Classes by Mrittika Megaraj

Python Objects and Classes

Python classes: A class is a blueprint or a template for creating objects.

In [1]:
```
#name of the class
class Car:
    name = ""   #defining class
    gear = 0
```

Python Objects: An object is called an instance of a class

For example: Car is class . So, we take Car1,Car2 as objects

In [2]:
```
# create class
class Car:
    name = ""
    gear = 0

# create objects of class
Car1 = Car()
```

Example Python program of Class and Objects

In [3]:
```
# define a class
class Car:
  name = ""
  gear = 0

# create object of class
Car1 = Car()

# access attributes and assign new values
Car1.gear = 11
Car1.name = "Mountain Bike"

print(f"Name: {Car1.name}, Gears: {Car1.gear} ")
```

```
Name: Mountain Bike, Gears: 11
```

Python Constructors

In [4]:
```
class person:
  def __init__(self, name, age):
    self.name = name
    self.age = age
person1 = person("Aswin",20)    #creating a person object
print(f"Name: {person1.name}")
print(f"Age: {person1.age}")
```

```
Name: Aswin
Age: 20
```

Destructor

In [5]:
```
class MyClass:
    def __init__(self, name):
        self.name = name

    def __del__(self):
        print(f"{self.name} is being destroyed!")

# Creating objects
obj1 = MyClass("Object 1")
obj2 = MyClass("Object 2")

# Deleting references to objects
del obj1
del obj2
```

```
Object 1 is being destroyed!
Object 2 is being destroyed!
```

Class and Static Variable

In [6]:
```
#Class and Static Variables:
class Voter:
    # Class variable for voting age
    voting_age = 18

    # Class variable to keep track of the total number of voters
    num_voters = 0
```

In [7]:
```python
#Class and Static Methods:
class Voter:
    voting_age = 18
    num_voters = 0

    def __init__(self, age):
        self.age = age
        Voter.num_voters += 1

    @staticmethod
    def is_eligible(age):
        return age >= Voter.voting_age

    @classmethod
    def get_total_voters(cls):
        return cls.num_voters

    def check_eligibility(self):
        if Voter.is_eligible(self.age):
            print(f"You are eligible to vote at age {self.age}.")
        else:
            print(f"Sorry, you are not eligible to vote at age {self.age}.'


# Create voter instances
voter1 = Voter(20)
voter2 = Voter(16)
voter3 = Voter(25)

# Check eligibility and total voters
voter1.check_eligibility()  # Output: You are eligible to vote at age 20.
voter2.check_eligibility()  # Output: Sorry, you are not eligible to vote a
voter3.check_eligibility()  # Output: You are eligible to vote at age 25.
print(f"Total voters: {Voter.get_total_voters()}")  # Output: Total voters:
```

```
You are eligible to vote at age 20.
Sorry, you are not eligible to vote at age 16.
You are eligible to vote at age 25.
Total voters: 3
```

Class vs Static Variables and Methods

1. Class and Static Variables

In [9]:
```python
class Account:
    '''Create Account Class'''
    count = 0
    acct_type = 'Saving'

    def __init__(self, acct:int, name:str) -> None:
        self.acct = acct
        self.name = name
        Account.count += 1
        # print('Object created..')
    def __str__(self) -> str:
        st = f'Count:{Account.count}, Type:{Account.acct_type}'
        inst = f'Acct No.:{self.acct}, Name:{self.name}'
        return f'{st},\n{inst}'

acct1 = Account(11, 'Snehal')
# print(acct1.__getstate__())
# print(acct1.count,acct1.acct,acct1.name,acct1.acct_type)
print(acct1)
acct2 = Account(12, 'Kal')
# print(acct2.__getstate__())
print(acct2)

acct3 = Account(13, 'Shiv')
# print(acct3.__getstate__())
print(acct3)

acct4 = Account(14, 'Shubh')
# print(acct4.__getstate__())
print(acct4)

print('Accounts:', Account.count)
```

```
Count:1, Type:Saving,
Acct No.:11, Name:Snehal
Count:2, Type:Saving,
Acct No.:12, Name:Kal
Count:3, Type:Saving,
Acct No.:13, Name:Shiv
Count:4, Type:Saving,
Acct No.:14, Name:Shubh
Accounts: 4
```

2. Class and Static Methods

In [10]:
```python
class Account:
    '''Create Account Class'''
    count = 0
    acct_type = 'Saving'
    # Instance method
    def __init__(self, acct:int, name:str) -> None:
        self.acct = acct
        self.name = name
        Account.count += 1
    def set_name(self: Account, name:str) -> None:
        self.name = name
    @classmethod
    def get_count(cls) -> int:
        return cls.count
    @staticmethod
    def get_type() -> str:
        return Account.acct_type
acct1 = Account(1, 'Snehal')
print(Account.get_count())
Account.set_name(acct1, name='Shubh')
print(acct1.get_count())

print(Account.get_type())
print(acct1.get_type())
```

```
1
1
Saving
Saving
```

In [ ]: