Title SEO: 5 Common Python Errors and How to Fix Them for Beginners

New learners of Python often make mistakes such as IndentationError, SyntaxError, IndexError, ValueError and AttributeError. This article explains in detail the causes and solutions to help you debug quickly, write neat and more efficient Python code.

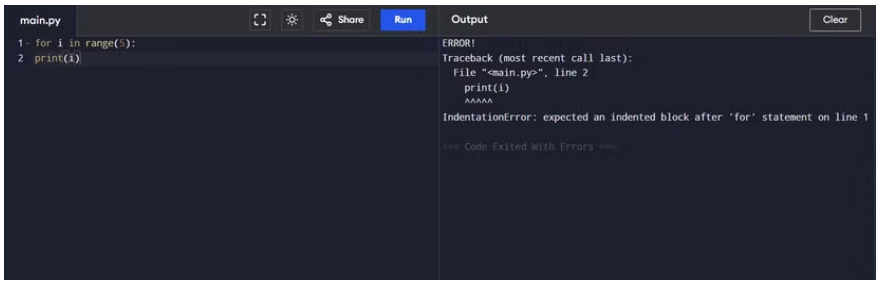
There are many different types of errors in Python. Whether you are new or experienced, you have probably encountered at least one annoying error. Some errors are easy to spot, while others make you scratch your head. Here are the 5 most common errors in Python and how to fix them.

1. IndentationError – Indentation error

If you are new to Python, or coming from C/C++ or Java, this is probably the most common mistake you will make.
Python uses **indentation** to define blocks of code, instead of punctuation marks `{}`like other languages.

For example:

```
for i in range(5): print(i)
```



Running this section will give **an IndentationError** because `print(i)`it is not properly indented. How to fix:

```
for i in range(5): print(i)
```

Note:

Don't mix **tabs** and **spaces** , as it's hard to spot errors with the naked eye. Also, when copying code from different editors, remember to double-check the indentation.

2. SyntaxError – Syntax error

Every language has its own set of syntax rules. If you write them wrong, Python won't understand what you want to do.

Some common syntax errors:

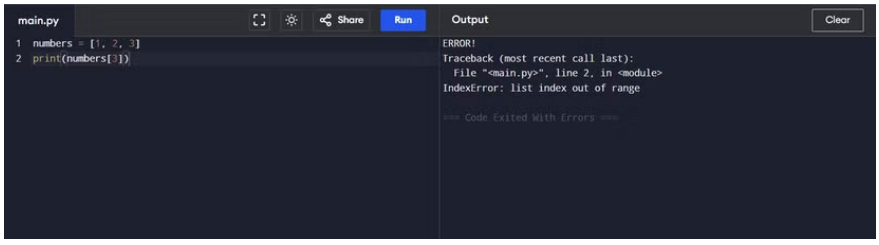
1. Forgot the `:` mark after `if`, `for`, `def`.
2. Mistyped keyword: write `esle` instead of `else`.
3. Use reserved keywords as variable names.
4. Confuse =(assign) with ==(compare).
5. Omitted parentheses or quotes.

The good news is that Python usually tells you which line is causing the error and what kind of error it is, making it easier to fix. If you use an IDE like PyCharm or VS Code, they will also suggest fixes very quickly.

3. IndexError – Index out of range error

Python indexes from **0** . If the list has `N`elements, a valid index is `0 ? N-1`.

```
numbers = [1, 2, 3] print(numbers[3]) # IndexError
```



How to avoid: always check the list length before accessing:

```
if index  len(numbers): print(numbers[index])
```

Also, don't iterate and edit the list in the loop at the same time, because it's easy to get an IndexError.

4. ValueError – Invalid value

This error occurs when **the data type is correct** , but **the value is invalid** .

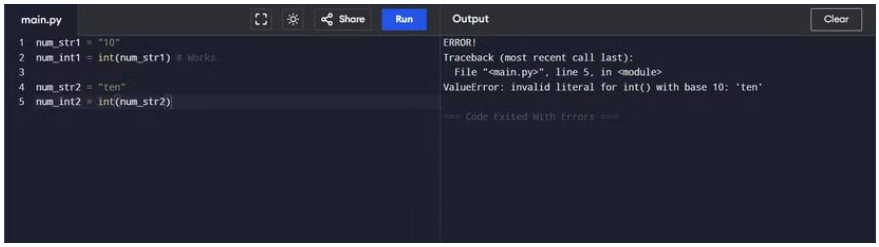
For example:

```
int("ten") # ValueError
```

The function `int()` receives a string as correct, but `"ten"` not a number so it fails.

Or:

```
import math math.sqrt(-5) # ValueError
```



The function `sqrt()` only accepts positive numbers.

How to handle: used `try-except` to avoid crash:

```
try: num = int(input("Nh?p s?: ")) except ValueError as e: print("Giá tr? 
 không h?p l?:", e)
```

5. AttributeError – Wrong attribute call

Raised when calling a **method or property that does not exist** on the object.

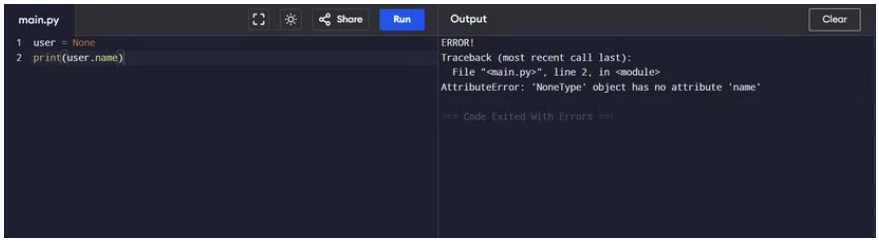
For example:

```
text = "hello" text.push() # AttributeError
```

String has no function `push()`.

Or:

```
user = None print(user.name) # AttributeError
```



How to handle:

1. Use `type()` or `isinstance()` to check the object type.

2. Used `dir(obj)` to see what properties an object has.

Conclude

Python is a friendly and easy-to-learn language, but there are still many errors that confuse beginners. Understanding **the 5 common errors** above will help you debug faster, write cleaner and more 'Pythonic' code.