

# How to Fix TypeError: Int Object Is Not Iterable in Python

Apr 11, 2023



## Table of Contents

Summarize and analyze this article with 🍎

🤖 Google AI Mode or 💬 ChatGPT or 💬 Perplexity or 🤖 Claude or 🦅 Grok (X).

The Python `TypeError: 'int' object is not iterable` is an exception that occurs when trying to loop through an integer value. In Python, looping through an object requires the object to be "iterable". Since integers are not iterable objects, looping over an integer raises the `TypeError: 'int' object is not iterable` exception.

## Python TypeError: Int Object Is Not Iterable Example

Ask AI 

Here's an example of a Python `TypeError: 'int' object is not iterable` thrown when trying to iterate over an integer value:

```
myint = 10

for i in myint:
    print(i)
```

In the above example, `myint` is attempted to be iterated over. Since `myint` is an integer and not an iterable object, iterating over it raises a `TypeError: 'int' object is not iterable`:

```
File "test.py", line 3, in <module>
    for i in myint:
TypeError: 'int' object is not iterable
```

## How to Fix `TypeError: Int Object Is Not Iterable`

In the above example, `myint` cannot be iterated over since it is an integer value. The Python `range()` function can be used here to get an iterable object that contains a sequence of numbers starting from 0 and stopping before the specified number.

Updating the above example to use the `range()` function in the `for` loop fixes the error:

```
myint = 10

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

Running the above code produces the following output as expected:

```
0
1
2
3
```

4  
5  
6  
7  
8  
9

# How to Avoid TypeError: Int Object Is Not Iterable

The Python `TypeError: 'int' object is not iterable` error can be avoided by checking if a value is an integer or not before iterating over it.

Using the above approach, a check can be added to the earlier example:

```
myint = 10

if isinstance(myint, int):
    print("Cannot iterate over an integer")
else:
    for i in myint:
        print(i)
```

A try-except block can also be used to catch and handle the error if the type of object not known beforehand:

```
myint = 10

try:
    for i in myint:
        print(i)
except TypeError:
    print("Object must be an iterable")
```

Surrounding the code in try-except blocks like the above allows the program to continue execution after the exception is encountered:

Object must be an iterable

# Track, Analyze and Manage Errors With Rollbar

Managing errors and exceptions in your code is challenging. It can make deploying production code an unnerving experience. Being able to track, analyze, and manage errors in real-time can help you to proceed with more confidence. Rollbar automates error monitoring and triaging, making fixing Python errors easier than ever. [Try it today!](#)

Python

## Related Resources

## **How to Handle TypeError: Unhashable Type 'Dict' Exception in Python**

## **How to Fix TypeError Exceptions in Python**

## How to Throw Exceptions in Python



**Build with confidence. Release with clarity.**

Rollbar helps you track what breaks, understand why, and improve what comes next.

- ✓ 5K free events per month, forever
- ✓ 14-day full feature trial
- ✓ Easy and quick installation

[Get started in minutes](#)

Plans starting at \$0. Take off with our 14-day full feature Free Trial.

## PRODUCT

	DOCUMENTATION
Product	Docs Overview
Pricing	Setting up Rollbar
Customers	Notifications
Platforms	Deploy Tracking
Integrations	Telemetry
Changelog	Security & Compliance
Roadmap	API

## LANGUAGES/FRAMEWORKS

JavaScript	Go
iOS	Java
Next.js	.NET
React	Node.js
React Native	PHP
Rails	Python
Ruby	Django
More...	

## COMPANY

About Us
Careers
Contact Us

