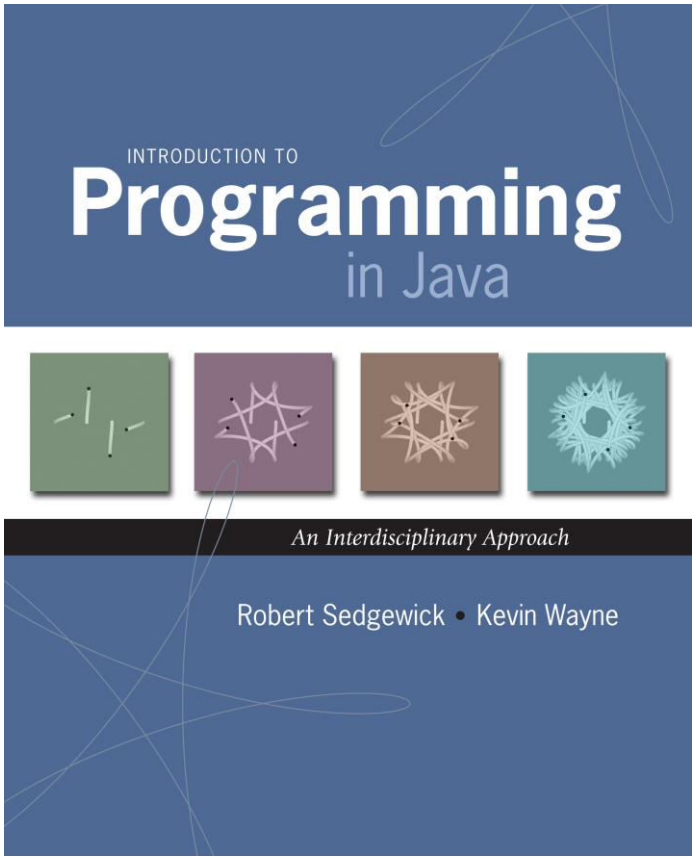
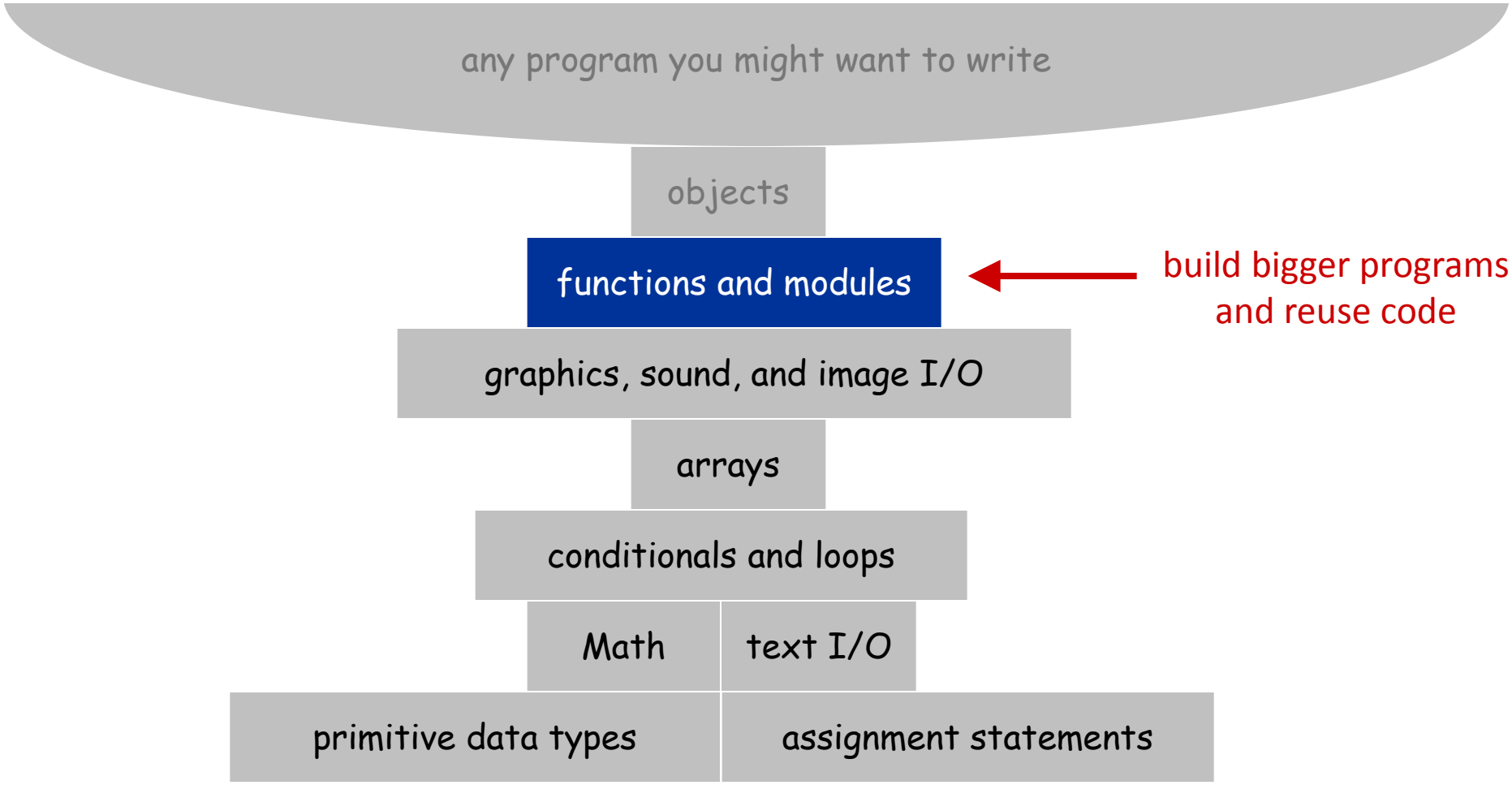
# 2.1 Functions



