1) Definition What is computer science? (1)
Computer Science (CS) is the study of computation, i.e. the automatic manipulation of information via algorithms
2) Definition What is an algorithm? (1)
An algorithm is a finite set of well-defined rules for the solution of a problem in a finite number of steps
3) Definition Difference among information, data, and representation (1)
Information is a concept which exists in the world (e.g. the meaning of a word, the value of a number, the color of a pixel)
Representation is a physical or abstract object which carries information
Data is a digital representation of information (e.g. a sequence of bits, bytes, characters, etc.)
4) Definition What is software? (1)
Computer programs, procedures, associated documentation and data related to the operation of a computer system
5) Definition What is software engineering? (1)
The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.
6) History What were software crises? (1)
Software crises is the difficulty of delivering useful and efficient software on time and within budget. Increasing the computational power of computers increases software crises due to the mismatch among the complexity of the software and the inadequacy of the methods used to develop it
7) Commonsense What makes software development costs rise? (1)
8) Commonsense What may delay software development? (1)
9) Commonsense Why could software miss requirements? (1)
10) Commonsense Why may software be inefficient? (1)
11) Commonsense What does it mean for software to be of low-quality? (1)
12) Commonsense What makes code (un)manageable or (un)maintainable? (1)
13) Definition What are the main SE phases? (2)
The main SE phases are: use case collection, requirement analysis, design, implementation, verification, release, deployment, documentation, maintenance
14) Definition In the context of SE phases what is the "Use case collection" phase about? (1)
Is the phase in which we negotiate expectations with customer or stakeholders. The output is a set of user stories
15) Definition In the context of SE phases what is the "Requirement Analysis" phase about? 1
The requirement analysis consists in produce a list of requirements the final product should satisfy. It involve a description about what problems should be solved and how, for each requirement define a clear and unambiguous acceptance criterion and assign to the requirements priorities and weights
16) Definition In the context of SE phases what is the "Design" phase about? (1)
The Design phase consists in produce a blueprint of the software. It include the modelling (what entities represent in the software) and architecture (how to organize the software) phases
17) Definition In the context of SE phases what is the "Implementation" phase about? (1)
The Implementation phase consists in write the code that relies the design into software
18) Definition In the context of SE phases what is the "Verification" phase about? (1)
The Verification phase consists in verify that the software meets the requirements through two steps: automated testing and acceptance testing
19) Definition In the context of SE phases what is the "Release" phase about? (1)
The Release phase consists in make one particular version of the software available to the customers
20) Definition In the context of SE phases what is the "Deployment" phase about? (1)
The Deployment phase consists in install and activate the software for the software
21) Definition In the context of SE phases what is the "Documentation" phase about? (1)
The Documentation phase consists in produce manuals (user manual and development manual) and guides for the software
22) Definition In the context of SE phases what is the "Maintenance" phase about? (1)
The Maintenance phase consists in fix bugs, improve the software, adapt it to new requirements

23) Definition What are the main SE Lifecycle Models? (2)
The main SE Lifecycle Models are: Waterfall Model, V-Model, Incremental Model, Iterative Model, Spiral Model, Scrum Model, Extreme Programming Model
24) Definition In the context of SE Lifecycle, what is the Waterfall Model? (1)
The Waterfall Models is a SE lifecycle model where the phases are sequential despite partially overlapping
25 Definition In the context of SE Lifecycle, what is the V-Model? (1)
The V-Model is a SE lifecycle model where phases are sequential, but with interleaved verification phases

26 Definition In the context of SE Lifecycle, what is the Incremental Model? (1)
The Incremental-Model is a SE lifecycle model where Starting from an abstract design, a number of iterations are perfomed, one per module
27 Definition In the context of SE Lifecycle, what is the Iterative Model? (1)
The Iterative-Model is a SE lifecycle model where Design and subsequent phases are repeated over and over again, for refinement
28 Definition In the context of SE Lifecycle, what is the Spiral Model? (1)
The Spiral-Model is a SE lifecycle model where the focus is on risk analysis, which is performed at each iteration
29 Definition In the context of SE Lifecycle, what is the Scrum Model? (1)
The Scrum-Model is the most famous model for agile development. The goal is balance time to market with agile reactivity to requirements changes.
30 Definition In the context of SE Lifecycle, what is the Extreme Programming Model? (1)
The Extreme Programming Model is a SE lifecycle model where the focus is on the delivery of working software in short time frames. Attempts to mitigate the consequent risk of low-quality code.
31 Definition Difference among shell, terminal, REPL (1)
Shell: is a program that has a simple job: REPL (Read, Evaluate, Print, Loop)
Terminal: is a text-based interface to the operative system (OS). Each terminal application is executing a shell program
REPL: Repl means Read, Evaluate, Print, Loop and are the simple jobs that a terminal can do. It reads a command from the user, evaluates the command, prints the result, it loops back to step 1 unless the user explicitly asks to exit
32 Commonsense What are the benefits of command-line interfaces over graphical ones?2
Because it allows the programmer to interact directly with the operating system, being more precise, it is much more stable compared to IDEs and GUIs, and it uses less CPU.
33 Definition In the context of software engineering, what is a script? (1)
A script is a file containing commands to be executed by a shell
34 Commonsense Pros and cons of automating activities via scripts (2)
Pros are:
Time savings: Automates repetitive tasks, saving time and effort
Cost reduction: Less human reducing operational costs
Scalability and customization: growing and modifying features is easy
Cons are:
Time and techical skills required: : Writing, testing, debugging scripts requires time and technical skills
Expose to vulnerability: Insecure scripts can expose the system to vulnerabilities
35 Terminal In a *nix system shell, how would you inspect the content of a directory? (1)
With **ls** command
36 Terminal In a *nix system shell, how would you to print the current directory location?
With **pwd** command
37 Terminal In a *nix system shell, how would you to remove the file foo? (1)
With **rm foo** command
38 Terminal In a *nix system shell, how would you to remove the directory bar? (1)
With **rm –r bar** command
39 Terminal In a *nix system shell, how would you to change disk? (1)

