Survey on Code Identifier Readability

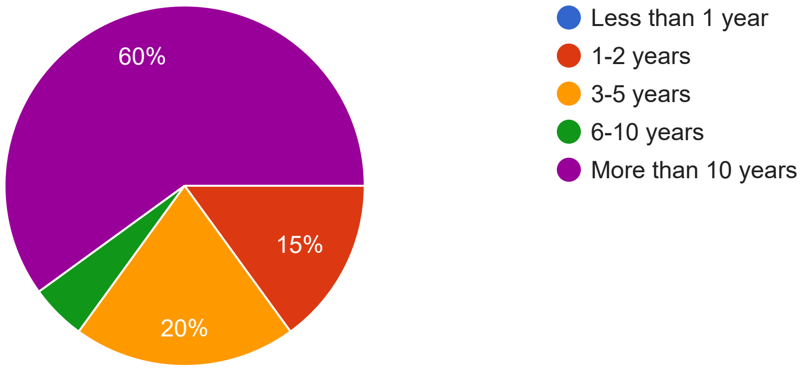
20 responses

Publish analytics

How many years of programming experience do you have?

Copy

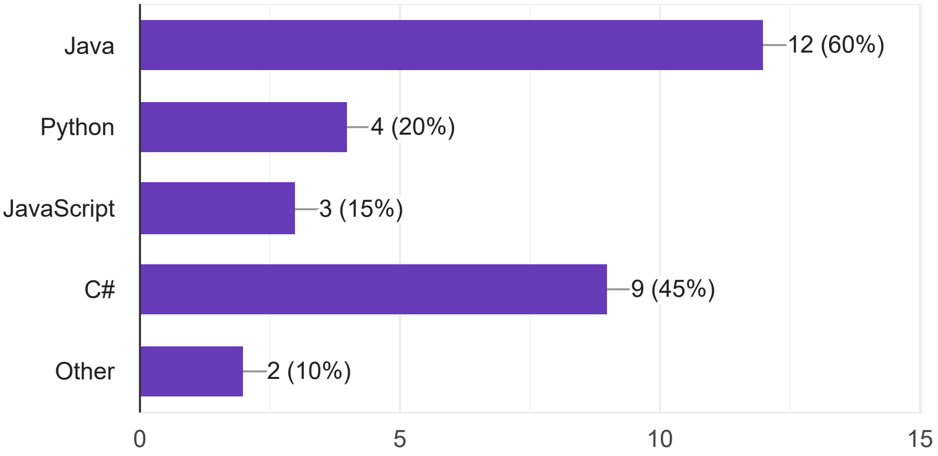
20 responses



What is your primary programming language?

Copy

20 responses

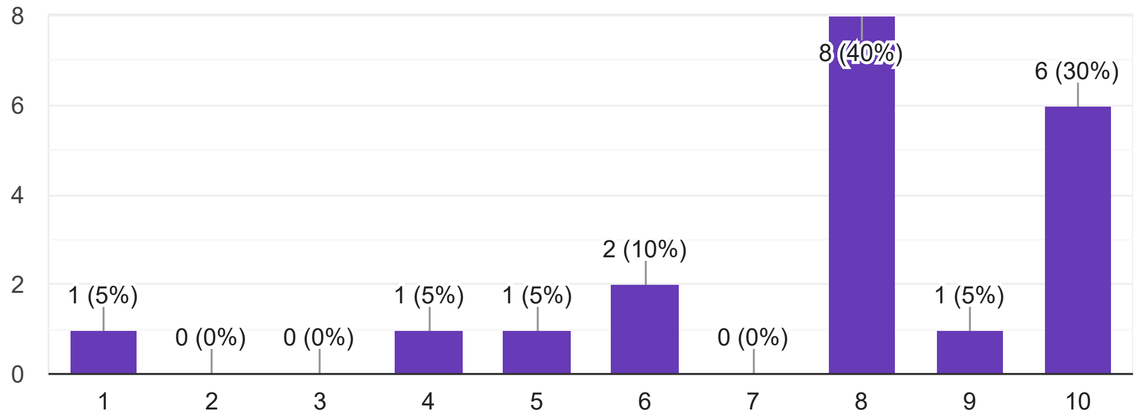


You will be asked to rate each identifier on a scale of 0–10 based on how readable you find it

Copy

output_dir

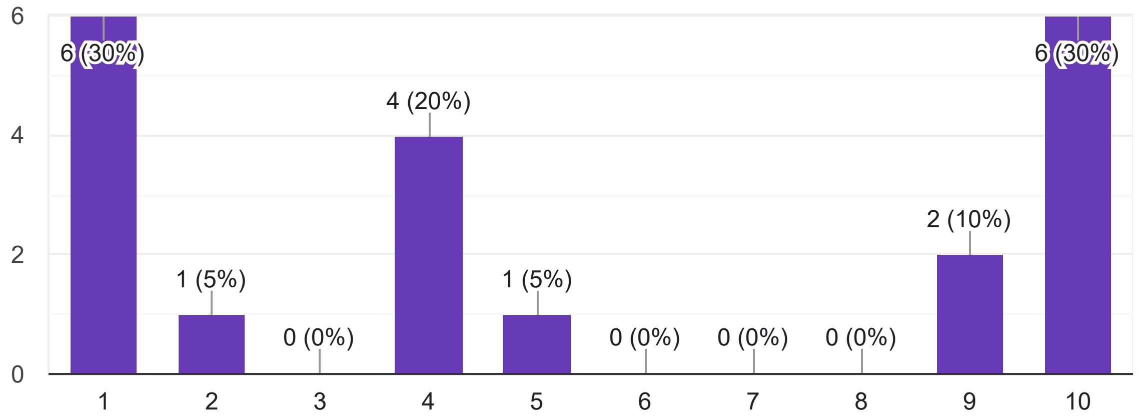
20 responses



Copy

doStuff()

