License Management System

Class License is created to handle individual license details.
Here we'll require some **attributes** for license

- product_name
- license_key
- status
- expiry_date

Methods

➔ **The __init__ method**: built-in function that initializes new objects when the class is created.

```
class License:
  def __init__(self, product_name, license_key, status, expiry_date):
        self.product_name = product_name
        self.license_key = license_key
        self.status = status
        self.expiry_date = expiry_date
```

➔ **The __str__ method**: returns a reader-friendly string of the class objects. The __str__ method called implicitly when the print function is used for **license.**

```
  def __str__(self):
    return f"{self.product_name} | {self.license_key} | {self.status} |
{self.expiry_date}"
```

➔ **check_expired():** Checks if the licence is expired by comparing current date versus expiry date.

```
  def check_expired(self):
    expiry_date=date.fromisoformat(self.expiry_date)
    today=date.today()
    return expiry_date<today
```

Class LicenseManager is created to handle multiple license functions. (Eg. Add License, View all License stc)

```
class LicenseManager:
  def __init__(self):
    self.licences=[]
```

The **__init__** method is called automatically when a new object of the class is created. By defining **self.licenses** in **__init__**, you make sure that every **LicenseManager** object starts with a clean, independent list of licenses.

Methods;

- ➔ **add_license()**: Adds a new license to the system.
  Appends license to licenses-list.

```
  def add_license(self,license):
    self.licenses.append(license)
```

- ➔ **view_licenses()**: Displays all licenses in the system.
  Checks if there's anything in the licenses-list- if yes, prints them using for loop.

```
  def view_license(self):
    if not self.licenses:
      print("No licenses")
    else:
      for license in self.licenses:
        print(license)
```

- ➔ **filter_by_status()**: Filters and displays licenses based on their status.
  Variable **found** stores the licenses whose status is the one provided in the parameter. If **found**-list isn't empty, licenses are printed using for loop which are in __str__ format.

```
  def filter_by_status(self,status):
    found=[license for license in self.licenses if
status.lower()==license.status.lower()]
    if not found:
      print(f"No {status} licenses")
    else:
      for license in found:
        print(license)
```

- ➔ **count_licenses()**: Counts the number of licenses for each product.
  Initialize dict **count**. Then check if **license** is already in **count**-dict, if yes-increment its count by 1. If not-initialize its value as 1. Then print licence's name along with its count using for loop.

```
  def count_license(self):
    count={}
    print("All the available licenses")
    for license in self.licenses:
      if license.product_name in count:
        count[license.product_name]+=1
      else:
        count[license.product_name]=1
    for product_name,product_count in count.items():
```

```
        print(f"{product_name} = {product_count}")
```

➔ **view_expired_licenses()**: Displays only the expired licenses.
Variable **expired** stores all the licenses that are expired using check_expired|() method. And If the **expired**-list is not empty, prints all the licenses using for loop.

```
  def view_expired(self):
    expired =[license for license in self.licenses if license.check_expired()]
    if expired:
      print("All expired licenses")
      for license in expired:
        print(license)
    else:
      print("No expired license")
```

---

system=LicenseManager() : This creates an object called system out of LicenseManager() class, and now it can use all the attributes & methods of LicenseManager() class.

Adding new licenses:

```
system=LicenseManager()
system.add_license(License("Adobe Premiere Pro", "APP-2024-ABC", "Active",
"2026-05-15"))
system.add_license(License("Microsoft Visual Studio", "VS-2024-ABC", "Used",
"2024-11-25"))
system.add_license(License("Autodesk Maya", "MAYA-2024-ABC", "Expired", "2023-08-30"))
system.add_license(License("Zoom Premium", "ZM-2024-ABC", "Active", "2025-04-10"))
system.add_license(License("Slack Professional", "SLACK-2024-ABC", "Used",
"2024-03-12"))
system.add_license(License("Tableau Professional", "TB-2024-ABC", "Expired",
"2023-12-01"))
system.add_license(License("Zoom Premium", "ZM-2024-ABC", "Active", "2025-04-10"))
```

Viewing all available license:

```
system.view_license()
```

```
Adobe Premiere Pro | APP-2024-ABC | Active | 2026-05-15
Microsoft Visual Studio | VS-2024-ABC | Used | 2024-11-25
Autodesk Maya | MAYA-2024-ABC | Expired | 2023-08-30
Zoom Premium | ZM-2024-ABC | Active | 2025-04-10
Slack Professional | SLACK-2024-ABC | Used | 2024-03-12
Tableau Professional | TB-2024-ABC | Expired | 2023-12-01
Zoom Premium | ZM-2024-ABC | Active | 2025-04-10
```

Using filter_by_status() method with parameter-”Active”

```
system.filter_by_status("Active")
```

```
Adobe Premiere Pro | APP-2024-ABC | Active | 2026-05-15
Zoom Premium | ZM-2024-ABC | Active | 2025-04-10
Zoom Premium | ZM-2024-ABC | Active | 2025-04-10
```

Viewing all the available licenses with their quantity:

```
system.count_license()
```

```
All the available licenses
Adobe Premiere Pro = 1
Microsoft Visual Studio = 1
Autodesk Maya = 1
Zoom Premium = 2
Slack Professional = 1
Tableau Professional = 1
```

Viewing all expired licenses:

```
system.view_expired()
```

```
All expired licenses
Microsoft Visual Studio | VS-2024-ABC | Used | 2024-11-25
Autodesk Maya | MAYA-2024-ABC | Expired | 2023-08-30
Slack Professional | SLACK-2024-ABC | Used | 2024-03-12
Tableau Professional | TB-2024-ABC | Expired | 2023-12-01
```

How functions from different class are able to interact with each other:
Composition is a design principle where a class contains instances of another class as its attributes or interacts with them.Instead of inheriting methods from the License class, the LicenseManager class works with License objects directly.
This approach allows the LicenseManager class to call methods on those objects.

Inheritance vs Composition
Inheritance should be used where there is a clear **‘is-a’ relationship** between two classes.
Eg.
A Dog is a Animal
A Car is a Vehicle

Here, **LicenseManager** is not a **License**, rather it manages a collection of **License** objects.
License: only handles individual license details.(eg. name, key, expiry date, status)
LicenseManager: only manages multiple licenses functions.(eg. Viewing all available license, adding license)
This separation endures clarity.
_____End_____