




# Welcome to the CoGrammar Recursion

**The session will start shortly...**

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.



## Cyber Security Session Housekeeping

---

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.  
**(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: [Questions](#)

## Cyber Security Session Housekeeping cont.

---

- For all **non-academic questions**, please submit a query: [www.hyperiondev.com/support](https://www.hyperiondev.com/support)
- We would love your **feedback** on lectures: [Feedback on Lectures](#)
- Find all the lecture **content** in you [Lecture Backpack](#) on GitHub.

# Safeguarding & Welfare

We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:



Ian Wyles  
Designated Safeguarding  
Lead



Simone Botes



Nurhaan Snyman



Rafiq Manan



Ronald Munodawafa



Tevin Pitts

Scan to report a  
safeguarding concern



or email the Designated  
Safeguarding Lead:  
Ian Wyles  
[safeguarding@hyperiondev.com](mailto:safeguarding@hyperiondev.com)

# ***Stay Safe Series:***

Mastering Online Safety One week at a Time

---

While the digital world can be a wonderful place to make education and learning accessible to all, it is unfortunately also a space where harmful threats like online radicalization, extremist propaganda, phishing scams, online blackmail and hackers can flourish.

As a component of this BootCamp the ***Stay Safe Series*** will guide you through essential measures in order to protect yourself & your community from online dangers, whether they target your privacy, personal information or even attempt to manipulate your beliefs.

## Digital Decorum:

### The Importance of Regular Software Updates

---

- Implement Automated Update Management
- Conduct Regular Vulnerability Assessments
- Establish a Patch Management Policy
- User Awareness and Training:



**SKILLS  
FOR LIFE**

**SKILLS BOOTCAMPS**



Department  
for Education

# CoGrammar

## Recursion

October 2024

# Learning Objectives & Outcomes

- Define recursion and identify its key components.
- Write simple recursive functions.
- Compare recursion with loops and analyse their use cases.
- Identify when to use recursion for solving problems.
- Develop recursive solutions for more advanced problems.



# Recursion

Have you ever used a set of nested mirrors, like the kind in a funhouse, where each mirror reflects the image of another mirror? What was the observation made?

# Recursion



# Polls

Please have a look at the poll notification and select an option.

Which of the following best describes the base case in a recursion function

- A. The point where the function stops calling itself
- B. A function that calls itself indefinitely
- C. A loop that repeats until a condition is met
- D. A function that doesn't need any parameters.

# Polls

Please have a look at the poll notification and select an option.

What comes to mind when you hear the term 'recursion' in programming?

- A. A function calling itself until a base condition is met
- B. A loop that repeats a certain number of times
- C. A concept I've heard of, but I'm not sure how it works.
- D. A confusing concept that often leads to stack overflow errors

# Recursion

- **Definition:** A function calling itself to solve smaller parts of a problem
- Key components:
  - **Base Case:** Stops recursion, prevents infinite loops.
  - **Recursive Step:** Function calls itself with modified input.

# Recursion: Factorial of a number

index.py

```
1 # Finding the factorial of a number
2 def factorial(n):
3     # Base case
4     if n == 0:
5         return 1
6     # Recursive call
7     return n * factorial(n-1)
```

Snipped



# Understanding Recursion Flow

- **Pushing calls onto the stack**
  - A new stack frame is pushed onto the stack for each call.
- **Reaching the base case**
  - When the function reaches this, it stops making further recursive calls.
- **Unwinding the stack (Combining results)**
  - each stack frame is popped off the stack, and control goes back to the previous function call

# Recursion vs Iteration

- **Recursion:** Ideal for problems like tree traversal, self-similar structures.
- **Iteration:** More efficient for problems with fixed repetitions.
- **When to Use:**
  - **Recursion:** Divide-and-conquer, dynamic programming.
  - **Iteration:** Simple loops, known number of iterations.

# Advanced Recursion Concepts

- **Tail Recursion:** Optimizes memory usage, treated like iteration by compilers.
- **Memoization:** Caches results of recursive calls to avoid redundant calculations.
- **Time Complexity:** Important for evaluating recursive function efficiency.

# Conclusion

- Recursion simplifies complex problems by breaking them into smaller steps.
- Understanding base cases and recursive logic is crucial to avoid errors.
- Mastering recursion opens up solutions for various algorithms and data structures.

# Polls

Please have a look at the poll notification and select an option.

Which of the following scenarios is most likely to cause a stack overflow when using recursion?

- A. A recursive function with no base case
- B. A recursive function with a base case but a large number of recursive calls
- C. A function that uses tail recursion
- D. A function that makes a single recursive call per step

# Polls

When is recursion typically more useful than a loop?

- A. When solving problems with a known number of iterations
- B. When solving problems that can be broken into smaller, similar subproblems
- C. When you want to use less memory
- D. When you don't know how to use loops



# Questions and Answers



# Thank you for attending



Department  
for Education

CoGrammar

