



Welcome to this CoGrammar lecture: Iteration and GitHub

The session will start shortly...

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



Software Engineering 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](#)

Software Engineering Session Housekeeping cont.

- For all non-academic questions, please submit a query: www.hyperiondev.com/support
- Report a **safeguarding** incident: www.hyperiondev.com/safeguardreporting
- We would love your **feedback** on lectures: [Feedback on Lectures](#)



Skills Bootcamp Progression Overview

To be eligible for a certificate of completion, students must fulfil three specific criteria. These criteria ensure a high standard of achievement and alignment with the requirements for the successful completion of a Skills Bootcamp.

✓ Criterion 1 - Meeting Initial Requirements

Criterion 1 involves specific achievements **within the first two weeks** of the program. To meet this criterion, students need to:

- Attend a minimum of 7-8 hours per week of guided learning (lectures, workshops, or mentor calls) within the initial two-week period, for a total minimum of **15 guided learning hours** (GLH), by no later than **15 September 2024**.
- Successfully complete the Initial Assessment by the end of the first 14 days, by no later than **15 September 2024**.



Skills Bootcamp Progression Overview



✓ Criterion 2 - Demonstrating Mid-Course Progress

Criterion 2 involves demonstrating meaningful progress through the successful completion of tasks **within the first half** of the bootcamp.

To meet this criterion, students should:

- Complete **42 guided learning hours** and the first half of the assigned tasks by the end of week 7, no later than **20 October 2024**.





Skills Bootcamp Progression Overview

✓ Criterion 3 - Demonstrating Post-Course Progress

Criterion 3 involves showcasing students' progress after completing the course. To meet this criterion, students should:

- Complete all mandatory tasks before the bootcamp's end date. This includes any necessary resubmissions, no later than 22 December 2024.
- Achieve at least 84 guided learning hours by the end of the bootcamp, 22 December 2024.



**SKILLS
FOR LIFE**

SKILLS BOOTCAMPS



Department
for Education

CoGrammar Iteration and GitHub

September 2024

Learning Outcomes

- Understand the syntax and structure of loops in Python.
- Implement loops to solve repetitive tasks.
- Implement nested for loops for more complex iterations.
- Understand the purpose and usage of the range() function in Python.
- Apply range() with different parameters to control the start, stop, and step values.

Learning Outcomes

- **Understand** the concept of trace tables and their importance in debugging.
- **Create** trace tables to manually track the execution of loops and conditional statements.
- **Understand** the basics of version control and the importance of using Git.
- **Initialise** a Git repository, commit changes, and manage branches.

Iterations



Poll

1. Which of the following statements about iteration in Python is true?

- A. A for loop is used to iterate over items of a sequence.
- B. A while loop executes only once regardless of the condition.
- C. Iteration is only possible with numeric data types.
- D. A loop cannot be used to iterate over a string.

Poll

2. What is the main advantage of using a while loop over a for loop?

- A. It's always faster.
- B. It's easier to read.
- C. It can run until a condition is no longer met, regardless of the number of iterations.
- D. None of the above.

Iterations

- Iterations refer to the process of repeatedly executing a set of instructions or a block of code. In programming, iterations are commonly associated with loops, where the same code block is executed multiple times, either for a specified number of times or until a certain condition is met.

Iterations in Coding

- Each execution of the code block within a loop is called an iteration. Iterations are essential for automating repetitive tasks and processing collections of data efficiently.
- *“Loops are like magic tricks in programming that help us avoid doing the same thing over and over again. Instead of writing the same code again, we use loops to make the computer do it for us. This saves time and makes our programs neat and tidy.”*
- In coding, there are two kinds of loops: for loops, which we use when we know exactly how many times we want to repeat something, and while loops, which we use when we want to keep doing something until a certain condition is true.

While Loops



While Loops

- While loops are control flow structures that repeatedly execute a block of code as long as a specified condition is true.
- These are used when you want to execute a block of code repeatedly as long as a specified condition is true. They continue iterating until the condition becomes false.

```
while condition:  
    # code block to be executed
```


While Loop Example

```
count = 0
while count < 5:
    print("Count is:", count)
    count += 1
# This will print numbers from 0 to 4.
```

- `while count < 5`: Start of a while loop. It checks if the value of `count` is less than 5. If this condition is true, the code block inside the loop will execute. If the condition is false, the loop will terminate.
- `print("Count is:", count)`: This line prints the current value of `count` along with the text "Count is:". Since `count` is initially 0, it will print "Count is: 0".
- `count += 1`: This line increments the value of `count` by 1 in each iteration of the loop. So, after the first iteration, `count` becomes 1, then 2, etc.

