Recursion

You will learn the definition of recursion as well as seeing how simple recursive programs work

What Is Recursion?

"*the determination of a succession of elements by operation on one or more preceding elements according to a rule or formula involving a finite number of steps*" (Merriam-Webster online)

What This Really Means

Breaking a problem down into a series of steps. The final step is reached when some basic condition is satisfied. ***The solution for each step is used to solve the previous step.*** *The solution for all the steps together form the solution to the whole problem.*

(The "Tam" translation)

Definition For Philosophy

"*...state of mind of the wise man; practical wisdom...*" [1]

See Metaphysics

1 The New Webster Encyclopedic Dictionary of the English Language

Metaphysics

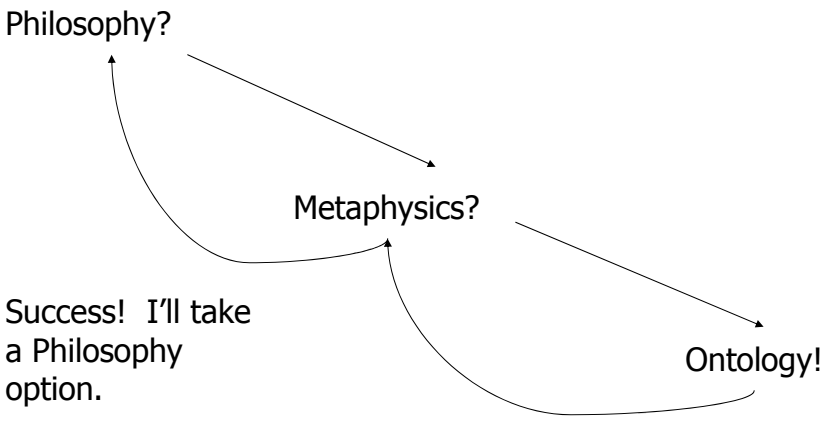
"*...know the ultimate grounds of being or what it is that really exists, embracing both psychology and* ***ontology.***" [2]

2 The New Webster Encyclopedic Dictionary of the English Language

Result Of Lookup , Possibility One: Success

- I know what Ontology means!

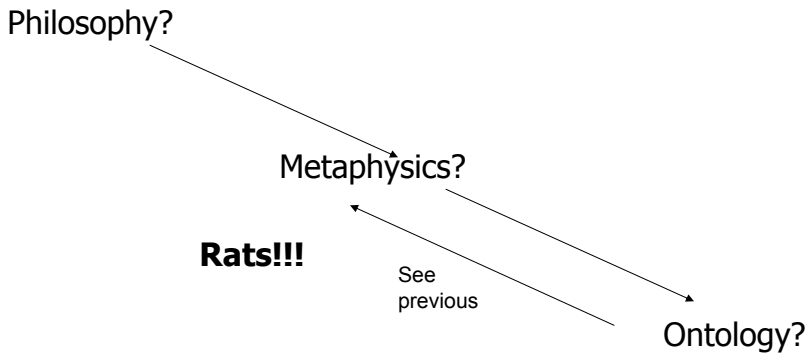
Result Of Lookup, Possibility One



Result Of Lookup, Possibility Two: Failure

- Lookups loop back.

Result Of Lookup, Possibility Two



James Tam

Ontology

"*…equivalent to metaphysics.*"[3]

3 The New Webster Encyclopedic Dictionary of the English Language

Wav file from "The Simpsons"

James Tam

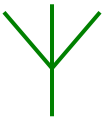
Result Of Lookup, Possibility Three: Failure

- You've looked up everything and still don't know the definition!