With / command

40 Terminal In a *nix system shell, how would you to move into some directory? (1)

With **cd pathdirectory** command

41 Terminal In a *nix system shell, how would you to move to the parent directory? (1)

With **cd ..** command

42 Terminal In a *nix system shell, how would you to move a file to another directory? (1)

With **mv source destination** command (sposta da source a destination)

43 Terminal In a *nix system shell, how would you to rename a file? (1)

With **mv foo baz** command (rinomina da foo a baz)

44 Terminal In a *nix system shell, how would you to copy a file into another directory?(1)

With **cp foo baz** command (copia da foo a baz)

45 Terminal In a *nix system shell, how would you to create a directory? (1)

With **mkdir bar** command (crea una directory chiamata bar)

46 Terminal In a Windows system shell, how would you inspect the content of a directory?

With **dir** command

47 Terminal In a Windows system shell, how would you to print the current directory location?

With **echo %cd%** command

48 Terminal In a Windows system shell, how would you to remove the file foo? (1)

With **del foo**command

49 Terminal In a Windows system shell, how would you to remove the directory bar?(1)

With **del bar** command

50 Terminal In a Windows system shell, how would you to change disk? (1)

With **D:** command

51 Terminal In a Windows system shell, how would you to move into some directory?(1)

With **cd pathdirectory** command

52 Terminal In a Windows system shell, how would you to move to the parent directory?

With **cd ..** command

53 Terminal In a Windows system shell, how would you to move a file to another directory? (1)

With **move source destination** command (sposta da source a destination)

54 Terminal In a Windows system shell, how would you to rename a file? (1)

With **move foo baz** command (rinomina da foo a baz)

55 Terminal In a Windows system shell, how would you to copy a file into another directory? (1)

With **copy foo baz** command (copia da foo a baz)

56 Terminal In a Windows system shell, how would you to create a directory? (1)

With **md bar** command (crea una directory chiamata bar)

57 OS In the context of operative systems' shells, what is a directory? (1)

Is a special type of file that is used to organize and store other files and directories. It acts as a container or folder in which files or other directories

58 OS In the context of operative systems' shells, what is a path? (1)

A path is a string that represents the location of a file or a directory in the file system

59 Commonsense What would you do if you don't remember how to use a terminal's command? (1)

I run **man** or **--help** command

60 Commonsense What makes a shell's command interactive? (1)

An interactive command is a command who wait for an user input when you start it. Because the command is starting a process.

61 OS What is the exit code of a command-line program? (1)

Is a code (usually a non negative integer) returned when a command ends.

62 OS In the context of operative systems, what is a process? (1)

A process is an instance of a running program that has its own execution environment. It have 3 essentials communication channels: stdin, stdout, stderr

63 OS In the context of operative systems, what is a stream? (1)

A stream is an unlimited sequence of bytes so may represent anything having a digital representation

64 OS In the context of operative systems, what are the default streams of a process? (1)

The default stream of a process are: stdin, stdout, stderr

65 Terminal Should you edit a textual file via the terminal, what would you do? (1)

I open a shell and run the **nano myfile.txt** command. So I can write what I want and then press **ctrl+O** to save and **ctrl+X** to exit from the editor. **nano** is a simple, interactive, text editor for the terminal.

