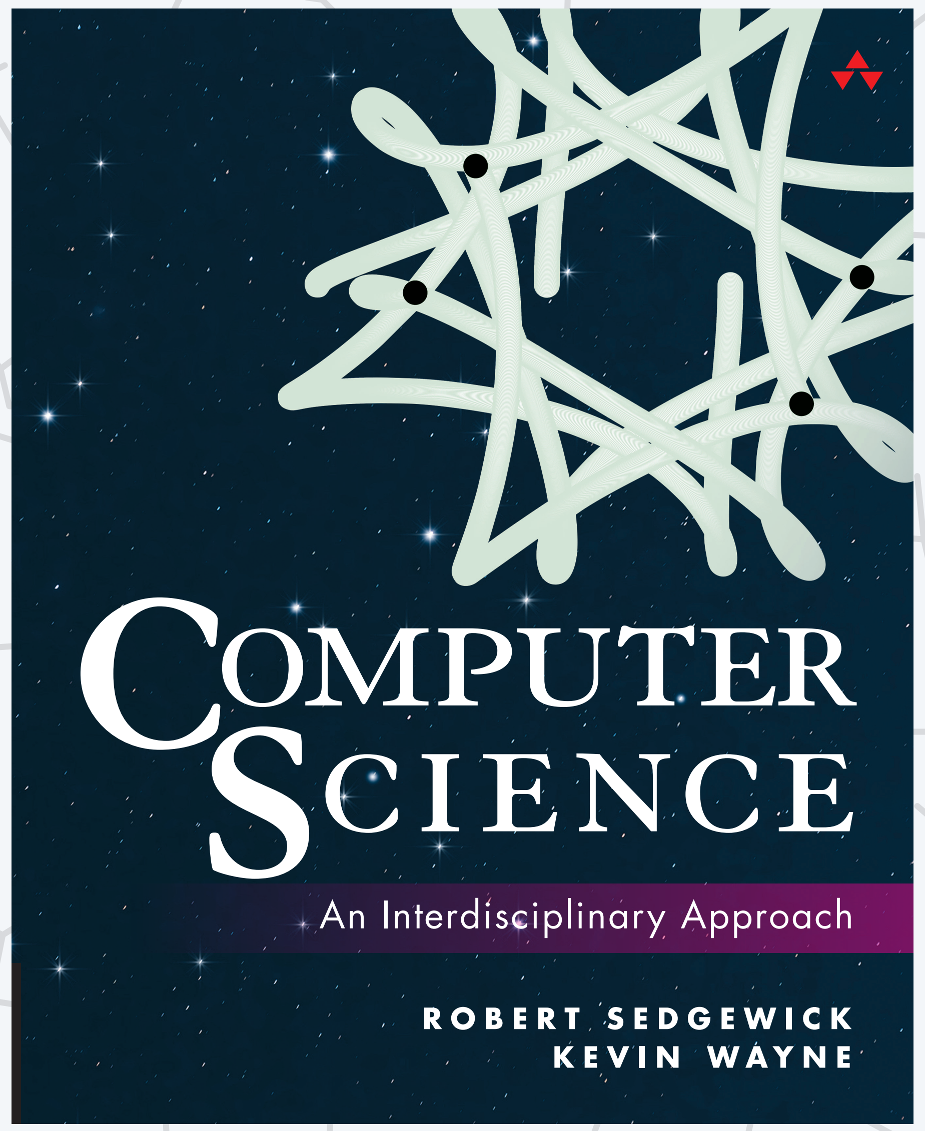
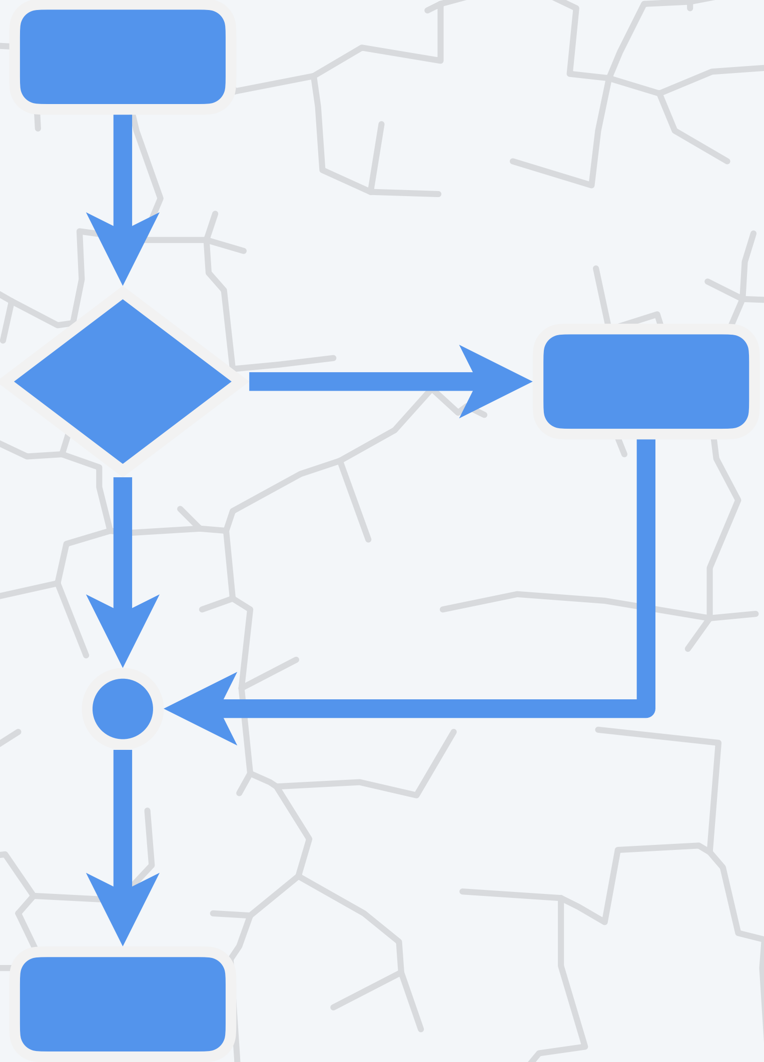


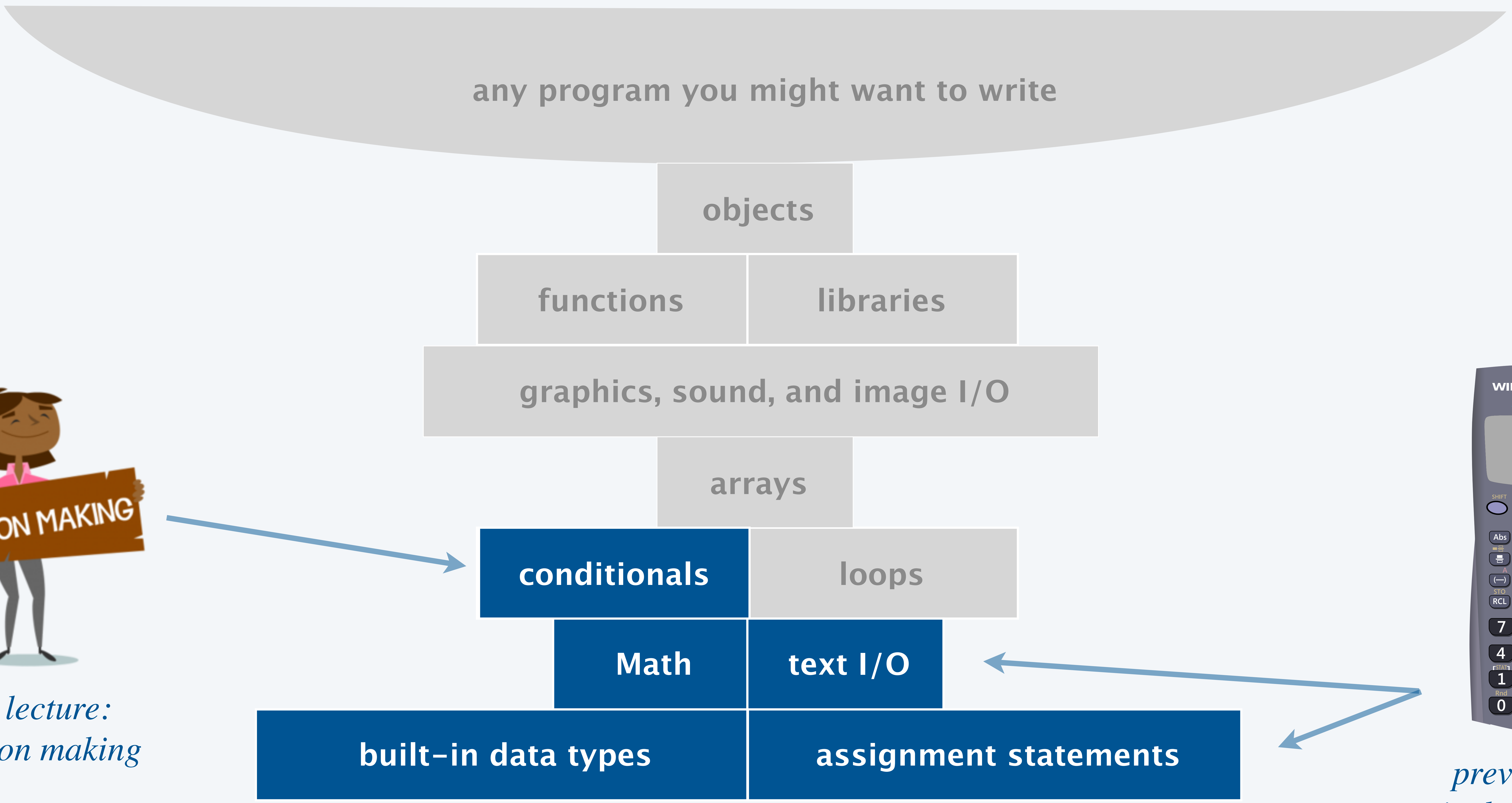
https://introcs.cs.princeton.edu

1.3 Conditionals

- *booleans*
- *if statements*
- *if–else statements*
- *nested conditionals*
- *year-to-speech*



