

A dark blue vertical bar is on the left. A blue arrow points right from it, containing the date.

4/8/2025

PORTFOLIO REPORT

Course: Top-Up Bachelor Of Informatics

Module Code: KOL389COM

Module: Open Source Development

Module Leader: Niels Müller Larsen

Several thin, curved lines in shades of blue and grey sweep upwards from the bottom left corner.

Sunil Shrestha

WORD COUNT:2535

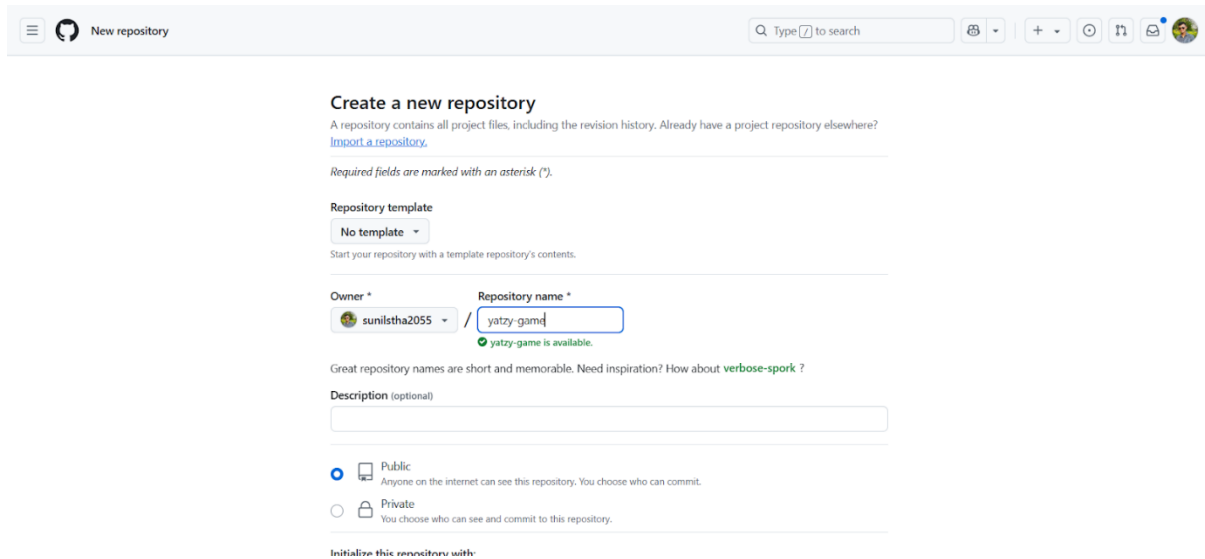
GITHUB LINK: <https://github.com/sunilstha2055/yatzy-game>

Table of Contents

Git and GitHub Workflow Documentation – Yatzy Game.....	1
Create a Repository on GitHub	1
Clone the Repository	1
Check Status	2
Stage Changes	3
Commit Changes	4
Push to GitHub.....	4
Pull Changes.....	5
What is Unit Testing?	6
Why is Unit Testing Important?	6
How Unit Testing Applies	6
For Worksheet 2, you’re tasked with:	6
Introduction.....	8
What is Open Source Development?.....	8
Collaboration of developers	9
Relevance to Me as a Student.....	9
Cost – Effectiveness.....	9
Customizability and Adaptability.....	10
Innovation.....	10
Community support.....	10
Licensing and legal considerations.....	10
Global Impact.....	11
Conclusion	11

Git and GitHub Workflow Documentation – Yatzy Game

Create a Repository on GitHub



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Required fields are marked with an asterisk ().*

Repository template
No template ▾
Start your repository with a template repository's contents.

Owner * sunilstha2055 / **Repository name *** yatzy-game
✓ yatzy-game is available.

Great repository names are short and memorable. Need inspiration? How about [verbose-spork](#) ?