66 Python What is Python? (1)

Python is a programming language

67 Python In which and how many ways can the python command be invoked? (2)

- **python** with no arguments starts an interactive Python shell
- **python FILENAME** starts a Python process that executes the Python code in the file FILENAME
- **python -m MODULE** starts a Python process that executes the module named MODULE
- **python -c "CODE"** starts a Python process that executes the Python code in the string "CODE"
- **python --help** to inspect the help of the python command
- **python –version** to check the version

68 Project In a Python project, what is the purpose of the requirements.txt file? (1)

Requirements.txt is a file which contains the dependencies of the application

69 Project In a Python project, what is the purpose of the .python-version file file? (1)

.python-version textual declaration of the Python version required by the application

70 Python In the context of the Python programming language, what is pip? (1)

pip is the Python package manager

71 Python In the context of the Python programming language, what is Kivy? (1)

Kivy is a third-party library

72 Python In the context of the Python programming language, what is the compilation cache? (1)

The process of translating a Python code into a machine code that can be executed

73 Definition What is a software library? (1)

Library is a collection of pre-cooked software be reused in different applications

74 Definition What is a dependency of a software project? (1)

Dependency is a piece of software that is required by another piece of software to run

75 Definition What is a transitive dependency of a software project? (1)

A transitive dependency refers to a dependency of a dependency. If A depends on B, and B depends on C, then A transitively depends on C

76 Definition What is the dependency graph of a software project?? (1)

The dependency graph is the graph spa that shows the relationships between different components within a project

77 Definition What is a cyclic dependency in a software project? (1)

78 Definition What is the runtime of a software? (1)

The runtime is the environment in which a piece of software is executed

79 Definition In the context of programming languages, what is a package manager? (1)

A package manager is a command-line-tool to install and manage dependencies

80 Definition What is the difference between compiled and interpreted programming languages? (1)

In the compiled languages, the program's source code is translated into machine code by a compiler before the program is executed. In the interpreted languages the source code is translated line-by-line or statement-by-statement into machine code by an interpreter at runtime.

81 Definition What is the Von Neuman architecture? (1)

The Von Neuman architecture describes a design architecture for an electronic digital computer that explain how computers with CPU, memory, and input/output devices works

82 Definition What are the most common instruction set architectures? (1)

The most common instruction set are: x86, amd64, arm,..

83 Commonsense Is Python a compiled or interpreted programming language? (1)

Python is both compiled and interpreted

84 Definition What is the standard library of a programming language? (1)

A standard library si a collection of reusable code that comes by default with the language

85 Definition In the context of programming languages, what is the difference between standard and third-party libraries? (1)

Standard libraries are part of the language, third-party libraries are developed by someone else. Third-party libraries needs to be installed (with pip command), standard libraries only imported (with import command)

86 Definition In the context of programming languages, what is a package repository? (1)

A package repository are collections of software and metadata about that software, commonly accessible via the web

87 Python In the context of the Python programming language, what is PyPI? (1)

PyPI is the default software repository, full of open-source Python software

88 Commonsense What are the main software platforms you are aware of? (1)

89 Commonsense What are the main package managers and repositories of the main software platforms? (2)

Windows has chocolatey, MacOS has Homebrew, for Linux depend on the distro (apt, centos, pacman,...), Java has Maven and Gradle, JS has npm,...

90 Definition In a software application involving a UI, what is the difference among the model, the view, and the controller? (2)

View: what is shown to the user

Model: what the application does or can do the controller

Controller: the glue between the view and the model

91 Definition In the context of programming, what is the purpose of code decomposition and modularity? (2)

The purpose of code decomposition and modularity is to be easier to develop, test, maintain, and scale

92 Definition In the context of programming, what is encapsulation? (1)

Whenever code is produced to solve a problem, let's encapsulate it into a module for later reuse

93 Definition In the context of programming, what is information hiding? (1)

Whenever a module is produced, only its interface is important for users, the implementation is hidden, and to most extents, irrelevant

94 Python What mechanisms does Python provide to support code decompositon? (1)

Python offers functions, classes, modules, packages

95 Definition What is the interface of a software? (1)

The user interface (UI) is the space by which users interact with the application in order to use it

96 Definition What is the API of a software? (1)

The application programming interface (API) is the space by which developers interact with the application in order to write code

97 Commonsense Describe the overall functioning of UI (2)

1. present the initial UI to the user
2. wait for inputs
3. process the inputs
4. present the output to the user
5. go back to 2

98 Definition In the context of software interfaces, what is the difference among public and private? (1)

Public is intended for external usage (to be used by other developers), private is the opposite of public

99 Python When, how, and why should you differentiate among public and private members in Pythons? (2)

WHEN: You want to hide internal details (encapsulation), control access to data, and maintain flexibility in your codebase.
HOW: By using public (no prefix), protected (single underscore), or private (double underscore) member names, following Python's naming conventions.
WHY: To enforce encapsulation, improve maintainability, control data access, prevent misuse, and make your code clearer and easier to understand.

100 Commonsense What are the classic, inefficient ways of tracking changes in software projects? Why are they inadequate? (3)

- Keep a manual record of changes in a text file (Problema: Forget to update the document, lack of details, time consuming,...)
- Copy the directory of the project to another location (Problema: is a waste of space and you can't see all the updates during the time)
- Copy the modified files to a shared network (Problema: there's no coordination with other members and no possibility to track the version)

101 DVCS What are the functionalities provided by Distributed Version Control Systems? (1)

Tracking the project history, Allowing roll-backs, Collecting meta-information on the changes, Merging information produced at different stages, facilitate parallel workflows

102 DVCS In the context of DVCS, what is a repository? (1)

A repository is a data structure that stores the project history, information on how to roll back changes, authors of changes, dates,...

