```
"""
Polymorphism
Poly = many
morphism = forms

Compile Time:
    Overloading:
        Method Overloading:
            Creating multiple m with the exact same name
            by changing the number of parameters passed to the method.

            Method Overloading is not supported in Python.
            But, we can mimic them with the use of default arguments.
        Operator Overloading:
            Same operator works in different forms for different data types.

Runtime:
    Overriding:
        Method Overriding:
            If we declare the same method which is present in parent class,

"""


class Mathematics:
    def add_numbers(self, a, b):
        print("Adding 2 numbers")
        print(a + b)

    def add_numbers(self, a, b, c):
        print("Adding 3 Numbers")
        print(a + b + c)

maths = Mathematics()
# maths.add_numbers(10, 20)
maths.add_numbers(10, 20, 30)


class A:
    def method(self):
        print("Inside Parent")

class B(A):
    def method(self):
        print("Inside Child")


a = A()
a.method()

b = B()
b.method()


# Mimic Method Overloading
class Mathematics:
    def add_numbers(self, a, b):
        print("Adding 2 numbers")
        print(a + b)
```

```
    def add_numbers(self, a=0, b=0, c=0):
        print("Adding 3 Numbers")
        print(a + b + c)

maths = Mathematics()
maths.add_numbers(10, 20)
maths.add_numbers(10, 20, 30)
maths.add_numbers(a=10, c=30)

# Examples of Method Overloading
print(len("Python")) # Number of Characters
print(len([1, 2, 3, 4, 5])) # Number of elements
print(len({1: "One", 2: "Two"})) # Number of key-value pairs

# Operator Overloading
# Same operator works in different forms for different data types.

# Examples of Operator Overloading
a = "Hello"
b = "World"
c = a + ' ' + b
print(c) # Hello World

a = 10
b = 20
c = a + b
print(c) # 30

a = [1, 2, 3, 4]
b = [5, 6, 7, 8]
c = a + b
print(c)

class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

p1 = Person("Person1", 20)
p2 = Person("Person2", 25)

# print(p1 < p2)
print(dir(p1))
print(p2.__dir__())


class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __repr__(self):
        return f"Hello!!!\n My Name is {self.name}. Nice to meet you..."

    def __lt__(self, other):
        return self.age < other.age

    def __ge__(self, other):
```

```
        return self.age >= other.age


p1 = Person("Person1", 20)
p2 = Person("Person2", 25)
print(p1)
print(p2)

print(p1 < p2)
print(p1 >= p2)
```