Description (optional)

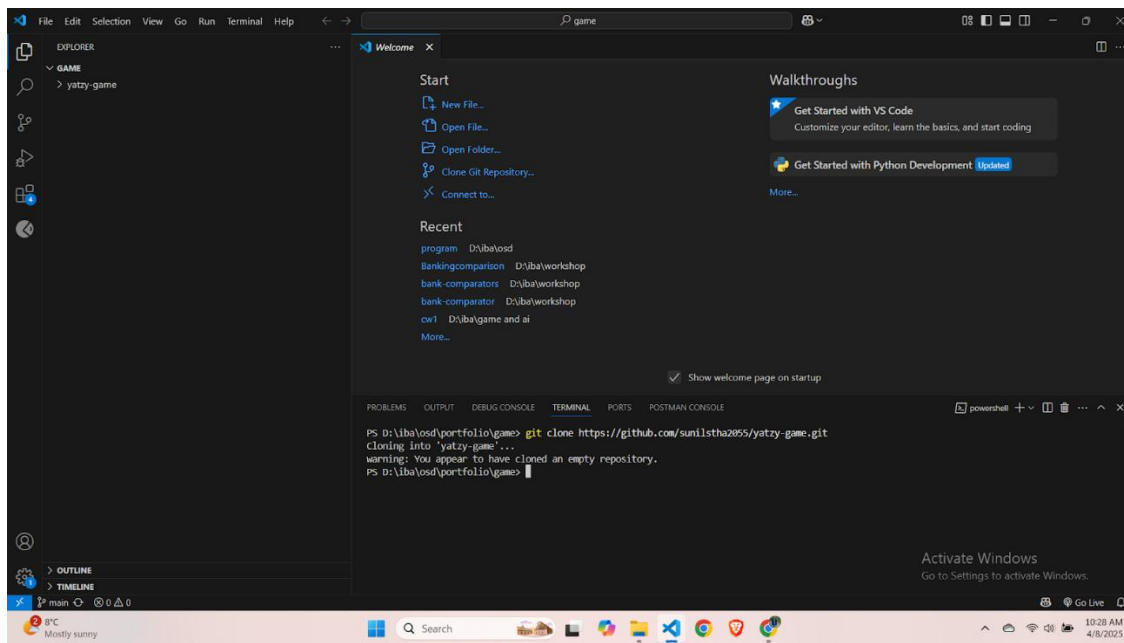
☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

Clone the Repository

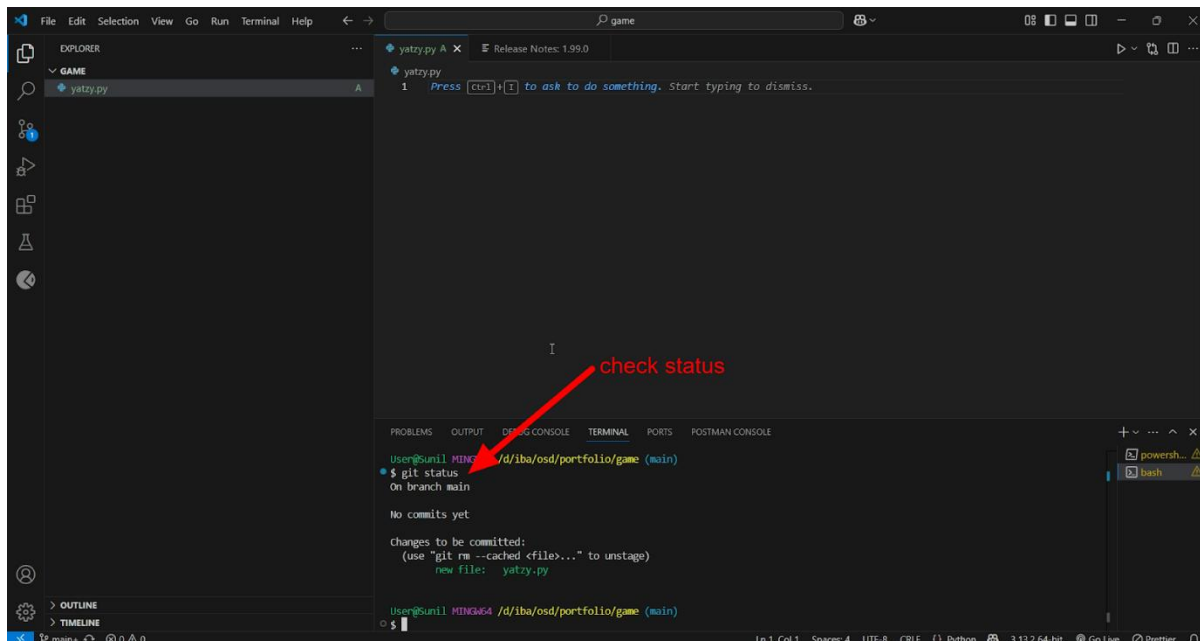
Git Clone is the command we use to create a copy of a remote repository (like one on GitHub or GitLab) on our local machine. It lets us download the entire project, including its history, to work on it locally. When we run `git clone`, it fetches the project's files from the remote repository and sets up a new local repository on our machine. This way, we have a complete copy of the project to work with, and we can make changes, commit them, and push them back to the remote repository.