103 DVCS In the context of DVCS, what is the working tree? (1)

A working tree is the collection of files (usually, inside a root folder) that constitute the project, excluding the meta-data

104 DVCS In the context of DVCS, what is a commit? (1)

A commit is a saved status of the project: collect the changes, creates a snapshot of the status of the worktree, records metadata-

105 DVCS In the context of DVCS, what is a merge commit? (1)

A merge commit is a commit with multiple parents

106 DVCS In the context of DVCS, what is a branch? (1)

A branch is a named sequence of commits

107 Git In the context of Git, what is a commit reference? (1)

A commit reference is a way to identify a specific commit in the version history of a repository

108 Git In the context of Git, what is the HEAD reference? (1)

HEAD reference is a special commit which refers to the current commit

109 Git In the context of Git, what is the difference between relative and absolute commit references? (1)

Absolute commit references refer directly to a specific, unchanging commit in the repository history. Relative commit references refer to commits in relation to the current commit

110 DVCS In the context of DVCS, what does the "checkout" operation mean and involve? (1)

The checkout operation is the operation of moving to another commit (moving to another branch, moving back in time)

111 DVCS In the context of DVCS, what does the "branching" operation mean and involve? (1)

The branch operation is the creation of an independent line of development allowing multiple developers to work on different features of the project

112 DVCS In the context of DVCS, what does the "merging" operation mean and involve? (1)

The branching operation is the integration of changes from a branch into another branch. Consists in combining independent lines of development

113 Git What is Git? (3)

Git is the main DVCS (Distributed Version Control System). It enables multiple developers to work on the same project in the same time keeping the history of changes.

114 Git What is the purpose of the command git config --global user.name "Your Real Name"? (1)

The command *git config --global user.name "Your Real Name"* sets the Git name globally on the system

115 Git What is the purpose of the command git config --global user.email "your.email@address"? (1)

The command sets the Git mail globally on the system

116 Git What is the purpose of the command git config --global core.editor nano? (1)

This command sets Nano as the default text editor for Git operations which requires a text editor

117 Git What is the purpose of the command git config --global init.defaultbranch master? (1)

This command sets how to name the default branch

118 Git What is the purpose and the effect of the command git init in directory my-project? (1)

This command initializes a new repository inside the directory my-project

119 Git In the context of Git, what is the stage? (1)

The stage is the virtual area where changes must be added to be committed. Commits save the changes included in the stage and the files changed after being added to the stage need to be restaged

120 Git What is the purpose of the command git add .? (1)

This command moves the state of all the files (included files who starts with .) into the stage as changes

121 Git What is the purpose of the command reset .? (1)

This command removes currently staged changes of all files from the stage

122 Git What is the purpose and the effect of the command git commit? (1)

This command creates a new changeset with the contents of the stage

123 Git What is the purpose of the command git status? (1)

This command prints the current state of the repository

124 Git In the context of Git, what is HEAD? (1)

HEAD is a name to refers to a current commit

125 Git In the context of Git, what is the purpose of ignoring files? How do you do that? (2)

To ignore some files we have to create a .gitignore file and put the file path to ignore into the .gitignore. The purpose of this is to ignore some files that could be temporary or regenerable or files that could contain private information

126 Git In the context of Git, what is the purpose of the .gitattributes files? (1)

A .gitattributes file is a text file that gives attributes to pathnames

127 OpSy In the context of operative systems, what is the line terminator? (2)

In the context of operating systems, a line terminator is a character or sequence of characters used to indicate the end of a line of text in a file or stream.

128 OpSy In the context of operative systems, what are the most common line terminators, and what systems use which one? (1)

Windows: \r\n Unix: \n Old mac: \r

129 DVCS Let's say you have a number of uncommitted changes in your working tree. What criteria would you adopt to decide which and how many commit to do?

I would commit together conceptual changes, if I make some changes in the project these different changes are divided in different commits, being sure that the project goes from a consistant and working state to another one.

130 Git What is the purpose of the command git log? (1)

The purpose of the **git log** command is to display the commit history of the repository including commit hash, author, date, commit message

131 Git What is the purpose and the effect of the command git checkout -b NAME? 1

This command creates a new branch and switches to it

132 Git What is the purpose and the effect of the command git checkout NAME? (1)

This command is used to switch to an existing branch

133 Git What is the purpose and the effect of the command git checkout XXX -- YYY? 2)

Restore the file YYY to its state in commit or branch XXX.

134 Git In the context of Git, what does "detached head" mean?" (2)

A "detached HEAD" means that the HEAD is pointing directly to a specific commit instead of a branch.

135 GitHub What is the difference between Git and GitHub? (2)

Git is a software you install locally to track changes in your code, GitHub is a service that hosts Git repositories online

136 GitHub In the context of GitHub, what is a fork? What's its purpose? (1)

A fork is a personal copy of someone else's project. The purpose is to p r o p o s e changes to the original project or to use the original project as a starting point for a new idea

137 GitHub In the context of GitHub, what is a pull request? What's its purpose? (1)

A pull request is a proposal to merge your changes from one branch or fork into another repository branch.
Purpose: To review, discuss, and approve code changes before integrating them.