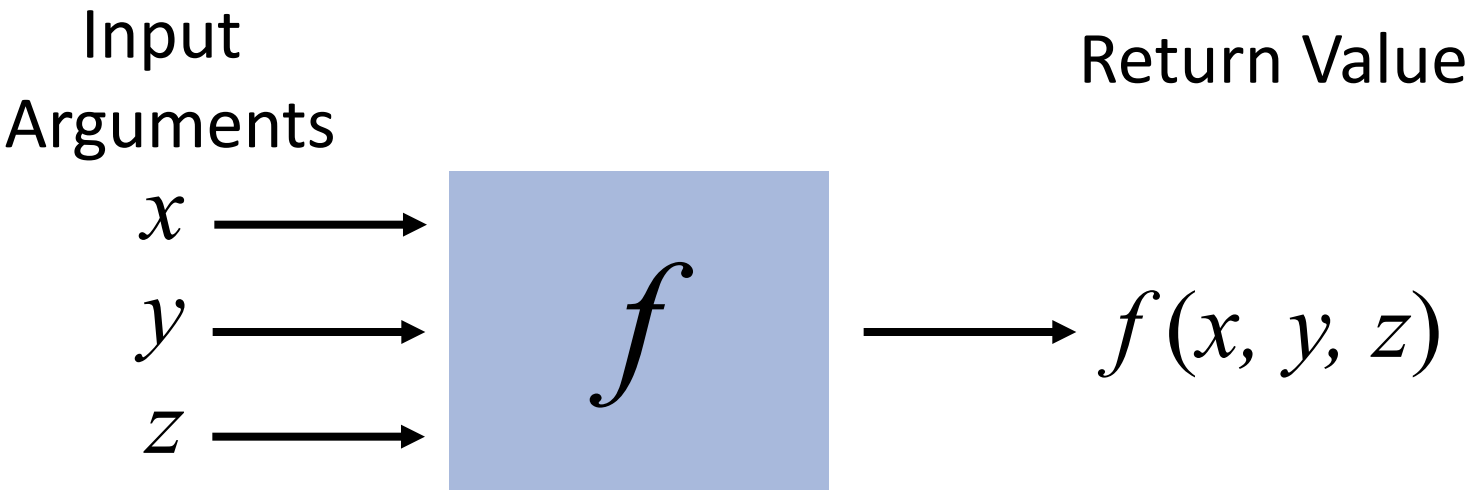
# A Foundation for Programming




Penn Engineering

# Functions

- Take in input arguments (zero or more)
- Perform some computation
  - May have side-effects (such as drawing)
- Return one output value



# Functions (Static Methods)

- Applications:
  - Use mathematical functions to calculate formulas
  - Use functions to build modular programs

- Examples:
  - Built-in functions:

    ```
    Math.random(), Math.abs(), Integer.parseInt()
    ```

  - I/O libraries:

    ```
    StdDraw.circle(), StdDraw.show()
    ```

  - User-defined functions:

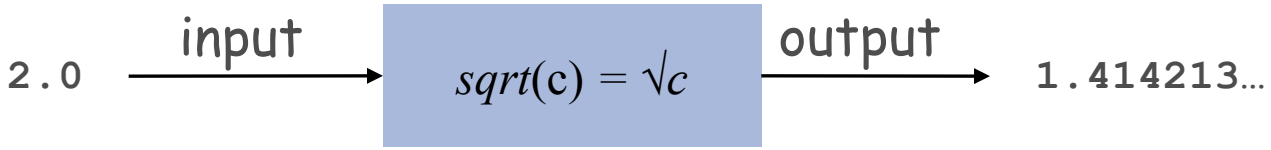
    ```
    main()
    ```

# Modularity

- Breaking programs into smaller pieces
- Avoid copy pasted code
- Each function can be checked separately

# Anatomy of a Java Function

- Java functions – It is easy to write your own
  - Example: `double sqrt(double c)`




```
public static double sqrt(double c) {
    ...
}
```

*return type* — `double`

*method name* — `sqrt`

*arguments* — `(double c)`

*method signature* (excludes return type) — `sqrt(double c)`

Please note that the method's signature is defined incorrectly in the figure on pg 188 of your textbook

Penn Engineering

# Anatomy of a Java Function

Example - absolute value

- think of what the function is doing. Finding absolute value
- give it a name. Name of a function = important!! abs
- What goes 'into' the function. What are the arguments . What type are the arguments?
  single integer. abs (int x)
- What does it return? what datatype comes out of the function.
  Another integer. int abs (int x)
- Put public static before it and create a method body