Check Status

git status

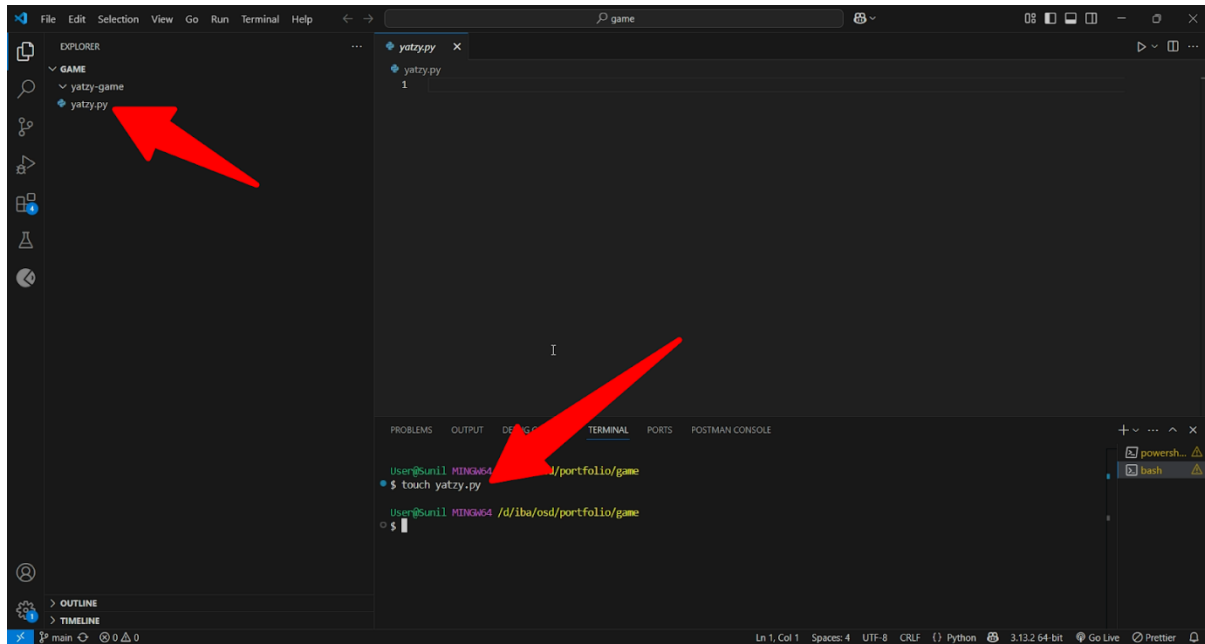
The git status command checks the current state of your Git working directory and staging area. It shows which files have been modified, added, or deleted since your last commit, and whether they're staged for the next commit or untracked (not yet added to Git).



Create Python File

`touch yatzy.py`

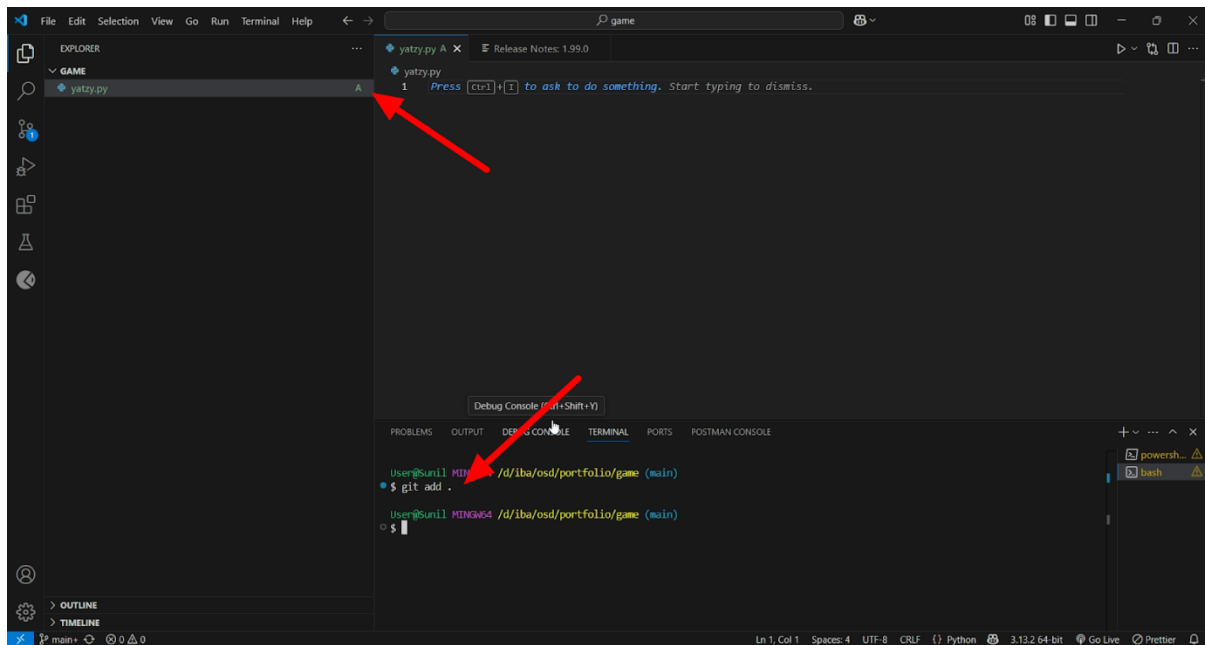
The `touch yatzy.py` command creates a new, empty file named `yatzy.py` in your current directory (assuming you're using a Unix-like system such as Linux or macOS). In the context of Worksheet 2, this is the initial step to create the Python file where you'll implement the Yatzy class with its dice and scoring methods.