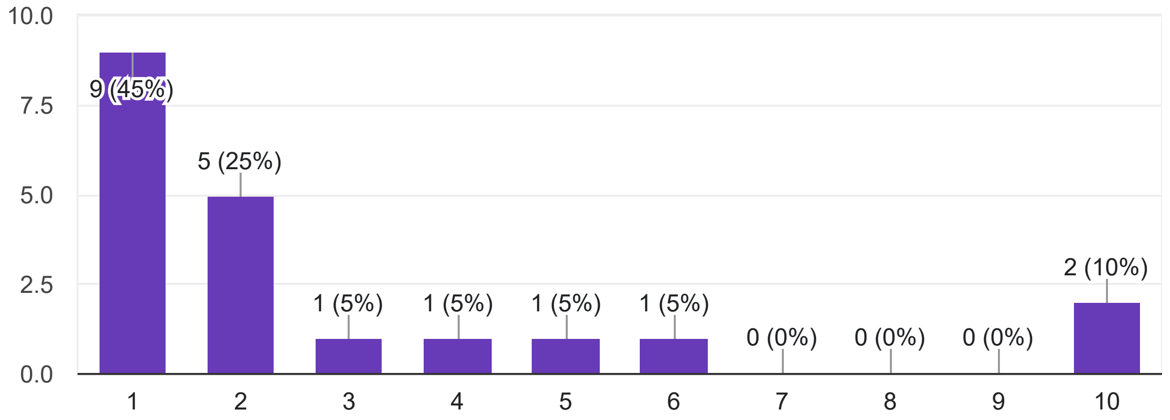
20 responses



Copy

x

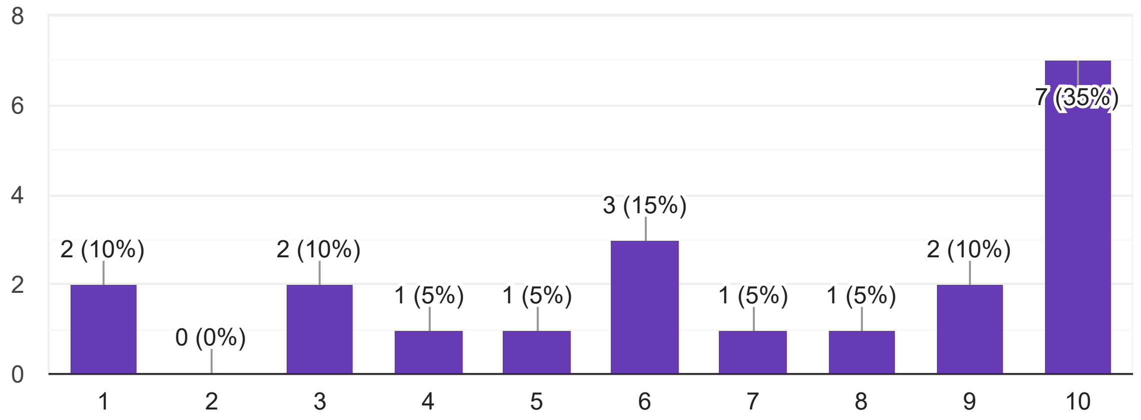
20 responses



Copy

VersionManager

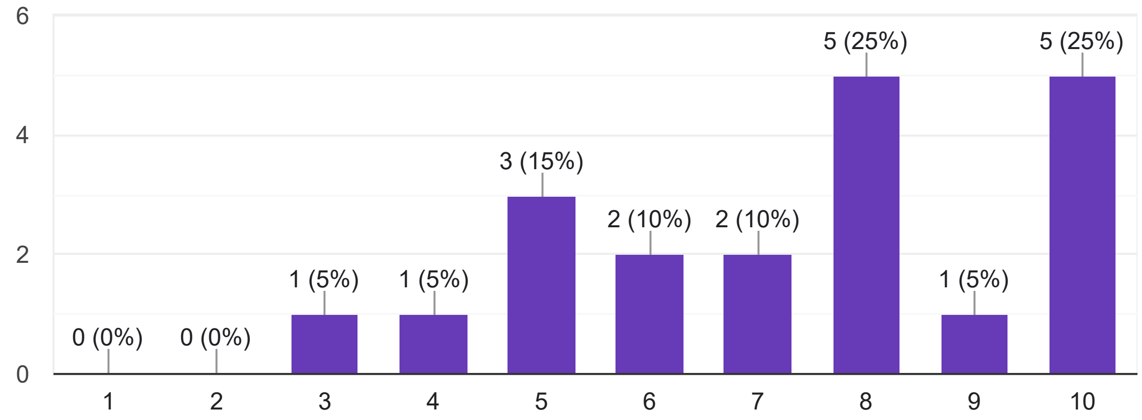
20 responses



Copy

render_templates()

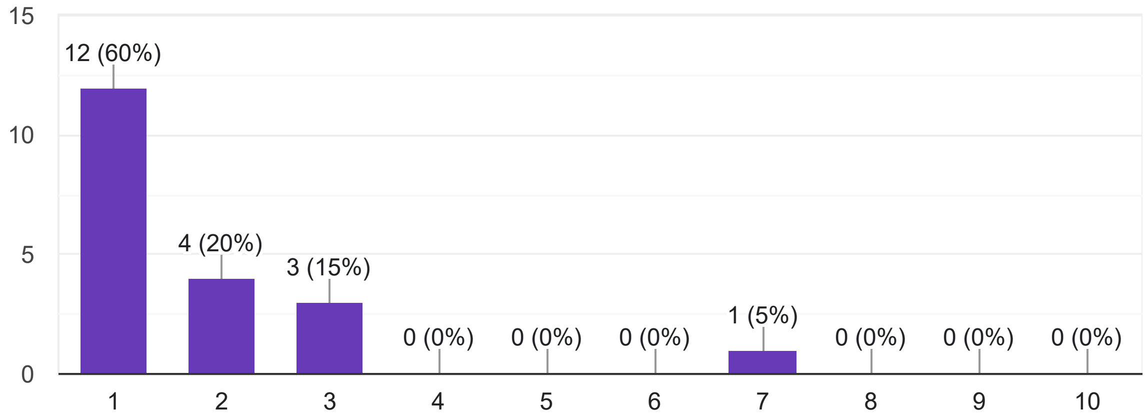
20 responses



Copy

_

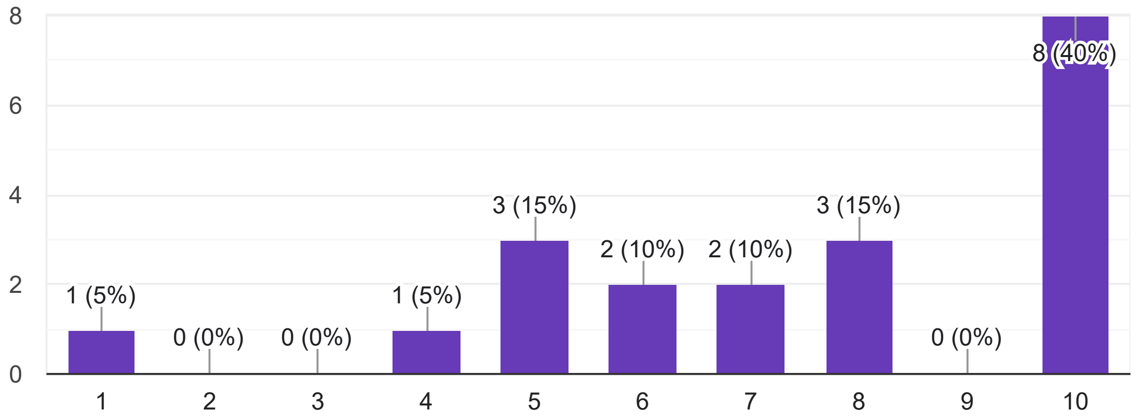
20 responses



Copy

validate_user_input()

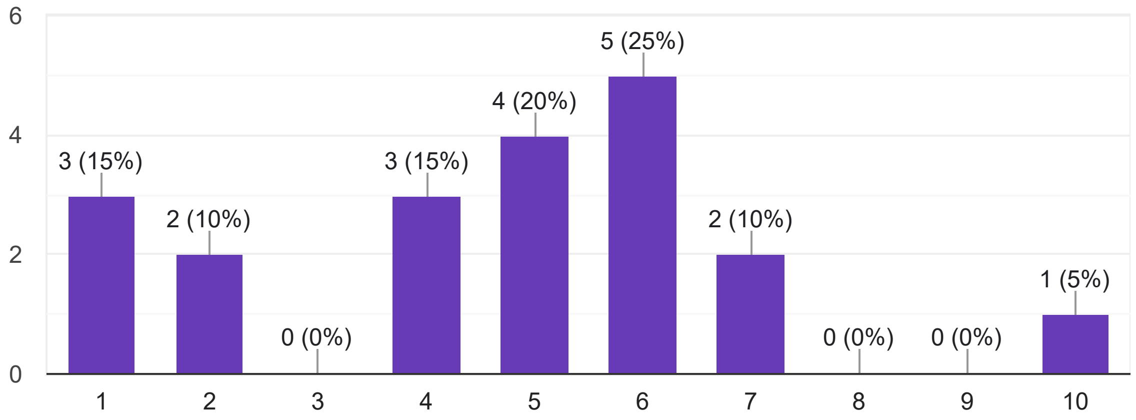
20 responses



Copy

DummyBuildSrc

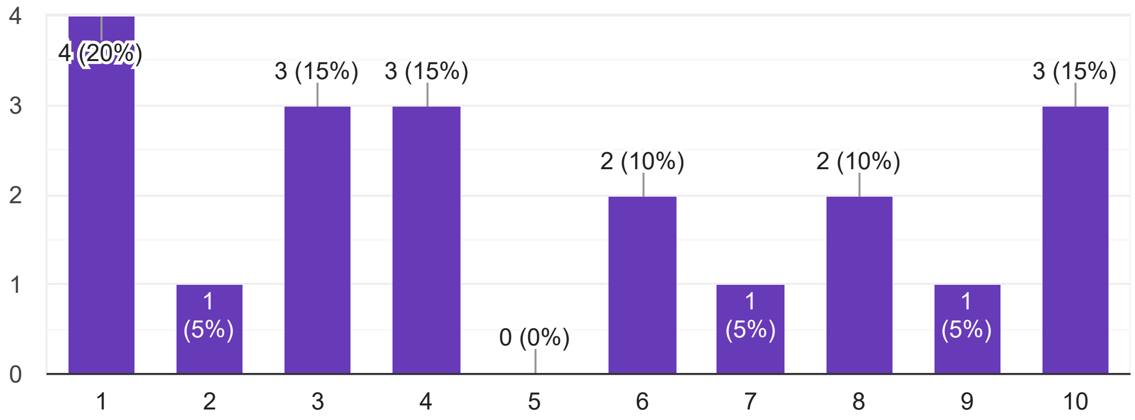
20 responses



Copy

tmp

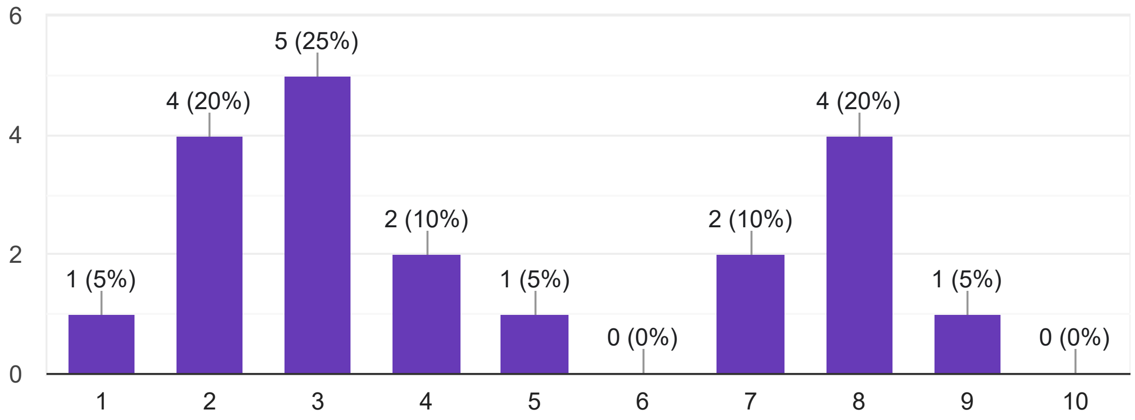
20 responses



Copy

maybe_cythonize()

