . .
    }
}
```

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

```
public class Newton
{
    public static double sqrt(double c)
    {
        double epsilon = 1e-15;
        if (c < 0) return Double.NaN;
        double t = c;
        while (Math.abs(t - c/t) > epsilon * t)
            t = (c/t + t) / 2.0;
        return t;
    }

    public static void main(String[] args)
    {
        double[] a = new double[args.length];
        for (int i = 0; i < args.length; i++)
            a[i] = Double.parseDouble(args[i]);
        for (int i = 0; i < a.length; i++)
        {
            double x = sqrt(a[i]);
            StdOut.println(x);
        }
    }
}
```

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

Summary of 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 the calling code.
- Control transfers back to the calling code.

Note. This technique (standard in Java) is known as "pass by value".

↑
other languages may use different methods

Scope (of a name). The code that can refer to that name.

Def. A variable's scope is code following the declaration in its block.

```
public class Newton
{
    public static double sqrt(double c)
    {
        double epsilon = 1e-15;
        if (c < 0) return Double.NaN;
        double t = c;
        while (Math.abs(t - c/t) > epsilon * t)
            t = (c/t + t) / 2.0;
        return t;
    }

    public static void main(String[] args)
    {
        double[] a = new double[args.length];
        for (int i = 0; i < args.length; i++)
            a[i] = Double.parseDouble(args[i]);
        for (int i = 0; i < a.length; i++)
            System.out.println(sqrt(a[i]));
    }
}
```

two different variables
with the same name i
each with two lines of scope

Best practice: declare variables so as to limit their scope.

Function Call Trace

```
public class Newton
{
    public static double sqrt(double c)
    {
        double epsilon = 1e-15;
        if (c < 0) return Double.NaN;
        double t = c;
        while (Math.abs(t - c/t) > epsilon * t)
            t = (c/t + t) / 2.0;
        return t;
    }

    public static void main(String[] args)
    {
        double[] a = new double[args.length];
        for (int i = 0; i < args.length; i++)
            a[i] = Double.parseDouble(args[i]);
        for (int i = 0; i < a.length; i++)
            System.out.println(sqrt(a[i]));
    }
}
```

Functions Challenge 1

What happens when you compile and run the following code?

```
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));
    }
}
```

Functions Challenge 2

What happens when you compile and run the following code?

```
public class Cubes2
{
    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));
    }
}
```

Functions Challenge 3

What happens when you compile and run the following code?

```
public class Cubes3
{
    public static int cube(int i)
    {
        i = 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));
    }
}
```

Functions Challenge 4

What happens when you compile and run the following code?

```
public class Cubes4
{
    public static int cube(int i)
    {
        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));
    }
}
```

Functions Challenge 5

What happens when you compile and run the following code?

```
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));
    }
}
```

Example: Gaussian Distribution

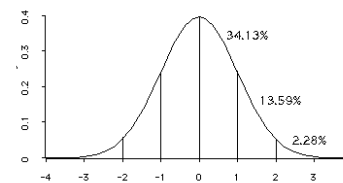


Gaussian Distribution

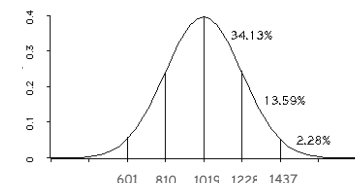
Standard Gaussian distribution.

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

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



$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$



$$\begin{aligned} \phi(x, \mu, \sigma) &= \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2 / 2\sigma^2} \\ &= \phi\left(\frac{x-\mu}{\sigma}\right) / \sigma \end{aligned}$$

Java Function for $\phi(x)$

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

```
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;
    }
}
```

$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$

$\phi(x, \mu, \sigma) = \phi\left(\frac{x-\mu}{\sigma}\right) / \sigma$

Overloading. Functions with different signatures are different.

Multiple arguments. Functions can take any number of arguments.

Calling other functions. Functions can call other functions. library or user-defined

Java function for $\Phi(z)$

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

    public static double Phi(double z)
    {
        if (z < -8.0) return 0.0;
        if (z > 8.0) return 1.0;
        double sum = 0.0, term = z;
        for (int i = 3; sum + term != sum; i += 2) {
            sum = sum + term;
            term = term * z * z / i;
        }
        return 0.5 + sum * phi(z);
    }

    public static double Phi(double z, double mu, double sigma)
    {
        return Phi((z - mu) / sigma);
    }
}
```

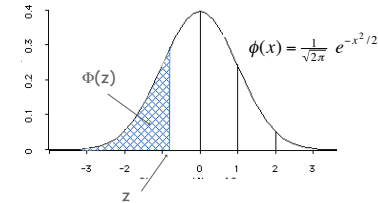
accurate with absolute error less than $8 * 10^{-16}$

$\Phi(z, \mu, \sigma) = \int_{-\infty}^z \phi(z, \mu, \sigma) = \Phi((z - \mu) / \sigma)$

Gaussian Cumulative Distribution Function

Goal. Compute Gaussian cdf $\Phi(z)$.