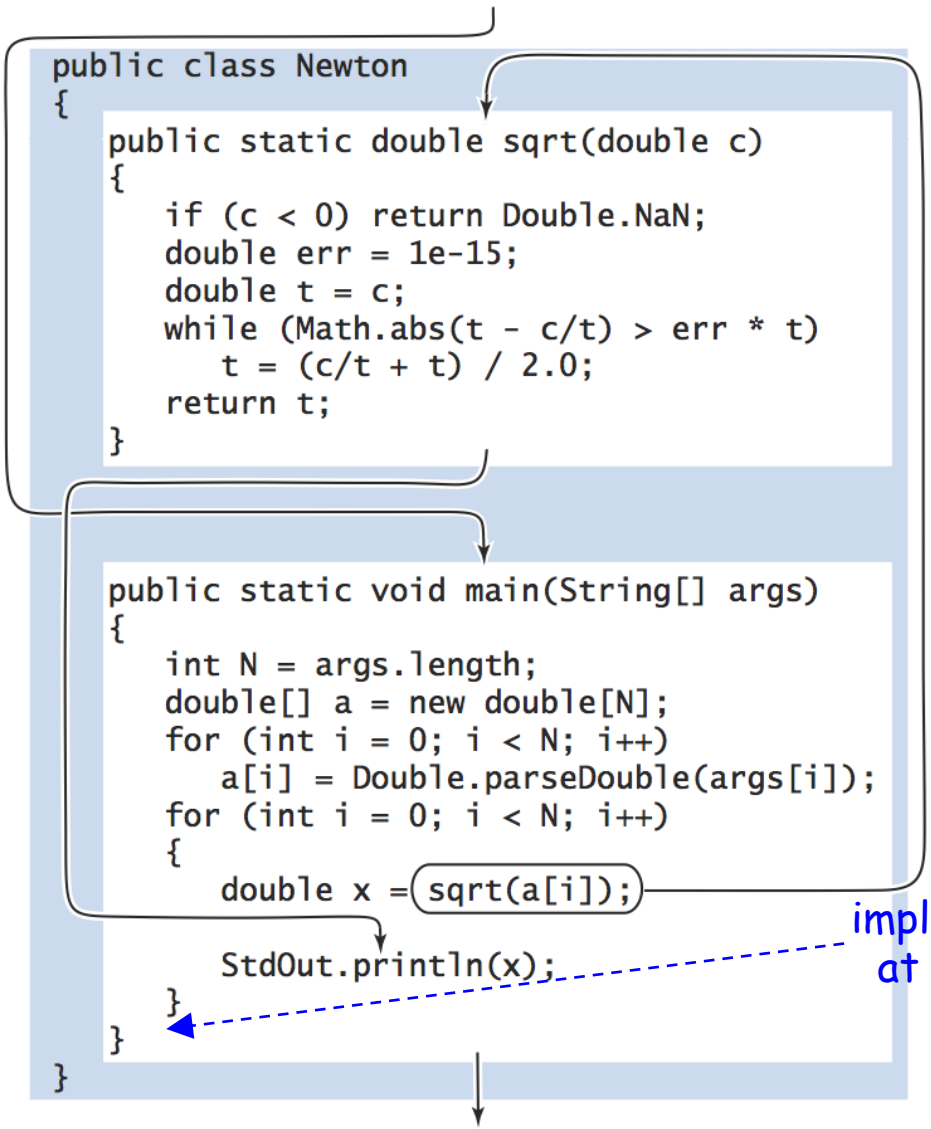
```
public static int abs(int x){

        // fill your code here
}
```

- Do not forget a **return** statement

# Flow of Control

Functions provide a new way to control the flow of execution




The sqrt function is written here using the Newton method. Do not worry about the actual code inside sqrt

