In [1]:
```
class computer:
    def config():
        print('i5,telugu')
com1=computer
com1.config()
```

```
i5,telugu
```

In [86]:
```
# Python3 program to
# demonstrate instantiating
# a class
class Dog:

    # A simple class
    # attribute
    attr1 = "mammal"
    attr2 = "dog"

    # A sample method
    def fun(self):
        print("I'm a", self.attr1)
        print("I'm a", self.attr2)


# Driver code
# Object instantiation
Rodger = Dog()

# Accessing class attributes
# and method through objects
print(Rodger.attr1)
Rodger.fun()
```

```
mammal
I'm a mammal
I'm a dog
```

In [5]:
```
# Sample string
my_string = "abcdef"

# Replace the first two characters with the last two characters
my_string = my_string[-2:] + my_string[2:]

print(my_string)  # Output: "efcdef"
```

```
efcdef
```

In [6]:
```
a='hemanth'
string=a[-2:]+a[2]
print(string)
```

```
thmanth
```

In [7]:
```
# Sample string
my_string = "hemanth"

# Interchange the first two and last two elements
interchanged_string = my_string[-2:] + my_string[2:-2] + my_string[:2]

print(interchanged_string)  # Output: "efbcda"
```

```
thmanhe
```

In [15]:
```
class computer:
    def __init__(self,cpu,ram):
        self.cpu=cpu
        self.ram=ram
    def config(self):
        print(self.cpu,self.ram)
com1=computer('i6',55)
com2=computer('i3',33)
com1.config()
com2.config()
```

```
i6 55
i3 33
```

In [52]:
```
class car:
    def __init__(self,mil,com):
        self.mil=mil
        self.com=com
c1=car('BMW',11)
print(c1.com,c1.mil)
```

```
11 BMW
11 BMW
```

In [26]:
```
class car:
    def __init__(self):
        self.mil=10
        self.com='BMW'
c1=car()
c2=car()
c1.mil=8
print(c1.com,c1.mil)
print(c2.com,c2.mil)
```

```
BMW 8
BMW 10
```

In [ ]:
```
class car:
    def __init__(self,mil,com):
        self.mil=mil
        self.com=com
c1=car('BMW',11)
print(c1.com,c1.mil)
print(c1.com,c1.mil)
```

In [74]:
```
class student:
    def __init__(self,m1,m2,m3):
        self.m1=m1
        self.m2=m2
        self.m3=m3
    def avg(self):
        return (self.m1+self.m2+self.m3)/3
s1=student(34,22,34)
s2=student(22,32,22)
print(s1.avg())
print(s2.avg())
```

```
30.0
25.333333333333332
```

In [78]:
```
class student:
    def __init__(self,m1,m2,m3):
        self.m1=m1
        self.m2=m2
        self.m3=m3
    def show(self):
        return (self.m1,self.m2,self.m3)
s1=student(34,22,34)
s2=student(22,32,22)
print(s1.show())
print(s2.show())
```

```
(34, 22, 34)
(22, 32, 22)
```

In [83]:
```
class student:
    def __init__(self,name,rollno):
        self.name=name
        self.rollno=rollno
    def show(self):
        print(self.name,self.rollno)

s1=student('Hemanth',1)
s2=student('venky',23)
s1.show()
s2.show()
```

```
Hemanth 1
venky 23
```

In [92]:
```python
# A Python program to demonstrate inheritance
class Person(object):

# Constructor
    def __init__(self, name, id):
        self.name = name
        self.id = id

# To check if this person is an employee
    def Display(self):
        return(self.name, self.id)


# Driver code
emp = Person('satyam', 102) # An Object of Person
print(emp.Display())
```

```
('satyam', 102)
```

In [55]:
```python

class Dog:
    attr1 = "mammal"
    attr2 = "dog"

    # A sample method
    def fun(self):
        print("I'm a", self.attr1)
        print("I'm a", self.attr2)


# Driver code
# Object instantiation
Rod= Dog()

# Accessing class attributes
# and method through objects
print(Rod.attr1)
Rod.fun()
```

```
mammal
```

In [38]:
```
# A Python program to demonstrate inheritance
class Person(object):

# Constructor
    def __init__(self, name, id):
        self.name = name
        self.id = id

# To check if this person is an employee
    def Display(self):
        print(self.name, self.id)


# Driver code
emp = Person("Satyam", 102) # An Object of Person
emp.Display()
```

```
Satyam 102
```

In [40]:
```
class student:
    def __init__(self):
        self.m1=11
        self.m2=10
s1=student()
s2=student()
print(s1.m1)
print(s2.m2)
```

```
11
10
```

In [56]:
```
# Get the number from the user
number = input("Enter a number: ")

# Initialize the sum of digits
total = 0

# Iterate through each character in the number
for digit_char in number:
    # Convert the character to an integer and add it to the total
    total += int(digit_char)

# Print the sum of the digits
print("Sum of digits:", total)
```

```
Enter a number: 23
Sum of digits: 5
```

In [60]:
```
n='123'
total=0
for i in n:
    total=total+int(i)
print(total)
```

```
6
```

