

#CodeYork

Session 3: Recursion and Examples

Recap

- Last time, we looked at:
 - Defining functions
 - Calling functions
 - For loops
 - The range function
 - While loops

Questions? Speak up now!

Part 1: Introduction to Recursion

Recursion

We can define how to solve both the simplest case and how to reduce the difficulty of solving a problem, bit by bit



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)

SAMPLE



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

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)

Course Website

Remember, all the slides and exercises are available at:

<https://york.gjcampbell.co.uk/>

Factorials Task Solution

```
def factorial(n):  
    if n == 1:  
        return 1  
    else:  
        return n * factorial(n - 1)  
  
print(factorial(4))
```

Base Case

Recursive Case

Palindromes Task Solution

```
def is_palindrome(word):  
    if len(word) < 2:  
        return True  
    elif word[0] == word[-1]:  
        return is_palindrome(word[1:-1])  
    else:  
        return False  
  
print(is_palindrome('hannah'))
```

Base Case 1

Recursive Case

Base Case 2

Part 2: Interesting Examples

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

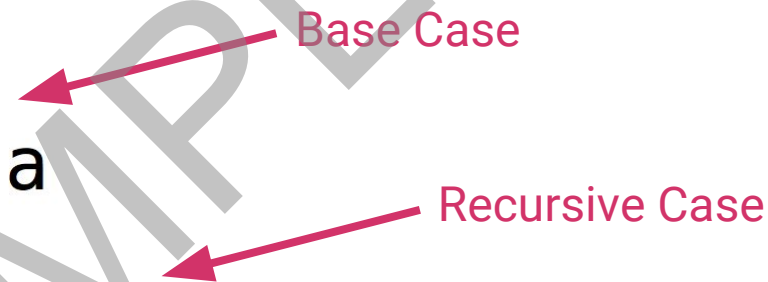


Euclid's Algorithm Implementation

```
def gcd(a, b):  
    if b == 0:  
        return a  
    else:  
        return gcd(b, a % b)
```

Base Case

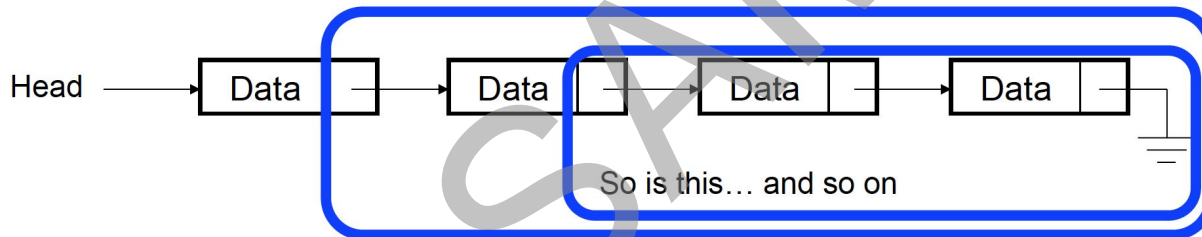
Recursive Case

The diagram consists of two pink arrows pointing to specific lines of code. The first arrow, labeled 'Base Case', points to the line 'return a' under the condition 'if b == 0:'. The second arrow, labeled 'Recursive Case', points to the line 'return gcd(b, a % b)' under the 'else:' condition.

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 see 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!

“To understand recursion, one must first understand recursion.”

- Stephen Hawking

Summary

- Today, we have looked at:

- Recursive definitions
- Factorials
- Palindromes
- GCD and Euclid's Algorithm
- Recursive Data Structures
- Interpreters and Compilers

} Don't worry about this stuff!
Really just for interest's sake!

Questions? Speak up now!

Thanks!

Contact us:

gjc510@york.ac.uk

jr1161@york.ac.uk

<https://york.gjcampbell.co.uk/>