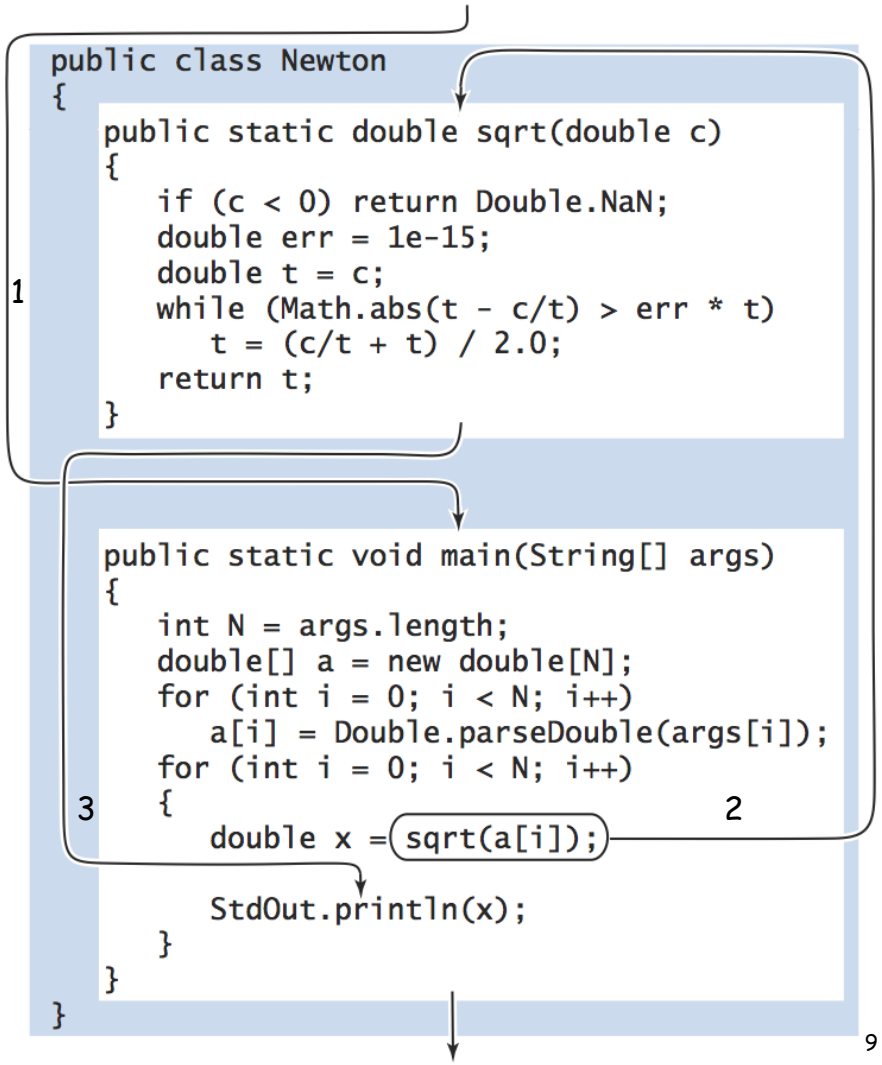
implicit return statement at end of void function

# Flow of Control

What happens when a function is called:

- Control transfers to the function
- Argument variables are assigned the values given in the call
- Function code is executed
- Return value is substituted in place of the function call in the calling code
