Polymorphism Overview

- Poly - Many, Morphs - forms
- Polymorphism refers to the ability of an object with the same name to carry out different functionality altogether.

Overriding v/s Overloading

- Overriding is a concept in which if objects are with same name then the latest one will be retained.
- Overloading is a concept in which if the functions/ methods with same name have different arguments, each can be called with respect to their arguments.

```
# operator overloading

print(2 + 3)      # 5
print("hai" + "hello")     # haihello

#######################################################################
# method overloading and method overriding

def add(a, b):
  return a + b

def add(a, b, c):
  return a + b + c


# print(add(1, 2))          # TypeError
# print(add(1, 2, 3))       # 6


#######################################################################
```

```python
# inheritance

class Banking:

  def __init__(self, name, balance):
    self.name = name
    self.balance = balance

  def deposit(self, amount):
    self.balance += amount
    print(f"deposited amount is: {amount}")


class SavingsAccount(Banking):

  def deposit(self, amount):
    if amount >= 1000:
      super().deposit(amount)


# b = Banking("steve", 20000)
# b.deposit(500)
#
# s = SavingsAccount("John", 10000)
# s.deposit(100)

#########################################################################
# method overloading using default arguments

def add(a, b, c=0):
  return a + b + c


print(add(1, 2))
print(add(12, 34, 6))
```

```
##########################################################################
# polymorphism in inbuilt function

# len()

print(len("hello"))          # counting the number of characters

print(len(["hello", "hai"]))    # counting the number of values
print(len(("hello", "hai")))    # counting the number of values

print(len({1, 2, 3, 4}))       # counting the number of keys
print(len({"a": 1, "b": 2}))    # counting the number of keys
```