Table of Contents

Chapter 14

Dunder/Magic Methods

Dunder or magic methods in Python are the methods having two prefix and suffix underscores in the method name. Dunder here means “Double Under (Underscores)”. These are commonly used for operator overloading. Few examples for magic methods are: __init__, __add__, __len__, __repr__ etc.
The __init__ method for initialization is invoked without any call, when an instance of a class is created, like constructors in certain other programming languages such as C++, Java, C#, PHP etc. These methods are the reason we can add two strings with ‘+’ operator without any explicit typecasting.
Ex.

```
class Employee:
    def __init__(self,name):
        self.name= name

    def __len__(self):
        return len(self.name)

    # def getLengthName(self):
    #     return len(self.name)

    def __str__(self):
        return f"The name of employee is str {self.name}"

    def __repr__(self):
        return f"The name of employee is repr {self.name}"

    # def getName(self):
    #     print(f"The name of employee is {self.name}")

    def __call__(self):
        print("I am a call method")

e = Employee("Omar")
print(e)
print(len(e)) # __len__
# print(e.getLengthName())
```

```
# print(str(e)) # __len__
# e.getName()
e() # I am a call method
```

Output:

```
# The name of employee is str Omar
# 4
# I am a call method
```

Some of the dunder methods:

Basic Customizations

- __new__(self) return a new object (an instance of that class). It is called before __init__ method.
- __init__(self) is called when the object is initialized. It is the constructor of a class.
- __del__(self) for del() function. Called when the object is to be destroyed. Can be used to commit unsaved data or close connections.
- __repr__(self) for repr() function. It returns a string to print the object. Intended for developers to debug. Must be implemented in any class.
- __str__(self) for str() function. Return a string to print the object. Intended for users to see a pretty and useful output. If not implemented, __repr__ will be used as a fallback.
- __bytes__(self) for bytes() function. Return a byte object which is the byte string representation of the object.
- __format__(self) for format() function. Evaluate formatted string literals like % for percentage format and ‘b’ for binary.
- __lt__(self, anotherObj) for < operator.
- __le__(self, anotherObj) for <= operator.
- __eq__(self, anotherObj) for == operator.
- __ne__(self, anotherObj) for != operator.
- __gt__(self, anotherObj)for > operator.
- __ge__(self, anotherObj)for >= operator.

Arithmetic Operators

- __add__(self, anotherObj) for + operator.
- __sub__(self, anotherObj) for – operation on object.
- __mul__(self, anotherObj) for * operation on object.
- __matmul__(self, anotherObj) for @ operator (numpy matrix multiplication).
- __truediv__(self, anotherObj) for simple / division operation on object.

- __floordiv__(self, anotherObj) for // floor division operation on object.

Type Conversion

- __abs__(self) make support for abs() function. Return absolute value.
- __int__(self) support for int() function. Returns the integer value of the object.
- __float__(self) for float() function support. Returns float equivalent of the object.
- __complex__(self) for complex() function support. Return complex value representation of the object.
- __round__(self, nDigits) for round() function. Round off float type to 2 digits and return it.
- __trunc__(self) for trunc() function of math module. Returns the real value of the object.
- __ceil__(self) for ceil() function of math module. The ceil function Return ceiling value of the object.
- __floor__(self) for floor() function of math module. Return floor value of the object.

Emulating Container Types

- __len__(self) for len() function. Returns the total number in any container.
- __getitem__(self, key) to support indexing. LIke container[index] calls container.__getitem(key)explicitly.
- __setitem__(self, key, value) makes item mutable (items can be changed by index), like container[index] = otherElement.
- __delitem__(self, key) for del() function. Delete the value at the index key.
- __iter__(self) returns an iterator when required that iterates all values in the container.