Basic building blocks for programming



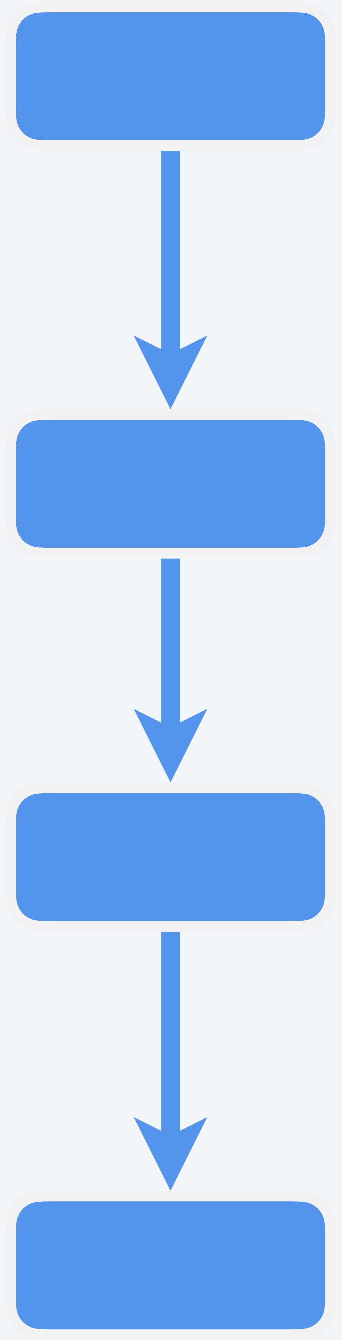
this lecture: decision making

previous lecture: equivalent to a calculator

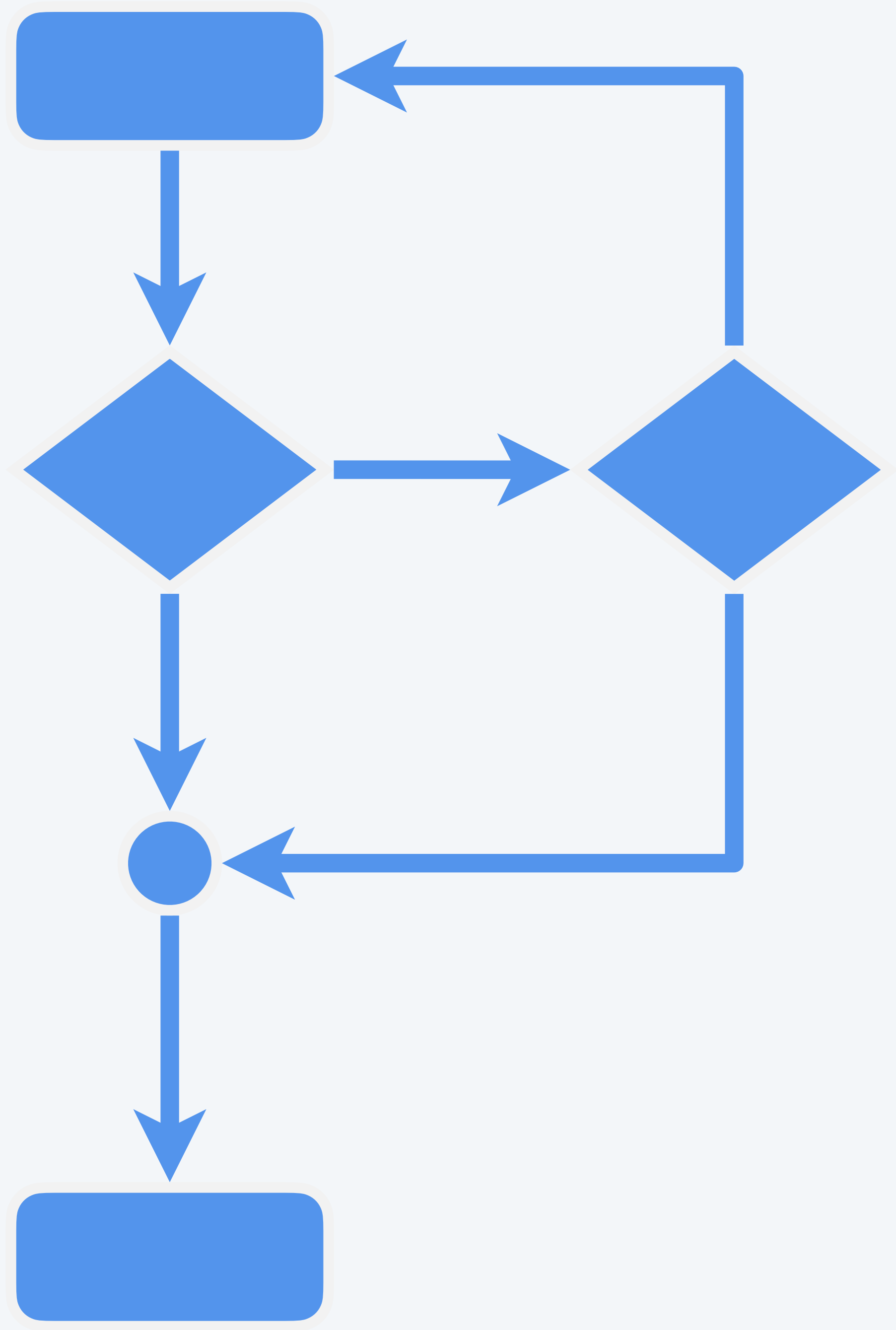
Conditionals and loops

Control flow. The sequence of statements that are actually executed in a program.

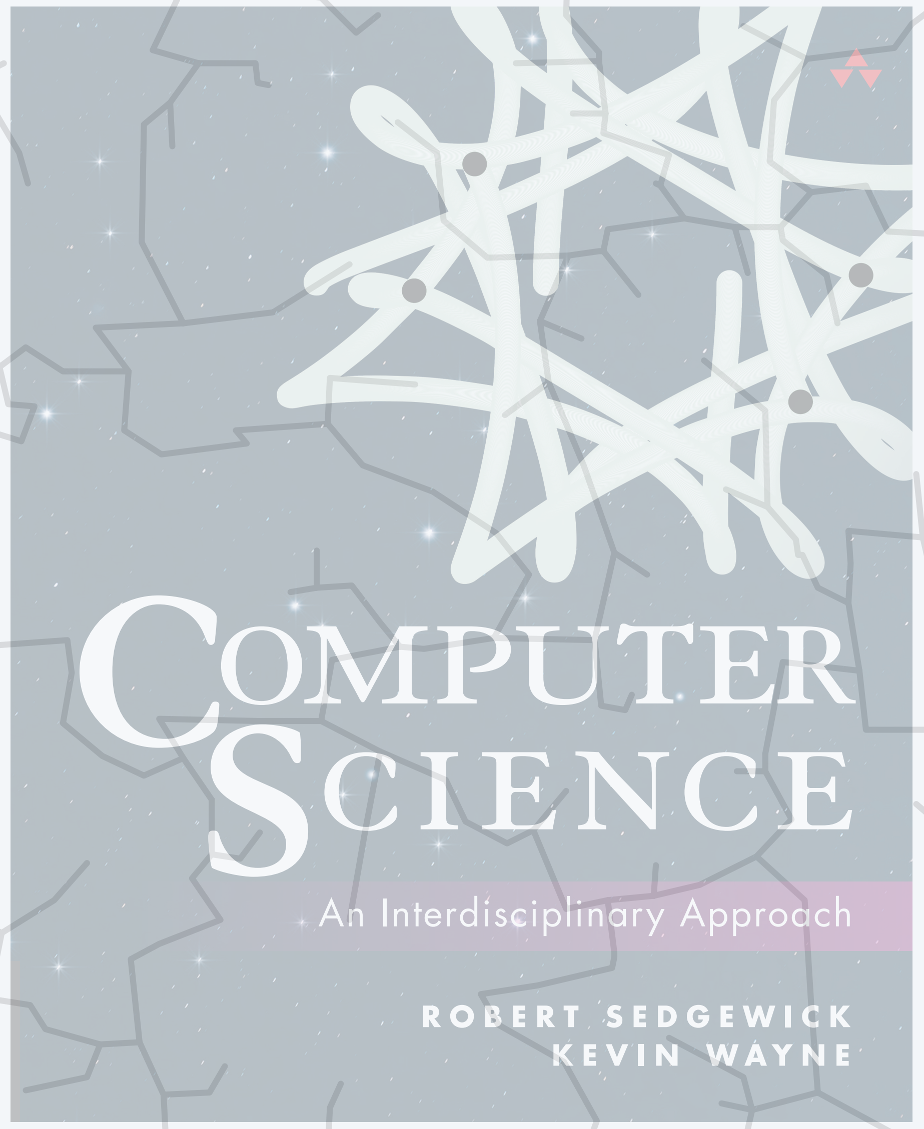
Conditionals and loops. Enable us to choreograph control flow.



straight-line control flow
(last lecture)



control flow with conditionals and loops
(this week)


https://introcs.cs.princeton.edu

1.3 Conditionals

- *booleans*
- *if statements*
- *if–else statements*
- *nested conditionals*
- *year-to-speech*

Built-in data types: review

A data type (type) is a set of values and a set of operations on those values.

type	set of values	example values	examples of operations
`String`	*sequences of characters*	`"Hello, World"` `"COS 126 is fun!"`	*concatenate*
`int`	*integers*	`17` `-12345`	*add, subtract, multiply, divide, compare, equality*
`double`	*floating-point numbers*	`2.5` `-0.125`	*add, subtract, multiply, divide, compare, equality*
`boolean`	*truth values*	`true` `false`	*and, or, not, equality*

Java's built-in data types (that we use regularly in this course)

The *boolean* data type

Typical usage: decision making in a program. ← *with conditionals and loops*

values	*true and false*		
literals	`true` `false`		
operations	*not*	*and*	*or*
operators	`!`	`&&`	`||`

← *logical operators*

expression	value
`!false`	`true`
`!true`	`false`

truth table for NOT

expression	value
`false && false`	`false`
`false && true`	`false`
`true && false`	`false`
`true && true`	`true`

truth table for AND

expression	value
`false || false`	`false`
`false || true`	`true`
`true || false`	`true`
`true || true`	`true`

truth table for OR

Equality and comparison operators: examples

zero denominator?	`denominator == 0`
non-negative discriminant?	`(b*b - 4.0*a*c) >= 0.0`
divisible by 60?	`(minutes % 60) == 0`
RGB color is not black?	`(red > 0) || (green > 0) || (blue > 0)`
valid month?	`(month >= 1) && (month <= 12)`
invalid month?	`!((month >= 1) && (month <= 12))`
string equality	args[0] ~~==~~ "Hello"

compound boolean expressions

don't compare strings with ==
(this expression evaluates to `false`*)*

The majority function

Majority function. True if at least two of *a*, *b*, and *c* are true; false otherwise.

a	b	c	majority
false	false	false	false
false	false	true	false
false	true	false	false
false	true	true	true
true	false	false	false
true	false	true	true
true	true	false	true
true	true	true	true

truth table for majority function

```
boolean majority = (a && b) || (b && c) || (a && c);
```

Amazing fact. Any boolean function can be constructed using &&, ||, and ! operators.

Example of computing with booleans: leap year test

Q. Is a given year a leap year? ← *Gregorian calendar*

A. Yes if **either**: (Case A:) divisible by 400 or (Case B:) divisible by 4 but not 100.

```
public class LeapYear {
   public static void main(String[] args) {

      int year = Integer.parseInt(args[0]);
      boolean isLeapYear;

      // Case B: divisible by 4 but not 100
      isLeapYear = (year % 4 == 0) && (year % 100 != 0);

      // ...or Case A: divisible by 400
      isLeapYear = isLeapYear || (year % 400 == 0);

      System.out.println(isLeapYear);

   }
}
```

```
~/cos126/datatypes> java LeapYear 2024
true

~/cos126/datatypes> java LeapYear 2023
false

~/cos126/datatypes> java LeapYear 1900
false

~/cos126/datatypes> java LeapYear 2000
true
```

if argument to `System.out.println()` *is of type* `boolean`, *it prints either* `true` *or* `false`

Conditionals: quiz 1

Which of the following code fragments check whether month is between 1 and 12?

I.

```
1 <= month <= 12
```

II.

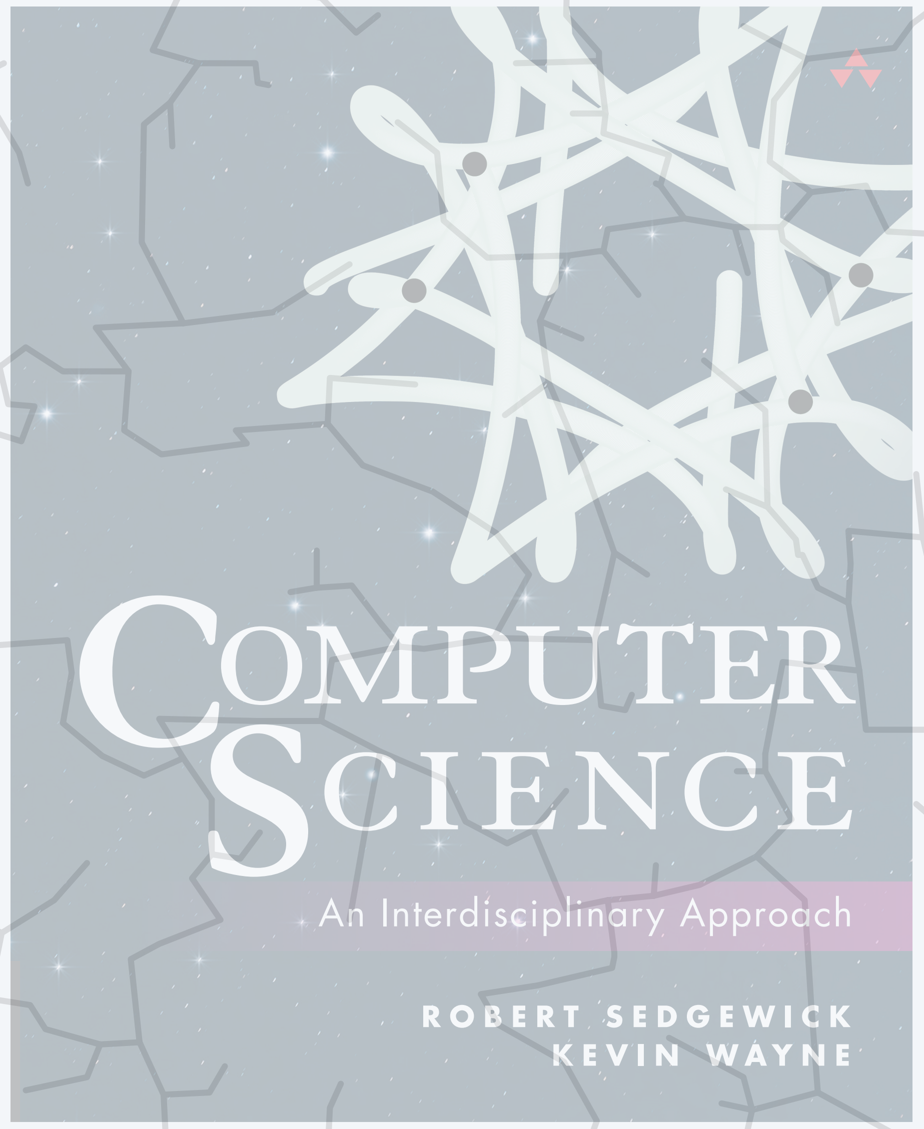
```
month >= 1 && month <= 12
```

A. I only.

B. II only.

C. I and II.

D. Neither I nor II.



https://introcs.cs.princeton.edu

1.3 Conditionals

- *if statements*
- *if–else statements*
- *nested conditionals*
- *year-to-speech*

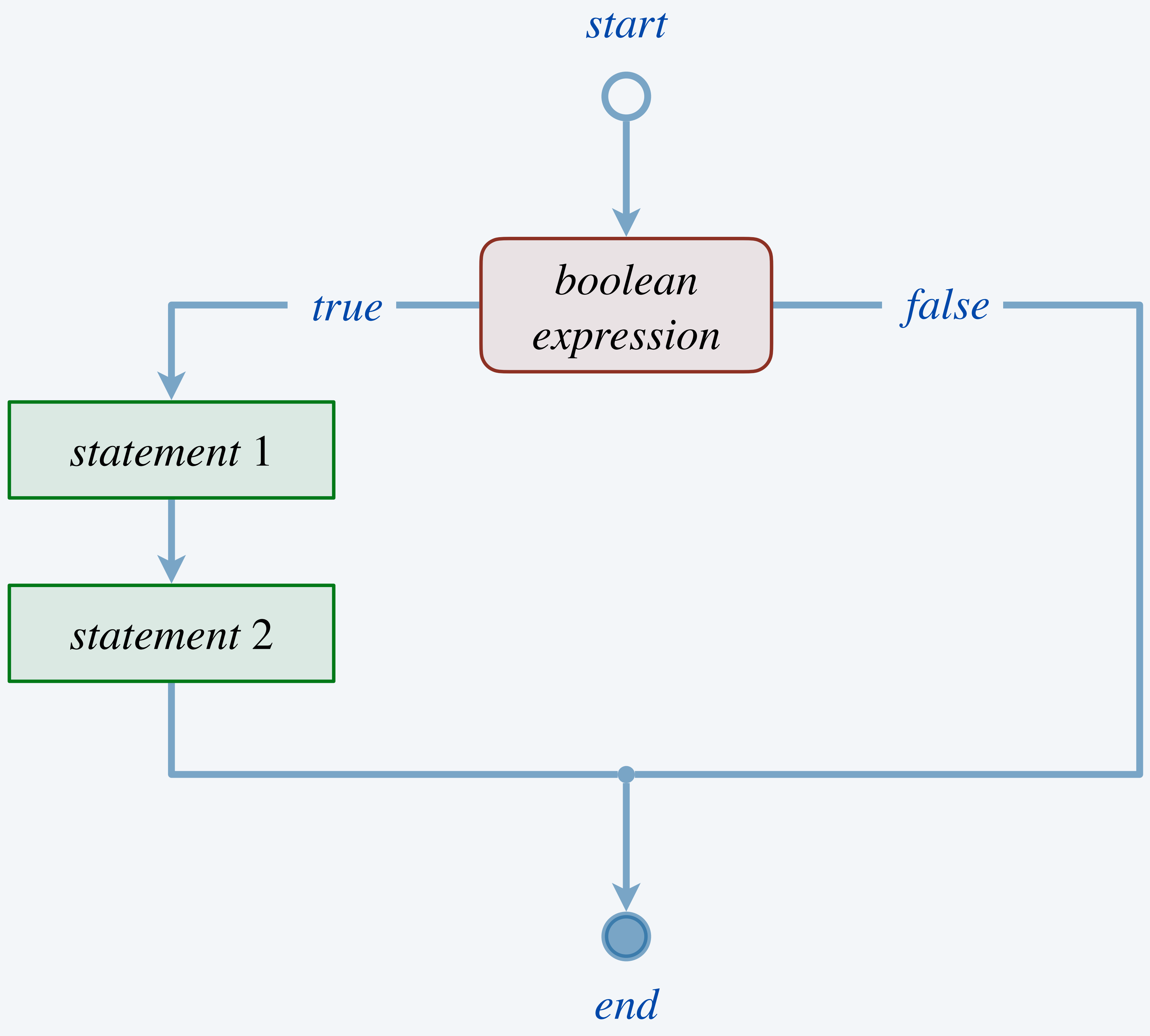
The `if` statement

Execute certain statement(s) depending on the value of a boolean expression.

