Python Class

- How to create classes and use them within Python
- How Object Oriented Concepts are applied within the language

1. Creating and instantiating classes
2. Inheritance
3. Instance Variables
4. Static methods
5. Class methods and
6. Other topics

Creating and instantiating classes

Classes are used to logically group data and functions in a way that's easy to reuse and easy to build upon if need to be

Data – attributes
Functions – methods

```
class Employee:
    pass

c = Employee()

print(c.__class__)
```

Example

```
class Employee:
    pass

emp1 = Employee()
emp2 = Employee()

emp1.fname = "Bala"
emp1.lname = "M"
emp1.salary = 30000

emp2.fname = "Ramesh"
emp2.lname = "K"
emp2.salary = 50000

print(emp1.__dict__)
print(emp2.__dict__)
```

```
print(emp1.salary)
```

Constructor / init method:

```
class Employee:
    def __init__(self, fname, lname, salary):
        self.fname = fname
        self.lname = lname
        self.salary = salary
        self.email = self.fname + '.' + self.lname + '@gmail.com'

emp1 = Employee("Bala", "Muthu", 30000)
emp2 = Employee("Ramesh", "Raghul", 50000)

print(emp1.email)
```

```
print(emp1.__dict__)
```

Output:

{'fname': 'Bala', 'lname': 'Muthu', 'salary': 30000, 'email': 'Bala.Muthu@gmail.com'}

```
class Employee:
    def __init__(self, fname, lname, salary):
        self.fname = fname
        self.lname = lname
        self.salary = salary
        self.email = self.fname + '.' + self.lname + '@gmail.com'

emp1 = Employee("Bala", "Muthu", 30000)
emp2 = Employee("Ramesh", "Raghul", 50000)

print(emp1.__dict__)

print(emp1.__sizeof__())

print(f'{emp1.fname} {emp1.lname}')
```

Output:

{'fname': 'Bala', 'lname': 'Muthu', 'salary': 30000, 'email': 'Bala.Muthu@gmail.com'}
32
Bala Muthu

```python
class Employee:
    def __init__(self, fname, lname, salary):
        self.fname = fname
        self.lname = lname
        self.salary = salary
        self.email = self.fname + '.' + self.lname + '@gmail.com'

    def fullname(self):
        return f"{self.fname} {self.lname}"

emp1 = Employee("Bala", "Muthu", 30000)
emp2 = Employee("Ramesh", "Raghul", 50000)

print(emp1.fullname())
print(emp2.fullname())
```

Output:

Bala Muthu
Ramesh Raghul

```python
class Employee:
    def __init__(self, fname, lname, salary):
        self.fname = fname
        self.lname = lname
        self.salary = salary
        self.email = self.fname + '.' + self.lname + '@gmail.com'

    def fullname(self):
        return f"{self.fname} {self.lname}"

emp1 = Employee("Bala", "Muthu", 30000)
emp2 = Employee("Ramesh", "Raghul", 50000)

print(emp1.fullname())
print(Employee.fullname(emp1))
```

Output:

Bala Muthu
Bala Muthu

Class Variables

Class Variables can be accessed by all instances

```
class Employee:
    raise_amt = 1.04
    def __init__(self, fname, lname, salary):
        self.fname = fname
        self.lname = lname
        self.salary = salary
        self.email = self.fname + '.' + self.lname + '@gmail.com'

    def fullname(self):
        return f"{self.fname} {self.lname}"

    def pay_raise(self):
        #self.salary = int(self.salary * self.raise_amt)
        self.salary = int(self.salary * Employee.raise_amt)

emp1 = Employee("Bala", "Muthu", 30000)
emp2 = Employee("Ramesh", "Raghul", 50000)

print(emp1.salary)
emp1.pay_raise()
print(emp1.salary)
```

Output:

30000
31200

Note: When you look at the method, “pay_raise”, the variable raise_amt can be accessed by using either with Class Name or self (“method”)

If it is accessed with self.raise_amt, the program looks for instance variable, if the instance doesn’t have the variable, then it looks for class variable

This can be checked by using the namespace.

```
print(emp1.__dict__)
print(Employee.__dict__)
```

Output:

```
{'fname': 'Bala', 'lname': 'Muthu', 'salary': 31200, 'email': 'Bala.Muthu@gmail.com'}
```

```
{'__module__': '__main__', 'raise_amt': 1.04, '__init__': <function Employee.__init__ at
0x000002223E17EEA0>, 'fullname': <function Employee.fullname at
0x000002223E18E378>, 'pay_raise': <function Employee.pay_raise at
0x000002223E18E400>, '__dict__': <attribute '__dict__' of 'Employee' objects>,
'__weakref__': <attribute '__weakref__' of 'Employee' objects>, '__doc__': None}
```

When you look at the output closely, the variable is available only at the "Employee (class)" namespace.

```
emp1.raise_amt = 1.05
print(emp1.__dict__)
print(Employee.__dict__)
```

Output:

```
{'fname': 'Bala', 'lname': 'Muthu', 'salary': 31200, 'email': 'Bala.Muthu@gmail.com',
'raise_amt': 1.05}
{'__module__': '__main__', 'raise_amt': 1.04, '__init__': <function Employee.__init__ at
0x0000028B0A86E378>, 'fullname': <function Employee.fullname at
0x0000028B0A86E400>, 'pay_raise': <function Employee.pay_raise at
0x0000028B0A86E488>, '__dict__': <attribute '__dict__' of 'Employee' objects>,
'__weakref__': <attribute '__weakref__' of 'Employee' objects>, '__doc__': None}
```

Note: Now when you look at the namespace, both Class and Method namespace have different values.

```
class Employee:
    raise_amt = 1.04
    no_of_employees = 0

    def __init__(self, fname, lname, salary):
        self.fname = fname
        self.lname = lname
        self.salary = salary
        self.email = self.fname + '.' + self.lname + '@gmail.com'
        Employee.no_of_employees += 1

    def fullname(self):
        return f"{self.fname} {self.lname}"

    def pay_raise(self):
        self.salary = int(self.salary * self.raise_amt)

emp1 = Employee("Bala", "Muthu", 30000)
emp2 = Employee("Ramesh", "Raghul", 50000)
```

```
print(Employee.no_of_employees)
```

Output:

```
2
```