```python
class SingletonInstance:
    _instance = None

    def __new__(cls, *args, **kwargs):
        if not cls._instance:
            cls._instance = super(SingletonInstance, cls).__new__(cls, *args, **kwargs)
        return cls._instance

    def __init__(self):
        self.essential_object = "This is an essential object"

    def get_essential_object(self):
        return self.essential_object

    def set_essential_object(self, new_value):
        self.essential_object = new_value

def get_global_instance():
    return SingletonInstance()

if __name__ == "__main__":
    # Getting the global access point
    instance1 = get_global_instance()
    print(instance1.get_essential_object())

    # Modifying the essential object
    instance1.set_essential_object("Modified essential object")

    # Accessing again
    instance2 = get_global_instance()
    print(instance2.get_essential_object())

    print(instance1 is instance2)
```

Singleton

Provide a global access point to that instance

- The **Singleton pattern** lets you **access some object from anywhere in the program**. However, it **also protects that instance** from **being overwritten** by other code.
- It's much better to **have it within one class**, especially if the rest of your code already depends on it.