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



$$\Phi(z) = \int_{-\infty}^z \phi(x) dx \quad \text{Taylor series}$$

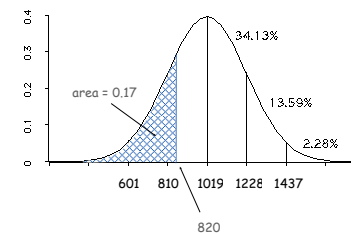
$$= \frac{1}{2} + \phi(z) \left(z + \frac{z^3}{3} + \frac{z^5}{3 \cdot 5} + \frac{z^7}{3 \cdot 5 \cdot 7} + \dots \right)$$

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 did not qualify?

A. $\Phi(820, \mu, \sigma) \approx 0.17051$. [approximately 17%]



```
double fraction = Gaussian.Phi(820, 1019, 209);
```

Gaussian Distribution

Q. Why relevant in mathematics?

A. Central limit theorem: under very general conditions, average of a set of 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

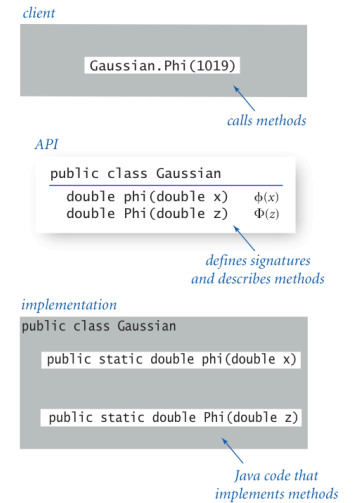
Libraries

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

Client. Program that calls library method(s).

API. Contract between client and implementation.

Implementation. Program that implements the methods in an API.



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.

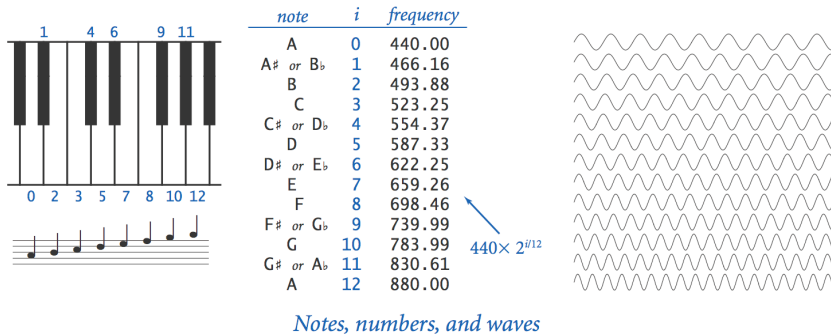
Digital Audio

Crash Course in Sound

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

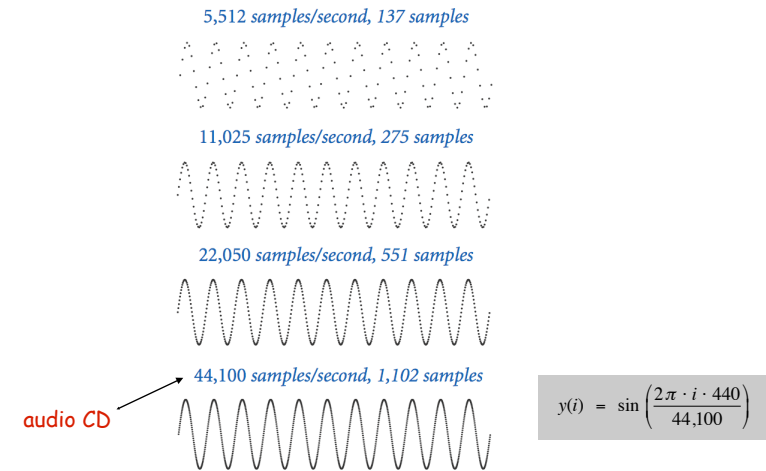
Concert A. Sine wave, scaled to oscillated at 440Hz.

Other notes. 12 notes on chromatic scale, divided logarithmically.



Digital Audio

Sampling. Represent curve by sampling it at regular intervals.