138 GitHub In the context of GitHub, what is an issue? What's its purpose? (1)

An issue is a record of a bugs, enhancement proposals and task to do.
Purpose: To track work, report problems, and organize project development.

139 Git Let's say you have an unpublished repository onto your computer. How can you publish it on GitHub? Specify the terminal commands. (3)

git remote add origin https://github.com/USERNAME/REPONAME.git
git branch -M main
git push -u origin main

140 Git Let's say you have access to a GitHub repository. How can you clone it? Specify the commands and their effects (1)

git clone https://github.com/USERNAME/REPOSITORY.git
Creates a local copy of the repository on your computer and Sets the remote origin so you can pull/push change

141 Git In the context of Git, what is a remote? (1)

Remotes are local names for the known copies of a repository that exist somewhere on the Internet

142 Git In the context of Git, what is the upstream of a branch? (1)

The upstream of a branch is the remote branch it is linked to for pulling and pushing changes

143 Git In the context of Git, what is the "divergent history" situation about? How can that happen? How to fix it? (2)

A divergent history occurs when both the local and remote branches have new different commits that the other doesn't have. It happens when you made commits locally and others pushed different commits to the remote before you pulled or if you try to push without pulling first.

144 Git In the context of Git, what are merge conflicts? How can they happen? How to fix them? (2)

Merge conflicts in Git happen when Git can't automatically merge changes because the same part of a file was modified in different branches. To fix them you manually edit the files to resolve the conflict, *git add <file>* and complete the merge with *git commit --no-edit*

145 Git In the context of Git, what is a branching workflow? (2)

A branching workflow is a strategy for using branches to manage work in a Git project.

146 Git In the context of Git, what is the GitFlow branching workflow? How does it work? (2)

GitFlow is a structured branching model for managing feature development, releases, and hotfixes. They are useful to ensure clear separation of work and stable releases.

147 Git What is the purpose and the effect of the command git push? (1)

To upload your local commits to a remote repository

148 Git What is the purpose and the effect of the command git pull? (1)

To download and integrate changes from the remote repository into your local branch

149 GitHub In the context of GitHub, what is an organization? (1)

An organization on GitHub is a space where multiple users can collaborate on repositories. It's used for companies, groups, or projects

150 GitHub In the context of GitHub, what is a Wiki? (1)

A Wiki is a space inside a GitHub repository where write and maintain **documentation** directly within the repo

151 QA In the context of SE, what is quality assurance? (1)

Quality assurance is the set of activities and practices aimed at ensuring that a software product works and it is of good quality.

152 QA When is software considered "correctly working"? (2)

Software works when it meets the requirements

153 QA When is software considered "of good quality"? (2)

Software is good when it is easy for developers to evolve or maintain it. The software should be reproducible, sustainable, evolvable, maintainable, scalable

154 Testing In the context of software engineering, what is testing? (1)

Testing is the process of executing a program to find bugs and to verify that the software behaves as expected

155 Testing In the context of software testing, what is the difference between an automated and manual test? (2)

Manual test: executed by a human, good for exploratory or visual tests, high time-consuming

Automated test: executed by a script, fast and repeatable, ideal for frequent checks

156 Testing In the context of software testing, what are the most common testing scopes? (2)

Unit testing: tests single components or functions in isolation.
Integration testing: checks how different modules work together.
System testing: verifies the behavior of the complete application.
End-to-end testing: validates that the system meets business/user requirements

157 Testing What is the difference among unit, integration, and end-to-end testing? (3)

Unit testing: tests single components (e.g. one function or class)
Integration testing: tests how components interact together
End-to-end testing: tests the entire system from the user's perspective

158 Commonsense What is the problem in skipping unit testing and just focus on integration testing? (1) If you skip unit testing and only do integration testing, you risk these problems: Harder to find bugs, Slower debugging, Weaker coverage

159 Testing Why one may want to have automated tests in a software project? (1)

They act as sentinels for the early detection of problems and They make testing cheap and fast, so developers can test often and early

160 Testing What issues may arise in the long run when a software project is lacking automated testing? (1)

Without automated tests, the code becomes fragile, hard to evolve, and more expensive to maintain over time.

161 Commonsense Why is reproducibility important in testing? How to achieve it? (2)

Reproducibility in testing is crucial to ensure that tests give consistent results and help reliably detect bugs. It allows developers to debug issues more easily and maintain trust in the test suite. To achieve it, tests should avoid randomness, use fixed data, and run in controlled environments

162 Testing What is test code? How to separate it from the main code? Why? (2)

Test code is the part of a project written to check that the main code behaves as expected. It should be separated from the main code by placing it in dedicated folders (`/tests`) and using clear naming. This separation keeps the main code clean and organized.

