exceptions

January 25, 2026

Common Built-in Exceptions

Exception When it occurs

ZeroDivisionError divide by zero

ValueError wrong value type

TypeError wrong data type

FileNotFoundError missing file

IndexError invalid index

KeyError missing dictionary key

[1]:
```python
try:
    print(10/0)
except:
    print("ZeroDivisionError: division by zero")
```

```
ZeroDivisionError: division by zero
```

[2]:
```python
try:
    x = int("abc")
except ValueError:
    print("ValueError: invalid literal for int() with base 10: 'abc'")
```

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

[5]:
```python
# Multiple Exceptions
try:
    a = int(input())
    b = int(input())
    print(a / b)
except ZeroDivisionError:
    print("ZeroDivisionError: division by zero")
except ValueError:
    print("ValueError: invalid literal for int() with base 10")
```

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

[6]:
```
# Generic exception handling
try:
    print(a)
except Exception as e:
    print(f"An error occurred: {e}")
```

```
10
```

[7]:
```
# 7  else Block (Runs if NO Exception)

try:
    a=20
    b=2
    print(a/b)
except ZeroDivisionError:
    print("Division by zero is not allowed.")
else:
    print("Division performed successfully.")
```

```
10.0
Division performed successfully.
```

[8]:
```
# 8  finally Block (ALWAYS Executes)

# Used for cleanup.

try:
    f=open("sample.txt", "r")
    print(f.read())
except FileNotFoundError:
    print("FileNotFoundError: The file does not exist.")
finally:
    print("Execution completed.")
```

```
FileNotFoundError: The file does not exist.
Execution completed.
```

[9]:
```
# 9  Complete Flow (try-except-else-finally)

try:
    x=int(input())
    print(10/x)
except Exception as e:
    print(f"An error occurred: {e}")
else:
    print("No error occurred.")
finally:
    print("Execution finished.")
```

```
1.0
No error occurred.
Execution finished.
```

[10]:
```python
#  Raising Exceptions (Manual Error Trigger)

# You can force an exception.
age=-5
if age<0:
    raise ValueError("Age cannot be negative.")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[10], line 6
      4 age=-5
      5 if age<0:
----> 6     raise ValueError("Age cannot be negative.")

ValueError: Age cannot be negative.
```

[11]:
```python
# 1 1  Custom Exceptions (ADVANCED)

# Create your own exception.

class invalidAgeError(Exception):
    pass

age=-1
if age<0:
    raise invalidAgeError("Age cannot be negative.")
```

```
---------------------------------------------------------------------------
invalidAgeError                           Traceback (most recent call last)
Cell In[11], line 10
      8 age=-1
      9 if age<0:
---> 10     raise invalidAgeError("Age cannot be negative.")

invalidAgeError: Age cannot be negative.
```

[12]:
```python
# 1 2  Exception Handling with Functions

def divide(a,b):
    try :
        return a/b
    except ZeroDivisionError:
        return "Error: Division by zero is not allowed."
print(divide(10,2))
print(divide(10,0))
```

```
5.0
Error: Division by zero is not allowed.
```

[ ]:
```
# Exception handling with files
try:
    with open("data.txt", "r") as file:
        data = file.read()
        print(data)
except FileNotFoundError:
    print("FileNotFoundError: The file does not exist.")
# with automatically closes files better than finally block
```

```
FileNotFoundError: The file does not exist.
```

[15]:
```
# Exception Handling with Lists & Dicts

lst=[1,2,3]
try:
    print(lst[5])
except Exception as e:
    print(f"An error occurred: {e}")

d={"name":"Tarun"}
try:
    print(d["age"])
except KeyError as e:
    print(f"KeyError: {e}")
```

```
An error occurred: list index out of range
KeyError: 'age'
```

[16]:
```
# Handling multiple exceptions in a single block
try:
    x=int("abc")
    print(10/x)
except (ValueError, ZeroDivisionError) as e:
    print(f"An error occurred: {e}")
```

```
An error occurred: invalid literal for int() with base 10: 'abc'
```

[17]:
```
# Nested try except
try:
    try:
        print(10/0)
    except ZeroDivisionError:
        print("Inner ZeroDivisionError caught")
except Exception as e:
    print(f"Outer exception caught: {e}")
```

```
Inner ZeroDivisionError caught
```