Common Python Mistakes Guide for Product Managers

Top 10 Beginner Mistakes (With Product Manager Examples)

1. Indentation Errors

Most common beginner mistake

What it looks like:

```python
# WRONG - Inconsistent indentation
if user_count > 1000:
print("High user base")    # Missing indentation
    print("Consider scaling")  # Too much indentation
```

Error message you'll see:

```
IndentationError: expected an indented block
```

What this means in plain English: Python uses indentation (spaces) to group code together, like how you indent bullet points. Everything that belongs inside an `if` statement, loop, or function must be indented the same amount.

How to fix it:

```python
# CORRECT - Consistent 4-space indentation
if user_count > 1000:
    print("High user base")
    print("Consider scaling")
```

Pro tip: Use 4 spaces consistently. Most code editors can show you invisible spaces and tabs.

---

2. Trying to Use Undefined Variables

What it looks like:

```python
# WRONG - Using a variable before creating it
total_revenue = monthly_revenue + quarterly_bonus  # monthly_revenue doesn't exist yet
monthly_revenue = 50000
```

Error message you'll see:

```
NameError: name 'monthly_revenue' is not defined
```

What this means in plain English: You're trying to use a variable that doesn't exist yet. It's like trying to open a filing cabinet before you've labeled it.

How to fix it:

```python
# CORRECT - Define variables before using them
monthly_revenue = 50000
quarterly_bonus = 15000
total_revenue = monthly_revenue + quarterly_bonus
```

---

3. Wrong Data Type Operations

What it looks like:

```python
# WRONG - Trying to do math with text
user_input = input("Enter user count: ")  # input() always returns text
total_users = user_input + 100            # Can't add number to text
```

Error message you'll see:

```
TypeError: can only concatenate str (not "int") to str
```

What this means in plain English: You're trying to mix different types of data in ways that don't make sense, like trying to add the word "hello" to the number 5.

How to fix it:

```python
# CORRECT - Convert text to number first
user_input = input("Enter user count: ")
user_count = int(user_input)              # Convert text to number
total_users = user_count + 100
```

---

4. List Index Out of Range

What it looks like:

```python
# WRONG - Trying to access an item that doesn't exist
monthly_signups = [120, 150, 98]  # Only 3 items (positions 0, 1, 2)
december_signups = monthly_signups[3]  # Position 3 doesn't exist
```

Error message you'll see:

```
IndexError: list index out of range
```

What this means in plain English: You're trying to access a position in a list that doesn't exist. It's like asking for the 5th item in a list that only has 3 items.

How to fix it:

```python
# CORRECT - Check list length or use proper index
monthly_signups = [120, 150, 98]
if len(monthly_signups) > 3:
    december_signups = monthly_signups[3]
else:
    december_signups = None  # Or handle missing data appropriately

# Or access the last item safely
last_month_signups = monthly_signups[-1]  # -1 always gets the last item
```

---

5. Dictionary Key Errors

What it looks like:

```python
# WRONG - Trying to access a key that doesn't exist
user_profile = {"name": "Sarah", "plan": "Premium"}
user_country = user_profile["country"]  # "country" key doesn't exist
```

Error message you'll see:

```
KeyError: 'country'
```

What this means in plain English: You're looking for information that isn't in the dictionary. Like looking for someone's phone number in an address book that only has their email.

How to fix it:

```python
# CORRECT - Use .get() method with a default value
user_profile = {"name": "Sarah", "plan": "Premium"}
user_country = user_profile.get("country", "Unknown")

# Or check if key exists first
if "country" in user_profile:
    user_country = user_profile["country"]
else:
    user_country = "Not specified"
```

---

6. Forgetting Parentheses in Function Calls

What it looks like:

```python
# WRONG - Missing parentheses
user_segments = ["New", "Active", "Churned"]
segment_count = len user_segments  # Missing parentheses after len
```

Error message you'll see:

```
SyntaxError: invalid syntax
```

What this means in plain English: Functions need parentheses to work, even if they don't need any inputs. It's like trying to call someone without dialing their number.

How to fix it:

```python
# CORRECT - Always include parentheses when calling functions
user_segments = ["New", "Active", "Churned"]
segment_count = len(user_segments)
print("Total segments:", segment_count)
```

---

7. Infinite Loops

What it looks like:

```python
# WRONG - Loop condition never becomes false
user_count = 0
while user_count < 1000:
    print("Adding more users...")
    # Forgot to increase user_count - this will run forever!
```

What happens: Your program gets stuck and keeps running the same code forever until you force it to stop.

How to fix it:

```python
# CORRECT - Make sure the loop condition will eventually become false
user_count = 0
while user_count < 1000:
    print(f"Current users: {user_count}")
    user_count += 100  # Increase the counter so loop will eventually end
```

---

8. Mixing Up Assignment (=) and Comparison (==)

What it looks like:

```python
# WRONG - Using assignment when you meant comparison
user_plan = "Premium"
if user_plan = "Premium":  # Should be == not =
    print("Premium user detected")
```

Error message you'll see:

```
SyntaxError: invalid syntax
```

What this means in plain English: You're trying to assign a value inside a condition where you should be comparing values.