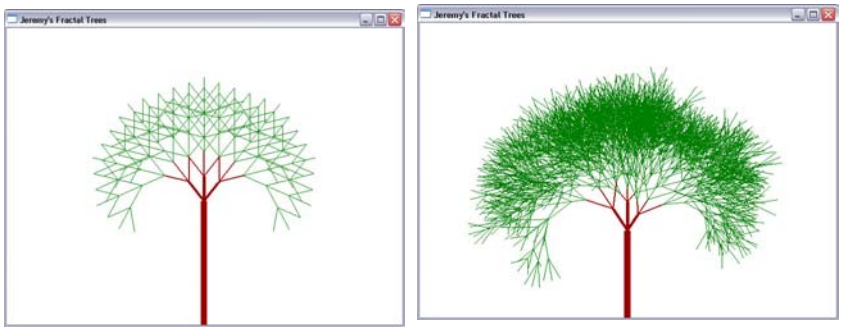
Looking Up A Word

```
if (you completely understand a definition) then
    return to previous definition (using the definition that's understood)
else
    lookup (unknown word(s))
```

Recursion: Can Be Used To Produce Graphics

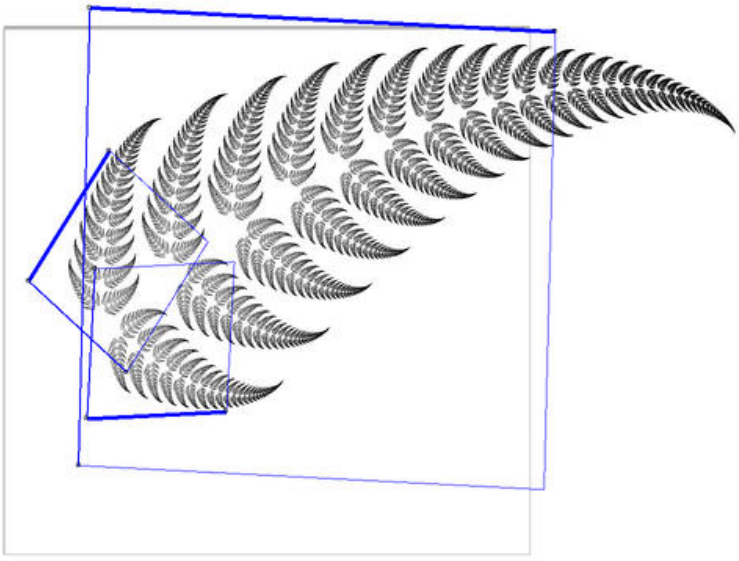


Produce a picture by repeating
a pattern



Images from http://www.csis.gvsu.edu/~marzkaj/CS367/project1.htm

Recursion: Can Be Used To Produce Graphics (2)

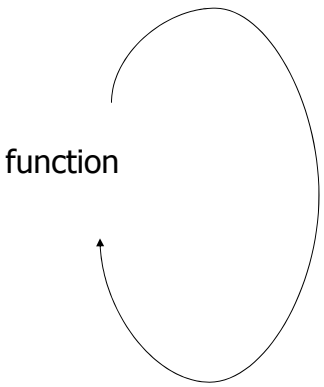


http://charm.cs.uiuc.edu/users/olawlor

Recursion In Programming

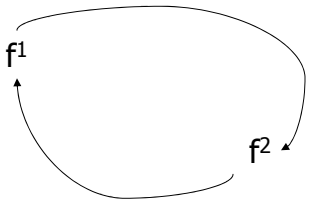
"*A programming technique whereby a function calls itself either directly or indirectly*."

Direct Call

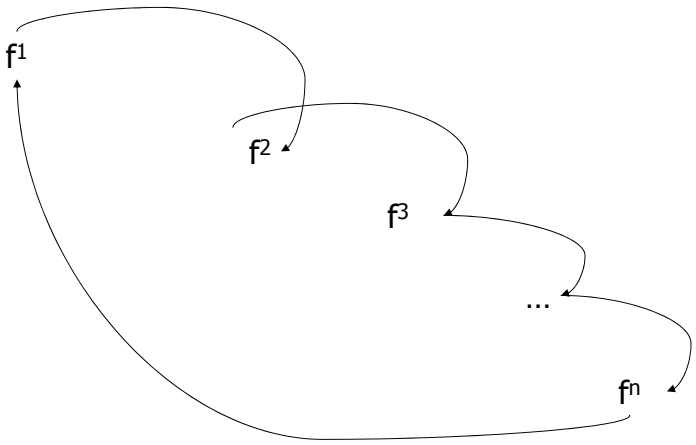



```
class Foo
{
   fun ()
   {
        fun ();
      :
    }
}
```

