Google STEP Interviews: What to Do, What Not to Do

07 December 2021

Before We Start…

The observations and opinions expressed here are mine alone, and most certainly not those of Google, which does not support or approve this talk in any way whatsoever.

Everything that you are about to hear may be wrong.

I will also probably contradict myself in multiple ways.

Do: Prepare

See the “things-to-study” section in the class STEP discord.

Every one of you has the potential to **completely crush** these interviews if you are prepared!

Don’t: Worry (Do: Be happy)

A good interviewer will always have more work for you to do than you will ever be able to finish in 45 minutes.

Relax and enjoy the ride. When you find yourself completely stumped, laugh and have fun with it.

Do: Remember that your interviewer also wants to do well

Your interviewer is there to make observations about you which can be **supported by evidence**.

Do: Ask questions

Ask (either yourself or your interviewer) about your inputs:

- Integers vs Floats
- Positive vs Negative
- Zero / Empty
- Duplicate Values
- Sorted vs Unsorted
- Stupidly Large Numeric Values
- Stupidly Large Containers
- Container Types

Don’t: Worry about looking stupid

Ask about anything you don’t completely understand.

If you still don’t understand, ask **again**.

Do: Trust but verify

It is probably safe to assume that your interviewer mostly knows what they are talking about.

However, everyone makes mistakes.

If you think your interviewer has said something incorrect, this could be an opportunity! (But proceed with caution.)

Don’t: Ask for permission

“Should I implement this with a dictionary?”

“Should I use a helper function?”

“Should I check for bad input?”

“Am I allowed to import this module which completely trivializes the problem?”

Don’t ask. Just do. If your interviewer wants you do to something else, they will tell you.

Do: Always Be Coding

Ultimately the single strongest signal is the code that you write, so write a lot of it.

Quantity is arguably more important than quality (as long as the quality is there eventually).

You **will not** be able to simply talk your way through your interview.

Don’t: Write pseudocode or comments

You are being evaluated on your **code**.

Don’t waste your precious time writing not-code.

Do: Summon useful functions into existence

When breaking down a problem, try to do it in terms of functions:

- Declare that there exists a function which does something you need.
- Specify its behavior precisely and then just use it.
- If time allows, implement it later.

Don’t: Destroy your input arguments

Always write non-destructively unless specifically instructed otherwise.

Do: Use common industry naming conventions

CONSTANT_NAME

variable_name

FunctionName()

Do: Demonstrate that you know your language

Write code that is free of syntax errors.

Write as legibly as you possibly can.

Capitalize True/true and False/false correctly for your language.

Use +=

(Python) Use for-loops wherever appropriate.

(Python) Generators and list comprehensions make good impressions.

(Python) Use proper indentation.

Do: Pretend to sanitize your inputs

```
def CheckData(arg1, arg2):
    pass


def Foo(arg1, arg2):
    error = CheckData(arg1, arg2):
    if error != None:
        ScreamAndDie(error, arg1, arg2)
```

Do: Use nice short variable names

for val in vals:

vs

for element in container:

Don’t: Destroy your work

After you finish a solution, always keep a backup copy of it somewhere. If you are asked to make modifications, make those changes to a **copy** so that you can instantly revert back to a known good state whenever needed.

This is especially important when interviewing virtually.

Do: Simplify your conditionals

```
if a > b:
	return a
return b
```

vs

```
if a > b:
	return a
else:
	return b
```

```
if a:
	return 3
```

vs

```
if a == True:
	return 3
```

```
if condition:
	foo()
else:
	bar()
```

vs

```
if not condition:
	bar()
else:
	foo()
```

```
return condition
```

vs

```
if condition:
	return True
else:
	return False
```

Don’t: Treat ‘return’ like a function

return value

vs

return(value)

(Python): The same goes for ‘while’ and ‘if’ and ‘elif’

Do: Pay attention to hints

When your interviewer gives you a hint:

- Don’t panic. This is an **opportunity**.
- Stop whatever you are doing. Immediately.
- Do not proceed until you understand 100% of whatever was just said.

Interlude - Finished!

Do: Test your code

Say it along with me:

> Testing your code is never optional.
> Testing your code is never optional.
> Testing your code is **never** optional.

Do: Choose useful test inputs

Things to test (as appropriate):

- Containers of length 0
- Containers of length 1
- The first element of a container
- The last element of a container
- Sorted and unsorted data
- Zero
- Negative numbers

Do: Actually test it (Dont: Just go through the motions)

- Don’t be fooled by how you **think** your code **should** work.
- Clear your mind, focus on one statement at a time.
- **Write down** what happens to your variables.

Do: Be prepared to do it all over again (Don’t: Despair)

- The interview process is specifically designed to favor false negatives over false positives.
- You are welcome to try again next year at the next level.
- This happens a lot.