



#CodeYork

Handout 3: Recursion and Examples

What's Recursion?

- Recursion is defining a solution in terms of itself
 - Our problem needs to have some cases where the solution is immediately obvious, and others where the problem can be simplified to a smaller one
 - Keep simplifying our problem again and again until we get a case with an obvious solution
- The simple cases (base cases)
- The other cases, where the problem must be simplified (recursive cases)



Factorials

- The factorial function is: $n! = n * (n - 1) * ... * 2 * 1$
- Some examples are:
 - $1! = 1$
 - $2! = 1 * 2 = 2$
 - $3! = 1 * 2 * 3 = 6$
 - $10! = 10 * 9 * 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1 = 3628800$
- This can be expressed recursively (see exercises)



Palindromes

- A palindrome is a word, phrase, number, or other sequence of characters which reads the same backward or forward.
 - eg. “madam” or “kayak”
- There are two ways to check if a string is a palindrome:
 - Directly from the definition, we can just reverse the string, and then check if it's equal
 - We can use recursion to check if the first and last character are the same, and then, check the if middle part is a palindrome, recursively (see exercises)



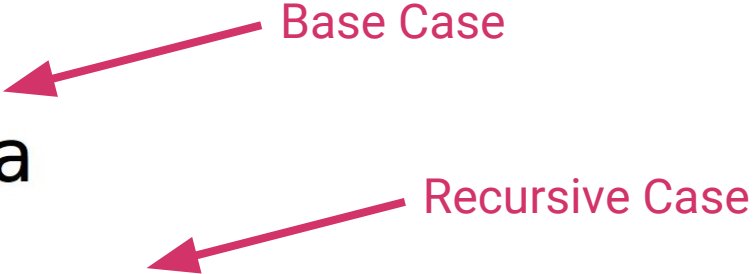
Intro to Euclid's Algorithm

- The GCD of two integers is their greatest common divisor/factor (HCF)
- For example:
 - $\text{gcd}(2, 3) = 1$
 - $\text{gcd}(4, 6) = 2$
 - $\text{gcd}(21, 18) = 3$
- The Euclidean Algorithm finds the GCD of two integers
- This is an example of a recursive algorithm



Euclid's Algorithm Implementation

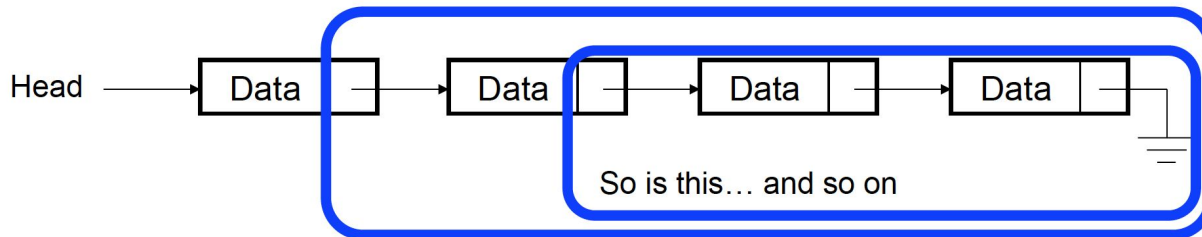
```
def gcd(a, b):  
    if b == 0:  ← Base Case  
        return a  
    else:      ← Recursive Case  
        return gcd(b, a % b)
```



Recursive Data Structures

- Just like algorithms, we can define data structures recursively.
- A linked list is an example of such a data structure.
 - Base case: The linked list is nothing, eg. *None* in Python.
 - Recursive case: The linked list has two items: the first element and the rest of the list.

This is a linked list:



This is also a linked list

Interpreters and Compilers

- Interpreters and compilers are programs that take programs as input!
- Interpreters then run these input programs, if they are valid, and if not they may give us information about why they are invalid, so we can fix them
 - You have seen this with the Python interpreter when using IDLE
- Compilers also take programs as input, but they output programs
 - Programs may be in the same language, but more commonly, are in some lower level language, such as assembly code for a CPU
- Interpreters and compilers use recursion!

