Day 42

Magic/Dunder Methods in Python:

What are Magic / Dunder Methods?

Magic methods (also called dunder methods) are special methods in Python:

- Start and end with double underscores: __method__
- Automatically called by Python
- Allow you to customize the behavior of objects

Example:

__init__, __str__, __add__, __len__

Why are Magic Methods Used?

They allow objects to:

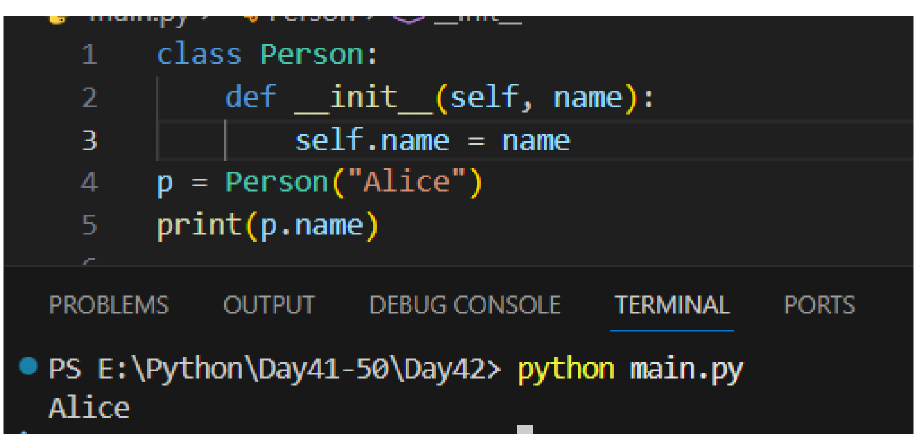
- Be printed nicely
- Behave like numbers
- Support operators (+, -, ==)
- Act like containers (len(), indexing)
- Work with loops
- Support comparisons

Categories of Magic Methods

Category	**Examples**
Object creation	__new__, __init__
String representation	__str__, __repr__
Operators	__add__, __sub__, __mul__
Comparison	__eq__, __lt__, __gt__
Container	__len__, __getitem__
Attribute access	__getattr__, __setattr__
Callable	__call__

Category	Examples
Context manager	__enter__, __exit__

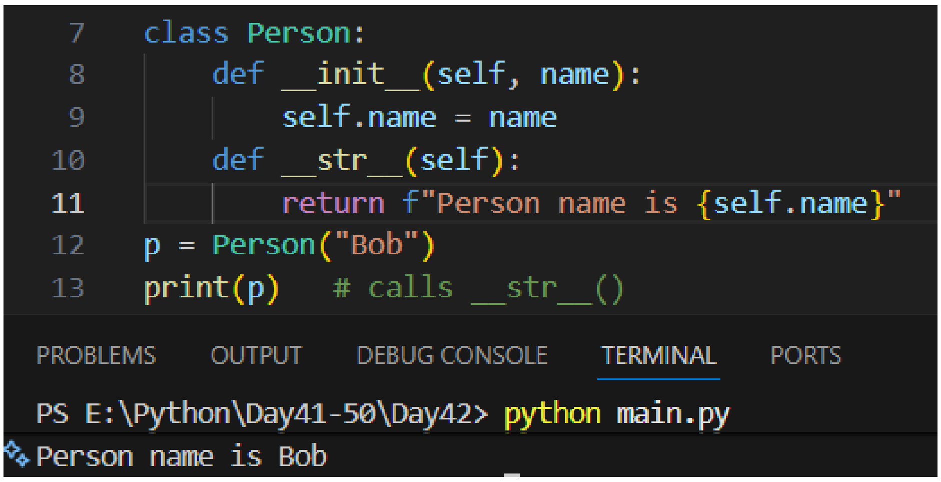
Example: __init__ – Object Initialization -> called automatically when an object is created.



```python
class Person:
    def __init__(self, name):
        self.name = name
p = Person("Alice")
print(p.name)
```

```
PS E:\Python\Day41-50\Day42> python main.py
Alice
```

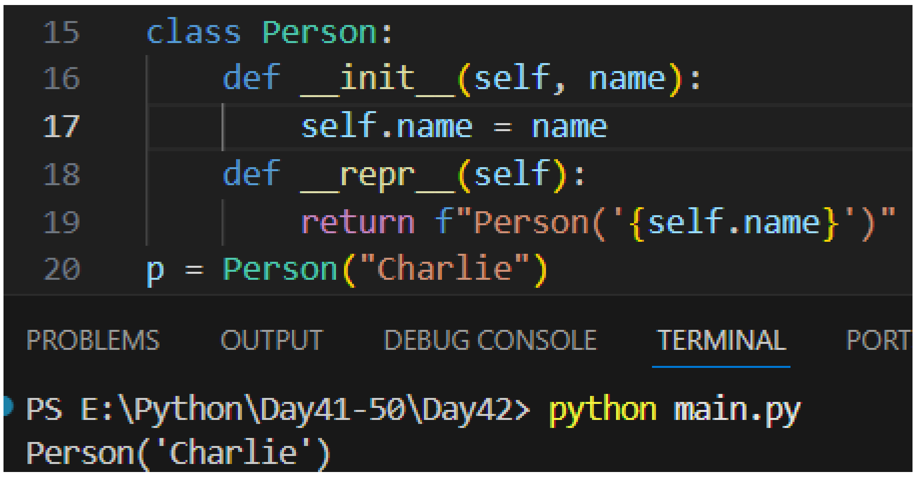
Example: __str__ – User-Friendly String Output



```python
class Person:
    def __init__(self, name):
        self.name = name
    def __str__(self):
        return f"Person name is {self.name}"
p = Person("Bob")
print(p)    # calls __str__()
```

```
PS E:\Python\Day41-50\Day42> python main.py
Person name is Bob
```