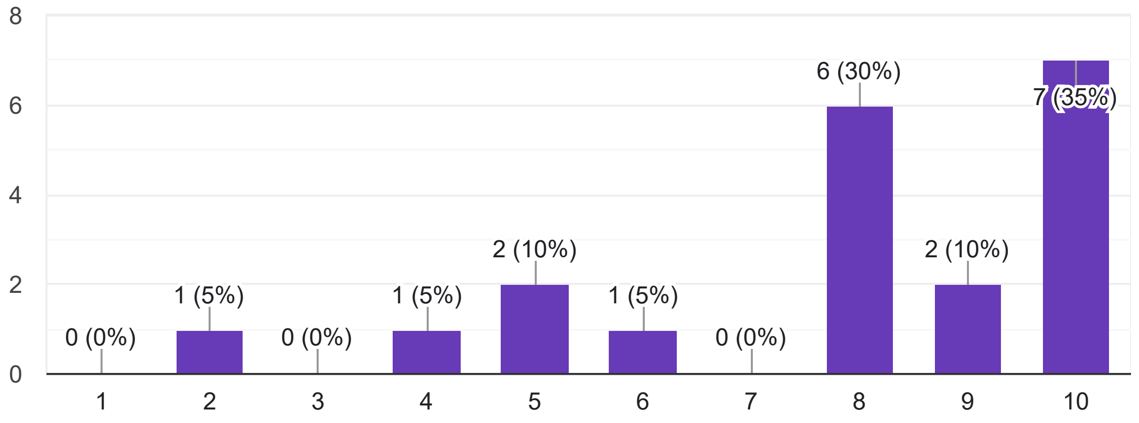
20 responses



Copy

is_debug_mode

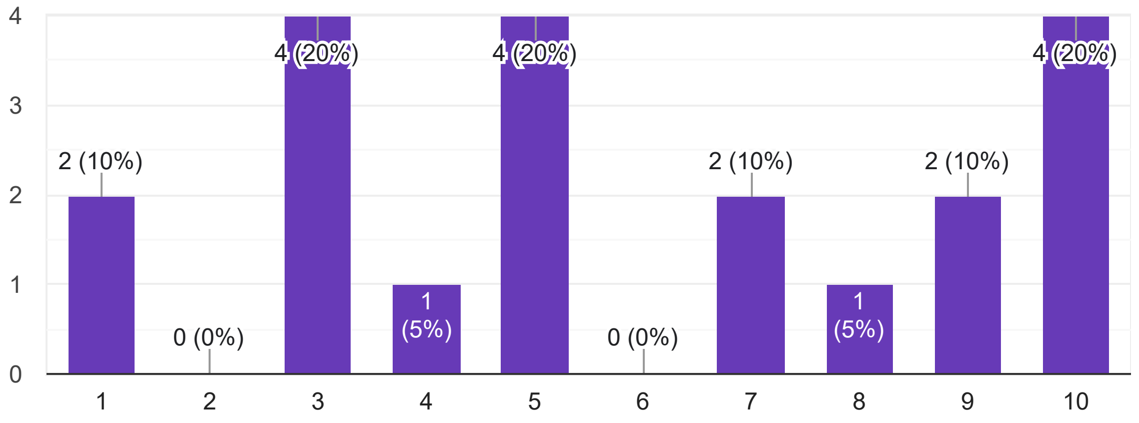
20 responses



Copy

Factorize

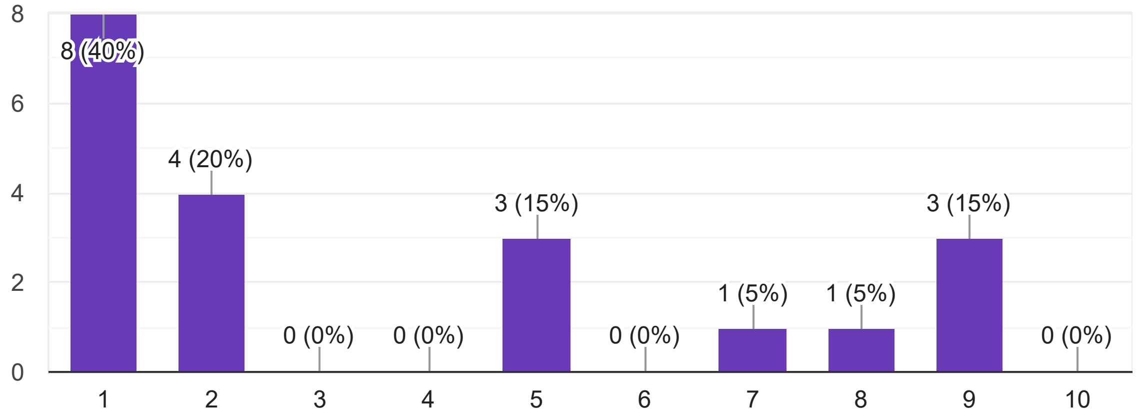
20 responses



Copy

i, j, k

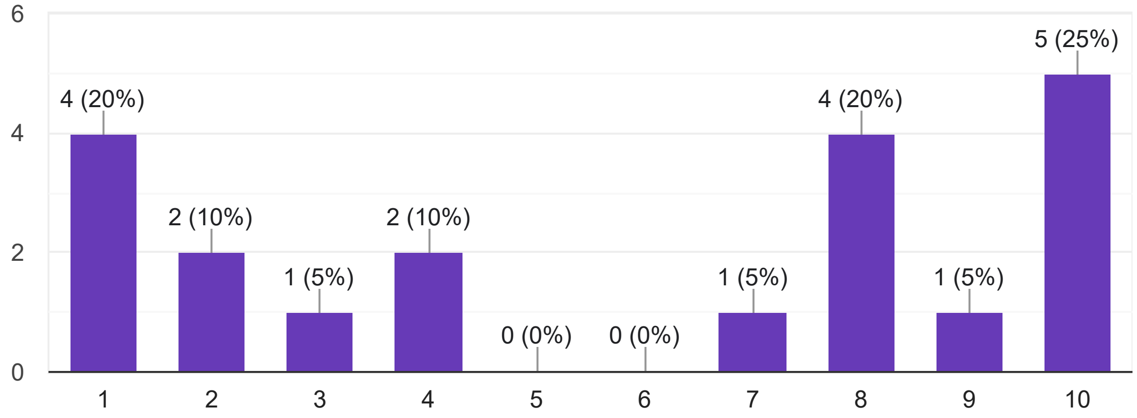
20 responses



Copy

get()

