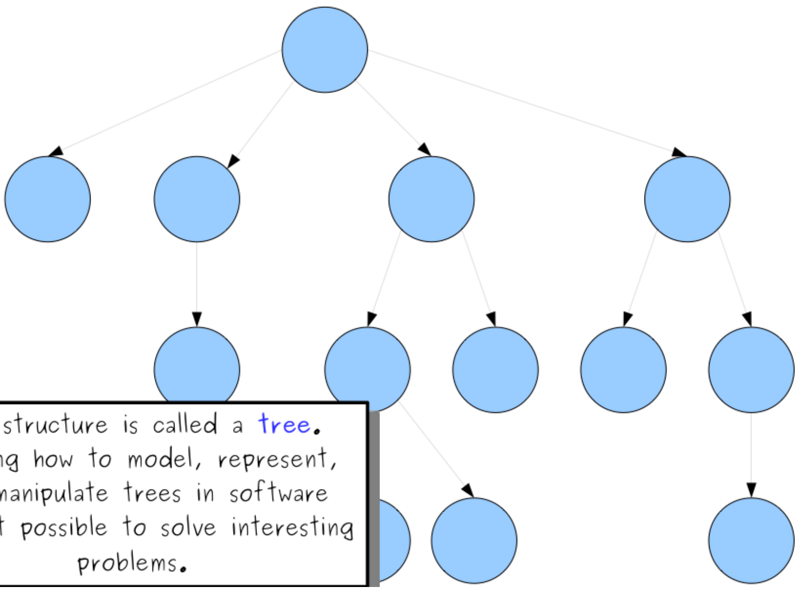


This structure is called a tree. Knowing how to model, represent, and manipulate trees in software makes it possible to solve interesting problems.

A recursive solution is a solution that is defined in terms of itself.

- Learn how to model and solve complex problems with computers
- To that end:
    - Explore common abstractions for representing problems
    - Harness recursion and understand how to think about problems recursively.
    - Quantitatively analyze different approaches for solving problems

Perfect Numbers: A positive integer n is called a perfect number if it's equal to the sum of its positive divisors (excluding itself)

To find the first four perfect numbers:

```cpp
#include <iostream>
using namespace std;

/* Returns the sum of the positive divisors of the number n >= 0. */
int sumOfDivisorsOf(int n) {
    int total = 0;

    for (int i = 1; i < n; i++) {
        if (n % i == 0) {
            total += i;
        }
    }

    return total;
}

int main() {
    int found = 0;  // How many perfect numbers we've found
    int number = 1; // Next number to test

    /* Keep looking until we've found four perfect numbers. */
    while (found < 4) {
        /* A number is perfect if the sum of its divisors is equal to it. */
        if (sumOfDivisorsOf(number) == number) {
            cout << number << endl;
            found++;
        }

        number++;
    }

    return 0;
}
```

In Python, indentation alone determines nesting.

In C++, indentation is nice, but ***curly braces*** alone determine nesting.

{ }

In Python, newlines mark the end of statements.

In C++, individual statements must have a semicolon (;) after them.

;

In Python, you print output by using `print()`.

In C++, you use the ***stream insertion operator*** (<<) to push data to the console. (Pushing `endl` prints a newline.)

In Python, you can optionally put parentheses around conditions in `if` statements and `while` loops.

In C++, these are mandatory.

<< endl ( )

Python and C++ each have **for** loops, but the syntax is different. (Check the textbook for more details about how this works!)

for(; ;)

C++ has an operator ++ that means "add one to this variable's value." Python doesn't have this.

i++

In Python, comments start with # and continue to the end of the line.

In C++, there are two styles of comments. Comments that start with /* continue until */. Comments that start with // continue to the end of the line.

//

/* */

In Python, each object has a type, but it isn't stated explicitly.

In C++, you *must* give a type to each variable. (The **int** type represents an integer.)