- Evaluate a boolean expression.
- If true, execute statements in code block delimited by curly braces.

```
if (<boolean expression>) {
   <statement 1>
   <statement 2>
}
```

if statement template



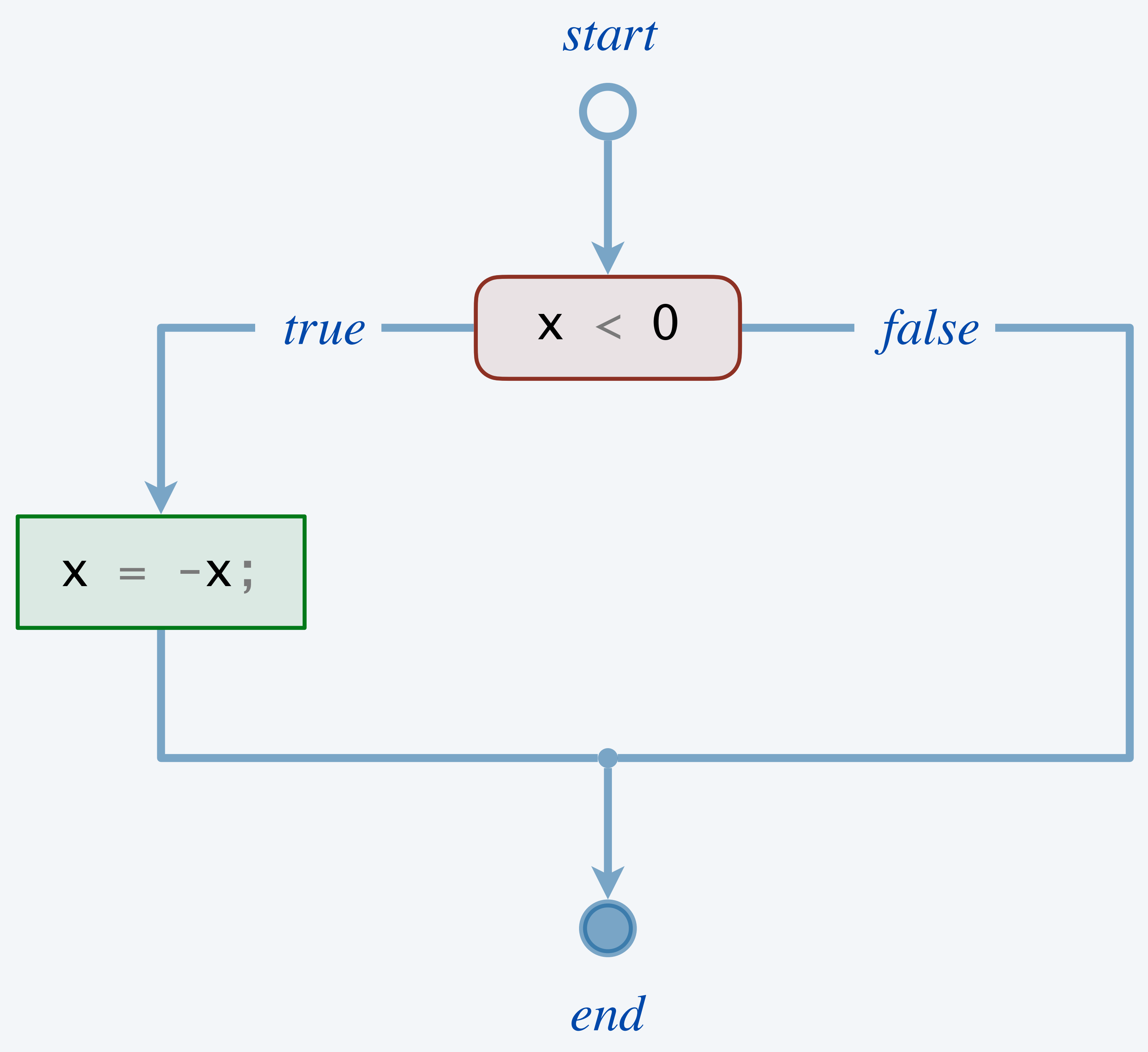
The *if* statement

Execute certain statement(s) depending on the value of a boolean expression.

- Evaluate a boolean expression.
- If true, execute statements in code block delimited by curly braces.

```
if (x < 0) {
   x = -x;
}
```

replaces x with the absolute value of x



Code blocks

A code block can contain a sequence of statements.

- Assignment statements.
- Declaration statements. ← *"local" variable accessible only within the block in which it is declared*
- Other *if* statements.
- ...

```
public class TwoSort {
   public static void main(String[] args) {
      int a = Integer.parseInt(args[0]);
      int b = Integer.parseInt(args[1]);
      if (b < a) {
         int temp = a;
         a = b;
         b = temp;
      }
      System.out.println(a);
      System.out.println(b);
   }
}
```

code block consists of a sequence of statements (swap values in a *and* b)

temp *not accessible here*

```
~/cos126/conditionals> java TwoSort 1234 126
126
1234

~/cos126/conditionals> java TwoSort 126 1234
126
1234
```

More examples of `if` statements

computation	for loop

singular vs. plural
(126 dollars vs. 1 dollar)
	`String result = price + " dollar";`
	`if (price != 1) {`
	`   result = result + "s";`
	`}`

check if donor is ineligible to donate blood	`if ((age < 16) || (weight < 110)) {`
	`   System.out.println("ineligible");`
	`}`

time normalization	`if (minutes >= 60) {`
	`   minutes = minutes - 60;`
	`   hours = hours + 1;`
	`}`

maximum of three integers	`int max = a;`
	`if (b > max) max = b;`
	`if (c > max) max = c;`

compound boolean expression

multiple statements in body of `if` statement

curly braces are optional since body of each `if` statement contains only one statement

Conditionals: quiz 2

What does the following code fragment print?

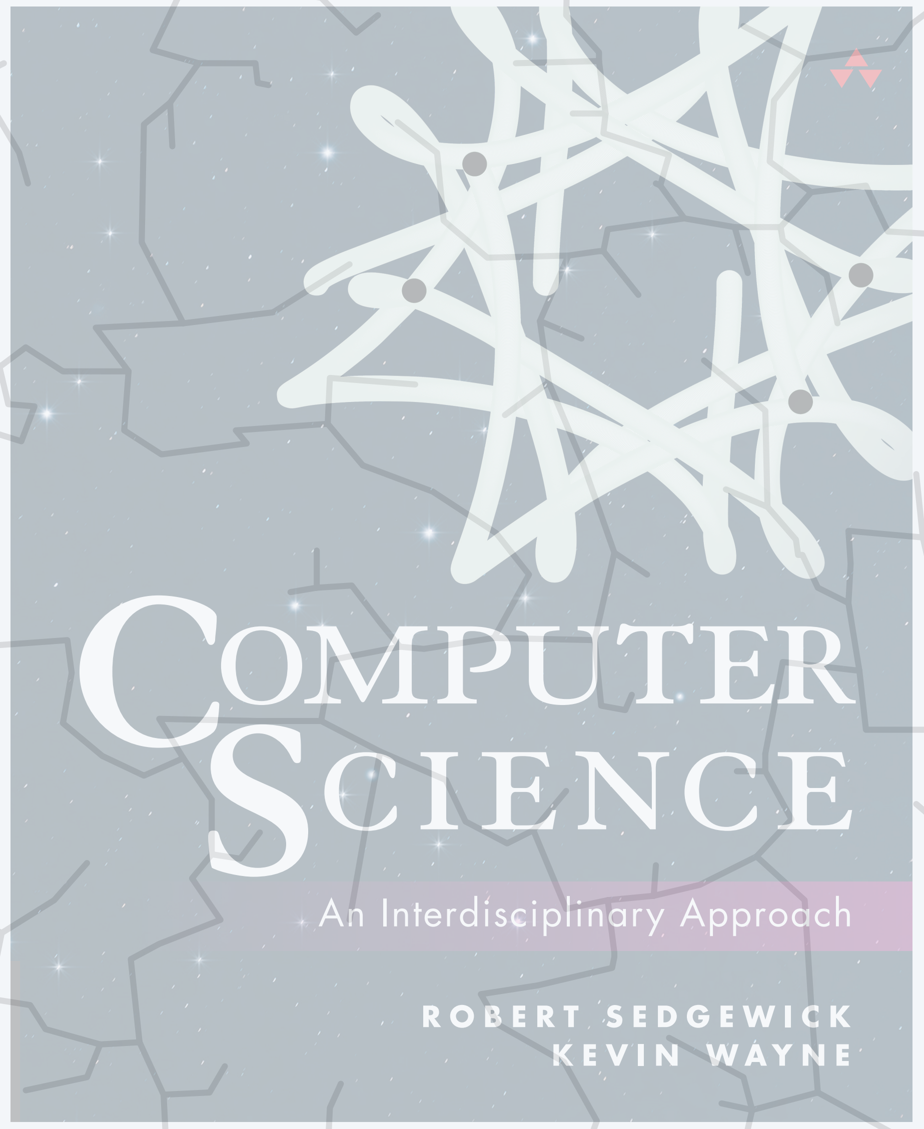
```
int x = -126;
if (x > 0); {
    System.out.println("positive");
};
```

A. `"positive"`

B. *nothing*

C. *compile-time error*

D. *run-time exception*



https://introcs.cs.princeton.edu

1.3 Conditionals

- *if statements*
- *if–else statements*
- *nested conditionals*
- *year-to-speech*