For Loops



For Loops

- For loops are control flow structures used to iterate over a sequence (such as a list, tuple, string, etc.) and execute a block of code for each element in the sequence.
- For loops are used when you know the number of times you want to execute a block of code.

```
for item in sequence:  
    # code block to be executed
```

For Loop Example

```
# Define a list of fruits
fruits = ["apple", "banana", "cherry", "date"]

# Use a for loop to iterate over the list
for fruit in fruits:
    print(fruit)
```

```
# Result
apple
banana
cherry
date
```

```
# This will print each fruit in the list
```

For Loops – Range Function

- Range is a built-in Python function used to generate a sequence of numbers. It is commonly used with for loops.
- Ranges in for loops are a way to specify a sequence of numbers that you want to iterate over. The range() function generates this sequence of numbers based on the arguments you provide.

```
range(start, stop, step)
```

For Loops – Range Function

- Range() takes three arguments: start, stop, and step.
- **start**: The starting value of the sequence (inclusive). If not provided, it defaults to 0.
- **stop**: The ending value of the sequence (exclusive). This is a required argument.
- **step**: The increment between each value in the sequence. If not provided, it defaults to 1.

Range Function Example

```
for i in range(start, stop, step):  
    # code block to be executed
```

```
for i in range(1, 6): # This will iterate from 1 to 5  
    print(i)
```

- This loop will print numbers 1 through 5. Remember, the stop value is exclusive, so the loop stops before reaching 6.

Trace Tables



Trace Tables

- A trace table is like a step-by-step record of your program's journey through each line of code. It helps you keep track of how the values of variables change as the program runs.

“Imagine you have a friend who's trying to bake a cake by following a recipe. A trace table is like a detailed checklist your friend uses to make sure they don't forget any steps and to see how the cake changes at each step.”

Trace Tables

- Consider the following code:

```
x = 5
y = 2
z = x + y
print(x, y, z)
```

- A trace table would track the values of `x`, `y`, and `z` at each step of execution.

Trace Tables

	A	B	C	D	E
1	Step	x	y	z	
2	1	5	2	-	
3	2	5	2	7	
4	3	5	2	7	
5					
6					

- In the first step, we assign the value 5 to the variable x and the value 2 to the variable y. At this point, we haven't calculated z yet, so its value is None.
- In the second step, we calculate the sum of x and y, which is $5 + 2 = 7$, and assign it to the variable z. Then, we print the values of x, y, and z, which are 5, 2, and 7 respectively.

Let's get coding!



Summary

- **for Loop:** Iterates over a sequence (like lists, strings, or ranges).
- **while Loop:** Repeats as long as a specified condition is true.
- Using **range() Function:** Generates a sequence of numbers for loop iterations. Parameters: start, stop, and step to control iteration.
- **Nested Loops:** Loops within loops to handle multi-dimensional data or complex problems.
- **Trace Tables:** Tools for tracking variable values and the flow of control in loops. Essential for debugging and understanding code execution.

**Let's take a short
break**

CoGrammar



Version Control



What is Version Control?

- Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.
- The Code base is stored in a central place.
- Format used: deltas.
- This means that only changes between versions are saved.
- You can therefor “roll back” your code to a previous version.

Why Version Control?

- Collaboration
 - Multiple people working on the same file at the same time.
 - Hard to keep track of what changes happen when.
 - Certain changes can be accidentally overwritten.
- Understanding What Happened
 - Full history of who made what changes.

Why Version Control?

- Storing Versions
 - Being able to rollback code becomes a great emergency tactic, when bugs become too difficult to handle.
 - Multiple versions and branches of a project can be managed.

Some Terminology

- **Version**: Code at a particular state.
- **Repository**: The collection of all files at all versions.
- **History**: The list of all changes made to a set of files.
- **Commit**: A wrapper for a set of changes.
- **Staging Area**: A file containing changes to be added to the next commit.

Git

CoGrammar



What is Git?

- Git is a distributed version control system for tracking changes in source code during software development.

Why Git?

- Most widely used version control system.
- Free and open-source. Designed to handle a large variety of systems.
- Distributed architecture:
 - When you download a repository, you download the full history of changes to your local computer.
- Everything is run from the command-line using the git application.

