# Python types of errors

Created By @**Mohamed Hamed**

**1st – year Artificial Intelligence student at KFS university, you can find me down:**

- **Facebook**
- **LinkedIn**
- **GitHub**
- **YouTube**

---

**You also can get 2 PDFs for some public methods of Data Structure (lists, sets, etc…) and Quiz in Basics down:**

- **Methods**
- **Quiz**

**When writing code in Python, you may encounter errors that prevent your program from running properly. These errors fall into three main categories: syntax errors, runtime errors, and logic errors. In this pdf, i'll try to explain each type of error.**

## 1. Syntax Errors:

### What It Is:

Syntax errors are like grammatical mistakes in a sentence. They occur when the code does not follow the rules of the Python language. The Python interpreter detects these errors before the program is executed.

### When It Occurs:

During the parsing phase, before the program runs.

**Examples:**

**Missing Colon (:)** at the end of a conditional sentence

```
if x == 5
    print("x is 5")
```

**Error Message:**

```
SyntaxError: invalid syntax
```

**Mismatched Parentheses ()** in built in function Print()

```
print("Hello, World!"
```

**Error Message:**

```
SyntaxError: unexpected EOF while parsing
```

---

## 2. Runtime Errors (Exceptions):

**What It Is:**

Runtime errors occur when the program is running. These errors are not detected during the parsing phase but arise when the program tries to perform an invalid operation.

**When It Occurs:**

During the execution of the program.

**Examples:**

**NameError:** the variable **y** is not defined before we call it

```
print(y)
```

**Error Message:**

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

**TypeError:**

```
result = "5" + 3
```

**Error Message:**

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

**IndexError:** when you are trying to access an index that is out of range of our variable

```
my_list = [1, 2, 3]
print(my_list[5])
```

**Error Message:**

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

**ZeroDivisionError:** Division by zero is mathematically undefined.

```
result = 10 / 0
```

**Error Message:**

```
ZeroDivisionError: division by zero
```

**KeyError:** when you are trying to access a key that does not exist in the dictionary

```
my_dict = {"name": "Alice"}
print(my_dict["age"])
```

**Error Message:**

```
KeyError: 'age'
```

**FileNotFoundError:** like you are trying to open file that does not exist

```
myFile = open("C:\AI\MohamedHamed.txt","r")
```

**Error Message:**

```
FileNotFoundError: [Errno 2] No such file or directory: 'nonexistent_file.txt'
```

**AttributeError:** like you are trying to use method with wrong object

```
my_list = [1, 2, 3]
my_list.append(4)  # Correct
my_list.add(5)     # Incorrect
```

**Error Message:**

```
AttributeError: 'list' object has no attribute 'add'
```

---

*Note: see the difference between **TypeError** and **AttributeError**

**TypeError:** Imagine you are trying to add an apple and a banana.That doesn't make sense,right? In Python, if you try to add a number (5) and a word ("hello"),Python gets confused because they are different types—just like an apple and a banana.

**AttributeError:** Imagine you have a toy car,and you tell it to "fly."But toy cars don't have wings! Python does the same thing—it gets confused when you ask something(like a list)to do something it wasn't designed to do. (read again and i hope you get it)

---

## 3. Logical Errors:

**What It Is:**

Logical errors occur when the program runs without crashing but produces incorrect results due to flawed logic.

**When It Occurs:**

During execution, but no exceptions are raised.

**Examples:**

**Incorrect Formula:**

```
def calculate_area(radius):
    return 2 * 3.14 * radius  #Logical error:formula for circumference instead of area
```

**Error:** the formula calculates the circumference of a circle instead of the area, even if we don't get an error, but that's still not our intention. the correct formula for area is **3.14 * radius ** 2**

**Incorrect Condition:**

```
def is_even(number):
    return number % 2 != 0  # Logical error: checks for odd instead of even
```

**Error:** this checks for odd numbers instead of even.

---

## Summary Table of Errors:

| Error Type | Description | When It Occurs | Example |
|---|---|---|---|
| **Syntax Error** | Violation of Python's syntax rules. | Before execution, during parsing. | Missing colon, incorrect indentation. |
| **Runtime Error** | Errors during execution. | During execution. | NameError, TypeError, IndexError. |
| **Logical Error** | Code runs but produces incorrect results. | During execution, but no exceptions are raised. | Incorrect formula, loop logic, or condition. |