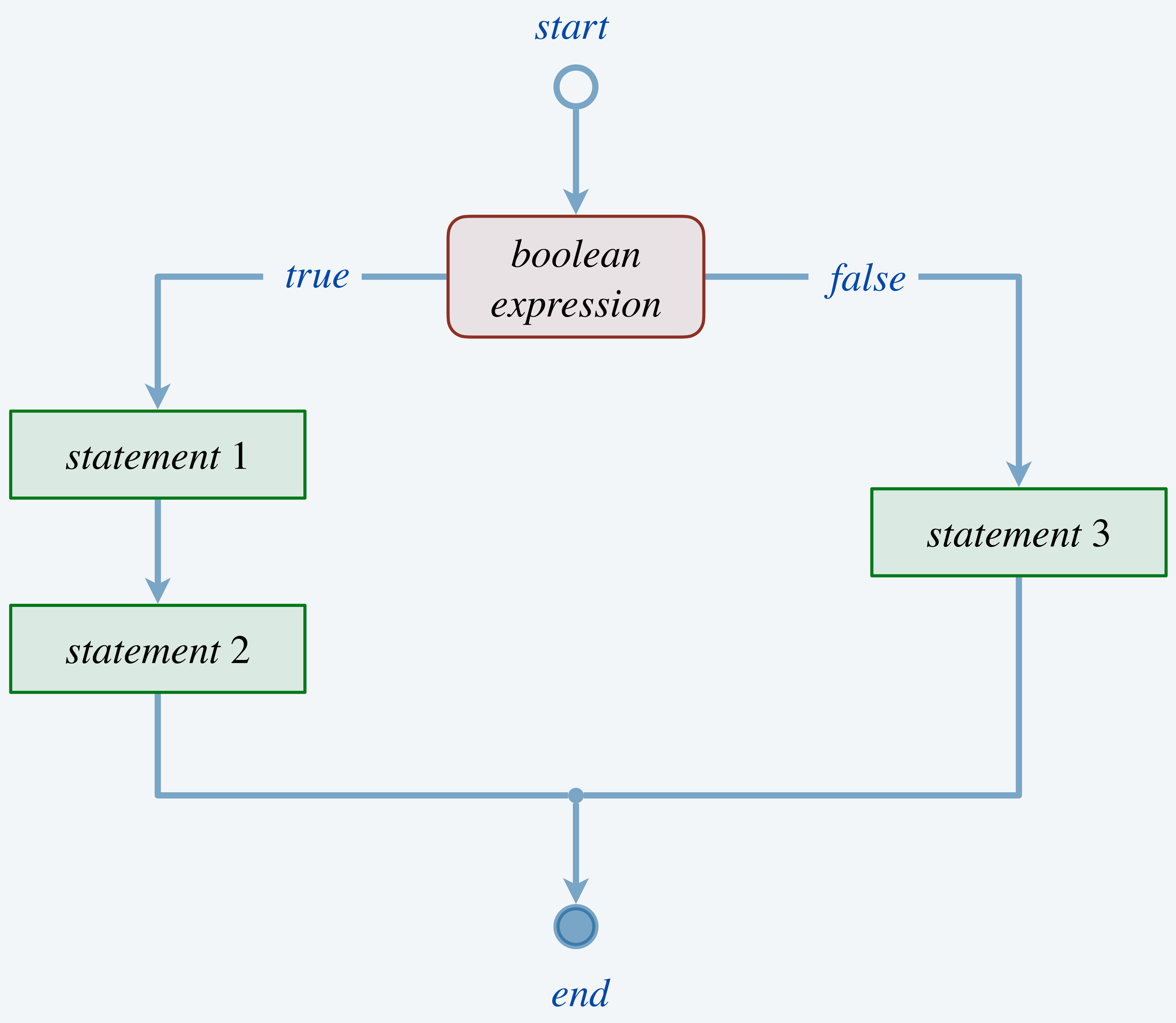
The *if-else* statement

Execute certain statements depending on the value of a boolean expression.

- Evaluate a boolean expression.
- If true, execute some statements.
- Otherwise, execute different statements. ← *the* `else` *clause*

```
if (<boolean expression>) {
    <statement 1>
    <statement 2>
}
else {
    <statement 3>
}
```

if-else statement template



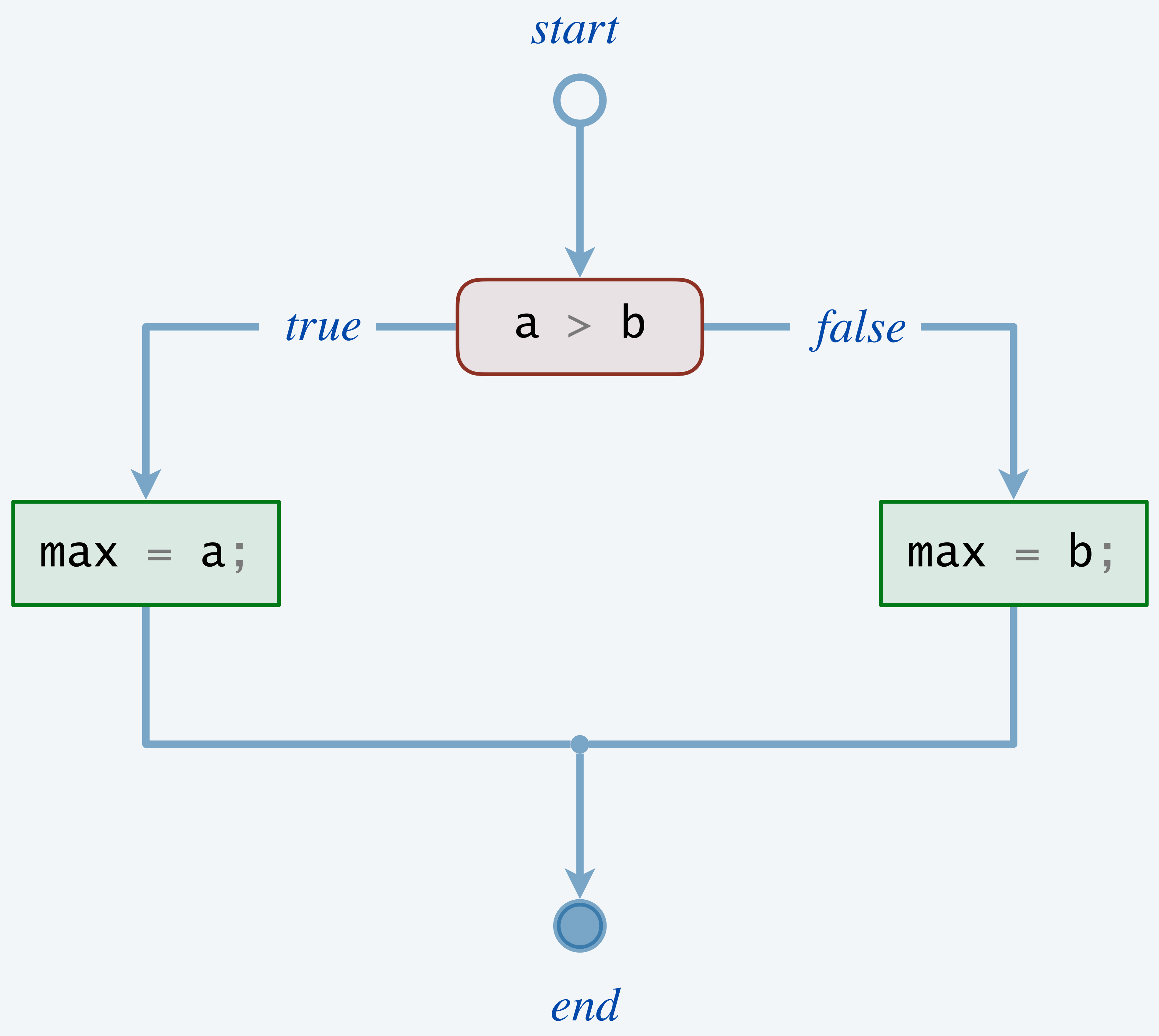
The *if-else* statement

Execute certain statements depending on the value of a boolean expression.

- Evaluate a boolean expression.
- If true, execute some statements.
- Otherwise, execute different statements. ← *the* `else` *clause*

```
int max;
if (a > b) {
   max = a;
}
else {
   max = b;
}
```

sets max to the maximum of a and b



Simulating a fair coin flip

Goal. Simulate a fair coin flip.

Remark. *Math.random*() returns a *double* value in the range $[0, 1)$.

```
public class CoinFlip {
   public static void main(String[] args) {
      double r = Math.random();
      if (r < 0.5) {
         System.out.println("Heads");
      }
      else {
         System.out.println("Tails");
      }
   }
}
```

```
~/cos126/conditionals> java CoinFlip
Heads

~/cos126/conditionals> java CoinFlip
Tails

~/cos126/conditionals> java CoinFlip
Tails
```

More examples of *if-else* statements

computation	if-else statement	

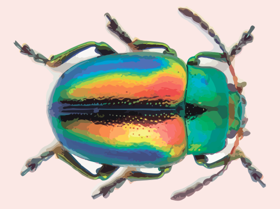
simulating a gambler's fair bet	`double r = Math.random();`
	`if (r < 0.5) cash = cash + bet;`
	`else         cash = cash - bet;`
		`if` *body consists of only one statement, so curly braces are optional*

parity	`String parity;`
	`if (n % 2 == 0) parity = "even";`
	`else            parity = "odd";`
		even: …, −4, −2, 0, 2, 4, …

integer remainder (with guard clause)	`if (denominator == 0) {`
	`   System.out.println("division by zero");`
	`}`
	`else {`
	`   int remainder = numerator % denominator;`
	`   System.out.println("remainder = " + remainder);`
	`}`
		good style to include curly braces even when optional

Conditionals: quiz 3

What does the following (buggy) code fragment print?



```
int x = -126;
boolean isPositive = (x > 0);
if (isPositive = true) System.out.println("positive");
else                   System.out.println("not positive");
```

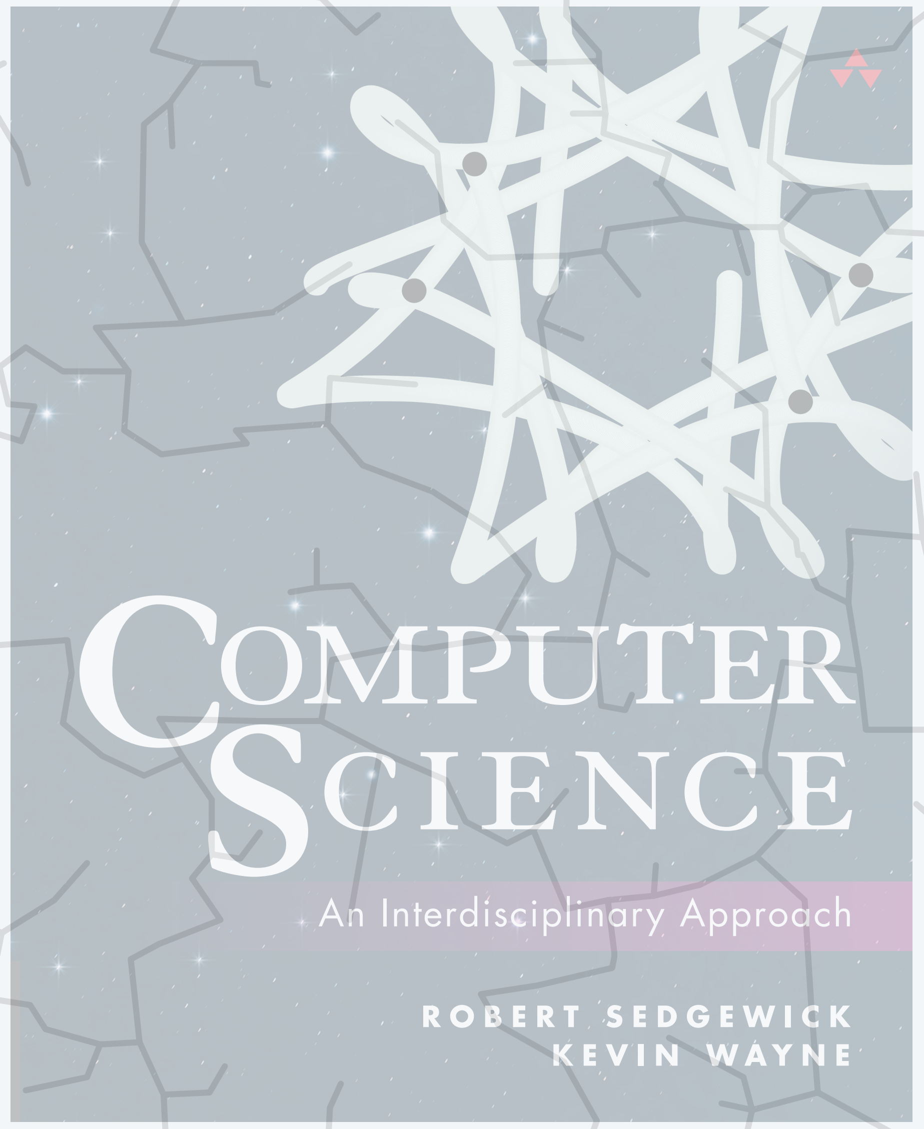
A. "positive"