Indirect Call



Indirect Call



Indirect Call (2)

Location of the online example:

- /home/219/examples/recursion/recursiveExample1
- www.cpsc.ucalgary.ca/~tamj/219/examples/recursion/recursiveExample1

```
public class Recursive1
{
   public void method1 ()
   {
      method2();
   }

   public void method2 ()
   {
      method2();
   }
}
```


Requirements For Sensible Recursion

1) Base case

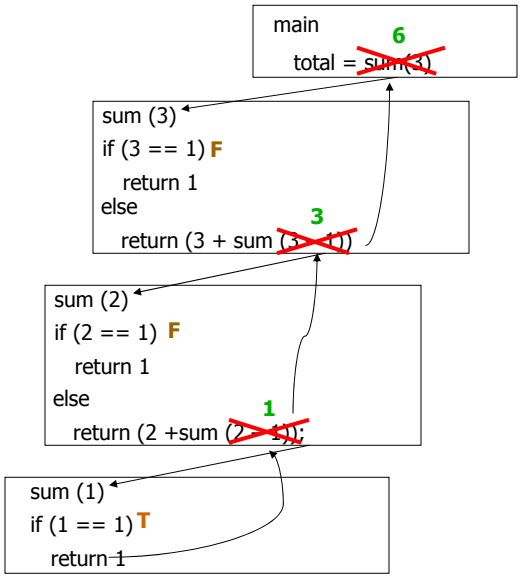
2) Progress is made (towards the base case)

Location of example employing sensible recursion:

- /home/219/examples/recursion/sensibleExample
- www.cpsc.ucalgary.ca/~tamj/219/examples/recursion/sensibleExample

Example Program

```
// Class: Recursive2
public int sum (int no)
    if (no == 1):
        return 1;
    else
        return (no + sum(no-1));

// Class: Driver
main ()
   int last = 0;
   int total = 0;
   Scanner in = new Scanner
                (System.in);
   Recursive2 aRecursive =
                new Recursive2 ();
   last = in.nextInt();
   total = aRecursive.sum (last);
   System.out.println("The sum of the
                series from 1 to "
                +last + " is "+total);
```



When To Use Recursion

- When a problem can be divided into steps.
- The result of one step can be used in a previous step.
- There is a scenario when you can stop sub-dividing the problem into steps (recursive calls) and return to previous steps.
- All of the results together solve the problem.

When To Consider Alternatives To Recursion

- When a loop will solve the problem just as well
- Types of recursion:
  - Tail recursion
    - A recursive call is the last statement in the recursive function.
    - This form of recursion can easily be replaced with a loop.
  - Non-tail recursion
    - A statement which is not a recursive call to the function comprises the last statement in the recursive function.
    - This form of recursion is very difficult (read: impossible) to replace with a loop.

Example: Tail Recursion

- Tail recursion: A recursive call is the last statement in the recursive function.
- Location of example:

  /home/courses/219/examples/recursion/tail

  www.cpsc.ucalgary.ca/~tamj/219/examples/recursion/tail

```
// Class RecursionTail
public void tail (int no) {
      if (no <= 5) {
         System.out.println(no);
         tail(no+1);
      }
      return;
   }

// In main
RecursionTail tailExample = new RecursionTail();
tailExample.tail(1);
```

Example: Non-Tail Recursion

- Non-Tail recursion: A statement which is not a recursive call to the function comprises the last statement in the recursive function.
- Location of example:
  /home/courses/219/examples/recursion/nonTail
  www.cpsc.ucalgary.ca/~tamj/219/examples/recursion/nonTail

```
// Class RecursionNonTail
public void nonTail (int no) {
  if (no < 5)
     nonTail(no+1);
  System.out.println(no);
  return;
}

// Main
RecursionNonTail tailExample = new RecursionNonTail();
tailExample.nonTail(1);
```


