Week-1

L1 Software Development (text editor, compiler)
- dev. envt. → Combinaⁿ of tools (write + run code)
- IDE → additional features for making SD effective
  Syntax check, find particular funcⁿ, renaming var.
  Eg. Pycharm, VS Code, Thonny
- Frameworks - common features bundled together in a framework, build + maintained by comm., helps structure projects
- Web app. framework - Flask
- to share & maintain code versions - VCS (Eg Git)

L2 REST APIs
- Use swagger → Insomnia.

L3 Version Control Systems
- how to handle concurrent As, manage several versions, track As, etc.
- VCS - A sys. that helps in tracking & managing changes to the source code or other documents & maintain versions of code. VCS falls under software configuraⁿ mgmt.
- Features - track & manage As, maintain diff. versions, maintain A history, tool to track issues, provide ways for collaborative dev.
- Centralised VCS - central repo maintained on server, based on client-server model, Eg. Perforce, SVN, highly dep. on server.
- Distributed VCS - entire repo is mirrored locally, everyone owns local copies of the repo Eg. Git.
- Git → free & open source VCS, focus is on speed, data integrity & parallel development.

- ways to use - CmdLine, IDEs, Github portal.
- Modified - after you Δ the file in your working direc
- Staged - you marked the file to be committed
- committed - Δs stored in local DB. Only staged file
- Create new repo — git init → git add file → git commit -m "descrip"
- clone repo - git clone <url>, git push.
- git pull → fetch & download Δs from remote repo & update the local repo.
- git status, git diff, git reset head <file>, git checkout <file>, git log.
- Branch → indep. versions of repo., default branch is master. git branch feature1 → git checkout
- Merge - A way to combine Δs made them in one or more branches to a single branch.
  git merge feature1.
- git rebase.

4 Issue Tracking & Code Review

- Issue tracking → An issue is reported by dev. or Tester or user when encountered, this needs to be saved & tracked. It is a tool that helps managing & tracking all issues.
- Impt. - deliver high quality prod., reduce cost of dev., keeping history of issues, better service, etc.
- do it on github.
- Code review - it is a software quality assurance in which 1 or more people examine the Δs done by dev., also c/d peer review.
- Impt → improves quality, min. debt, risk reduc^{n}, etc.

- Pair programming – agile way, 2 prog. work together in 1 ws. 1 writes code, other reviews, they switches roles frequently.

L5 Debugging.

- error - discrepancy btw. actual & intended behaviour
- failure - observable error
- fault - where the failure has occurred
- debugging - determining the cause of failure.
- steps - reproduce problem, find defect, investigate fix, implement fix, test fix.
- techniques - logging (insert print statements), dump & diff (use diff tool to compare log data btw. execn), stepping in debugger, profiling tool (how often, how long various parts of prog. are executed)
- strategies -
  1. Input manipuln - edit inputs, observe outputs
  2. backwards - find statement that generated incorrect output, follow data backwards.
  3. forwards - find event that triggered incorrect behaviour, follow control flow forward.
  4. black-box debug - find documentn, code egs. to understand correct usage.
- pdb (Python debugger) → import pdb, pdb.set_trace()
  options → n(ext), s(tep), p expression, b(reak), c(ont(inue))

16 Softwares Metrics

- reasons to write clean code - easier to understand by other members, testing bugs, easier to maintain & extend.
- software metric - quantitative way to measure the quality of your code. Tool — Radon
- Cyclomatic complexity (cc) = no. of decisions + 1.
  if(+1), elif(+1), else(0), for(+1), while(+1), except(+1)
  rel - 5(A), 6-10(B), 11-20(C), 21-30(D), 31-40(E), 41+(F)
- Raw metrics → LOC(total lines of code), LLOC(logical), SLOC(source), comments
- Halstead's metrics → Vocab: $n = n_1 + n_2$
  n_1 → distinct operator Len: $N = N_1 + N_2$
  n_2 → " operands calc. len: $\hat{N} = n_1 \log_2 n_1 + n_2 \log_2 n_2$
  N_1 → total operator Vol.: $V = N \log_2 n$
  N_2 → " operands difficulty: $D = \frac{n_1}{2} \times \frac{N_2}{n_2}$

time req.: $T = \frac{E}{18}$ seconds effort: $E = D \cdot V$

no. of bugs: $B = \frac{V}{3000}$

17 Writing Clean Code

- Code Smells → certain problematic charac. of code.
  'Clean code' → 63 code smells.
- Comments → redundant comments, commented out code
- Func^{ns} → should do 1 thing, too long, too many args, flag arg., dead func^{n}
- gen → DRY, incorrect behaviour at boundary, use explanatory variables.
- Refactoring → fixing the code by improving its struc. w/o fixing behaviour. Eg. Pylint.