Python Debugging Checklist for Product Managers

When Your Code Doesn't Work: A Step-by-Step Debugging Guide

Step 1: Don't Panic! 🧘

- **Take a deep breath** - Every programmer deals with bugs daily
- **Remember**: The computer is doing exactly what you told it to do, just not what you *meant* to tell it
- **Read the error message** - It's trying to help you, not confuse you

---

Phase 1: Quick Visual Check (30 seconds)

✅ Syntax Basics Checklist

- [ ] **Indentation**: Are all lines inside functions/if statements/loops indented consistently?
- [ ] **Parentheses**: Do all opening `(` have matching closing `)`?
- [ ] **Quotes**: Do all opening `"` or `'` have matching closing quotes?
- [ ] **Colons**: Does every `if`, `for`, `while`, and function definition end with `:`?
- [ ] **Variable names**: Are they spelled consistently throughout your code?

Quick Fix Examples:

```python
# WRONG: Missing colon
if user_count > 1000
    print("High volume")

# RIGHT: Added colon
if user_count > 1000:
    print("High volume")
```

Phase 2: Read the Error Message (1 minute)

📖 Error Message Decoder

Step 1: Find the actual error

- Look for the **last line** of the error message - that's usually the real problem
- Ignore the long traceback for now

Step 2: Identify the error type

If you see...	It probably means...
`NameError`	You used a variable that doesn't exist
`TypeError`	You mixed wrong data types (like adding text + number)
`IndentationError`	Your spacing is wrong
`SyntaxError`	You have a typo or missing punctuation
`IndexError`	You tried to access a list position that doesn't exist
`KeyError`	You looked for a dictionary key that doesn't exist

Step 3: Find the line number

- Look for `line XX` in the error message
- Go to that exact line in your code
- The problem is usually on that line or the line right before it

Example Error Analysis:

```
Traceback (most recent call last):
  File "analytics.py", line 15, in <module>
    total_users = active_users + new_users
NameError: name 'new_users' is not defined
```

Translation: "On line 15, you tried to use a variable called 'new_users' but you never created it."

Phase 3: Investigate the Problem (2-3 minutes)

🔍 Detective Work Checklist

For Variable Errors:

- [ ] **Check spelling**: Is `user_count` spelled the same everywhere?
- [ ] **Check order**: Did you define the variable before using it?
- [ ] **Check scope**: If it's inside a function, did you define it inside that function?

For Data Type Errors:

- [ ] **Print the variables**: Add `print(variable_name, type(variable_name))`
- [ ] **Check input sources**: Did `input()` give you text when you needed a number?
- [ ] **Check calculations**: Are you mixing strings and numbers?

For List/Dictionary Errors:

- [ ] **Print the data**: Add `print(my_list)` or `print(my_dict)`
- [ ] **Check length**: Add `print(len(my_list))`
- [ ] **Check keys**: Add `print(my_dict.keys())`

🕵️ Debugging Print Statements

Add these diagnostic prints to understand what's happening:

```python
# Check what your variables contain
print(f"user_count = {user_count}, type = {type(user_count)}")

# Check list contents and length
print(f"my_list = {my_list}")
print(f"list length = {len(my_list)}")

# Check dictionary contents
print(f"my_dict = {my_dict}")
print(f"available keys = {list(my_dict.keys())}")

# Track loop progress
for i, item in enumerate(my_list):
    print(f"Processing item {i}: {item}")
```

Phase 4: Common Problem Patterns & Solutions

🚨 Pattern 1: "Variable Not Defined" Errors

Symptoms:

```python
NameError: name 'conversion_rate' is not defined
```

Debugging steps:

- [ ] Search your code for where you first created this variable
- [ ] Make sure that line runs before the line that uses it
- [ ] Check if it's inside a function or if statement that might not run

Common fix:

```python
# WRONG: Using before defining
total_revenue = monthly_revenue * 12  # Error: monthly_revenue not defined
monthly_revenue = 5000

# RIGHT: Define before using
monthly_revenue = 5000
total_revenue = monthly_revenue * 12
```

🚨 Pattern 2: "Type Mismatch" Errors

Symptoms:

```python
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Debugging steps:

- [ ] Add `print(type(variable))` for each variable in the problem line
- [ ] Check if you need to convert types with `int()`, `str()`, or `float()`
- [ ] Look for `input()` functions - they always return strings

Common fix:

```python
# WRONG: Mixing types
user_input = input("Enter number: ")  # This is always a string
result = user_input + 100  # Error: can't add string + number

# RIGHT: Convert types
user_input = input("Enter number: ")
user_number = int(user_input)  # Convert string to number
result = user_number + 100
```

🚨 Pattern 3: "Index Out of Range" Errors

Symptoms:

```
python

IndexError: list index out of range
```

Debugging steps:

- [ ] Print the list: `print(my_list)`
- [ ] Print the list length: `print(len(my_list))`
- [ ] Check what index you're trying to access
- [ ] Remember: lists start at 0, not 1

Common fix:

