Python Typing Cheat Sheet: Any & Callable

Why Enum is Useful

1. Avoids 'magic values' (e.g., 1, "RUNNING") and makes code more readable.
2. Prevents typos since invalid enum access raises errors.
3. Editors offer autocomplete and validation with Enum members.
4. Safer comparisons using enum identity (not raw values).
5. More scalable and clean for managing fixed states or roles.
6. Works safely as dictionary keys or in logic checks.

Basic Usage

```
from enum import Enum


class Status(Enum):
    PENDING = 1
    RUNNING = 2
    COMPLETED = 3
```

Accessing Enum Values

```
print(Status.PENDING)       # Status.PENDING
print(Status.PENDING.name)  # 'PENDING'
print(Status.PENDING.value) # 1
```

Looping Through Enum

```
for status in Status:
    print(status.name, status.value)
```

String-Based Enum

```
class Color(Enum):
    RED = "red"
    GREEN = "green"
    BLUE = "blue"
```

Python Typing Cheat Sheet: Any & Callable

Enum Comparison

```
Status.PENDING == Status.PENDING   # True
Status.PENDING == 1                # False
```

Example Dictionary Use

```
class UserRole(Enum):
    ADMIN = "admin"
    GUEST = "guest"
    MODERATOR = "moderator"

discounts = {
    UserRole.ADMIN: 50,
    UserRole.GUEST: 10,
    UserRole.MODERATOR: 30
}
```