Warmup: Musical Tone

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

```
public class Tone
{
    public static void main(String[] args)
    {
        int sps = 44100;
        double hz = Double.parseDouble(args[0]);
        double duration = Double.parseDouble(args[1]);
        int N = (int) (sps * duration);
        double[] a = new double[N+1];
        for (int i = 0; i <= N; i++)
            a[i] = Math.sin(2 * Math.PI * i * hz / sps);
        StdAudio.play(a);
    }
}
```

```
% java Tone 440 1.5
[ concert A for 1.5 seconds]
```

$$y(i) = \sin\left(\frac{2\pi \cdot i \cdot hz}{44,100}\right)$$



Play That Tune

Goal. Play pitches and durations from standard input on standard audio.

```
public class PlayThatTune
{
    public static void main(String[] args)
    {
        int sps = 44100;
        while (!StdIn.isEmpty())
        {
            int pitch = StdIn.readInt();
            double duration = StdIn.readDouble();
            int N = (int) (sps * duration);
            double[] a = new double[N+1];
            for (int i = 0; i <= N; i++)
                a[i] = Math.sin(2 * Math.PI * i * hz / sps);
            StdAudio.play(a);
        }
    }
}
```

```
% more elise.txt
7 .125
6 .125
7 .125
6 .125
2 .125
5 .125
3 .125
0 .25
...
```

```
% java PlayThatTune < elise.txt
```



Musical Tone Function

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

```
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;
}
```

$$y(i) = \sin\left(\frac{2\pi \cdot i \cdot \text{hz}}{44,100}\right)$$

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

Digital Audio in Java

Standard audio. Library for playing digital audio.

| | |
|---|---------------------------------------|
| <code>public class StdAudio</code> | |
| <code>void play(String file)</code> | <i>play the given .wav file</i> |
| <code>void play(double[] a)</code> | <i>play the given sound wave</i> |
| <code>void play(double x)</code> | <i>play sample for 1/44100 second</i> |
| <code>void save(String file, double[] a)</code> | <i>save to a .wav file</i> |
| <code>double[] read(String file)</code> | <i>read from a .wav file</i> |

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

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



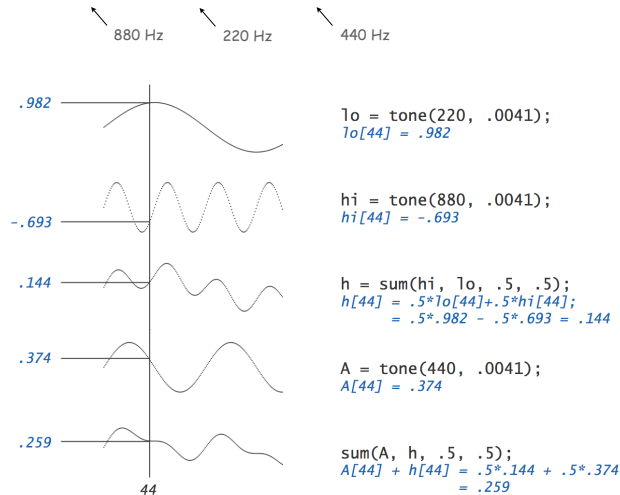
Remark. Java arrays passed "by reference" (no copy made).

library developed
for this course
(also broadly useful)



Harmonics

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



Harmonics

```
public class PlayThatTune // improved version with Harmonics
{
    // Return weighted sum of two arrays.
    public static double[] sum(double[] a, double[] b, double awt, double bwt) {
        double[] c = new double[a.length];
        for (int i = 0; i < a.length; i++)
            c[i] = a[i]*awt + b[i]*bwt;
        return c;
    }

    // Return a note of given pitch and duration.
    public static double[] note(int pitch, double duration) {
        double hz = 440.0 * Math.pow(2, pitch / 12.0);
        double[] a = tone(1.0 * hz, duration);
        double[] hi = tone(2.0 * hz, duration);
        double[] lo = tone(0.5 * hz, duration);
        double[] h = sum(hi, lo, .5, .5);
        return sum(a, h, .5, .5);
    }

    public static double[] tone(double hz, double t)
        // see previous slide

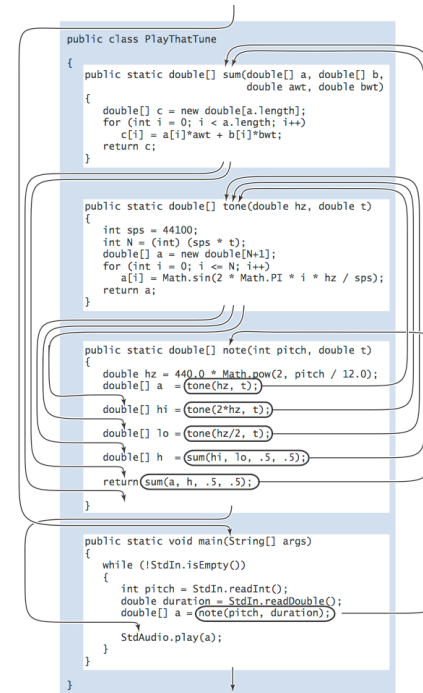
    public static void main(String[] args)
        // see next slide
}
```


Harmonics

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

```
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);
    }
}
```

```
% more elise.txt           % java PlayThatTune < elise.txt
7 .125
6 .125
7 .125
6 .125
7 .125
2 .125
5 .125
3 .125
0 .25
```



L. V. BEETHOVEN - FÜR ELISE

Both Hands Right Hand Left Hand Slow

WWW.YOUCANPLAYIT.COM