Drawbacks Of Recursion

Function calls can be costly

- Uses up memory
- Uses up time

Benefits Of Using Recursion

- Simpler solution that's more elegant (for some problems)
- Easier to visualize solutions (for some people and certain classes of problems – typically require either: non-tail recursion to be implemented or some form of "backtracking")

Common Pitfalls When Using Recursion

- These three pitfalls can result in a runtime error
  - No base case
  - No progress towards the base case
  - Using up too many resources (e.g., variable declarations) for each function call

No Base Case

```
public int sum (int no)
    return (no + sum(no-1));
```


No Base Case

```
public int sum (int no)
    return (no + sum(no-1));
```

When does it stop???

No Progress Towards The Base Case

```
public int sum (int no)
    if (no == 1):
        return 1;
    else
        return (no + sum(no));
```

The recursive case doesn't make any progress towards the base (stopping) case

Using Up Too Many Resources

•Location of example:

/home/courses/219/examples/recursion/resourceExample

www.cpsc.ucalgary.ca/~tamj/219/examples/recursion/resourceExample

```
// RecursionResource
public class RecursionResource
{
  public void method (int no)
  {
        System.out.println(no);
        char [] list = new char[1000000];
        method (++no);
  }
}

// Main
RecursionResource resourceExample = new RecursionResource();
resourceExample.method(1);
```

Another Example Of Recursion

- In order display of Linked list
- Location of the example:
  - /home/219/examples/recursion/linked
  - www.cpsc.ucalgary.ca/~tamj/219/examples/recursion/linked

Another Example Of Recursion: Driver

```
// Main
 Manager aManager = new Manager();
 aManager.display();

 aManager.add();
 aManager.add();
 aManager.add();

 aManager.displayRecursive();
 System.out.println();
```

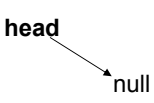
Another Example Of Recursion: Manager

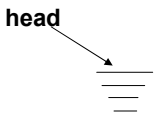
```
public Manager ()
{
     head = null;
}

public void displayRecursive()
{
     System.out.println("Displaying list");
     if (head == null)
     {
        System.out.println("\tList is empty");
        return;
      }
     else
     {
        final int FIRST = 0;
        doRecursiveDisplay (head,FIRST);
     }
  }
```


Another Example Of Recursion: Manager (2)

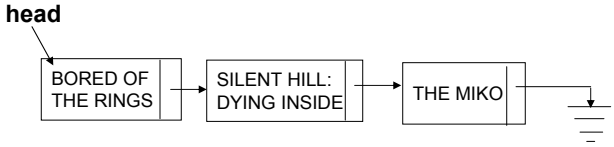
```
private void doRecursiveDisplay (BookNode temp, int index)
{
    if (temp == null)
        return;
   else
   {
        index++;
        System.out.println("\tTitle No. " + index + ": "+
                                  temp.getData().getName());
        temp = temp.getNext();
        doRecursiveDisplay(temp,index);
    }
}
```

Traversing The List: Display



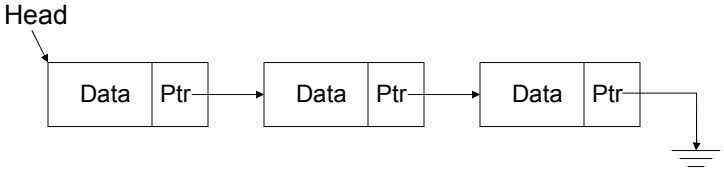


Traversing The List: Display (2)



A Good Example Of When To Use Recursion

•Displaying a linked list in reverse order.



Undergraduate Definition Of Recursion

Word: **re·cur·sion**

Pronunciation: ri-'k&r-zh&n

Definition: See recursion

Wav file from "The Simpsons"

You Should Now Know

- What is a recursive computer program
- How to write and trace simple recursive programs
- What are the requirements for recursion/What are the common pitfalls of recursion