- Control transfers back to the calling code

Note: This is known as "pass by value"



# Scope

Scope: the code that can refer to a particular variable

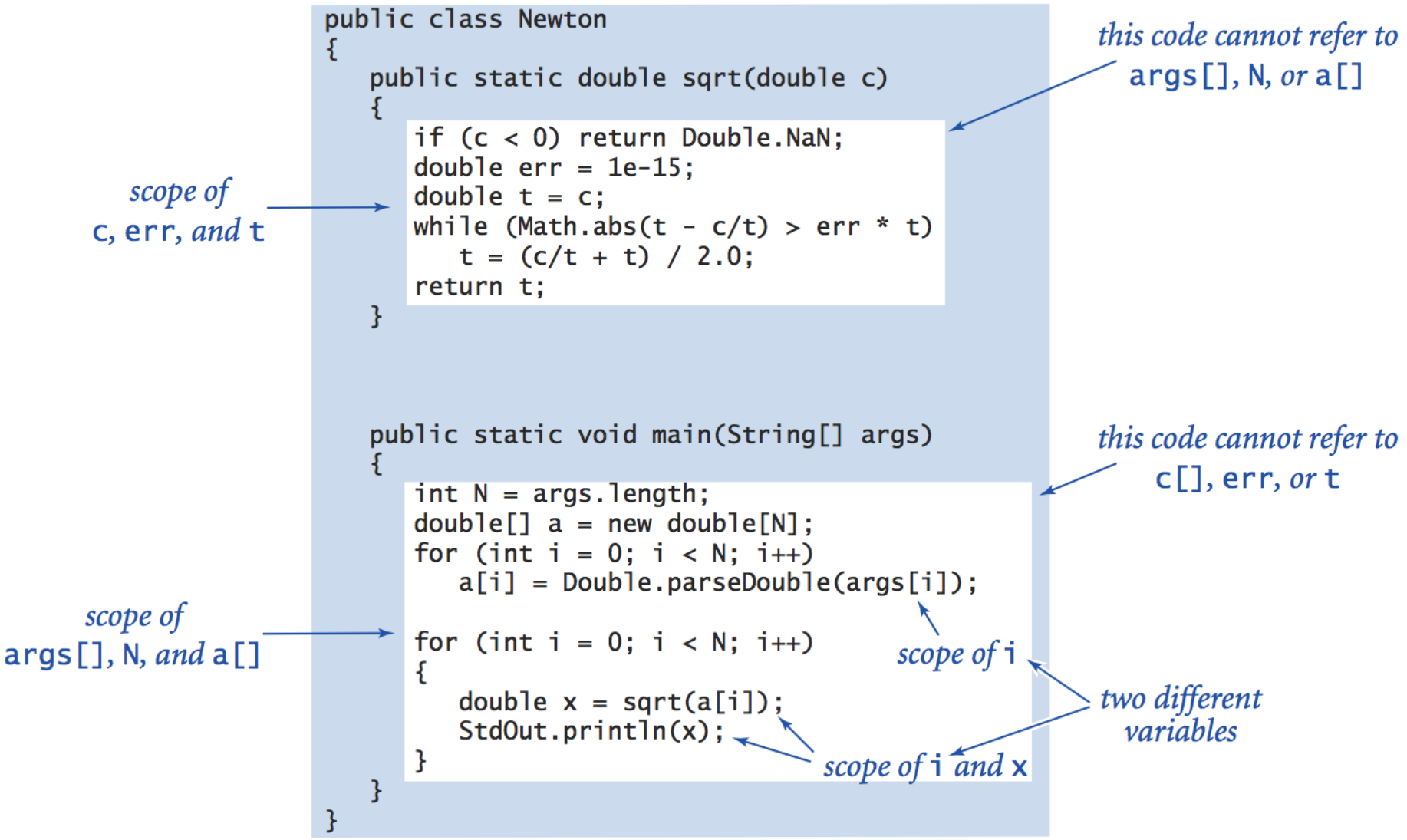
- A variable's scope is the entire code block (any any nested blocks) after its declaration