Stage Changes

`git add .`

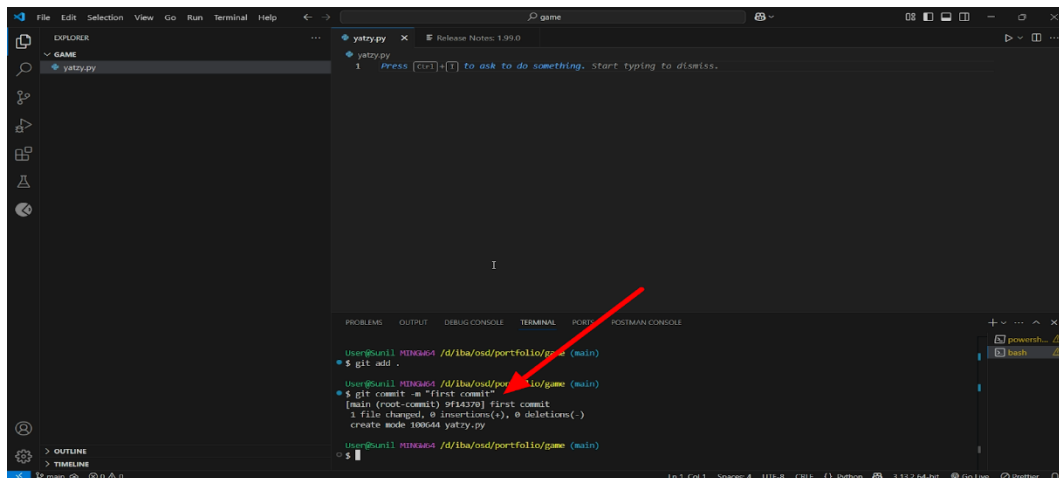
`Git add` is a command we use to tell Git which changes we want to save in your project. When we make changes to files in your project, we use to select the specific files or modifications we want to keep. Once we've added them, they are staged and ready to be saved.



Commit Changes

git commit -m "first commit"

