

# Recursion



---

## The Three Laws of Recursion

1. A recursive algorithm **must have a base case**. This means there is a value for the function that is easy to calculate directly, without a recursive call. This identifies a point where the recursive calls end and you can solve the entire problem.
2. Each time a recursive algorithm is invoked, **it must move closer to the base case**. This means that the algorithm does a little work now, and makes a recursive call that solves a slightly easier problem (“easier” meaning “closer to the base case”).
3. A recursive algorithm **must call itself**. That’s what makes it recursive.

# Random Numbers



---

## Outcomes from Rolling Two Dice

<b>Total</b>	<b>Individual Dice</b>	<b># Ways</b>	<b>Probability</b>
2	(1,1)	1	1/36
3	(1,2), (2,1)	2	2/36
4	(1,3), (2,2), (3,1)	3	3/36
5	(1,4), (2,3), (3,2), (4,1)	4	4/36
6	(1,5), (2,4), (3,3), (4,2), (5,1)	5	5/36
7	(1,6), (2,5), (3,4), (4,3), (5,2), (6,1)	6	6/36
8	(2,6), (3,5), (4,4), (5,3), (6,2)	5	5/36
9	(3,6), (4,5), (5,4), (6,3)	4	4/36
10	(4,6), (5,5), (6,4)	3	3/36
11	(5,6), (6,5)	2	2/36
12	(6,6)	1	1/36

# File Input / Output



---

## File Modes

- "r" opens a file for reading.
- "w" opens a file for writing. If the file already exists, its old contents are destroyed.
- "a" opens a file for appending data to the end of the file.
- "rb" opens a file for reading binary data.
- "wb" opens a file for writing binary data.

The last two are important on Windows if you're opening or modifying a file created by a different application.

---

## Reading and Writing From a File

<code>read(<i>number</i>)</code>	read the specified number of characters from the file and return them as a string. If no integer is provided, then return all unread characters.
<code>readline()</code>	read the next line of the file up to a <code>\n</code> and return it as a string.
<code>readlines()</code>	return a list of the remaining lines of a file.
<code>write(<i>outString</i>)</code>	write the specified string to a file.
<code>close()</code>	close the file.



# Object-Oriented Programming





---

## Procedural Programming vs. OOP

In Procedural programs, you:

- break down a task into variables, data structures, and subprograms
- you use subprograms to operate on data structures

In Object-Oriented programs, you:

- define *objects* that expose behavior (*methods*) and data (*attributes*) using well-defined interfaces
- bundle everything together, so that an object only operates on its own attributes using methods

---

## Four Basic Programming Concepts in OOP

- ***Encapsulation:*** hiding implementation details of a class from other objects.
- ***Abstraction:*** simplifying complex reality by modeling classes appropriate to the problem.
- ***Inheritance:*** a way to define new classes using parts of classes that have already been defined.
- ***Polymorphism:*** the process of interpreting an operator or function in different ways for different data types.