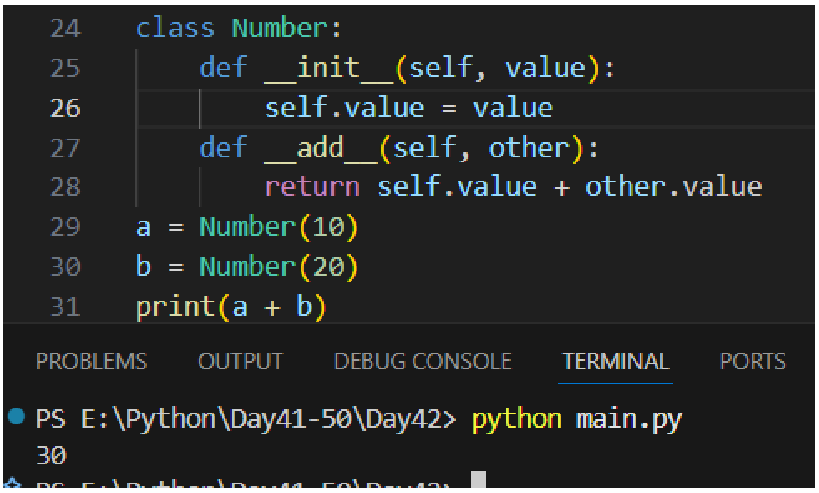
Example: __repr__ – Developer-Friendly Representation



```python
class Person:
    def __init__(self, name):
        self.name = name
    def __repr__(self):
        return f"Person('{self.name}')"
p = Person("Charlie")
```

```
PS E:\Python\Day41-50\Day42> python main.py
Person('Charlie')
```

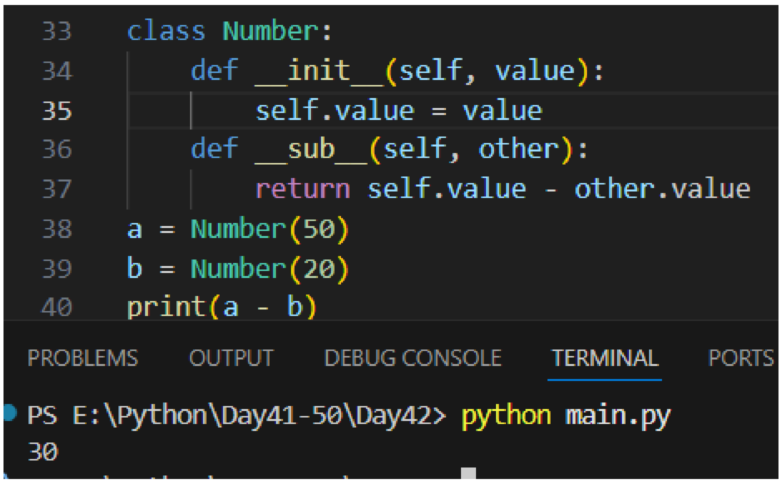
Example: __add__ – Addition Operator (+)



```python
class Number:
    def __init__(self, value):
        self.value = value
    def __add__(self, other):
        return self.value + other.value
a = Number(10)
b = Number(20)
print(a + b)
```

```
PS E:\Python\Day41-50\Day42> python main.py
30
```

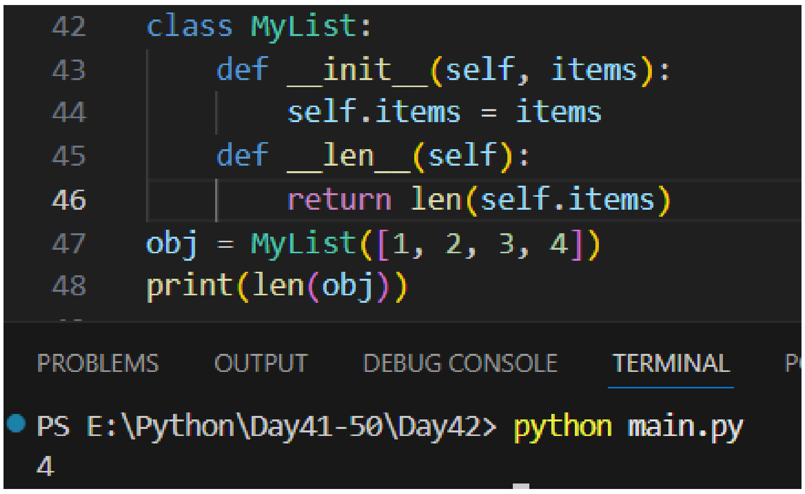
Example: __sub__ – Subtraction Operator (-)



```python
class Number:
    def __init__(self, value):
        self.value = value
    def __sub__(self, other):
        return self.value - other.value
a = Number(50)
b = Number(20)
print(a - b)
```

```
PS E:\Python\Day41-50\Day42> python main.py
30
```

Example: __len__ – Length of Object



```python
class MyList:
    def __init__(self, items):
        self.items = items
    def __len__(self):
        return len(self.items)
obj = MyList([1, 2, 3, 4])
print(len(obj))
```

```
PS E:\Python\Day41-50\Day42> python main.py
4
```

--The End--