

# Argument of type "Literal['x']" cannot be assigned to parameter "\_\_s" of type "slice" in function "\_\_setitem\_\_"

Asked 2 months ago Modified 2 months ago Viewed 474 times

My code could be simplified to this:

```
from typing import Union, Any, Dict, List, Tuple

def myFunc() -> Union[Dict, List, str,
                    int, float, bool, None]:
    something = { 'prop': 123 }
    return something

obj = myFunc()
obj['k'] = isinstance(obj, Dict) and 321
```

and I got the exception from Pylance like this:

```
(variable) obj: object | List | str | int | float | bool | Dict | None
"__setitem__" method not defined on type "object" PylancereportGeneralTypeIssues
"__setitem__" method not defined on type "str" PylancereportGeneralTypeIssues
"__setitem__" method not defined on type "int" PylancereportGeneralTypeIssues
"__setitem__" method not defined on type "float" PylancereportGeneralTypeIssues
"__setitem__" method not defined on type "bool" PylancereportGeneralTypeIssues
Argument of type "Literal['k']" cannot be assigned to parameter "__s" of type
"slice" in function "__setitem__"
"Literal['k']" is incompatible with "slice" PylancereportGeneralTypeIssues
```

What could be the issue?

python

python-3.x

Share Edit Follow Flag

edited Dec 8, 2022 at 20:30

asked Dec 8, 2022 at 20:00



Roman

17.7k

14

85

89

2 `Union[object, ...]` is pretty much identical to `object`, since *everything* is an instance of object via inheritance. – chepner Dec 8, 2022 at 20:24

The code is going to fail at runtime even if `isinstance(obj, Dict)` is false. `and` only applies to the evaluation of the right-hand side, not to the attempt to assign to `obj['k']`. – chepner Dec 8, 2022 at 20:29

What you want is a regular `if` statement: `if isinstance(obj, dict): obj['k'] = 123`. – chepner Dec 8, 2022 at 20:30

1 Answer

Sorted by:

Highest score (default) 

0



The issue is in the assumption that `obj['k']` can always be valid, but this is not the case for the obj of the types `List | str | int | float | bool | None`. The solution is to rewrite the last line as follows:

```
obj = isinstance(obj, Dict) and {**obj, 'k': 123}
```

The complete example is below:




```
from typing import Union, Any, Dict, List, Tuple

def myFunc() -> Union[object, Dict, List, str,
                    int, float, bool, None]:
    something = { 'prop': 123 }
    return something

obj = myFunc()
obj = isinstance(obj, Dict) and {**obj, 'k': 123}
```

[Share](#) [Edit](#) [Follow](#) [Flag](#)[edited Dec 8, 2022 at 20:28](#)[answered Dec 8, 2022 at 20:00](#)**Roman**

17.7k ● 14 ● 85 ● 89

- 1  Why make a new `dict`? You could just as easily perform the type check and then do the item assignment if it passes. With `and`, you'll throw away any non-`dict` and replace it with `False`, further confusing the static typing. `if isinstance(obj, dict): obj['k'] = 123` would do the trick without type confusion and needless copies. – [ShadowRanger](#) Dec 8, 2022 at 20:30 
-  Thank you @ShadowRanger, `if isinstance(obj, dict): obj['k'] = 123` is another working option – [Roman](#) Dec 8, 2022 at 20:34 