Simple example:

```
int count = 1;
for (int i = 0; i < 10; i++){
      count *= 2;
}
// using 'i' here generates
// a compiler error
```

Best practice: declare variables to limit their scope

# Scope with Functions



# Tracing Functions

```
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++)
         System.out.println(i + " " + cube(i));
   }
}
```

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

Penn Engineering

# Function Examples

*absolute value of an* `int` *value*

```
public static int abs(int x)
{
   if (x < 0) return -x;
   else       return  x;
}
```

*absolute value of a* `double` *value*

```
public static double abs(double x)
{
   if (x < 0.0) return -x;
   else         return  x;
}
```

overloading

*primality test*

```
public static boolean isPrime(int N)
{
   if (N < 2) return false;
   for (int i = 2; i <= N/i; i++)
      if (N % i == 0) return false;
   return true;
}
```

multiple arguments

*hypotenuse of a right triangle*

```
public static double hypotenuse(double a, double b)
{  return Math.sqrt(a*a + b*b);  }
```

Penn Engineering

# What if a function doesn't return anything?

- void - special word that says this function will not return anything
- If your return type is void you do not need a return statement
- Remember public static void main(String[] args)?

# What if a function doesn't return anything?

- functions that just print something
- functions that just draw . Most (all?) of your StdDraw functions
- Every function declaration has to have something as its return type! If a function is not returning anything you DO have to say 'void'

# Function Challenge 1a

Q. 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++)
         System.out.println(i + " " + cube(i));
   }
}
```

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

Penn Engineering

# Function Challenge 1b

Q. 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++)
         System.out.println(i + " " + cube(i));
   }
}
```

Penn Engineering

# Function Challenge 1c

Q. 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++)
         System.out.println(i + " " + cube(i));
   }
}
```

# Function Challenge 1d

Q. 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++)
         System.out.println(i + " " + cube(i));
   }
}
```

# Function Challenge 1e

Q. 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++)
         System.out.println(i + " " + cube(i));
   }
}
```