```
python

# WRONG: Assuming list has certain length
monthly_data = [100, 150, 200]
q4_total = monthly_data[3]  # Error: only positions 0, 1, 2 exist

# RIGHT: Check length or use safe methods
monthly_data = [100, 150, 200]
if len(monthly_data) > 3:
    q4_total = monthly_data[3]
else:
    q4_total = 0  # or handle missing data appropriately
```

🚨 Pattern 4: "Key Not Found" Errors

Symptoms:

```
python

KeyError: 'email'
```

Debugging steps:

- [ ] Print the dictionary: `print(my_dict)`
- [ ] Print available keys: `print(my_dict.keys())`
- [ ] Check for typos in key names
- [ ] Use `.get()` method for safer access

Common fix:

python

```python
# WRONG: Assuming key exists
user_profile = {"name": "John", "plan": "Premium"}
email = user_profile["email"]  # Error: 'email' key doesn't exist

# RIGHT: Use safe access
user_profile = {"name": "John", "plan": "Premium"}
email = user_profile.get("email", "No email provided")
```

Phase 5: Advanced Debugging Techniques

🔧 Step-Through Debugging

When you're really stuck, trace through your code line by line:

python

```python
def calculate_engagement_score(sessions, total_users):
    print(f"Input: sessions={sessions}, total_users={total_users}")  # Check inputs

    if total_users == 0:
        print("Warning: total_users is 0, returning 0")  # Check edge cases
        return 0

    avg_sessions = sessions / total_users
    print(f"Calculated avg_sessions: {avg_sessions}")  # Check intermediate results

    engagement_score = avg_sessions * 100
    print(f"Final engagement_score: {engagement_score}")  # Check final result

    return engagement_score

# Test with debug prints
result = calculate_engagement_score(1500, 300)
print(f"Function returned: {result}")
```

🧪 Test with Simple Data

If your code works with simple data but fails with real data:

```python
# Start with simple test data
test_users = [
    {"name": "Alice", "active": True},
    {"name": "Bob", "active": False}
]

# Test your function with simple data first
result = process_users(test_users)
print(f"Test result: {result}")

# If that works, gradually use more complex data
```

🎯 Isolate the Problem

Comment out parts of your code to find where the problem starts:

```python
# Comment out everything except the basics
user_count = 1000
# conversion_rate = calculate_conversion(signups, user_count)  # Comment out
# send_report(conversion_rate)  # Comment out
print(f"User count: {user_count}")  # This should work

# Then uncomment one line at a time until you find the problem
```

Phase 6: Prevention Strategies

✅ Code Review Checklist (Before Running)

Before you run your code, quickly check:

- [ ] **Did I spell all variable names consistently?**
- [ ] **Did I define variables before using them?**
- [ ] **Are my indentations consistent (4 spaces)?**
- [ ] **Do I have colons after if/for/while/def statements?**
- [ ] **Are my quotes and parentheses balanced?**
- [ ] **Did I convert input strings to numbers if needed?**

🧪 Test Early and Often

- **Run your code frequently** - don't write 50 lines then test
- **Test with simple data first** - use small, predictable examples
- **Add print statements as you go** - see what your variables contain
- **Test edge cases** - what happens with empty lists or zero values?

📝 Keep a Debugging Journal

Write down:

- What error you encountered
- What caused it
- How you fixed it

This helps you recognize patterns and solve similar problems faster.

Emergency Debugging Protocol

When Nothing Else Works:

1. 🔄 **Start Fresh**
   - Save your current code
   - Create a new file
   - Rewrite the problematic section from scratch (often reveals the issue)
2. 🔍 **Rubber Duck Debugging**
   - Explain your code line by line to someone else (or a rubber duck)
   - Often you'll spot the problem while explaining
3. 🌐 **Search for Help**
   - Copy the exact error message into Google
   - Add "python" to your search
   - Look for Stack Overflow results
4. ✂️ **Divide and Conquer**
   - Cut your code in half
   - Test each half separately
   - Find which half has the problem
   - Repeat until you isolate the issue

🚨 When to Ask for Help

Ask for help when:

- You've spent more than 30 minutes on the same error
- You've tried all the steps in this checklist
- You understand the error message but can't figure out why it's happening

💡 How to Ask for Help Effectively

When asking for help, provide:

1. **The exact error message** (copy and paste it)
2. **The code that's causing the problem** (the specific lines)
3. **What you expected to happen**
4. **What actually happened**
5. **What you've already tried**

Remember: Debugging is a Skill

- **Every programmer debugs constantly** - you're not alone!
- **Bugs are learning opportunities** - each one teaches you something new
- **You'll get faster with practice** - what takes 30 minutes now will take 30 seconds later
- **The computer is never wrong** - it's doing exactly what you told it, just not what you meant

Keep calm and debug on! 🐛➡️✨