VERSION CONTROL TOOLS

C O D E S T A N D A R D S

Introduction
Version Control Tools

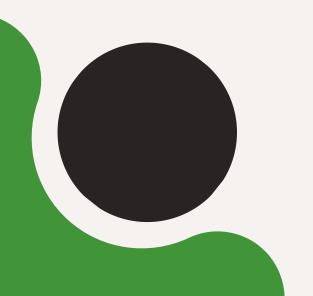
Version control is an essential tool in modern software development, allowing teams to track changes, collaborate efficiently, and maintain code integrity. Alongside version control, adhering to code standards ensures consistency, readability, and maintainability of the codebase.



TWO TYPES

Centralized Version Control Systems (CVCS)

Distributed Version Control Systems (DVCS)



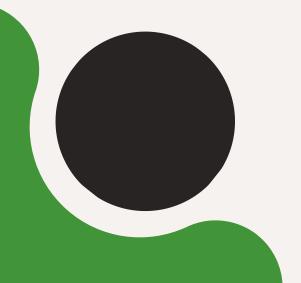
CVCS

- Subversion (SVN): A widely used CVCS where all code resides in a central repository, and developers check out the latest version before making modifications.
- Perforce: Commonly used in enterprise environments for managing large codebases.



DVCS

- Git: The most popular DVCS, offering flexibility, branching, and merging capabilities. Platforms such as GitHub, GitLab, and Bitbucket provide cloud-based hosting services.
- Mercurial: Similar to Git, used for distributed development with a focus on ease of use and performance.





ADVANTAGES

Branching Merging Recovery Backup History Tracking Collaboration

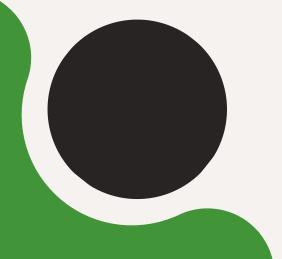
Few Commands!



git init – Initialize a Git repository git clone <repository-url> - Clone a remote repository git status – Check the status of your working directory git add <file-name> - Add a file to the staging area (git add . to add all) git commit -m "message" - Commit changes with a message git branch
 branch-name > - Create a new branch git checkout <branch-name> / git switch <branch-name> - Switch to a branch git merge

 branch-name> – Merge a branch into the current one git push origin
 branch-name> – Push commits to remote git pull origin

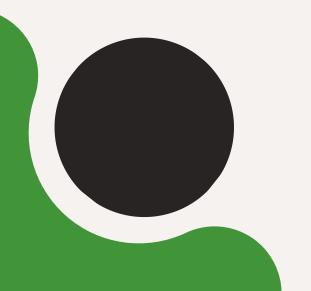
 branch-name> – Fetch and merge changes from remote



Code Standards

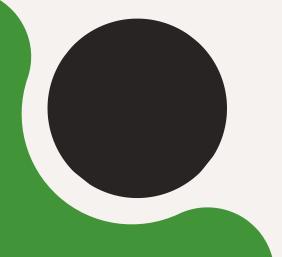
Code standards are essential for maintaining consistent, readable, and secure code across a project.

- Naming Conventions
- Code Formatting
- Documentation
- Error Handling
- Security Best Practices
- Code Review Process



Initial Code

```
Learning-GEN-AI > Tutorials > VCT > 💠 test_script.py > ...
      import os, sys # Unused imports
      def add_numbers(a, b):
         """Adds two numbers and returns the result."""
         return a+b # Formatting issue (spacing around operator)
      def divide_numbers(a, b):
          """Divides two numbers, handling division by zero."""
          try:
              return a / b
 10
          except ZeroDivisionError:
 11
              print("Cannot divide by zero") # No logging used
 12
 13
      print(add_numbers(5,10))
 14
      print(divide_numbers(10, 0))
 15
 16
```



Black

```
Learning-GEN-AI > Tutorials > VCT > 🜵 test_script.py > ...
      import os, sys # Unused imports
      def add_numbers(a, b):
          """Adds two numbers and returns the result."""
          return a + b # Formatting issue (spacing around operator)
  8
      def divide_numbers(a, b):
          """Divides two numbers, handling division by zero."""
 10
 11
          try:
              return a / b
 12
          except ZeroDivisionError:
 13
               print("Cannot divide by zero") # No logging used
 14
 15
 16
      print(add_numbers(5, 10))
 17
      print(divide_numbers(10, 0))
```

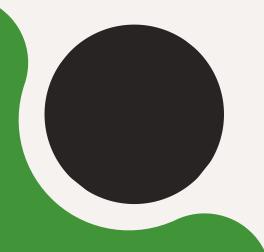


Ruff

```
    (base) sanju@sanju-linux:~/Intern_VH/Learning-GEN-AI/Tutorials/VCT$ ruff chec
    k test_script.py
    test_script.py:1:1: E401 [*] Multiple imports on one line
    test_script.py:1:8: F401 [*] `os` imported but unused
    test_script.py:1:12: F401 [*] `sys` imported but unused
    Found 3 errors.
    [*] 3 fixable with the `--fix` option.

    (base) sanju@sanju-linux:~/Intern_VH/Learning-GEN-AI/Tutorials/VCT$ ruff chec
    k test_script.py --fix
    Found 3 errors (3 fixed, 0 remaining).
```

```
Learning-GEN-AI > Tutorials > VCT > 🜵 test_script.py > ...
  1
      def add_numbers(a, b):
         """Adds two numbers and returns the result."""
         return a+b # Formatting issue (spacing around operator)
      def divide_numbers(a, b):
           """Divides two numbers, handling division by zero."""
  8
          try:
  9
              return a / b
          except ZeroDivisionError:
 10
               print("Cannot divide by zero") # No logging used
 11
 12
      print(add_numbers(5,10))
 13
      print(divide_numbers(10, 0))
 14
 15
```



Flake8

```
(base) sanju@sanju-linux:~/Intern_VH/Learning-GEN-AI/Tutorials/VCT$ flake8 te
st_script.py
test_script.py:1:1: F401 'os' imported but unused
test_script.py:1:1. F401 'sys' imported but unused
test_script.py:1:10: E401 multiple imports on one line
test_script.py:3:1. E302 expected 2 blank lines, found 1
test_script.py:4:4: E111 indentation is not a multiple of 4
test_script.py:5:4: E111 indentation is not a multiple of 4
test_script.py:7:1: E302 expected 2 blank lines, found 1
test_script.py:14:1: E305 expected 2 blank lines after class or function definition, found 1
test_script.py:14:20: E231 missing whitespace after ','
```





Comparision

Feature	Poetry	Ruff	Black	Flake8
Purpose	Manages dependencies and packaging	Lints Python code	Formats Python code	Lints Python code
Configuration	pyproject.toml	.ruff.toml	pyproject.toml	.flake8
Installation	pip install poetry	pip install ruff	pip install black	pip install flake8
Auto-fix	No	Yes	Yes	No
Scope	Project setup and dependencies	Linting code	Formatting code	Linting code
Use Case	Manage Python projects	Check and enforce code quality	Auto-format Python code	Check for style issues





Thank you