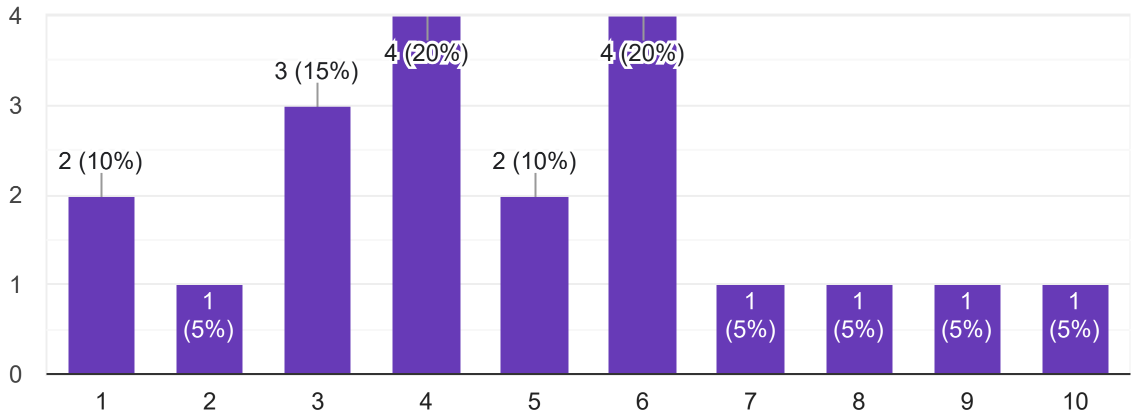
20 responses



Copy

Hashing

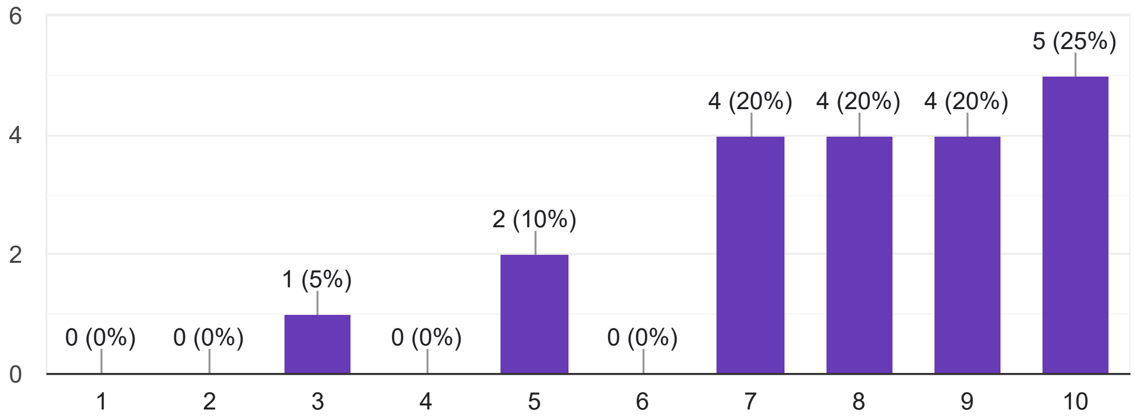
20 responses



Copy

write_version_info()

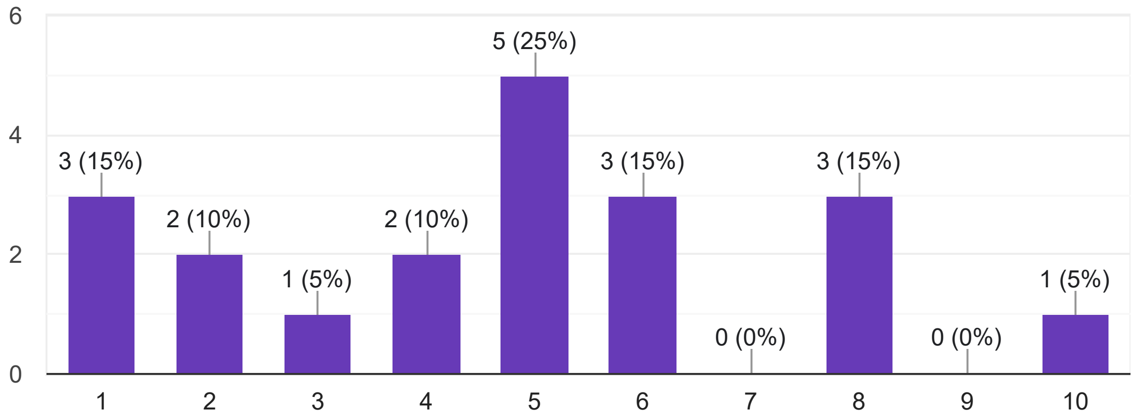
20 responses



Copy

process_tempita()

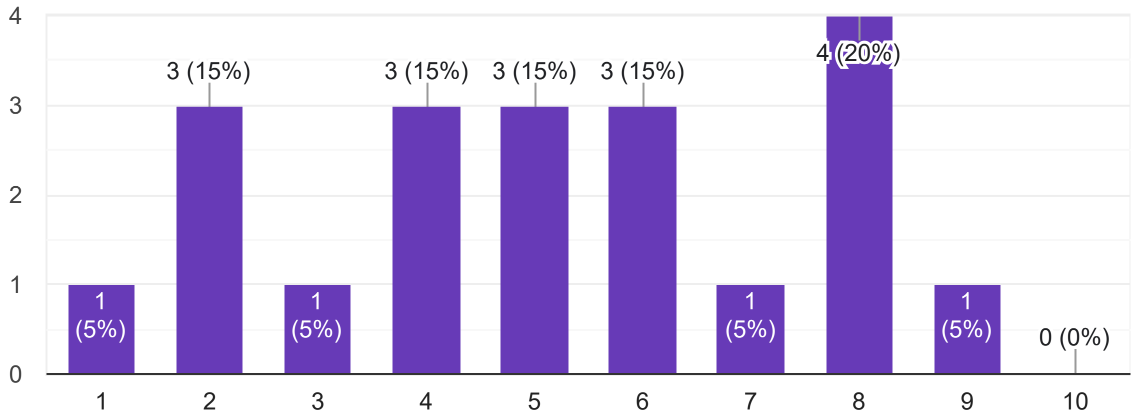
20 responses



Copy

ext_data

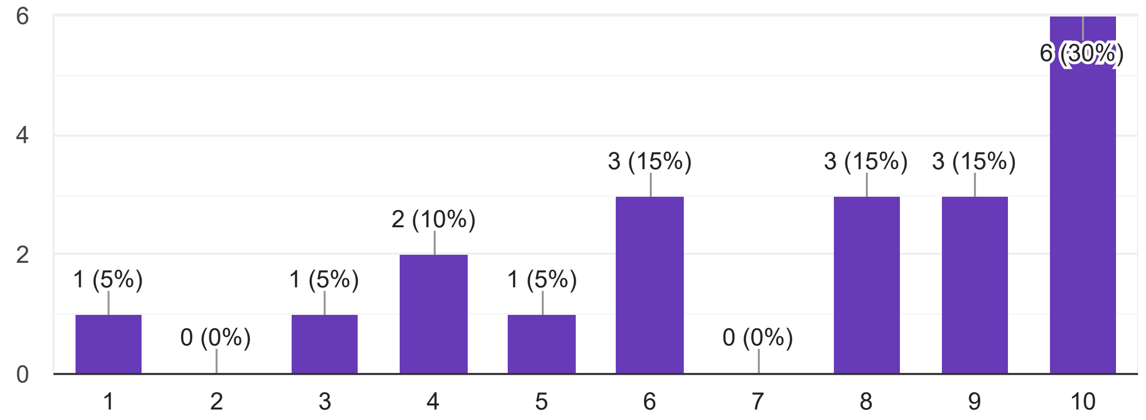
20 responses



Copy

run()