163 Testing What is test driven development (TDD)? (2)

TDD is a practice in which tests are not only a form of validation, but also a form of specification.
The practice of: converting requirements to test cases, preparing tests before development, define the expected behavior via test cases, track all development by always testing all cases

164 Testing In what sense can software test act as a form of specification? (1)

Software tests can act as a form of specification because they describe the expected behavior of the code in a precise and executable way

165 Testing What is technical debt? How is it related to software testing? (1)

Technical debt is the cost of quick, limited solutions that require extra work later. Without good testing, it's harder to catch bugs and safely change code, increasing that debt over time.

166 Testing How to deal with a project which was not following TDD since the very beginning? 3

Begin by writing tests for the most critical parts of the code (especially those that are often modified or prone to bugs) and gradually increase coverage by adding tests when fixing bugs or adding new features.

167 Testing In the context of software testing, what is a regression? (1)

A regression is a bug that reappears in code that previously worked correctly

168 Testing What are test doubles and what problem do they address? (3)

Test doubles are simulated versions of components used in place of real ones during testing. They help isolate the unit under test and reduce testing complexity and cost. Types include dummies (placeholders), stubs (basic responses), spies (track usage), mocks (verify expected interactions), and fakes (simplified working versions not for production).

169 Testing In the context of software testing, what is test coverage? (2)

Code coverage is a set of metrics that measure how much of the source code of a program has been executed when testing.

170 Testing What are the common metrics for test coverage? (2)

Function coverage: did the flow control get through this function?
Branch coverage: did the flow control tried both branches of this condition?
Line-of-code coverage: did the flow control get through this line during tests?

171 Testing How to measure the test coverage of a Python project? (2)

It can be used the coverage tool. First, install it with *pip install coverage*. Then run your tests with coverage tracking using *coverage run -m pytest*, and generate a report with coverage report (text) or coverage html (browser). This shows how much of your code is executed by tests.

172 Testing If a project has 100% test coverage and 100% success rate for tests, can we declare it bug free? Can we safely say it satisfies all requirements? (2)

We cannot declare a project bug-free or fully compliant with requirements. Tests may miss edge cases, logic errors, or incorrect assumptions or simply few requirements covered.

173 QA Aside from testing, what is quality assurance about? (2)

Aside from testing, quality assurance (QA) is about defining processes, standards, and practices to ensure that software is developed correctly. It includes code reviews, documentation, process audits, and tool adoption to prevent defects before they occur.

174 QA In the context of software engineering, what is static analysis? (1)

Static Analysis is the Code analysis without execution

175 QA What static analysis tool may you exploit when working on a Python project? What's their purpose? (2)

Mypy: static analysis for bug detection **Pyflakes**: effective programming, excluding style
Pylint: reverse engineering, style, and effective programming **Bandit**: security scanner

176 Build Automation What is the build life cycle of a software project? (2)

The build life cycle is the process of creating tested deployable artifacts from source code

177 Build Automation In the context of Software Engineering, what is build automation? (2)

In Software Engineering, build automation refers to Create software that automates the building of some software

178 Build Automation What can you expect, in general, from a build automation tool? (2)

Help for all the general ue case of building software. Time saving and less repetition of things to do

179 Build Automation In the context of build automation, what is dependency locking? Why is it necessary? (2)

Dependency locking is the practice of fixing the exact versions of a project's dependencies to ensure consistent and repeatable builds. It is necessary because without locking, updates or changes in dependencies can introduce bugs or incompatibilities, making the build unreliable and harder to reproduce.

180 Poetry In a Python project, what would you use Poetry for? (3)

In a Python project, you would use Poetry to manage dependencies, packaging, and publishing

181 Python Where would you release a Python project on Test PyPI? When on PyPI? (2)

You release a Python project on Test PyPI to test the packaging and publishing process without making it publicly available in the main index. Once everything works correctly, you release it on PyPI to distribute it officially to all users

182 Python What is Test PyPI? What's its purpose? (1)

Test PyPI is a separate instance of the Python Package Index used for testing the packaging and distribution of Python projects. Its purpose is to let developers verify that their packages build and upload correctly before publishing them to the official PyPI.

183 Poetry What are the steps to release a Python project via Poetry? (2)

- Configure the project with *pyproject.toml*
- Build the package using *poetry build*
- Publish to Test PyPI with *poetry publish --repository test-pypi*
- Publish to PyPI with *poetry publish* after configuring credentials

184 Versioning In the context of SE, what is versioning? (1)

Versioning is the process of assigning a unique identifier to a unique state of some software

185 Versioning At what levels can versioning occur? (2)

Versioning can happen at different levels: Code (DVCS), Feature version, software release

186 Versioning What are the admissible scopes for software versioning? (1)

The admissible scopes are:
- Internal: Identifies a point in development, Changes do not impact the "outer world"
- External: A publicly visible release of the software, Changes are disruptive

187 Versioning In the context of software versioning, what is code naming? What's its purpose? (1)

Is a word, short name or acronym which the versioni s represented by. Its purpose is to make the version easily identifiable

188 Versioning What are the versioning approaches you are aware of? (2)

- SemVer: Uses a format like MAJOR.MINOR.PATCH
- Data based versioning: The version is represented by a string representing the release date
- Unary nnumbering: The version is represented by a string whose length grows at each version
- Degree of retro compatibility: The version is represented one or more numbers, separately incremented, that reflect incrementally widespread changes in the product

189 Versioning Provide an overview of date-based versioning: purpose, functioning, pros, and cons (2)

The version is represented by a string representing the release date.
Its purpose is to clearly indicate when a version was released
The main advantage is its clarity in showing the software's age and novelty. The cons: it can be misleading because the amount of development or changes doesn't always match the time elapsed.