B. "not positive"

C. *nothing*

D. *compile-time error*

E. *run-time exception*



https://introcs.cs.princeton.edu

1.3 Conditionals

- *if statements*
- *if–else statements*
- *nested conditionals*
- *year-to-speech*

Nesting conditionals: rock, paper, scissors

Three-way selection. Rock, paper, scissors.

```java
public class RockPaperScissors {
   public static void main(String[] args) {
      int r = (int) (Math.random() * 3);

      if (r == 0) {
         System.out.println("Rock");
      }
      else {
         if (r == 1) {
            System.out.println("Paper");
         }
         else {
            System.out.println("Scissors");
         }
      }
   }
}
```

0, 1, *or* 2
(see precept)

`if-else` *statement nested within the* `else` *clause of an* `if` *statement*

```
~/cos126/conditionals> java RockPaperScissors
Rock

~/cos126/conditionals> java RockPaperScissors
Scissors
```

Nesting conditionals: types of triangle

Triangle. Given three angles of a triangle, is it invalid, acute, obtuse, right?

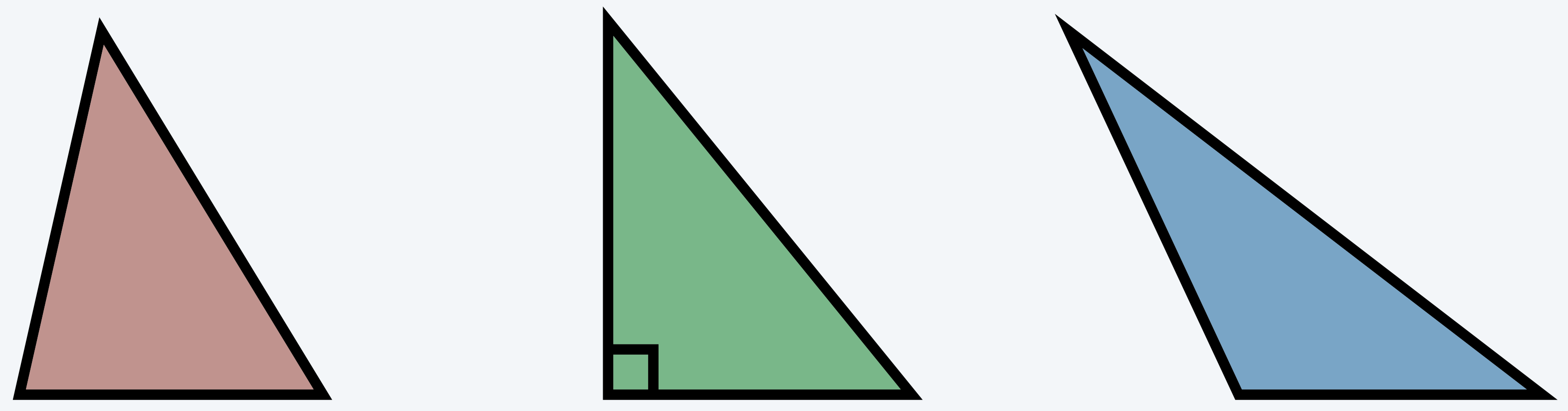
```
public class Triangle {
   public static void main(String[] args) {
      int a = Integer.parseInt(args[0]);
      int b = Integer.parseInt(args[1]);
      int c = Integer.parseInt(args[2]);

      if (a <= 0 || b <= 0 || c <= 0 || (a + b + c != 180))
         System.out.println("invalid");
      else {
         if (a < 90 && b < 90 && c < 90)
            System.out.println("acute");
         else {
            if (a > 90 || b > 90 || c > 90)
               System.out.println("obtuse");
            else
               System.out.println("right");
         }
      }
   }
}
```

`if` statement nested within an `if` statement

`if` statement nested within an `if` statement within an `if` statement

type	description
invalid	angles don't sum to 180°
acute	all angles less than 90°
obtuse	an angle greater than 90°
right	a 90° angle

mutually exclusive alternatives



Multiway selection shorthand

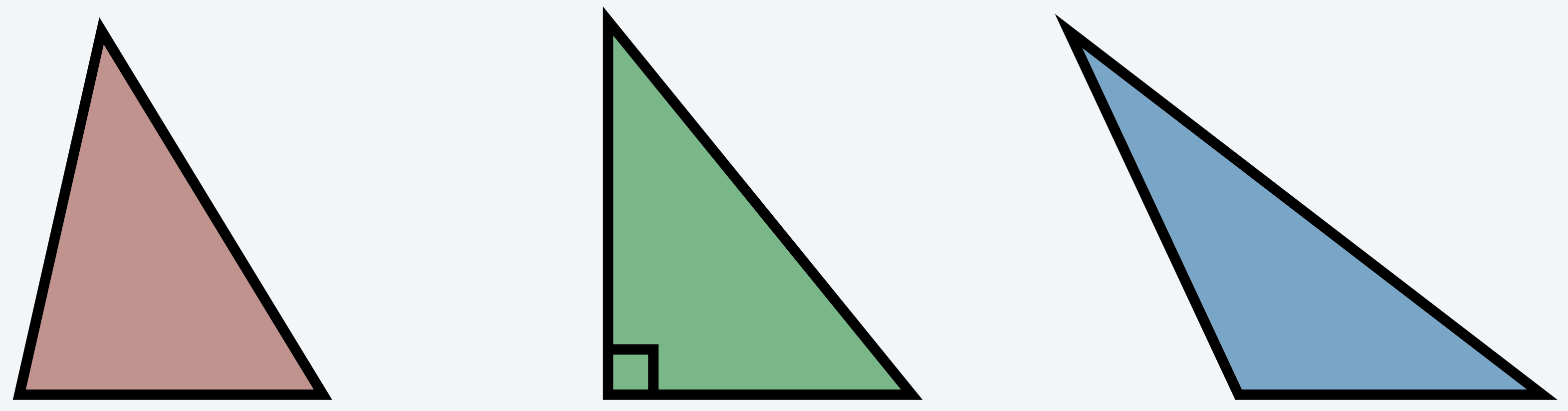
Note. Curly braces not needed here since each body consists of a single (compound) statement.

```
public class Triangle {
   public static void main(String[] args) {
      int a = Integer.parseInt(args[0]);
      int b = Integer.parseInt(args[1]);
      int c = Integer.parseInt(args[2]);

      if (a <= 0 || b <= 0 || c <= 0 || (a + b + c != 180))
         System.out.println("invalid");
      else if (a < 90 && b < 90 && c < 90)
         System.out.println("acute");
      else if (a > 90 || b > 90 || c > 90)
         System.out.println("obtuse");
      else
         System.out.println("right");

   }
}
```

4 mutually exclusive alternatives

type	description
invalid	angles don't sum to 180°
acute	all angles less than 90°
obtuse	an angle greater than 90°
right	a 90° angle