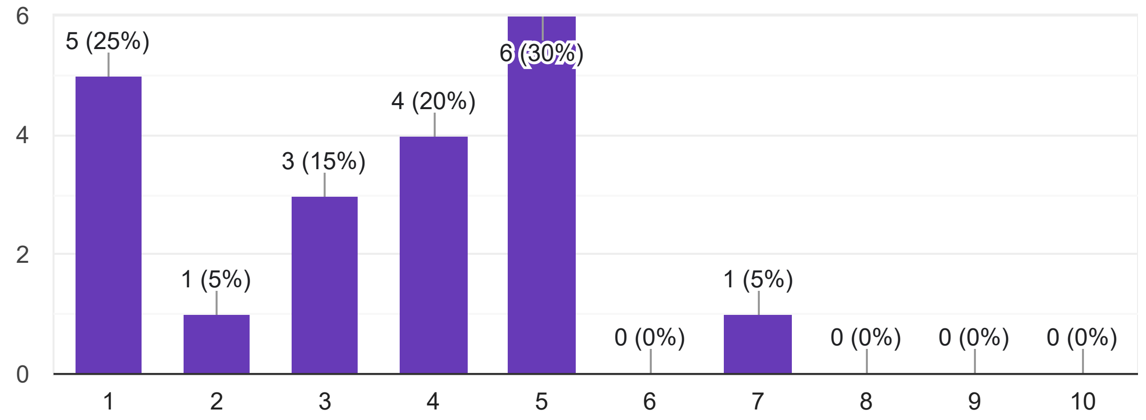
20 responses



Copy

pxifile

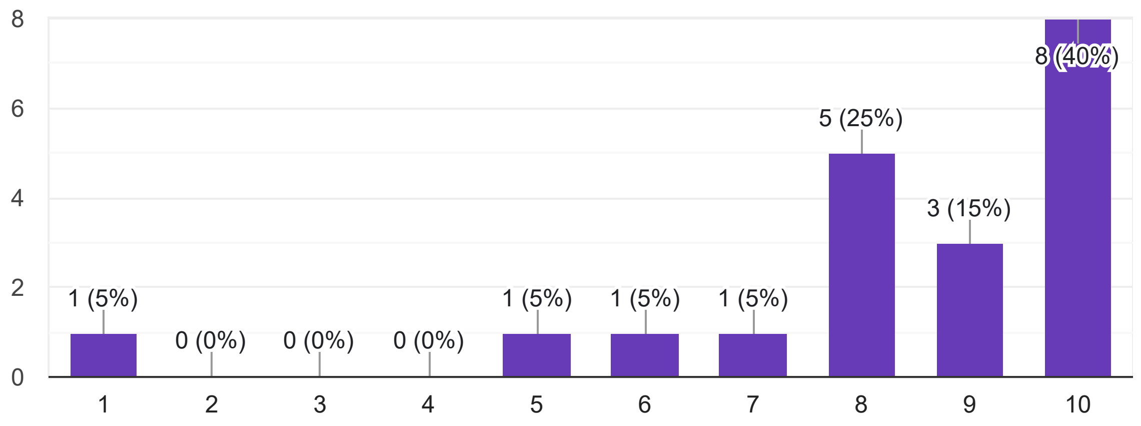
20 responses



Copy

InputValidator

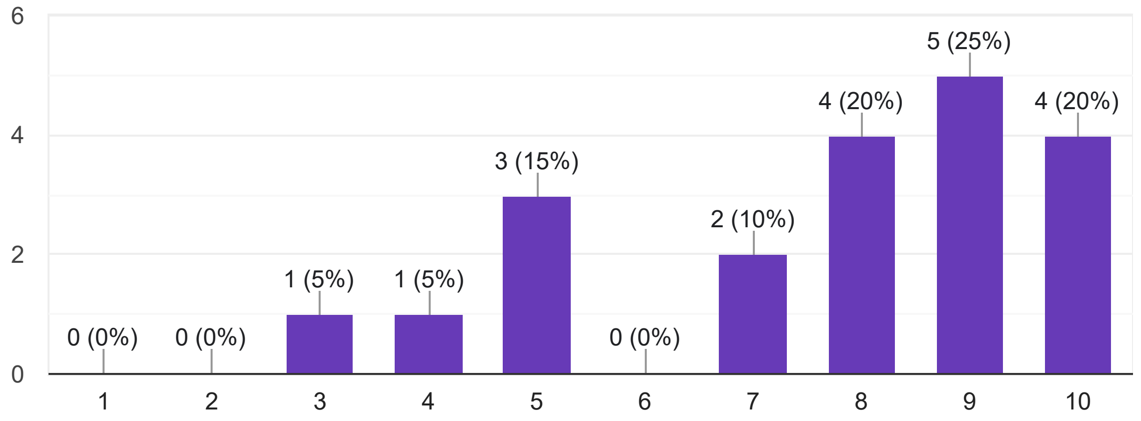
20 responses



Copy

testIsInitialized()

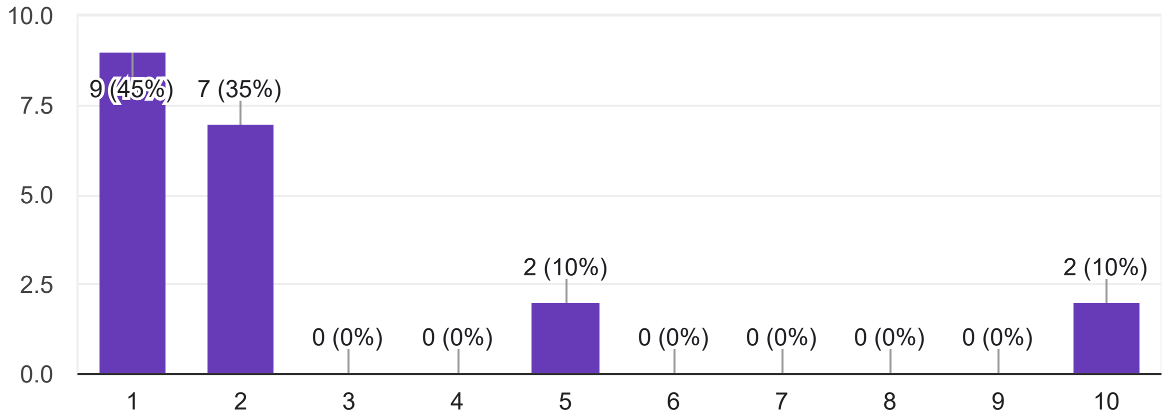
20 responses



Copy

d

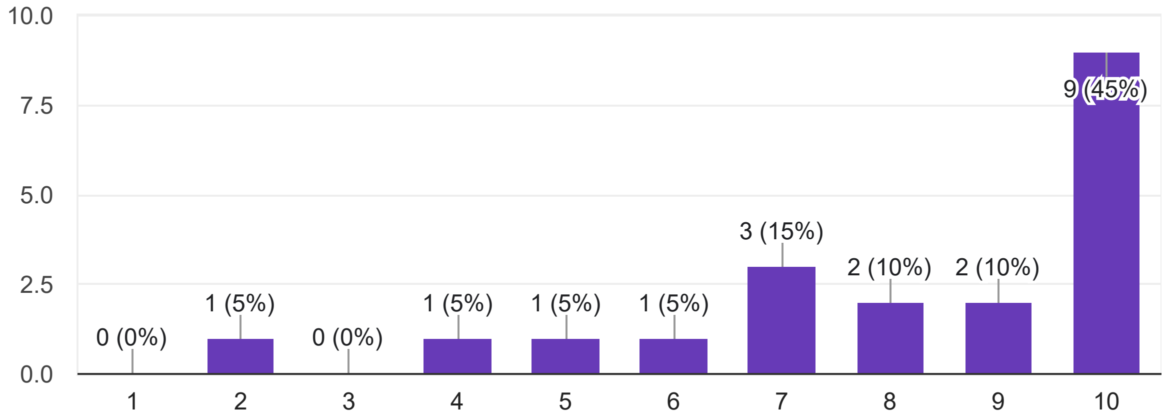
20 responses



Copy

calculate_tax()

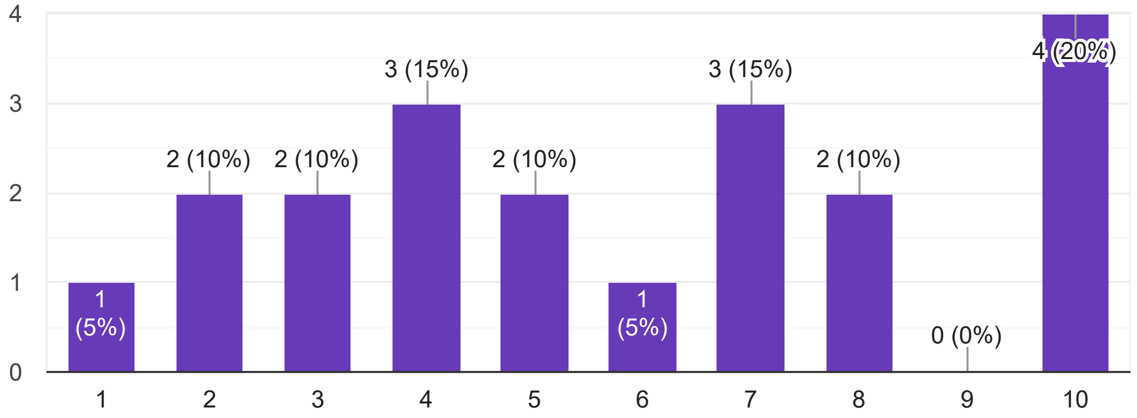
20 responses



Copy

LazyInitializerTestImpl

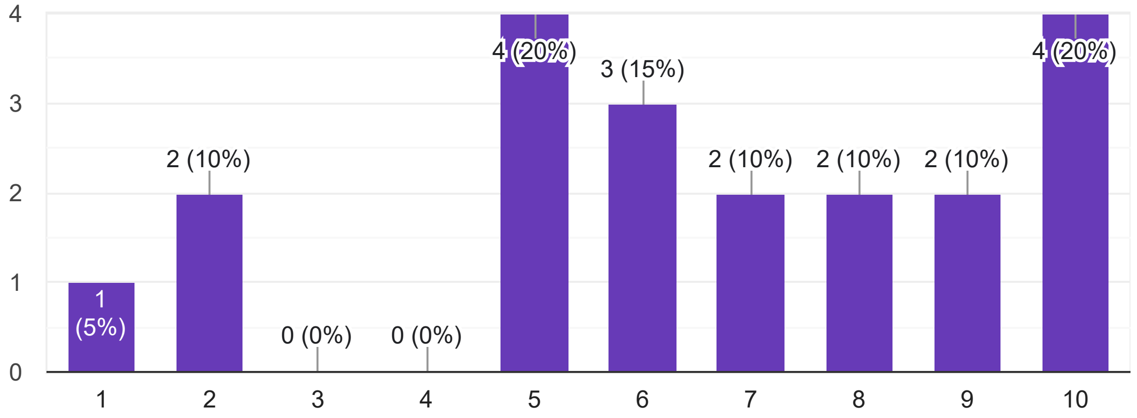
20 responses



Copy

clean_me()