In [63]:
```
class GFG:
    def __init__(self, name, company):
        self.name = name
        self.company = company

    def __str__(self):
        return f"My name is {self.name} and I work in {self.company}."


my_obj = GFG("John", "GeeksForGeeks")
print(my_obj)
```

```
My name is John and I work in GeeksForGeeks.
```

In [95]:
```
# Sample class with init method
class Person:

    # init method or constructor
    def __init__(self, name):
        self.name = name

    # Sample Method
    def say_hi(self):
        print('Hello, my name is', self.name)


p = Person('Nikhil')
p.say_hi()
```

```
Hello, my name is Nikhil
```

In [96]:
```
# Python3 program to show that the variables with a value
# assigned in the class declaration, are class variables and
# variables inside methods and constructors are instance
# variables.

# Class for Dog


class Dog:

    # Class Variable
    animal = 'dog'

    # The init method or constructor
    def __init__(self, breed, color):

        # Instance Variable
        self.breed = breed
        self.color = color


# Objects of Dog class
Rodger = Dog("Pug", "brown")
Buzo = Dog("Bulldog", "black")

print('Rodger details:')
print('Rodger is a', Rodger.animal)
print('Breed: ', Rodger.breed)
print('Color: ', Rodger.color)

print('\nBuzo details:')
print('Buzo is a', Buzo.animal)
print('Breed: ', Buzo.breed)
print('Color: ', Buzo.color)

# Class variables can be accessed using class
# name also
print("\nAccessing class variable using class name")
print(Dog.animal)
```

```
Rodger details:
Rodger is a dog
Breed:  Pug
Color:  brown

Buzo details:
Buzo is a dog
Breed:  Bulldog
Color:  black

Accessing class variable using class name
dog
```

In [103]:
```
class Dog:
    animal='dog'
    def __init__(self,breed,color):
        self.breed=breed
        self.color=color
rod=Dog("PuG","Brown")
buz=Dog("Bull","black")
print(rod.animal)
print(rod.breed)
print(Dog.animal)
```

```
dog
PuG
dog
```

In [104]:
```
class Dog:
    def __init__(self,breed,color):
        self.breed=breed
        self.color=color
rod=Dog("PuG","Brown")
buz=Dog("Bull","black")
print(rod.breed)
print(rod.color)
```

```
PuG
Brown
```

In [112]:
```
class Dog:
    animal='dog'
    def __init__(self,breed):
        self.breed=breed
    def get(self):
        print(self.breed)

m=Dog("Brown")
m.get()
```

```
Brown
```

In [117]:
```
class Person:
    def __init__(self, name, sex, profession):
        # data members (instance variables)
        self.name = name
        self.sex = sex
        self.profession = profession

    # Behavior (instance methods)
    def show(self):
        print('Name:', self.name, 'Sex:', self.sex, 'Profession:', self.profes

    # Behavior (instance methods)
    def work(self):
        print(self.name, 'working as a', self.profession)

# create object of a class
jessa = Person('Jessa', 'Female', 'Software Engineer')

# call methods
jessa.show()
jessa.work()
```

```
Name: Jessa Sex: Female Profession: Software Engineer
Jessa working as a Software Engineer
```

In [127]:
```
class addition:
    def __init__(self,f,s):
        self.first=f
        self.second=s
    def display(self):
        print(self.first)
        print(self.second)
    def calculate(self):
        print(self.second)
m=addition(100,200)
m2=addition(222,333)
m.display()
m2.calculate()
```

```
100
200
333
```

In [120]:
```
class MyClass:
    def __init__(self, name=None):
        if name is None:
            print("Default constructor called")
        else:
            self.name = name
            print("Parameterized constructor called with name", self.name)

    def method(self):
        if hasattr(self, 'name'):
            print("Method called with name", self.name)
        else:
            print("Method called without a name")

# Create an object of the class using the default constructor
obj1 = MyClass()

# Call a method of the class
obj1.method()

# Create an object of the class using the parameterized constructor
obj2 = MyClass("John")

# Call a method of the class
obj2.method()
```

```
Default constructor called
Method called without a name
Parameterized constructor called with name John
Method called with name John
```

In [129]:
```
class employee:
    def __init__(self):
        print('employee created')
    def __del__(self):
        print('destructor deleted')
obj=employee()
del obj
```

```
employee created
destructor deleted
```

In [132]:
```
class employee:
    def __init__(self):
        print("employee created")
    def __del__(self):
        print("destructor deleted")
def create():
    print('making')
    obj=employee()
    print('function')
    return obj
print('calling')
ob=create()
print('program')
```

```
calling
making
employee created
function
destructor deleted
program
```

In [133]:
```
class employee:
    def __init__(self):
        print("employee created")
    def show(self):
        print("hello")
        print("world")
m=employee()
m.show()
```

```
employee created
hello
world
```

In [ ]:
```
class recursive:
    def __init__(self,n):
        self.n=n
        print("recursive function",n)
    def run(self,n=None):
        if n is None:
            n=self.n
        if n<=0:
            return
```