How to fix it:

```python
# CORRECT - Use == for comparison, = for assignment
user_plan = "Premium"           # Assignment: setting the value
if user_plan == "Premium":      # Comparison: checking the value
    print("Premium user detected")
```

---

9. String and Number Concatenation

What it looks like:

```python
# WRONG - Trying to combine text and numbers directly
user_count = 1500
message = "We have " + user_count + " active users"
```

Error message you'll see:

```
TypeError: can only concatenate str (not "int") to str
```

What this means in plain English: You can't directly glue together text and numbers. You need to convert the number to text first.

How to fix it:

```python
# CORRECT - Convert number to string or use f-strings
user_count = 1500

# Option 1: Convert to string
message = "We have " + str(user_count) + " active users"

# Option 2: Use f-strings (recommended)
message = f"We have {user_count} active users"

# Option 3: Use .format()
message = "We have {} active users".format(user_count)
```

---

10. Forgetting to Return Values from Functions

What it looks like:

```python
# WRONG - Function does calculation but doesn't return the result
def calculate_conversion_rate(signups, visitors):
    conversion_rate = (signups / visitors) * 100
    # Missing return statement - result is lost!

result = calculate_conversion_rate(50, 1000)
print(result)  # Prints: None
```

What happens: Your function does the work but doesn't give you back the answer, so you get `None` instead of the result.

How to fix it:

```python
# CORRECT - Always return the result you want to use
def calculate_conversion_rate(signups, visitors):
    conversion_rate = (signups / visitors) * 100
    return conversion_rate  # Give back the result

result = calculate_conversion_rate(50, 1000)
print(result)  # Prints: 5.0
```

Error Message Translator

When you see this error → It usually means this:

Error Message	Plain English Meaning	Quick Fix
`IndentationError`	Your code spacing is wrong	Fix your indentation (use 4 spaces consistently)
`NameError: name 'x' is not defined`	You used a variable that doesn't exist	Create the variable before using it
`TypeError`	You mixed incompatible data types	Convert types or check your data
`IndexError: list index out of range`	You tried to access a list position that doesn't exist	Check list length or use different index
`KeyError`	You looked for a dictionary key that doesn't exist	Use `.get()` method or check if key exists
`SyntaxError: invalid syntax`	Python can't understand your code	Check for typos, missing parentheses, or wrong symbols
`ValueError`	The value is the right type but wrong format	Check your input data format

Before & After Examples

Example 1: Processing User Feedback

Before (with errors):

```python
# Multiple mistakes in this code
feedback_list = ["Great app", "Too slow", "Love it"]
positive_count = 0

for feedback in feedback_list
    if feedback = "Great app" or feedback = "Love it":  # Missing colon, wrong operato
    positive_count += 1

print("Positive feedback: " + positive_count)  # Type error
```

After (corrected):

python

```python
# Fixed version
feedback_list = ["Great app", "Too slow", "Love it"]
positive_count = 0

for feedback in feedback_list:  # Added missing colon
    if feedback == "Great app" or feedback == "Love it":  # Fixed comparison operators
        positive_count += 1

print(f"Positive feedback: {positive_count}")  # Fixed string formatting
```

Example 2: Calculating User Metrics

Before (with errors):

python

```python
# Problematic code
def calculate_metrics(users):
monthly_active = 0  # Wrong indentation
for user in users:
if user[active]:  # Missing quotes, wrong indentation
monthly_active += 1
# Missing return statement

users = [{"name": "John", "active": True}, {"name": "Jane", "active": False}]
result = calculate_metrics(users)
print("MAU:", result)  # Will print None
```

After (corrected):

```python
# Fixed version
def calculate_metrics(users):
    monthly_active = 0  # Correct indentation
    for user in users:
        if user["active"]:  # Added quotes, fixed indentation
            monthly_active += 1
    return monthly_active  # Added return statement

users = [{"name": "John", "active": True}, {"name": "Jane", "active": False}]
result = calculate_metrics(users)
print(f"MAU: {result}")  # Will print: MAU: 1
```

Prevention Tips

1. **Write code step by step** - Don't try to write everything at once
2. **Test frequently** - Run your code often to catch errors early
3. **Use descriptive variable names** - `conversion_rate` is better than `cr`
4. **Read error messages carefully** - They usually tell you exactly what's wrong
5. **Keep functions small** - Easier to debug when things go wrong
6. **Use print statements** - Add `print()` statements to see what your variables contain
7. **Check your data types** - Use `type()` to see what kind of data you're working with

Quick Debugging Questions to Ask Yourself

1. **Is my indentation consistent?** (Count the spaces)
2. **Did I define all my variables before using them?**
3. **Are my parentheses and quotes balanced?**
4. **Am I using the right comparison operators?** (== not =)
5. **Do my lists and dictionaries have the data I think they do?**
6. **Did I forget to return a value from my function?**
7. **Are my data types compatible for the operation I'm trying to do?**

Remember: Every programmer makes these mistakes, even experienced ones! The key is learning to recognize and fix them quickly.