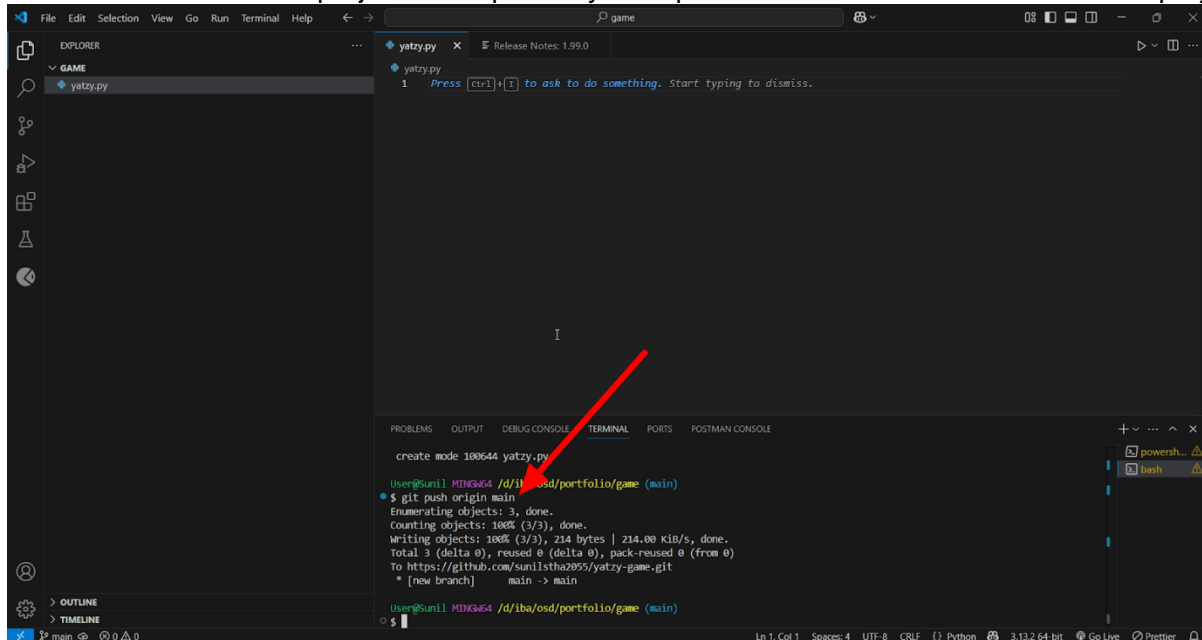
When we use the command git commit, we are telling Git to permanently save the changes we've made and staged with git add. This means the current state of our 4 files, including the changes we've selected, will be saved as a new version or snapshot of our project. Each commit includes a message to describe the changes, making it easier to track the history of our project and understand what was changed and why. Once committed, these changes are safely recorded in the project's history and can be revisited or reverted to if needed.



Push to GitHub

git push origin main

Git push, is the command we use to send our local project changes to a remote repository (like GitHub or GitLab). After we've made changes to our project, staged them with `git add`, and saved them with `git commit`, we use `git push` to share those changes with others. This command uploads our commits to the remote repository so that other team members can see and access the latest version of the project. It keeps everyone up to date with the current state of the project.



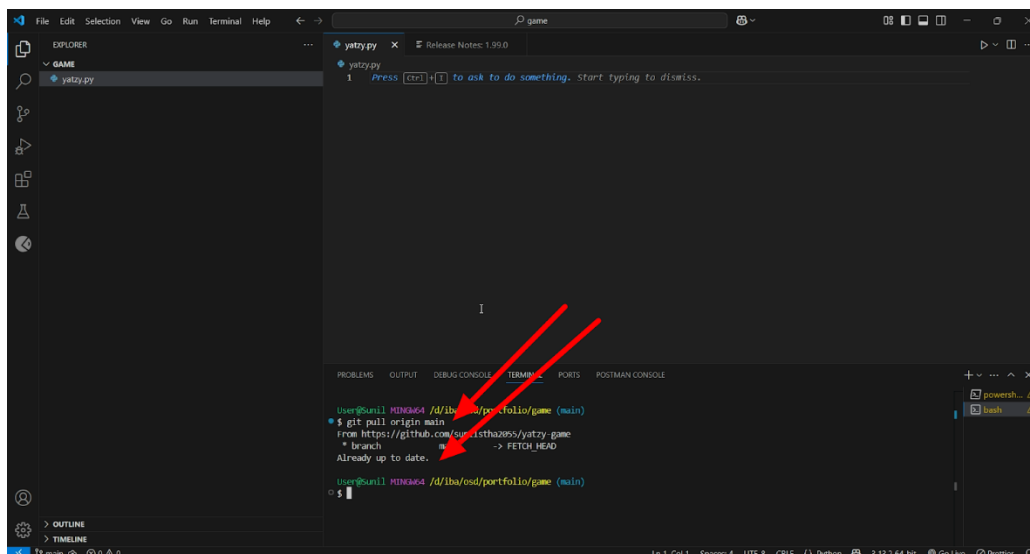
```
File Edit Selection View Go Run Terminal Help
game
EXPLORER
GAME
yatz.py
yatz.py
1 Press [ctrl]+[I] to ask to do something. Start typing to dismiss.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
create mode 100644 yatz.py
User@sunil MINGW64 /d:/liba/osd/portfolio/game (main)
$ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 214 bytes | 214.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/sunilsth2095/yatz-game.git
 * [new branch] main -> main
User@sunil MINGW64 /d:/liba/osd/portfolio/game (main)
$
```