mutually exclusive alternatives

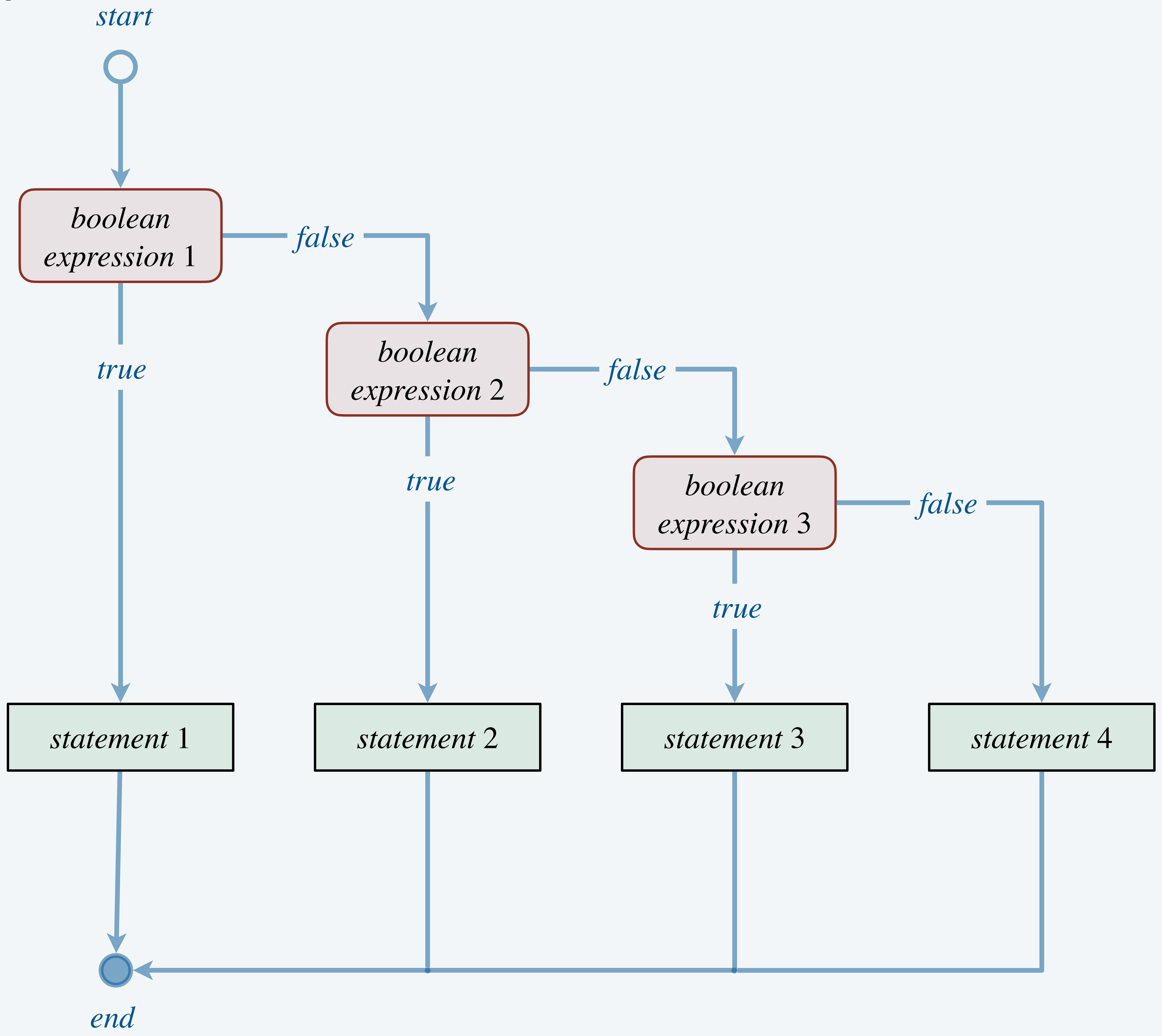


A ladder of nested *if-else* statements

Multiway selection. Mutually exclusive alternatives.

```
if (<boolean expression 1>) {
   <statement 1>
}
else if (<boolean expression 2>) {
   <statement 2>
}
else if (<boolean expression 3>) {
   <statement 3>
}
else {
   <statement 4>
}
```

if-else ladder template



More examples of multiway selection

computation	nested if-else statements

signum function
$$signum(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ +1 & \text{if } x > 0 \end{cases}$$
	`int signum;`
	`if      (x < 0) signum = -1;`
	`else if (x > 0) signum = +1;`
	`else if         signum =  0;`

Reynold's number
(ratio of inertial to viscous forces)
	`if (reynolds <= 2000.0) {`
	`   System.out.println("laminar flow");`
	`}`
	`else if (reynolds >= 3500.0) {`
	`   System.out.println("turbulent flow");`
	`}`
	`else {`
	`   System.out.println("transitional flow");`
	`}`

$\longleftarrow$ *3 mutually exclusive alternatives*

$\longleftarrow$ *3 mutually exclusive alternatives*

Conditionals: quiz 4

What will the following (buggy) code fragment print? Assume `income` is `100000`.

A. `0.22`

B. `0.25`

C. `0.28`

D. `0.33`

E. `0.35`

```
double rate = 0.35;
if (income <  47450) rate = 0.22;
if (income < 114650) rate = 0.25;
if (income < 174700) rate = 0.28;
if (income < 311950) rate = 0.33;
System.out.println(rate);
```

income	rate
0 – $47,450	22%
$47,450 – $114,649	25%
$114,650 – $174,699	28%
$174,700 – $311,949	33%
$311,950 +	35%

marginal tax rate

Nested *if* statements

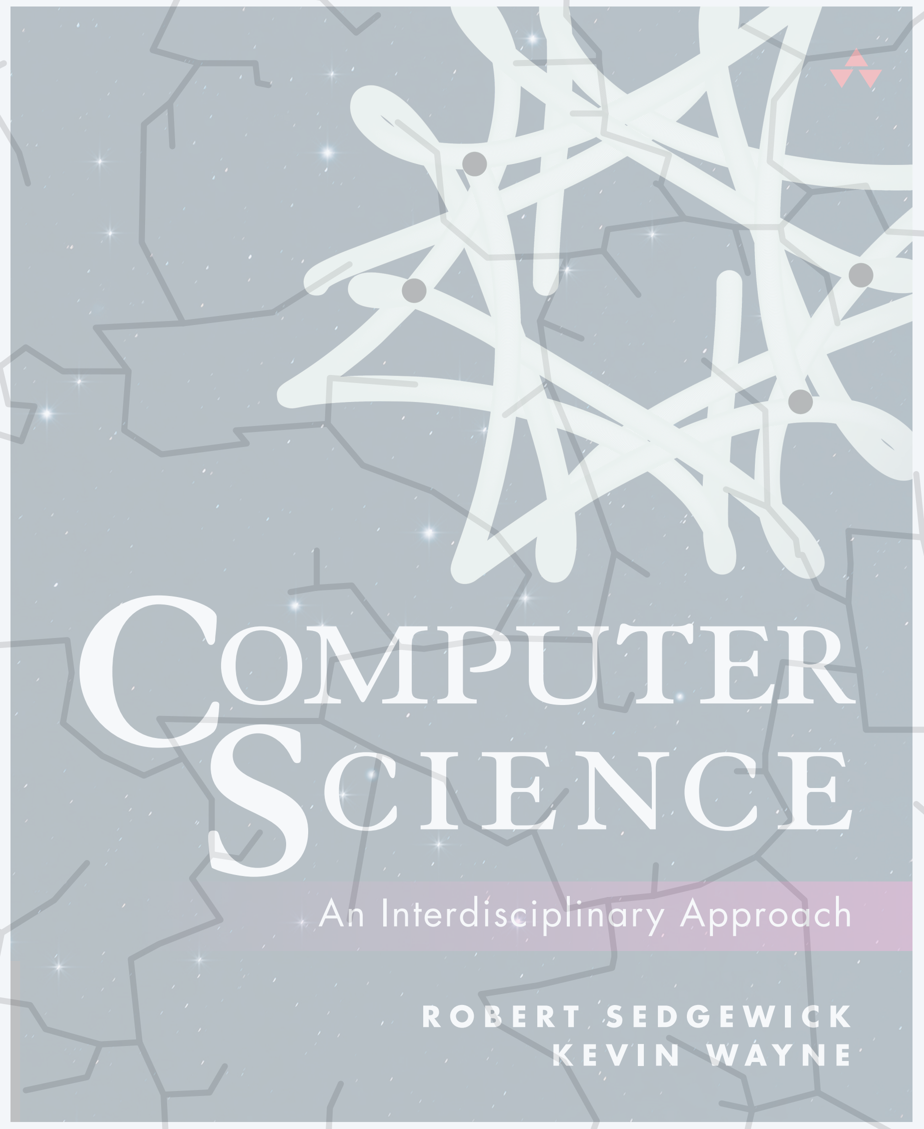
Design principle. Avoid unnecessary/gratuitous nesting of *if* statements.

```
if (r == 0) {
   if (g == 0) {
      if (b == 0) {
         System.out.println("black");
      }
   }
}
```

bad design (gratuitous nesting)

```
if (r == 0 && g == 0 && b == 0) {
   System.out.println("black");
}
```

easier to read and debug



https://introcs.cs.princeton.edu

1.3 Conditionals

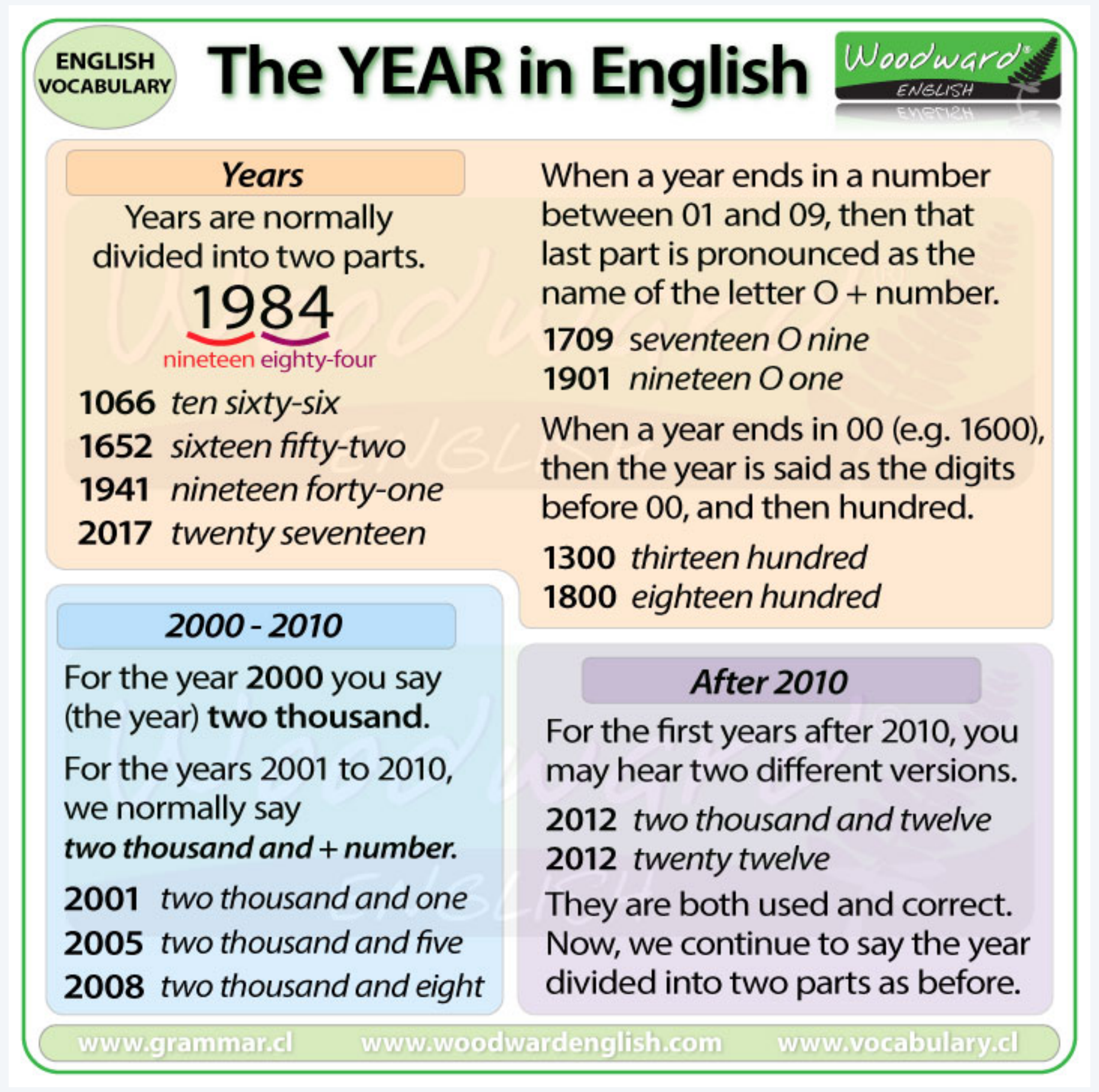
- *if statements*
- *if–else statements*
- *nested conditionals*
- *year-to-speech*

Text-to-speech year

Rules for speaking a year (1–9999) in English.

