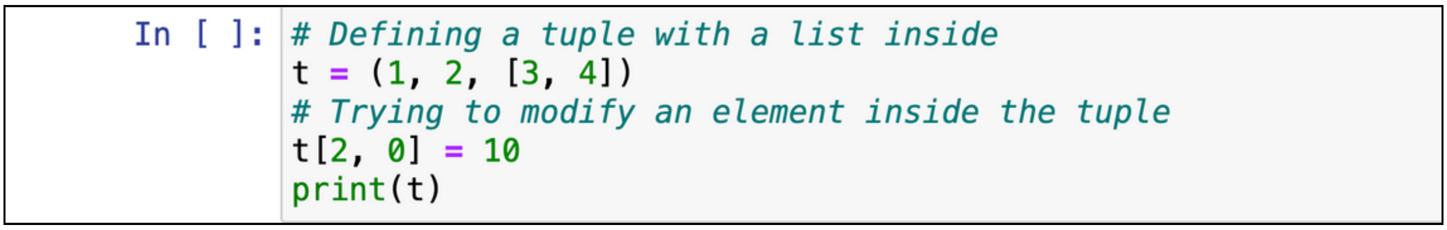
Tuple with a List Inside – Why This Code Fails?

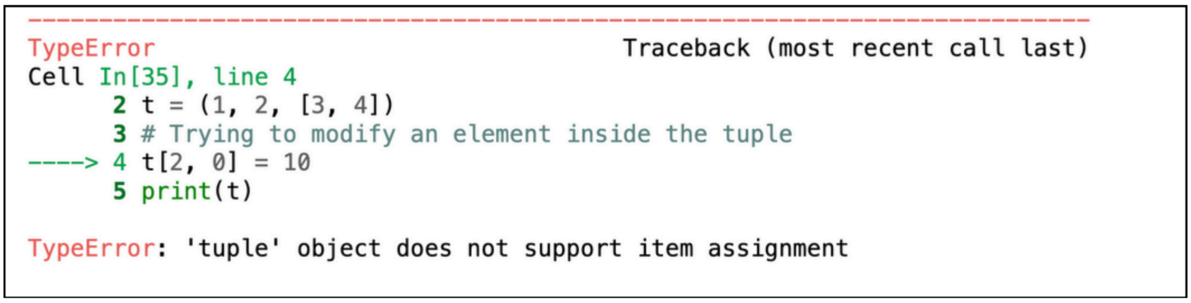
Python Brain Teaser

💻 Code:



```
In [ ]: # Defining a tuple with a list inside
        t = (1, 2, [3, 4])
        # Trying to modify an element inside the tuple
        t[2, 0] = 10
        print(t)
```

❌ Error Alert! 🚨



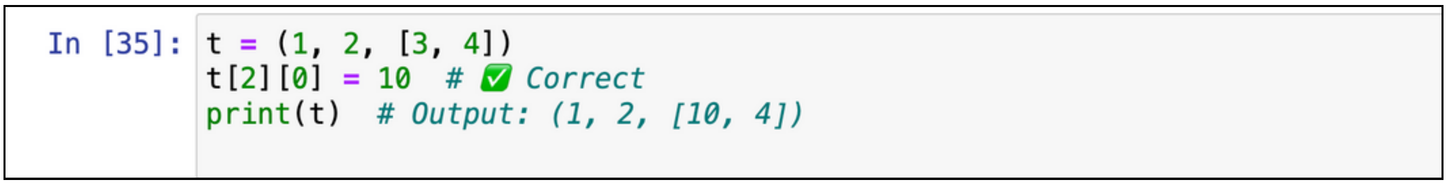
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[35], line 4
      2 t = (1, 2, [3, 4])
      3 # Trying to modify an element inside the tuple
----> 4 t[2, 0] = 10
      5 print(t)

TypeError: 'tuple' object does not support item assignment
```

👀 Why the Error Happens?

📌 t → Tuple (**Immutable**) 🚫
📌 t[2] → List Inside Tuple (**Mutable**) ✅
📌 t[2,0] → But ,Python sees this expression as modifying the tuple itself(since the syntax is wrong), which isn't allowed 🚨

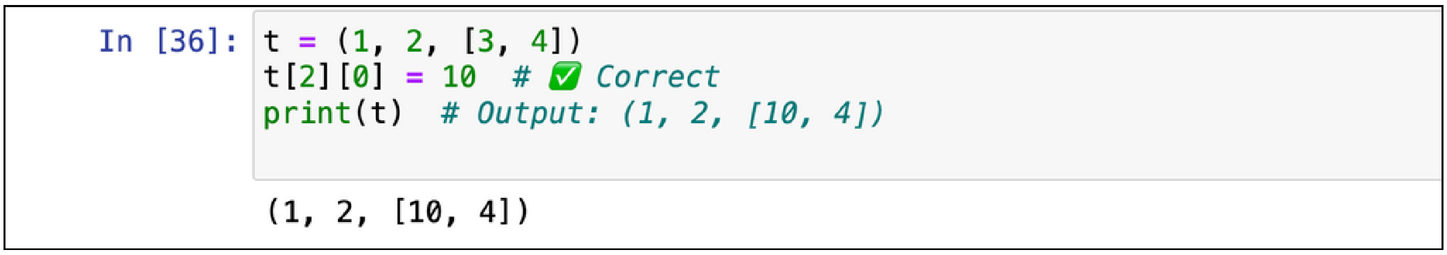
🛠️ Fixing the Code!

Modify the **list** inside, **not the tuple**!