Pull Changes

git pull origin main

Git pull is the command we use to download changes from a remote repository (like GitHub or GitLab) to our local project. When other team members make updates to the project and push their changes to the remote repository, we can use git pull to get those updates and apply them to our local files. This keeps our project up to date with the latest changes made by others. It's a way to make sure our local copy of the project is in sync with the remote repository.



```
File Edit Selection View Go Run Terminal Help
game
EXPLORER
GAME
yatz.py
yatz.py
1 Press [ctrl]+[I] to ask to do something. Start typing to dismiss.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
User@sunil MINGW64 /d:/liba/osd/portfolio/game (main)
$ git pull origin main
From https://github.com/sunilsth2095/yatz-game
 * branch            main       -> FETCH_HEAD
Already up to date.
User@sunil MINGW64 /d:/liba/osd/portfolio/game (main)
$
```


What is Unit Testing?

Unit testing is a software testing method where individual components or "units" of a program are tested in isolation to ensure they work as expected. A "unit" is typically the smallest testable part of an application, such as a function, method, or class. In your case, the units are the methods of the Yatzy class (e.g., Ones(), TwoPairs(), Yatzy()).

- **Key Characteristics:**
 - **Isolation:** Each unit is tested independently of other parts of the program. For example, testing TwoPairs() doesn't rely on the game's full logic, just the dice values.
 - **Automation:** Unit tests are usually automated, meaning they can be run repeatedly without manual intervention (e.g., via GitHub Actions in your assignment).
 - **Granularity:** Focuses on specific functionality, not the entire system.
- **Purpose:**
 - Verify that each method behaves correctly for given inputs.
 - Catch bugs early in development.

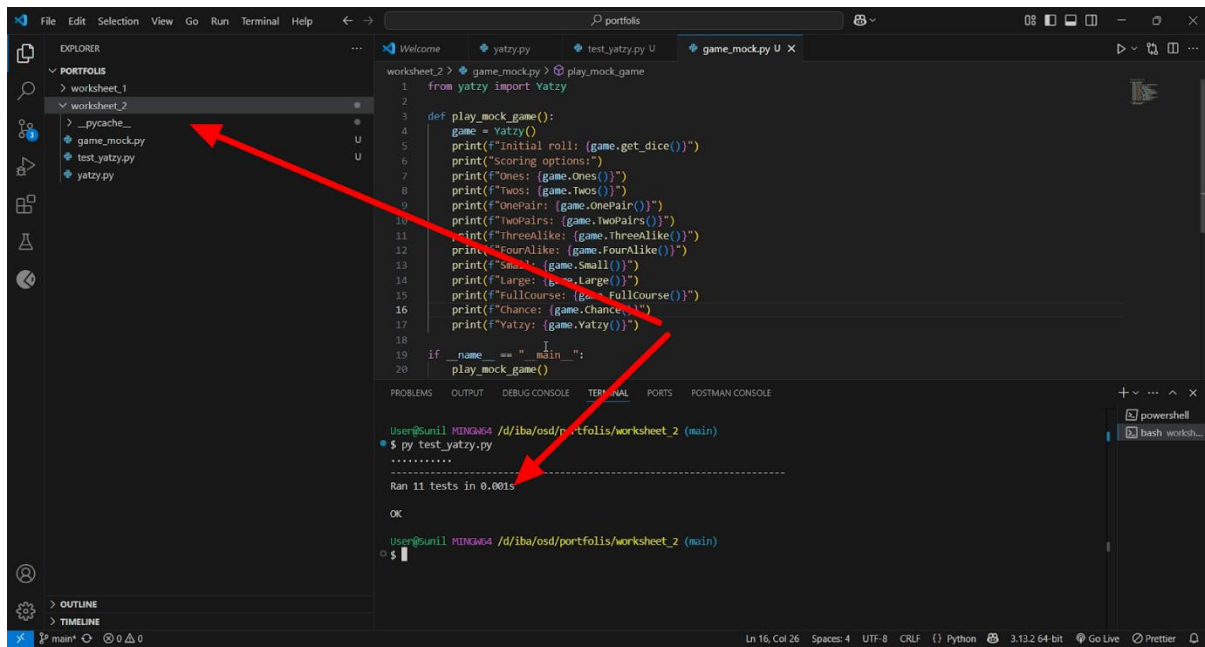
Why is Unit Testing Important?

