⚠️

Errors

AssertionError

```
AssertionError: Should be 6
```

AttributeError

> `AttributeError` occur when using incorrect or unsupported list operations or methods. Attempting to sort a list that contains elements of different data types can result in an `AttributeError`.

> Ensure that you're using the appropriate methods and operations supported by lists.

> Check data types: If you encounter an `AttributeError` or unexpected behavior, review the data types in your list and ensure they are compatible with the operations you're performing

```
fruits = ["apple", "banana", "cherry", 5]
fruits.sort()  # AttributeError: 'int' object has no attribute 'lower'

-----Output-----
AttributeError: 'int' object has no attribute 'lower'
```

> Ensure that the indices you are using to access or modify list elements are correct and within the valid range

> Verify the presence of values: Before using methods like `remove()` or `index()`, confirm that the value exists in the list to avoid `ValueError`

IndexError

> `IndexError` occurs when you try to access an index that is outside the valid range of indices for a list. This typically happens when the index is negative or greater than or equal to the length of the list.

> Ensure that the index you're trying to access is within the valid range of indices for the list. Remember that list indices start from `0` and go up to `len(list) - 1`.

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

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

KeyError

> This error occurs when you try to access a key that does not exist in the dictionary.

```
student_scores = {"Alice": 90, "Bob": 85}
print(student_scores["Charlie"])

-----Output-----
KeyError: 'Charlie'
```

- Check if you are using the correct key spelling and case sensitivity

- Use the `in` operator to check if the key exists in the dictionary before accessing it
- Use the `get()` method to retrieve the value for a key and provide a default value to handle cases where the key is not present

NameError

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

variable = 55
print(varaible)
```

TypeError

This error occurs when you perform dictionary operations with incompatible data types.

```
student_scores = {["Alice", "Bob"]: 90, "Charlie": 95}

-----Output-----
TypeError: unhashable type: 'list'
```

- Ensure that the keys and values are of compatible data types
- Avoid using mutable data types like lists as dictionary keys since they are not hashable. In the context of dictionaries in Python, the term "hashable" refers to an object's ability to be uniquely identified and mapped to a fixed-size integer value, known as a hash value.

```


-----Output-----
TypeError: can't convert 'int' object to string

try this
print ('Hi '+str(5))
```

```
TypeErrpr: unsupported operand type(s) for int +
-------------------------------------
try this
print ('Hi '+str(5))
```

```
TypeError: missing 1 required positional argument
TypeError: takes 2 positional arguments but 3 were given
-------------------------------------------------------
def exampleFunction(x,y,z):
exampleFunction(1) '''Error 1'''
exampleFunction(1,2,3,4) '''Error 2'''
exampleFunction(1,2,3) '''Correct'''
```

ValueError

> There is an issue with the value stored in a particular object

```
fruits = ["apple", "banana", "cherry"]
fruits.remove("orange")  # ValueError: list.remove(x): x not in list

-----Output-----
ValueError: list.remove(x): x not in list
```

> A `ValueError` may occur when using methods like `remove()` or `index()` if the specified value is not found in the list. Check if the value exists in the list before performing the operation

```
ValueError: invalid literal for int() with base 10
--------------------------------------------------
try this
print(float('8.5')+5)
```

```
x, y (1,2,3)
ValueError: too many values to unpack
```

```
year = "O6" # O instead of 0. O is a letter not a number
year = type(int(year))
year = type(float(year))
```

```
-----Output-----
ValueError: invalid literal for int() with base 10: 'O6'
ValueError: could not convert string to float: 'O6'
```

UnboundLocalError

```
UnboundLocalError: local variable 'x' referenced before assignment
```