Debugging: Reading Error Messages and Fixing Bugs

Python for Absolute Beginners · Lesson 14

What you'll learn

- Why error messages are help, not failure
- How to read an error message from the bottom up
- The seven errors beginners meet most, and what each one means
- A four-step routine that works on any bug
- How to find bugs that give no error at all, using print()



Every programmer meets bugs

- A bug is a mistake in a program
- Every programmer, beginner or expert, writes bugs every day
- Some bugs stop the program with an error message
- Others run, but give the wrong answer
- When you see red text, don't panic and don't guess: read it



Reading an error message

```
                                score = 5
print(scroe)

Output:
Traceback (most recent call last):
  File "C:\python-course\bug1.py", line 2, in <module>
    print(scroe)
          ^^^^^
NameError: name 'scroe' is not defined. Did you mean:
'score'?
```

Read it from the bottom up

- The last line says what went wrong: a NameError, Python doesn't know the name scroe
- The lines above say where: line 2, and the line of code
- Start at the bottom, then find the line number
- Your message may look slightly different, depending on your version of Python

SyntaxError: the missing colon

```
                              age = 12
if age >= 10
    print("Old enough")

Output:
SyntaxError: expected ':'
```

- Every if, elif, else, for and while line ends with a colon
- Python reads the whole file first, so a SyntaxError appears before anything prints
- If the line it points to looks fine, check the line above



SyntaxError: parentheses and = versus ==

```
print("Hello"

Output:
SyntaxError: '(' was never closed

if age = 12:

Output:
SyntaxError: invalid syntax. Maybe you meant '==' or ':='
instead of '='?
```

- Every opening parenthesis needs a closing one
- One = stores a value. Two == ask a question
- In an if line, use ==

IndentationError

```
                                age = 12
if age >= 10:
print("Old enough")

Output:
IndentationError: expected an indented block after 'if'
statement on line 2
```

- The line after a colon must be indented, usually four spaces

NameError

```
print(Hello)

Output:
NameError: name 'Hello' is not defined
```

- A typo in a variable name
- Forgetting the quotes around text
- Using a name before you created it
- Capital letters count: true is not True



TypeError

```
print("Age: " + 12)

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

- You tried to join text and a number with +
- Use a comma instead: print("Age:", 12)
- Or convert the number with str()
- Also what you get when you forget int() on an answer from input()

IndexError and KeyError

```
                fruits = ["apple", "banana", "cherry"]
print(fruits[3])

Output:
IndexError: list index out of range

student = {"name": "Mia"}
print(student["age"])

Output:
KeyError: 'age'
```

- Positions start at 0, so the last position is one less than len()
- Check the spelling of dictionary keys, and that the entry was added first

ValueError

```
age = int("twelve")

Output:
ValueError: invalid literal for int() with base 10:
'twelve'
```

- int() can only convert text that looks like a whole number
- "twelve" and "12.5" both fail

Error cheat sheet

Error	It usually means	Check first
SyntaxError	Python can't read the code	Check colon, quotes, parentheses, = vs ==
IndentationError	The spaces are wrong	Indent the line after a colon
NameError	Python doesn't know the name	Check typos, quotes, capital letters
TypeError	Wrong kinds of things together	Text plus a number?
IndexError	That list position doesn't exist	Count from 0
KeyError	That dictionary label doesn't exist	Check the spelling
ValueError	Right type, wrong value	Does the text look like a number?

A four-step routine

1. Read the last line
2. Go to the line number, and look at the line above it
3. Say what you expected and what happened
4. Change one thing, then run it again

One change at a time, so you always know which change helped

Bugs with no error message

```
                                a = 10
b = 20
average = a + b / 2
print(average)

Output:
20.0
```

- Expected 15.0, got 20.0
- Python did exactly what you asked, just not what you meant
- No error to read, so look inside the program

print() is your flashlight

```
                                    total = 0
for number in range(1, 4):
    total = total + number
    print("number is", number, "total is", total)
print(total)

Output:
number is 1 total is 1
number is 2 total is 3
number is 3 total is 6
6
```

- Show what a value really is, then compare it with what you expected
- The average bug: division happens first. The fix: average = (a + b) / 2
- Delete the extra print() lines once the bug is fixed

When you're stuck

- Read your code aloud, line by line
- Shrink the problem: test the smallest piece that still goes wrong
- Search the last line of the error message
- Take a short break
- Go back to the last version that worked, then change one thing at a time



Hands-on exercise

- Make each of the seven errors on purpose, and write down each last line in your own words
- Run the practice file and fix its three bugs, one at a time
- Fix the average program so it prints 15.0
- Add print() lines inside a loop you wrote earlier, then remove them
- Write the four-step routine on a sticky note

Key takeaways

- An error message is Python helping you: read it
- Read from the bottom up: the last line says what, the line number says where
- The seven common errors: SyntaxError, IndentationError, NameError, TypeError, IndexError, KeyError, ValueError
- Four steps fix any bug: read the last line, go to the line, say what you expected, change one thing
- No error but the wrong answer? Use print() to look inside



Take the quiz

Take the Lesson 14 quiz: 5 quick questions to check what you learned.

Next lesson

Functions: Reusable Blocks