




Python Debugging Checklist

Professional Reference Guide for Code Quality & Problem Resolution

 **CRITICAL FIRST STEPS:** Always read error messages completely, check recent changes, and test one fix at a time.

1. Error Identification & Classification

☐ Syntax Errors (Code won't run)

- ☐ Missing colons after if/for/def statements
- ☐ Incorrect indentation (use 4 spaces consistently)
- ☐ Typos in keywords (def, if, for, while, etc.)
- ☐ Unmatched parentheses (), brackets [], or braces {}

☐ Runtime Errors (Code crashes during execution)

- ☐ TypeError: Wrong data type used in operation
- ☐ ValueError: Correct type but invalid value
- ☐ IndexError: List/string index out of range
- ☐ KeyError: Dictionary key doesn't exist
- ☐ NameError: Variable not defined or misspelled

☐ Logic Errors (Wrong results)

- ☐ Function returns unexpected values
- ☐ Infinite loops or early termination
- ☐ Incorrect conditional statements

☐ Import Errors

- ☐ Missing modules or incorrect import statements
- ☐ File path issues or module not in Python path

2. Input Validation Issues

☐ Data Type Validation

Check if inputs match expected data types

```
isinstance(value, (int, float)) # Check for numbers
```

☐ Numeric Value Validation

Verify numeric values are positive when required

☐ String Validation

Validate string inputs aren't empty or None

```
if not string.strip(): raise ValueError("String cannot be empty")
```

☐ Boolean Logic Validation

Ensure boolean conditions evaluate correctly

3. Function-Specific Debugging

☐ Single Responsibility Check

Does each function do one thing well?

☐ Parameter Validation

Are all required parameters provided and correct types?

☐ Return Value Verification

Does the function return what the docstring promises?

☐ Edge Cases Handling

Empty inputs, zero values, boundary conditions

4. Professional Code Standards

☐ Naming Conventions

- ☐ Variables use snake_case
- ☐ Functions are descriptive verbs

☐ Documentation Standards

- ☐ Docstrings with Args, Returns, Raises

☐ Type Hints

Parameter and return types specified

☐ Error Messages

Informative and user-friendly

5. Common Python Pitfalls

☐ Indentation Issues

Consistent spacing (4 spaces vs tabs)

☐ Variable Scope Problems

Variables accessible where you're using them?

☐ Data Type Mismatches

String vs numeric operations

```
# Avoid: "5" + 10 # Use: int("5") + 10
```

☐ Mutable Default Arguments

Don't use lists/dicts as default parameters

```
# Bad: def func(items=[]): # Good: def func(items=None):
```

6. Testing & Validation Steps

Testing Order: Valid inputs → Edge cases → Invalid inputs → Error handling

☐ Test with valid inputs first

Ensure basic functionality works

☐ Test edge cases

Empty strings, zero, negative numbers

☐ Test invalid inputs

Verify error handling works properly

☐ Use debugging techniques

Print statements or debugger to trace execution

7. Professional Debugging Tools

☐ VS Code Debugger

Setup breakpoints and watch variables

☐ Print Statement Debugging

```
print(f"Debug: {variable_name} = {variable_value}")
```

☐ Exception Handling

Try-except blocks for graceful error management

☐ Production Logging

Use logging module for production-level debugging