1. **Reliability:** Ensures your Yatzy class methods return correct scores (e.g., Yatzy() returns 50 only when all dice are identical).
2. **Maintainability:** Makes it easier to update code (e.g., fixing TwoPairs() in Worksheet 3) without introducing new errors, as tests confirm the fix works.
3. **Automation in CI/CD:** In Worksheet 2, GitHub Actions runs your tests automatically on every push, ensuring consistent quality.
4. **Documentation:** Tests act as examples of how methods should work (e.g., test_small() shows Small() expects a 1-2-3-4-5 sequence).
5. **Collaboration:** In Worksheet 3, unit tests help your classmate verify your fix resolves their reported issue.

How Unit Testing Applies

For Worksheet 2, you're tasked with:

- Creating a Yatzy class with specific methods.
- Writing tests for all methods (e.g., Ones(), TwoPairs(), etc.).
- Automating these tests with GitHub Actions.



Open Source Development and Its Relevance

Introduction

Open-source development represents a paradigm shift in how software is created, shared, and maintained. Unlike proprietary software, where code is closely guarded, open-source software (OSS) is freely available for anyone to use, modify, and distribute, typically under licenses like the MIT or GNU General Public License. This collaborative model has fueled some of the most impactful technologies today, from operating systems like Linux to version control systems like Git. As a final-year student in the Faculty of Engineering, Environment, and Computing at Coventry University, enrolled in the KOL389COM Open Source Development module, I've had the opportunity to engage directly with this approach through practical coursework. This essay examines the nature of open-source development, its broader implications, and its significance to me as I prepare to transition from academia to the professional world. Through this module's portfolio—building a Yatzy class, using GitHub, and collaborating with peers—I've gained skills and perspectives that underscore the value of open source in my education and future career.

What is Open Source Development?

Open-source development is fundamentally about community and transparency. It begins with a developer or group making their source code publicly accessible, often on platforms like GitHub, GitLab, or Bitbucket. Others can then contribute by fixing bugs, adding features, or adapting the code for new purposes. A prime example is Linux, an open-source operating system initiated by Linus Torvalds in 1991. Today, it powers everything from servers to smartphones, thanks to contributions from thousands of developers worldwide. Similarly, Git, created by Torvalds in 2005, revolutionized version control, enabling collaborative coding at scale—a tool I've used extensively in this coursework.

The open-source model thrives on principles like peer review and iterative improvement. Code is scrutinized by a diverse community, leading to higher quality and security over time. Tools like Python, with its vast ecosystem of open-source libraries (e.g., NumPy, Flask), demonstrate how this approach accelerates innovation. However, it's not without challenges—coordinating contributors, maintaining consistency, and ensuring funding can be complex. Yet, the benefits often outweigh these hurdles, as evidenced by the dominance of OSS in modern technology.

For me, engaging with open source in this module has meant applying these principles practically. The assignment required me to use GitHub for version control, automate testing with GitHub Actions, and collaborate via Issues—mirroring real-world open-source workflows. This hands-on experience has demystified the process and highlighted its practical utility.

Collaboration of developers

One of the most amazing features of open source is that it unites developers across the world. It connects engineers, developers, as well as enthusiasts who want to share their thoughts on projects of interest. This will help to get the result faster and more accurately, as well as give the developers a sense of ownership. It attracts individuals with different skills and experiences. Multiple developers can work on different aspects of the same project making faster progress. It also provides a platform for that individual who wants to learn and always gives a guide to new developers with less experience which will help them to gain valuable skills for a lifetime.

Relevance to Me as a Student

As a computing student, open-source development is directly relevant to my academic and professional growth. This coursework has equipped me with technical skills that are in high demand. Learning Git, for instance, has been transformative. In Worksheet 1, I practiced commands like ``add``, ``commit``, ``push``, and ``branch`` to manage my Yatzy project on GitHub. These skills are foundational for collaborative software development, where tracking changes and merging contributions are daily tasks. Beyond syntax, Git has taught me the importance of structured workflows—creating branches for features or fixes ensures my code remains organized and recoverable, a lesson I'll carry into group projects or employment.

Unit testing, another key component of Worksheet 2, has deepened my understanding of software quality. Writing tests for my ``Yatzy`` class methods (e.g., ensuring ``TwoPairs()`` returns 0 unless two distinct pairs exist) forced me to think critically about edge cases and expected behavior. Automating these tests with GitHub Actions introduced me to continuous integration (CI), a practice ubiquitous in open-source projects. CI ensures that every code change is validated, reducing errors—a discipline I now appreciate as essential for reliable software.