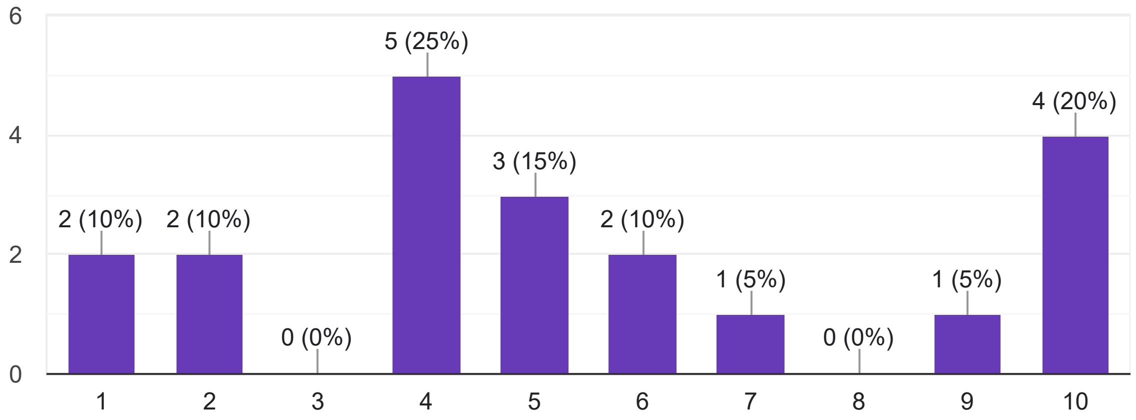
20 responses



Copy

macros

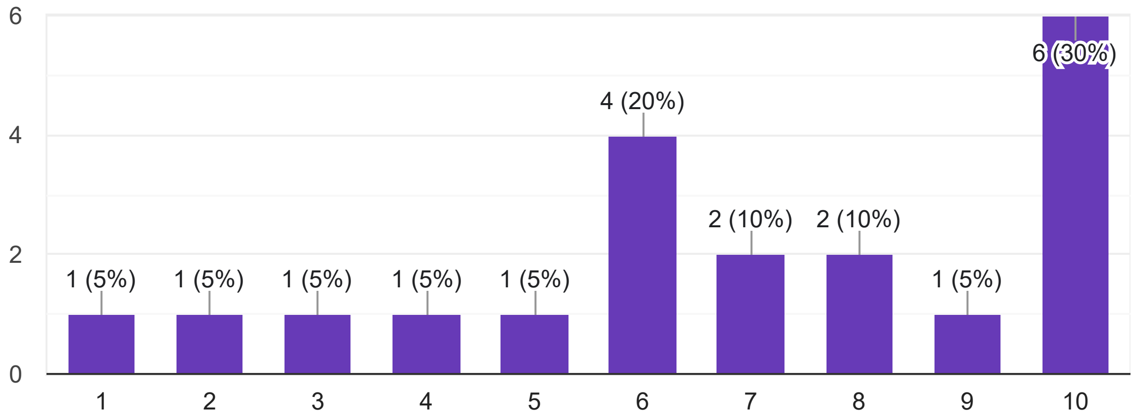
20 responses



Copy

parse()

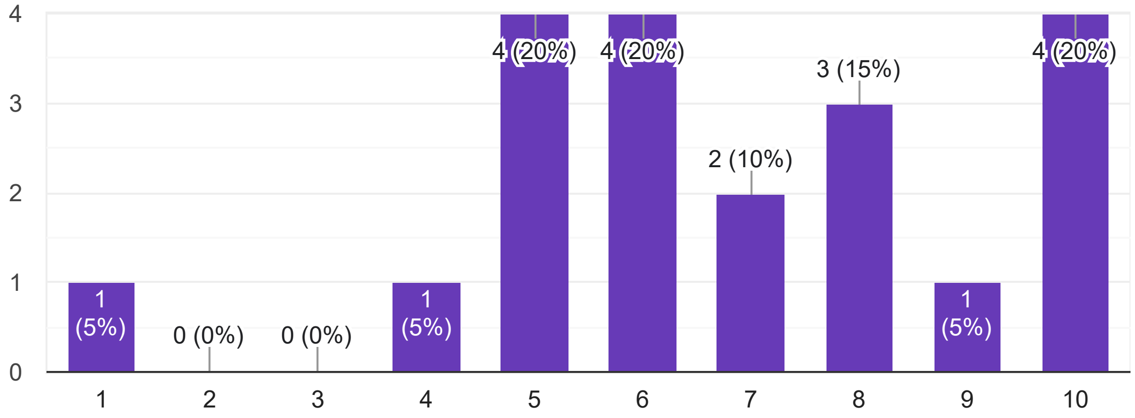
20 responses



Copy

MemoizerComputableTest

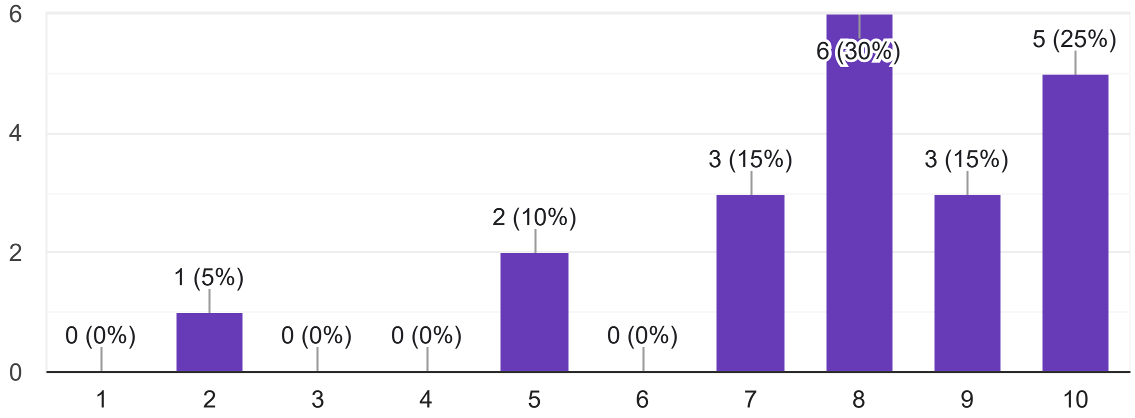
20 responses



Copy

setup_cache()

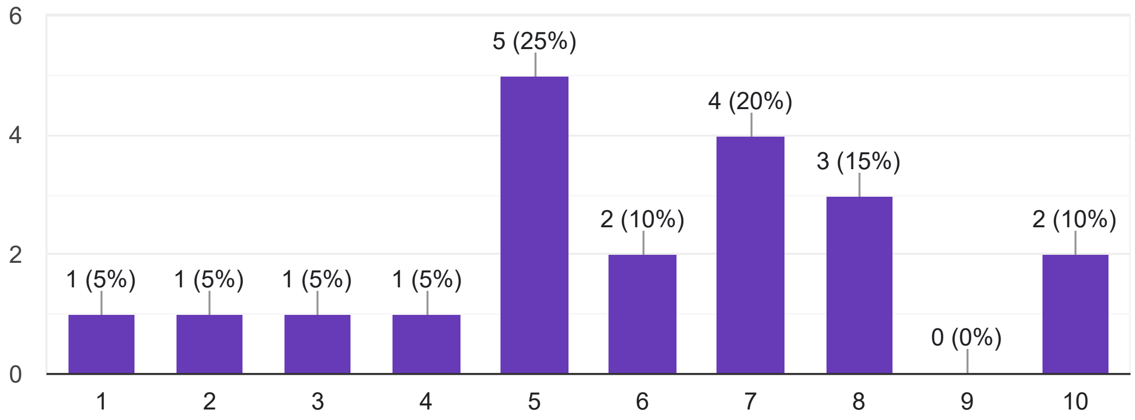
20 responses



Copy

linetrace()