190 Versioning Provide an overview of unary numbering (in the context of software versioning): purpose, functioning, pros, and cons (2)

The version is represented by a string whose length grows at each version. Its purpose is to show incremental progress by extending the version number.
The pro is that is mainly useful for mature projects with slow, steady updates. The cons is: can cause version length to explode, making it impractical for most projects.

191 Versioning Provide an overview of Semantic Versioning: purpose, functioning, pros, and cons 3

Semantic Versioning (SemVer) uses a format like X.Y.Z (Major.Minor.Patch).
Its purpose is to clearly communicate the impact of changes: major versions introduce backward-incompatible changes, minor versions add backward-compatible features, and patch versions fix bugs without breaking compatibility.

The pros include clarity in version meaning and easier dependency handling, while the cons are the need for strict discipline in versioning and API management.

192 Versioning How can one tie semantic versioning to DVCS commits? (1)

193 Versioning In the context of DVCS, what are conventional commits? What's their purpose? (2)

```
<type>[optional scope]: <description>

[optional body]

[optional footer(s)]
```

Their purpose is improve readability of commit history and Support semantic versioning automation

194 Versioning In the context of conventional commits, what is a breaking change? (1)

A commit that has a footer BREAKING CHANGE: introduces a breaking API change

195 Versioning In the context of conventional commits, what is a feat? (1)

A commit of the type feat introduces a new feature to the codebase

196 Versioning In the context of conventional commits, what is a fix? (1)

A commit of the type fix patches a bug in your codebase

197 Versioning In the context of conventional commits, what is a chore? (1)

A commit of the type chore refers to routine tasks that do not affect the application's functionality

198 Versioning In the context of semantic versioning, what is the difference among a major, minor, or patch change? (2)

Patch change (Z): is incremented when backward-compatible bug fixes are introduced
Minor change (Y): is incremented when new, backward-compatible functionality is added to the public API or when existing functionality is deprecated
Major change (X): is incremented when backward-incompatible changes are made to the public API.

199 Versioning What is the idea behind semantic release? (2)

The core idea behind semantic release is to automate the release process of a software project by relying on semantic versioning and structured commit messages

200 Versioning How are semantic versioning, conventional commits, and semantic release related? 2

These three concepts work together to automate and standardize the software release process in a consistent way.

201 Versioning Soppose that, in your Python project, you add one more public method to a class or module. Is this a major, minor, or patch change? (1)

MINOR

202 Versioning Soppose that, in your Python project, you add one more private method to a class or module and you use it inside another public function of that class or module. Is this a major, minor, or patch change? (1)

PATCH

203 Versioning Soppose that, in your Python project, you rename (all occurrences of) a public function. Is this a major, minor, or patch change? (1)

MAJOR

204 Versioning Soppose that, in your Python project, you rename (all occurrences of) a private function. Is this a major, minor, or patch change? (1)

PATCH

205 Versioning Soppose that, in your Python project, you rename (all usages of) a public function's parameter name. Is this a major, minor, or patch change? (1)

MAJOR

206 Versioning Soppose that, in your Python project, you rename (all usages of) a private function's parameter name. Is this a major, minor, or patch change? (1)

PATCH

207 Versioning Soppose that, in your Python project, you rename (all occurrences of) a public class. Is this a major, minor, or patch change? (1)
MAJOR
208 Versioning Soppose that, in your Python project, you rename (all usages of) a public class's constructor's parameter name. Is this a major, minor, or patch change? (1)
MAJOR
209 Versioning Soppose that, in your Python project, there's function f which is slow. After some edits to its body, you manage to make it much faster. Is this a major, minor, or patch change? (1)
PATCH
210 Versioning Soppose that, in your Python project, you add one more public method to a class or module. You commit the changes using conventional commit. What type would you use for the commit? 1
FEAT
211 Versioning Soppose that, in your Python project, you add one more private method to a class or module and you use it inside another public function of that class or module. You commit the changes using conventional commit. What type would you use for the commit? (1)
CHORE
212 Versioning Soppose that, in your Python project, you rename (all occurrences of) a public function. You commit the changes using conventional commit. What type would you use for the commit?
FEAT
213 Versioning Soppose that, in your Python project, you rename (all occurrences of) a private function. You commit the changes using conventional commit. What type would you use for the commit?
REFACTOR
214 Versioning Soppose that, in your Python project, you rename (all usages of) a public function's parameter name. You commit the changes using conventional commit. What type would you use for the commit? (1)
FEAT
215 Versioning Soppose that, in your Python project, you rename (all usages of) a private function's parameter name. You commit the changes using conventional commit. What type would you use for the commit? (1)
CHORE
216 Versioning Soppose that, in your Python project, you rename (all occurrences of) a public class. You commit the changes using conventional commit. What type would you use for the commit? (1)
FEAT
217 Versioning Soppose that, in your Python project, you rename (all usages of) a public class's constructor's parameter name. You commit the changes using conventional commit. What type would you use for the commit? (1)
FEAT
218 Versioning Soppose that, in your Python project, there's function f which is slow. After some edits to its body, you manage to make it much faster. You commit the changes using conventional commit. What type would you use for the commit? (1)
REFACTOR
219 Versioning Suppose your Python project is currently at version 1.2.3. You add one more public method to a class or module. What should be the next version number if this change is going to be released immediately? (1)
1.3.0
220 Versioning Suppose your Python project is currently at version 1.2.3. You add one more private method to a class or module and you use it inside another public function of that class or module. What should be the next version number if this change is going to be released immediately? (1)
1.2.4