Collaboration, emphasized in Worksheet 3, mirrors open-source community dynamics. Working with a classmate to identify and fix an issue in my code (e.g., refining ``TwoPairs()``) taught me how feedback loops improve outcomes. Using GitHub Issues to document and resolve problems parallels how open-source maintainers handle contributions. This experience honed my communication and problem-solving skills, preparing me for team-based environments where diverse perspectives drive progress.

Beyond technical skills, open source fosters a mindset of sharing and learning. By engaging with tools and practices used in OSS, I've gained confidence in my ability to contribute to real projects. This coursework has also built a tangible portfolio on GitHub, showcasing my abilities to future employers or collaborators—an asset as I approach graduation.

Cost – Effectiveness

Compared to proprietary software, open source software is more affordable. Open source solutions are available for use by both individuals and organizations without the need to pay licensing fees,

which can be quite expensive, particularly for new and small businesses. These savings on expenses can be put to better use in the form of team growth, marketing, or innovation. Furthermore, volunteers frequently contribute to open-source projects, which lessens the need for businesses to pay more employees for development. Free and open-source tools can also make technology more accessible to a wider audience by reducing the entry hurdle for entrepreneurs and new creators.

Customizability and Adaptability

Customizability is one of the main benefits of open-source software. Developers can alter the source code to meet their unique requirements because it is available for download. Because of this adaptability, businesses can customize the software to meet their own needs, producing more focused and effective solutions. As developers try out various strategies and produce new features and functionalities that may not have occurred to the original designers, customizability also fosters innovation. This dynamic atmosphere promotes innovation and advances the cause.

Innovation

New ideas and concepts are frequently developed in open source projects. Without being restricted by proprietary software, developers are free to experiment with various coding methods, architectures, and technologies. This freedom to experiment has produced groundbreaking ideas and innovations that have completely changed the sector. For instance, Linus Torvalds' Linux operating system, which began as a side project, has developed into one of the most popular operating systems in the world, running embedded equipment as well as desktops and servers. Similar to this, open source initiatives like Mozilla Firefox, MySQL, and the Apache HTTP Server have significantly changed the technological environment.

Community support

Active communities that assist contributors and users are common in open source projects. These communities frequently produce thorough tutorials, documentation, and guides to aid in others' comprehension and efficient use of the product. Beyond only offering technical help, community support encompasses conversations on project roadmaps, feature requests, and best practices. Over time, this collaborative atmosphere aids in preserving the project's relevance and life. Opportunities for networking and career advancement are also generated by the community component of open source development. Through networking, knowledge sharing, and project collaboration, developers can work together on projects and perhaps further their careers.

Licensing and legal considerations

Different licenses are used by open-source projects to specify the usage, modification, and distribution of the program. These licenses vary from copyleft (like the GNU General Public 25 Licence) to permissive (like MIT and Apache). Contributors and consumers must be aware of the

many kinds of open-source licenses and their ramifications. Permissive licenses, on the other hand, provide greater flexibility in how the code can be used and disseminated, whereas copyleft licenses mandate that derivative works be distributed under the same license. Respecting open-source licenses is essential to avoiding legal problems and maintaining the project's integrity. It is important for organizations and developers to be aware of the licenses attached to the software they contribute to and use.

Global Impact

The worldwide reach of open-source development makes it possible for developers to work together on projects from all over the world. A wider variety of needs are addressed by more inclusive and significant solutions as a result of the diversity of viewpoints. In underserved areas, open-source software has also been essential in granting access to technology. Open-source software has been used by initiatives like One Laptop per Child to provide children in underdeveloped nations with access to inexpensive computers.

Conclusion

Open-source development is more than a technical practice—it's a philosophy of collaboration and accessibility that resonates deeply with my journey as a student. Through this module, I've gained proficiency in Git, unit testing, and CI, skills that enhance my employability in a tech industry increasingly reliant on OSS. My personal experience building, testing, and refining the `Yatzy` class on GitHub has demonstrated to me the power of collaborative effort and iterative improvement, aligning with the ethos of open-source communities.

Looking ahead, open source will remain relevant as I enter the workforce. Whether I join a company using OSS tools or contribute to projects myself, the lessons of transparency, quality, and teamwork will guide me. This coursework has not only prepared me technically but also instilled a lifelong learning mindset—open source is ever-evolving, and I'm eager to evolve with it. As I graduate, my GitHub portfolio stands as a testament to this growth, a stepping stone to a career where I can both leverage and contribute to the open-source ecosystem.