- Break up year into first-two and last-two digits; say each two-digit number.
- Special cases:
  - year ends in 000: say *thousand* for last three digits
  - year ends in 00 (but not 000): say *hundred* for last two digits
  - year ends in 01 to 09: say *oh* followed by single digit
  - year begins with 00: skip first two digits

year	spoken
2024	*twenty twenty-four*
1776	*seventeen seventy-six*
2000	*two thousand*
1700	*seventeen hundred*
1901	*nineteen oh one*
0026	*twenty-six*
12345	*invalid year*



ENGLISH VOCABULARY

The YEAR in English

Years

Years are normally divided into two parts.

1984
nineteen eighty-four

1066 *ten sixty-six*
1652 *sixteen fifty-two*
1941 *nineteen forty-one*
2017 *twenty seventeen*

When a year ends in a number between 01 and 09, then that last part is pronounced as the name of the letter O + number.

1709 *seventeen O nine*
1901 *nineteen O one*

When a year ends in 00 (e.g. 1600), then the year is said as the digits before 00, and then hundred.

1300 *thirteen hundred*
1800 *eighteen hundred*

2000 - 2010

For the year **2000** you say (the year) **two thousand**.

For the years 2001 to 2010, we normally say ***two thousand and + number.***

2001 *two thousand and one*
2005 *two thousand and five*
2008 *two thousand and eight*

After 2010

For the first years after 2010, you may hear two different versions.

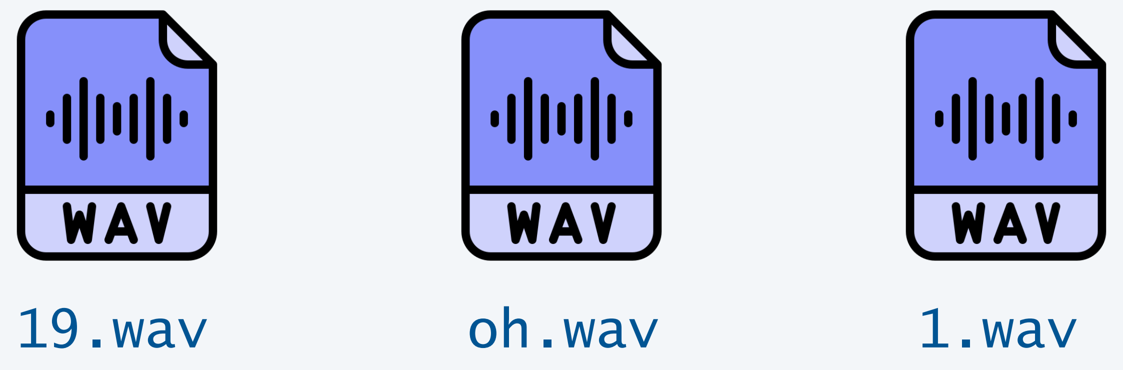
2012 *two thousand and twelve*
2012 *twenty twelve*

They are both used and correct. Now, we continue to say the year divided into two parts as before.

www.grammar.cl www.woodwardenglish.com www.vocabulary.cl

Text-to-speech approach

Domain-specific synthesis. Concatenate pre-recorded words to form desired output.



speaking the year 1901

word	audio file
1–99	`1.wav`, `2.wav`, `3.wav`, …
hundred	`hundred.wav`
thousand	`thousand.wav`
oh	`oh.wav`

vocabulary

Applications.

- Talking clocks.
- Train schedule announcements.
- Interactive telephone voice response systems.

Note. Limited to words in vocabulary.

Live coding

```java
public class SayYear {
   public static void main(String[] args) {
      int year = Integer.parseInt(args[0]);
      int firstTwoDigits = year / 100;
      int lastTwoDigits  = year % 100;

      if (year % 1000 == 0) {
         int firstDigit = year / 1000;
         StdAudio.play(firstDigit + ".wav");
         StdAudio.play("thousand.wav");
      }

      else {
         if (firstTwoDigits > 0)
            StdAudio.play(firstTwoDigits + ".wav");

         if (lastTwoDigits == 0)
            StdAudio.play("hundred.wav");

         else {
            if (lastTwoDigits < 10)
               StdAudio.play("oh.wav");

            StdAudio.play(lastTwoDigits + ".wav");
         }
      }
   }
}
```

assumes year is between 1 and 9999

parse first and last two digits of year

special case for years ending in 000

say first two digits (unless 00)

special case for years ending in 00 (but not 000)

special case for years ending in 01 to 09

say last two digits

Testing

Principle. Supply inputs that activate all possible execution paths through program. ← *so that all code gets tested*

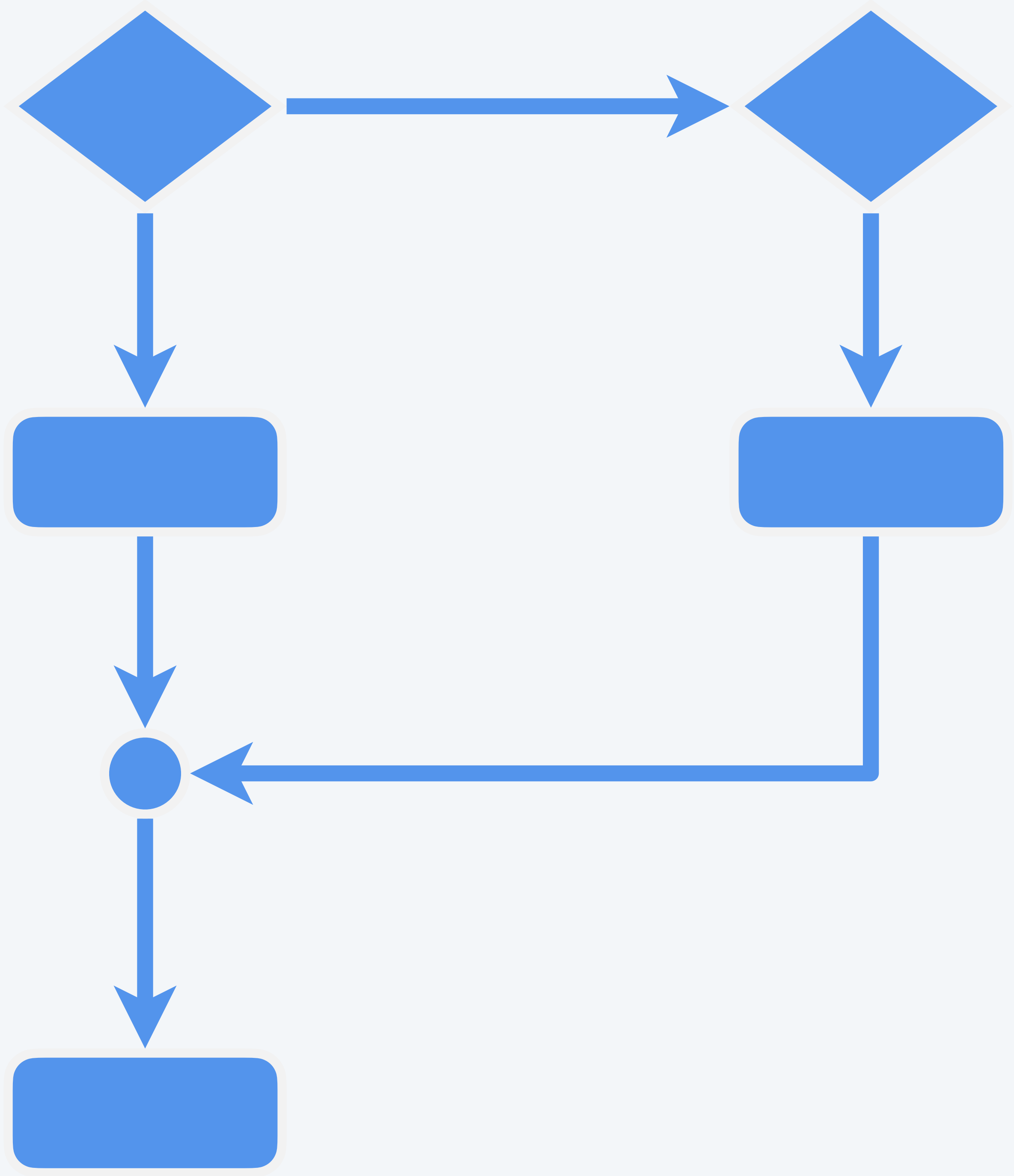
```
~/cos126/conditionals> java-introcs SayYear 2024   ←  typical case
🔊 [speaks "twenty twenty-four"]

~/cos126/conditionals> java-introcs SayYear 1776   ←  typical case
🔊 [speaks "seventeen seventy-six"]

~/cos126/conditionals> java-introcs SayYear 2000   ←  year ends in 01 to 09
🔊 [speaks "two thousand"]

~/cos126/conditionals> java-introcs SayYear 1700   ←  year ends in 000
🔊 [speaks "seventeen hundred"]

~/cos126/conditionals> java-introcs SayYear 1901   ←  year ends in 00 (but not 000)
🔊 [speaks "nineteen oh one"]

~/cos126/conditionals> java-introcs SayYear 26     ←  year begins with 00
🔊 [speaks "twenty-six"]
```

Summary

One–way selection. The *if* statement.

Binary selection. The *if–else* statement.

Multiway selection. Ladder of nested *if–else* statements.



control flow with conditionals

Credits

media	source	license
Decision Making	nextlevelscoaching.com	non-commercial use
Scientific Calculator	Fornax at Wikimedia	CC BY-SA 3.0
Coin Toss	clipground.com	CC BY 4.0
Types of Triangles	Adobe Stock	education license
Bugs	Adobe Stock	education license
Russian Nesting Dolls	Adobe Stock	education license
Rock, Paper, Scissors	Adobe Stock	education license
Watering Can	Katerina Kamprani	
Digital Clock	Chrkl at Wikimedia	CC BY 3.0
Live Coding Icon	Adobe Stock	education license
Code Testing Icon	Adobe Stock	education license
The Year in English	Woodward English	