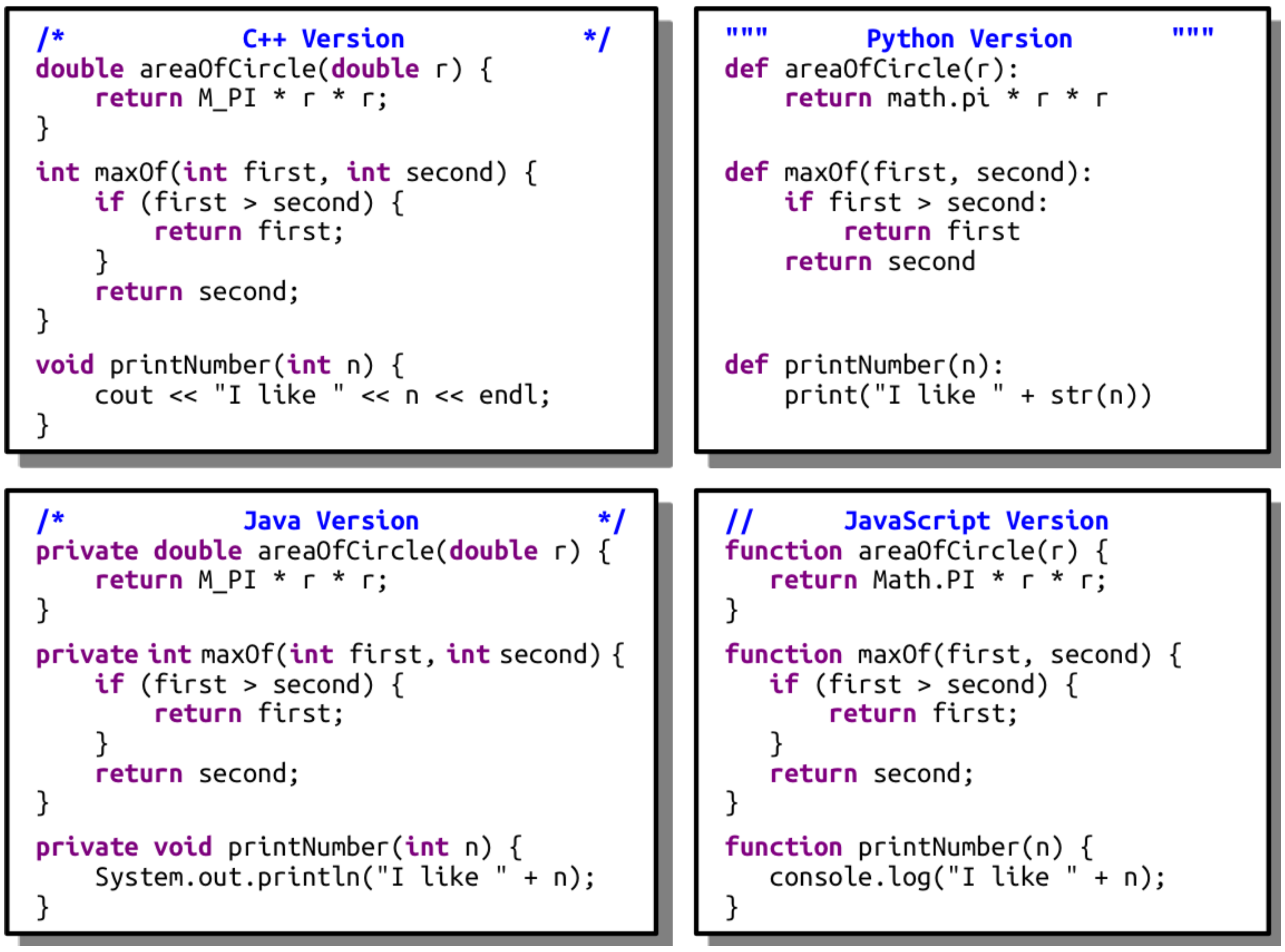
int

double

In Python, statements can be either in a function or at the top level of the program.

In C++, all statements must be inside of a function.

- C++ is a great language for writing high-performance code that takes advantage of underlying hardware.
- Compiling C++ code introduces some delays between changing the code and running the code.
- C++ programs, generally, run much faster than Python programs



```cpp
/*              C++ Version                */
double areaOfCircle(double r) {
    return M_PI * r * r;
}
int maxOf(int first, int second) {
    if (first > second) {
        return first;
    }
    return second;
}
void printNumber(int n) {
    cout << "I like " << n << endl;
}
```

```python
"""          Python Version              """
def areaOfCircle(r):
    return math.pi * r * r

def maxOf(first, second):
    if first > second:
        return first
    return second


def printNumber(n):
    print("I like " + str(n))
```

```java
/*              Java Version                */
private double areaOfCircle(double r) {
    return M_PI * r * r;
}
private int maxOf(int first, int second) {
    if (first > second) {
        return first;
    }
    return second;
}
private void printNumber(int n) {
    System.out.println("I like " + n);
}
```

```javascript
//        JavaScript Version
function areaOfCircle(r) {
   return Math.PI * r * r;
}
function maxOf(first, second) {
   if (first > second) {
       return first;
   }
   return second;
}
function printNumber(n) {
   console.log("I like " + n);
}
```

```cpp
/*              C++ Version              */
double areaOfCircle(double r) {
    return M_PI * r * r;
}

int maxOf(int first, int second) {
    if (first > second) {
        return first;
    }
    return second;
}

void printNumber(int n) {
    cout << "I like " << n << endl;
}
```

```python
"""          Python Version          """
def areaOfCircle(r):
    return math.pi * r * r

def maxOf(first, second):
    if first > second:
        return first
    return second


def printNumber(n):
    print("I like " + str(n))
```

```java
/*              Java Ve
private double areaOf
    return M_PI * r *
}
private int maxOf(int
    if (first > secon
        return first;
    }
    return second;
}
private void printNum
    System.out.println("I like " + n);
}
```

```javascript
    console.log("I like " + n);
}
```

Functions in C++ work like functions in Python/JavaScript or like methods in Java. They (optionally) take in parameters, perform a calculation, then (optionally) return a value.

```cpp
/*              C++ Version              */
double areaOfCircle(double r) {
    return M_PI * r * r;
}

int maxOf(int first, int second) {
    if (first > second) {
        return first;
    }
    return second;
}

void printNumber(int n) {
    cout << "I like " << n << endl;
}
```

```python
"""          Python Version          """
def areaOfCircle(r):
    return math.pi * r * r

def maxOf(first, second):
    if first > second:
        return first
    return second


def printNumber(n):
    print("I like " + str(n))
```

```java
/*              Java Ve
private double areaOf
    return M_PI * r *
}
private int maxOf(int
    if (first > secon
        return first;
    }
    return second;
}
private void printNum
    System.out.println("I like " + n);
}
```

```javascript
    console.log("I like " + n);
}
```

You define a function by writing

```
return-type fn-name(args) {
    // … code goes here …
}
```

All variables in C++ need a type. Some common types include **int** (integer), **double** (real number), and **bool** (true/false),

If a function does not return a value, its return type should be the cool-but-scary-sounding **void**.