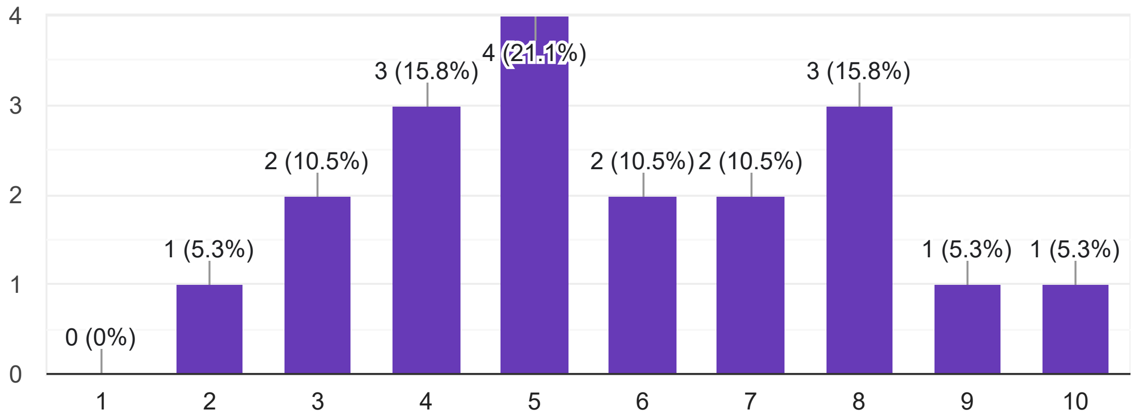
20 responses



Copy

parsed

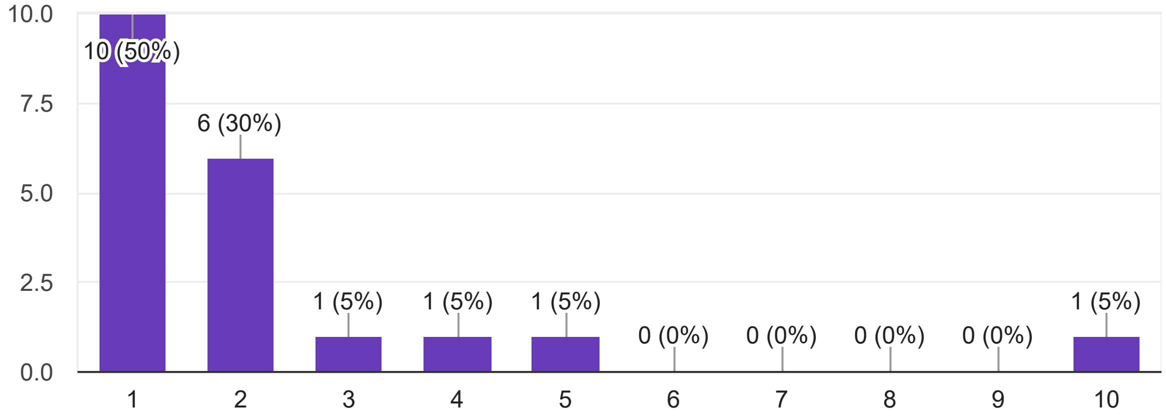
19 responses



Copy

f()

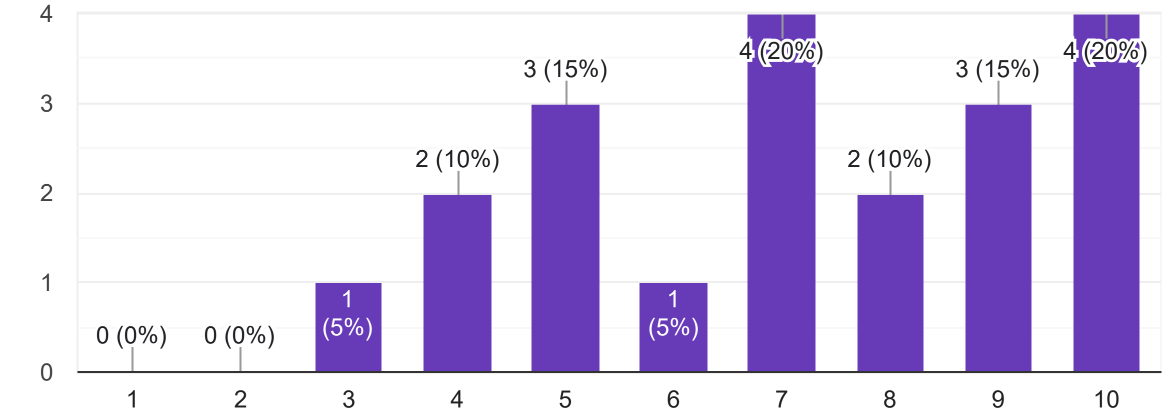
20 responses



Copy

user_data

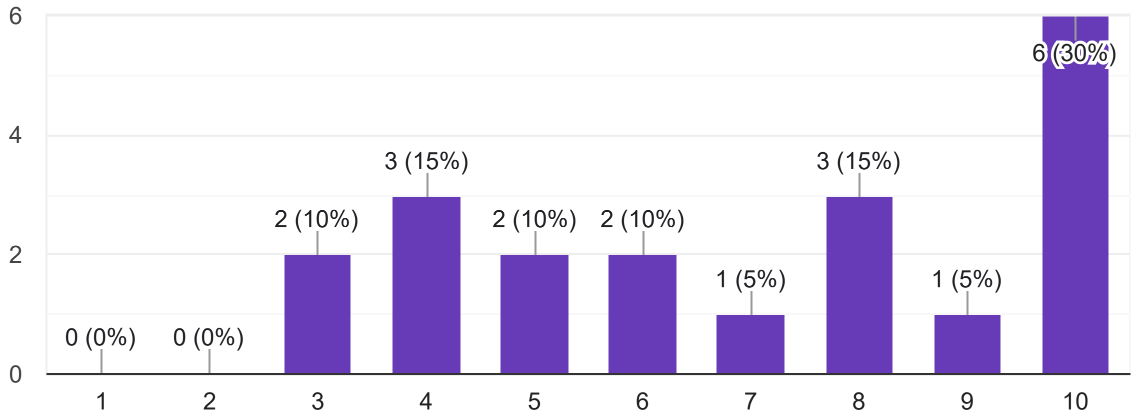
20 responses



Copy

CleanCommand

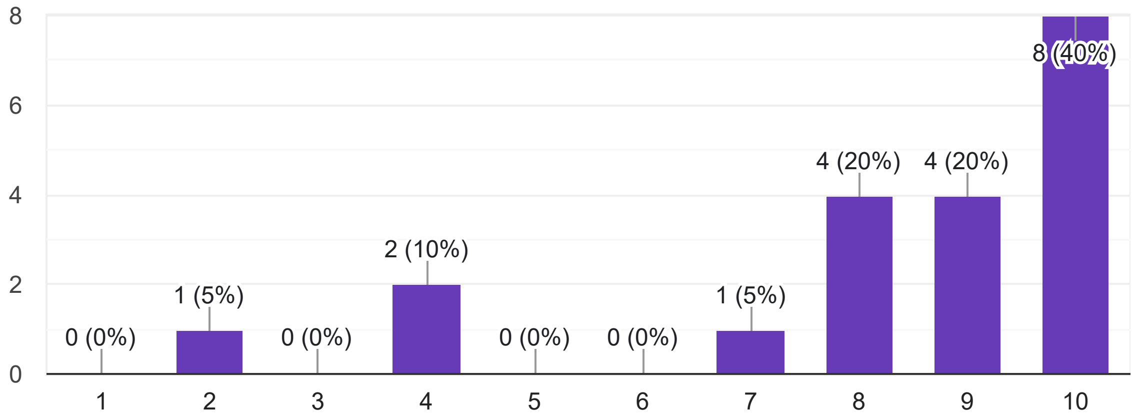
20 responses



Copy

initialize()