```
In [35]: t = (1, 2, [3, 4])
         t[2][0] = 10  # ✅ Correct
         print(t)  # Output: (1, 2, [10, 4])
```

👀 Output:



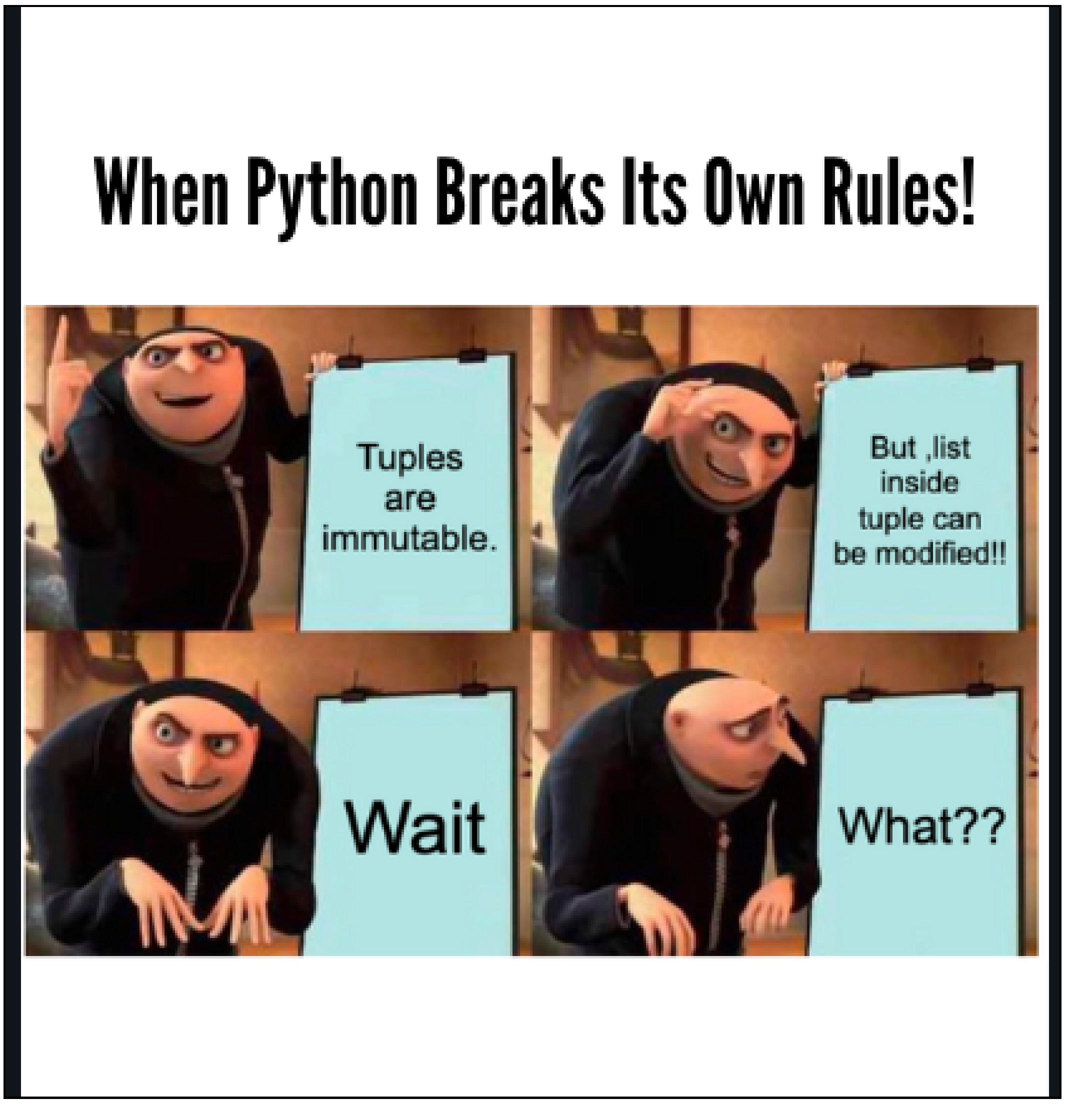
```
In [36]: t = (1, 2, [3, 4])
         t[2][0] = 10  # ✅ Correct
         print(t)  # Output: (1, 2, [10, 4])

         (1, 2, [10, 4])
```

💡 Key Difference:

❌ t[2,0] → Tries to modify the tuple (Not Allowed!)
✅ t[2][0] → Modifies the list inside (Allowed!)

🔥 Takeaway: Tuples are immutable, but lists inside them can be modified! 🚀

💬 Did you spot the error? Comment below! 😜👇
🔔 Follow for more Python tricks! 🐍✨


When Python Breaks Its Own Rules!
Tuples are immutable.
But ,list inside tuple can be modified!!
Wait
What??

🔍Exact Reasoning for above meme :

🔹 **Tuples are immutable** 🚫, meaning their structure (the references they hold) **cannot be changed.** 🔗❌

🔹 **BUT**... if a tuple contains a **mutable object** (like a list 📝), the **reference to the list remains the same(which is tuple itself)**, but its **contents can be modified**! ✏️✅

```
In [39]: #Example with Memory Addresses:

         t = (1, 2, [3, 4])

         print(id(t[2]))  # Memory address of the list inside tuple

         t[2][0] = 10  # Modify list content

         print(id(t[2]))  # Address remains the same, proving list is modified in place
         print(t)
```

```
4425823424
4425823424
(1, 2, [10, 4])
```