Git Installation and Setup

- Download and install Git from git-scm.com.
- Configure user name and email:
 - `git config --global user.name "Your Name"`
 - `git config --global user.email "your.email@example.com"`

Repositories

- Two types: `local` and `remote`.
- All changes stored in a hidden file called `“.git”`.
- Two ways to get a repository:
 - Create a new one using `git init`.
 - Get a remote one using `git clone <repository-url>`.

Initialising a New Local Repository

- `mkdir my-project`
- `cd my-project`
- `git init`
 - Create a `.git` directory that contains all the repository's metadata and object database.

Viewing the Commit Status

- `git status`
 - Shows all new files, changed files, and files added to the current commit.
- E.g:
 - On branch master
 - Your branch is up-to-date with 'origin/master'.
 - Changes to be committed:
 - (use "git reset HEAD <file>..." to unstage)

new file: newFile.py

Staging Changes

- First, you need to add your files in the working directory to the staging area.
 - `git add <file-name>`
- The file is now being tracked and staged for commit.

Committing Changes

- Once you have added all files to the staging area, then you can commit your code.
 - `git commit -m <commit-message>`
 - NB: Each commit has to have a message attached to it.
 - The message just explains what changed.

Viewing the Version History

- `git log`
 - Shows the commit hash (a unique identifier for the commit), Author, Date and the commit message.
- E.g:
`commit a9ca2c9f4e1e0061075aa47cbb97201a43b0f66f`
`Author: HyperionDev Student <hyperiondevstudent@gmail.com>`
`Date: Mon Sep 8 6:49:17 2017 +0200`

`Initial commit.`

Branching

- Sometimes, a developer needs to work independently on the same code base.
- For example: adding a new feature.
- With other changes constantly being made, this can sometimes be difficult and cause many merge conflicts.
- Solution: branching

Branching (Continue)

- To create a new branch:
 - `git branch <branch-name>`
- To switch branches:
 - `git checkout <branch-name>`
- By default, Git uses `main` as the name of the main branch.
 - This used to be called `master`, until Git decided that was a bad idea.

Stashing Changes

- When switching branches, Git will throw up a fuss if you have uncommitted changes.
- However, sometimes your changes are not yet ready for a commit.
- You can use `git stash` to temporarily save your changes to a clipboard without committing.
- To get your changes back, `git stash pop` will get the latest stash on the clipboard.

Merging

- There is no use in branching code to make a new feature without being able to make it a part of the main branch.
- Merging allows you to take the changes that you have made in your branch and apply them to the main branch (or another branch of your choice).
- To merge bug-fix branch into main branch:
 - `git checkout main`
 - `git merge bug-fix`

Handling Merge Conflicts

- Merge conflicts occur when changes in two branches conflict.
- Resolution Steps:
 - Identify conflict files using `git status`.
 - Manually resolve conflicts in the files.
 - Stage the resolved files using `git add`.
 - Complete the merge with `git commit`.

Working with Remote Repositories

- Commands:
 - `git remote add origin <remote-url>` : Add a remote repository.
 - `git push -u origin main` : Push changes to the remote repository.
 - `git pull origin main` : Pull changes from the remote repository.
- This will synchronise the local repository with the remote repository.

Git - Key Commands

<code>git init</code>	Initialise a new Git repository in your project directory.
<code>git add</code>	Add files to the staging area, preparing them to be committed.
<code>git commit</code>	Record changes to the repository with a descriptive commit message.
<code>git status</code>	View the status of files in your repository, including untracked, modified, or staged files.
<code>git push</code>	Push commits from your local repository to a remote repository, such as GitHub.
<code>git pull</code>	Fetch changes from a remote repository and merge them into your local repository.
<code>git clone</code>	Clone a remote repository to your local machine.

GitHub

CoGrammar



What is GitHub ?

- **GitHub** is an online hub for your code where you can work on projects with others, keep track of changes, and showcase your work to the world!
- GitHub improves **Git**'s version control capabilities by offering an intuitive platform equipped with collaborative tools such as pull requests, issue tracking, and project management features.
- Additionally, GitHub serves as a showcase for your work, enabling you to share your projects with a global audience, receive feedback, and contribute to the wider community of developers.

Setting Up a Remote Repository

- Create a Remote Repository on GitHub
 - Go to GitHub and log in to your account
 - [GitHub Account](#)
 - Click on the "+" icon in the top-right corner and select "New repository."
- Link Local Repository to Remote Repository as illustrated before
 - Copy the URL of your newly created remote repository.