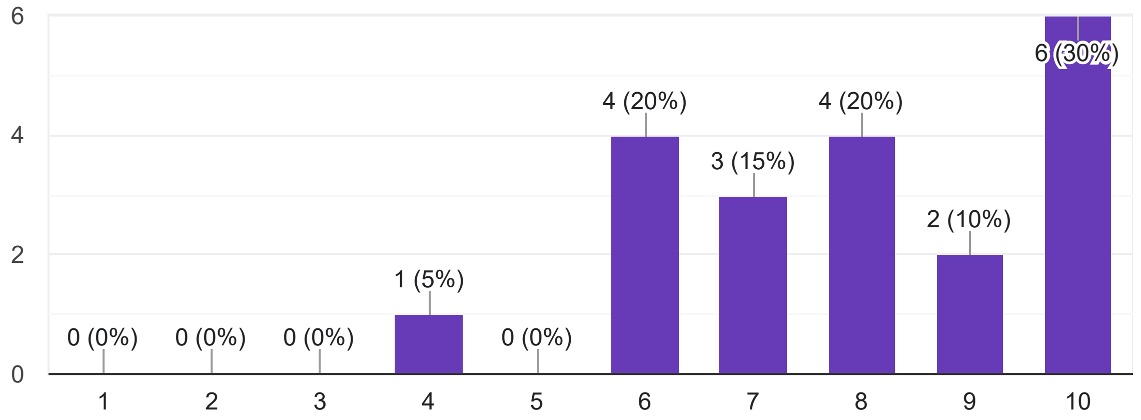
20 responses



Copy

version_info

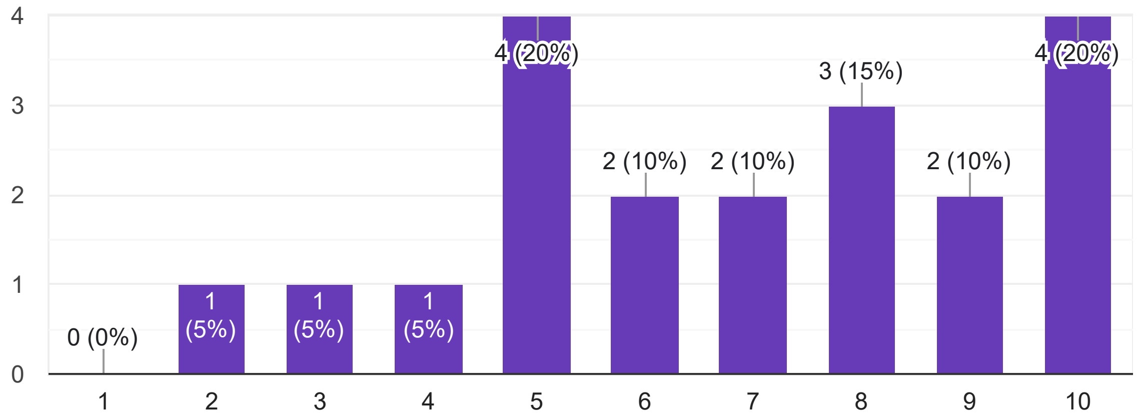
20 responses



Copy

db_query()

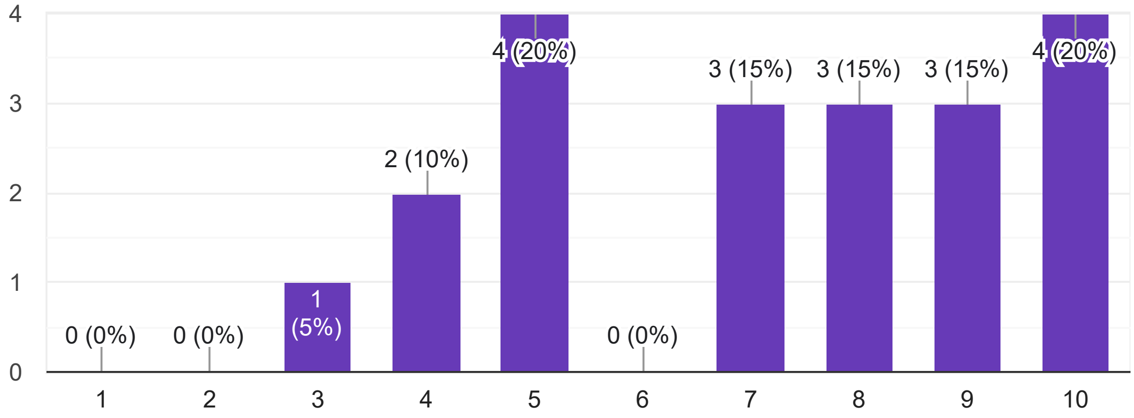
20 responses



Copy

ConfigParser

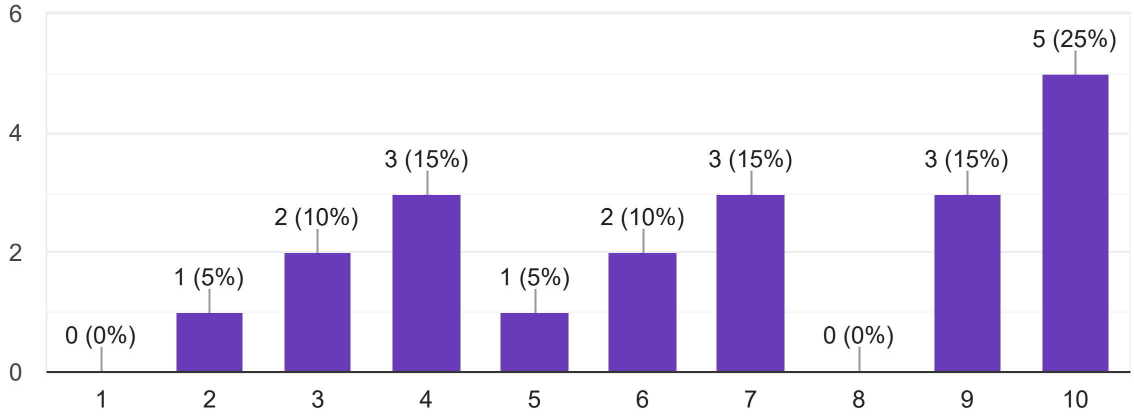
20 responses



Copy

handle()

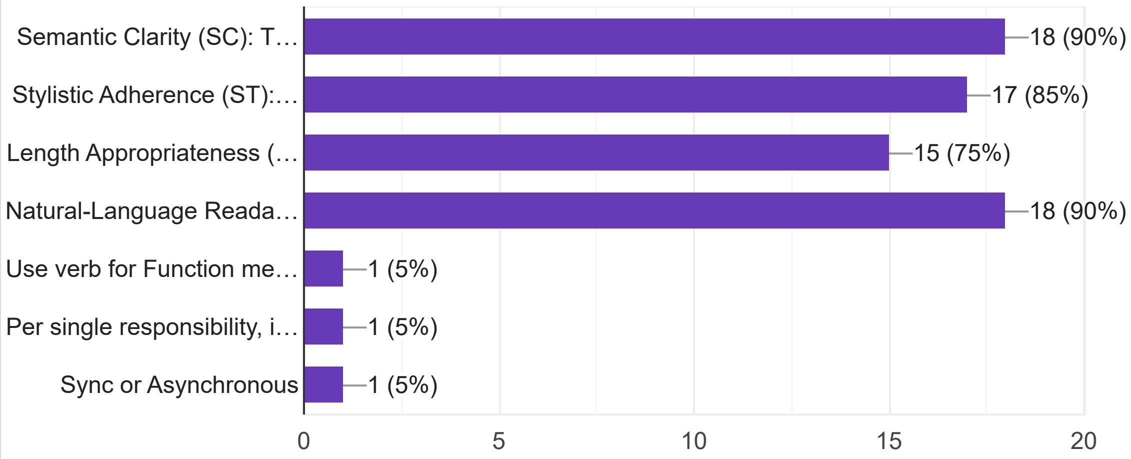
20 responses



Copy

Which of the following factors do you think contribute to identifier readability? (Select all that apply)

20 responses



Beyond the options above, what other factors influence how readable you find an identifier? (e.g., consistency, domain-specific terms, etc.)

7 responses

Intent of the variable or function in a context inportant

1) Consistency in naming throughout the codebase.
2) Use of domain specific names.
3) avoid unnecessary abbreviation while naming.
4) avoid same name for different type (avoid confusion).
5) proper use of prefix and suffix while naming.

Consistency:
It's crucial to use identifiers consistently throughout the codebase. If getUserData() is used in one place, don't use fetchUserInfo() elsewhere for the same purpose.
Consistency in naming style (e.g., always using camelCase for methods) is vital.
Context:
The readability of an identifier can depend on its context. For example, i might be perfectly acceptable as a loop counter, but it's poor for a variable representing an index in a more complex algorithm.
Domain-Specific Terms:
Using terms that are common and well-understood within the specific domain of the software can greatly enhance readability for developers working on that project. However, it can hinder readability for those unfamiliar with the domain.
Abbreviations:
While some abbreviations are widely accepted (e.g., id for identifier, btn for button), excessive or obscure abbreviations can significantly reduce readability. It's important to use abbreviations sparingly and ensure they are unambiguous.
Pronounceability:
Identifiers that are easy to pronounce are often easier to remember and discuss, which can indirectly contribute to readability.
Clarity vs. Brevity:
There's often a trade-off between clarity and brevity. While shorter identifiers can be quicker to type, longer, more descriptive identifiers are generally more readable. It's important to strike a balance between the two.
Use of Acronyms:
Similar to abbreviations, acronyms should be used with caution. Common acronyms like HTTP or URL are fine, but less common ones should be avoided or spelled out for clarity.
Avoiding Negation in Boolean Names:
It's generally better to use positive boolean names (e.g., isActive, isEnabled) rather than negative ones (e.g., isNotActive, isDisabled). This makes the logic easier to follow.
Clarity of Purpose:
The identifier should clearly indicate the purpose of the variable, function, class, etc. This helps

developers understand how the code works and reduces the need to refer to documentation or other code.

We should not sacrifice clarity for brevity at the same time identifier names should take an entire line (screen width) :)

Domain language always matters and it definitely makes the code more readable

Consistency throughout the classes , functional naming conventions

Domain specific terms, consistency