221 Versioning Suppose your Python project is currently at version 1.2.3. You rename (all occurrences of) a public function. What should be the next version number if this change is going to be released immediately? (1)
2.0.0
222 Versioning Suppose your Python project is currently at version 1.2.3. You rename (all occurrences of) a private function. What should be the next version number if this change is going to be released immediately? (1)
1.2.4
223 Versioning Suppose your Python project is currently at version 1.2.3. You rename (all usages of) a public function's parameter name. What should be the next version number if this change is going to be released immediately? (1)
2.0.0
224 Versioning Suppose your Python project is currently at version 1.2.3. You rename (all usages of) a private function's parameter name. What should be the next version number if this change is going to be released immediately? (1)
1.2.4
225 Versioning Suppose your Python project is currently at version 1.2.3. You rename (all occurrences of) a public class. What should be the next version number if this change is going to be released immediately? (1)
2.0.0
226 Versioning Suppose your Python project is currently at version 1.2.3. You rename (all usages of) a public class's constructor's parameter name. What should be the next version number if this change is going to be released immediately? (1)
2.0.0
227 Versioning Suppose your Python project is currently at version 1.2.3. There's function f which is slow. After some edits to its body, you manage to make it much faster. What should be the next version number if this change is going to be released immediately? (1)
1.2.4
228 Versioning Since the last release (1.2.3), your semantic-versioned project's main branch contains the following commit types: fix, fix, fix. Should you release now, what's the next version number? (1)
1.2.4
229 Versioning Since the last release (1.2.3), your semantic-versioned project's main branch contains the following commit types: fix, feat, feat. Should you release now, what's the next version number? (1)
1.3.0
230 Versioning Since the last release (1.2.3), your semantic-versioned project's main branch contains the following commit types: feat, feat, feat. Should you release now, what's the next version number? (1)
1.3.0
231 Versioning Since the last release (1.2.3), your semantic-versioned project's main branch contains the following commit types: chore, feat!, fix. Should you release now, what's the next version number? (1)
2.0.0
232 Versioning Since the last release (1.2.3), your semantic-versioned project's main branch contains the following commit types: fix, feat!, feat. Should you release now, what's the next version number? (1)
2.0.0
233 CI In your own words, what is continuous integration? (1)
Continuous Integration (CI) is a development practice where developers frequently merge their code changes into a shared repository

234 CI In your own words, what is integration hell? (1)

When multiple team members work in isolation and try to integrate their work late in the development cycle, they often face numerous merge conflicts, unexpected bugs, and compatibility issues

235 CI In the context of continuous integration, what is a pipeline? (2)

A pipeline is an automated sequence of steps that run each time code is integrated into a shared repository (Fetching the code, Installing dependencies, Running automated tests,…)

236 GHA What is GitHub Actions? (1)

GitHub Actions is a CI/CD platform built into GitHub.

237 CI In the context of continuous integration, provide an overview of the abstract pipeline design (2)

Designing a CI pipeline means organizing all the necessary steps to build, verify, and deliver a software project. These operations are structured as a dependency graph, where each step depends on others. The graph is then modeled using the abstractions of the chosen CI provider. The goal is to express the entire CI logic.

238 GHA In the context of GitHub Actions, what is the difference among workflow, jobs, steps (2)

Workflow: A YAML-defined automation triggered by events (e.g., push, pull request). It can contain multiple jobs and runs independently from other workflows.

Job: A set of steps executed sequentially in a fresh virtual machine. Jobs run in parallel unless dependencies are specified. Matrix strategy allows running jobs with different OS/runtimes.

Step: An individual shell command within a job (e.g., install dependencies, run tests). Steps run in the order they're defined

239 GHA How would you design a GitHub Actions workflow for a Python project? (3)

240 GHA In the context of GitHub Actions, what is a runner? (1)

A runner is a virtual machine that executes GitHub Actions jobs.