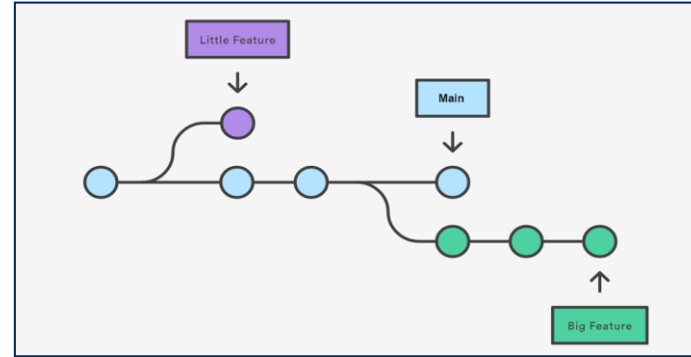
```
git remote add origin <your_remote_repository_url>
```

Directed Acyclic Graph (DAG)

- Imagine GitHub's DAG as a family tree for your code. It visually shows how changes (commits) are connected, branching out and merging back together over time. This helps you understand the history of your project.
- Understanding branches and the DAG is crucial for working together on projects using Git. It keeps everyone organised and allows you to track the history of your code easily.
- On GitHub, you can visualise the DAG by navigating to the "Insights" tab of a repository and selecting "Network."

DAG Branching Visualisation

- Each circle (``o``) represents a commit.
- The ``main`` branch has four commits, and the ``Big Feature`` branch has three commits.
- The branch pointers (``main`` and ``feature``) point to the latest commits in each branch.

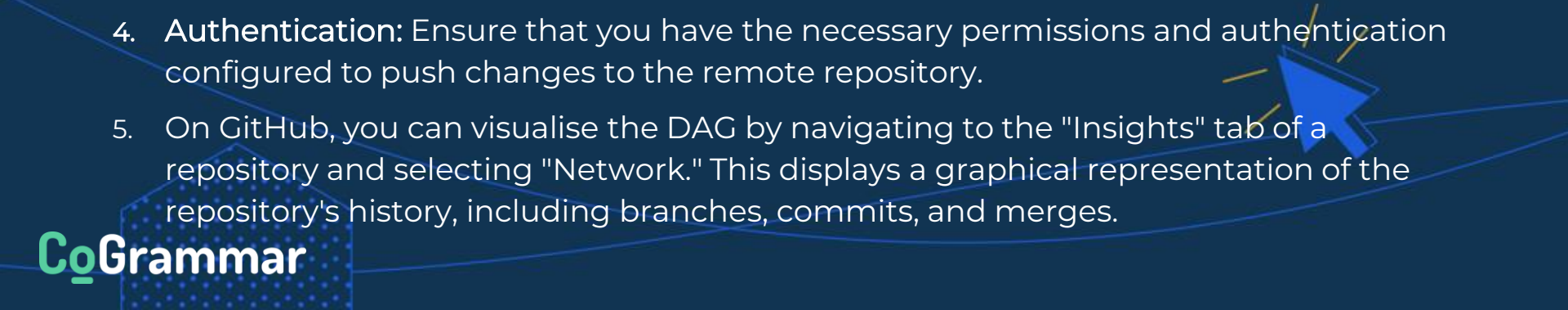


Demo Time!





Git Tips and Best Practices

1. **Branch Tracking:** It's helpful to set up tracking between your local branches and remote branches. This allows Git to automatically determine the remote branch when pushing or pulling changes.
 2. **Review Changes Before Pushing:** Always review your changes before pushing them to the remote repository to ensure they are correct and meet project standards.
 3. **Pull Before Push:** It's a good practice to pull changes from the remote repository before pushing your changes. This helps avoid conflicts and ensures that you're working with the latest codebase.
 4. **Authentication:** Ensure that you have the necessary permissions and authentication configured to push changes to the remote repository.
 5. On GitHub, you can visualise the DAG by navigating to the "Insights" tab of a repository and selecting "Network." This displays a graphical representation of the repository's history, including branches, commits, and merges.
- 

Tasks To Complete

- T04 - Iteration

Additional Resources

- Iterations

<https://docs.python.org/3/library/stdtypes.html#iterator-types>

- Python For Loops

https://www.w3schools.com/python/python_for_loops.asp

<https://realpython.com/python-for-loop/>

- Python While Loops

<https://www.geeksforgeeks.org/python-while-loop/>

- Git CheatSheet

<https://education.github.com/git-cheat-sheet-education.pdf>

Questions and Answers



Thank you for attending



Department
for Education

CoGrammar

