|  |
| --- |
| Polymorphism  Polymorphism means having many forms  In simple words polymorphism allows us to perform same action in many different ways.  Note: Advantage of method overriding  '''The main advantage of method overriding is that the class can give its own specific implementation to an inherited method without even modifying the parent class code.''' |

|  |
| --- |
| Polymorphism in Build-in function or Function Polymorphism  The Build-in function length() calculates the length of an  object depending on its type.  If object is string it returns count of characters  If object is list, it returns count of items in list  Note: The len() method treats an 'object' as per it`s type''' |

|  |
| --- |
| # Finding the length of Items/Elements # Notes: Find's the Items in List names = ["nameOne", "nameTwo", "nameThree"] print(len(names)) # 3  # Find the length of the String # Note: Find's the number of characters in String mobile = "Samsung Note 10" print(len(mobile)) # 15  # Find the length of the Dictionary # Note: Dictionary, finds the length of keys rollNo = {101: "userOne", 102: "userTwo"} print(len(rollNo)) # 2 |

|  |
| --- |
| # Polymorphism in + Operator # The + Operator have multiple uses  # + Operator used for concatenation s1 = "Sajeed" s2 = "Kiran" print(s1+" "+ s2) # Sajeed Kiran  # + Operator is used for addition n1 = 10 n2 = 20 print(n1+n2) # 30 |

|  |
| --- |
| # Polymorphism in Class Methods class Parent(object):  def d1(self):  print('d1 Function Parent')  class Child(object):  def d1(self):  print('d1 Function Child')  p = Parent() c = Child()  # Now, Pack the both objects in Tuple. This is due to polymorphism for i in (p, c):  i.d1()  d1 Function Parent  d1 Function Child |

|  |
| --- |
| # Method Overriding # Polymorphism and Single Level Inheritance class Parent: # Is subclassed by:   Child  def d1(self):  print('d1 Function Parent')  class Child(Parent): # Overrides method in Parent  def d1(self):  print('d1 Function Child')  c = Child() c.d1() # Here, Child method overrides parent method  d1 Function Child |

|  |
| --- |
| # Method overriding # Polymorphism and Multiple Inheritance class GrandParent: # Is subclassed by:   Child  def d1(self): # Is overridden in:   Child  print("d1 method grandparent")  class Parent: # Is subclassed by:   Child  def d1(self): # Is overridden in:   Child  print("d1 method parent")  class Child(GrandParent, Parent):  def d1(self): # Overrides method in GrandParent  print("d1 method child")  g = GrandParent() g.d1() # d1 method grandparent p = Parent() p.d1() # d1 method parent c = Child() c.d1() # d1 method child  # Now, Pack the both objects in Tuple. This is due to polymorphism for i in (g,p,c):  i.d1()  d1 method grandparent  d1 method parent  d1 method child  d1 method grandparent  d1 method parent  d1 method child |

|  |
| --- |
| # Method overriding # Single Level Inheritance and Hierarchical  class Bank: # Is subclassed by: SBI AXIS ICICI  def getRateOfInterest(self): # Is overridden in: SBI AXIS ICICI  return 0 class SBI(Bank):  def getRateOfInterest(self): # Overrides method in Bank  return 6 class AXIS(Bank):  def getRateOfInterest(self): # Overrides method in Bank  return 7 class ICICI(Bank):  def getRateOfInterest(self): # Overrides method in Bank  return 8 s = SBI() print(s.getRateOfInterest()) # 6 a = AXIS() print(a.getRateOfInterest()) # 7 i = ICICI() print(i.getRateOfInterest()) # 8 |