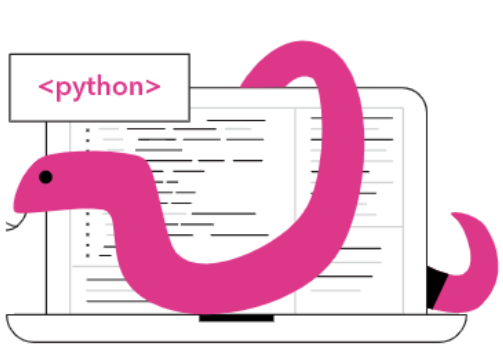
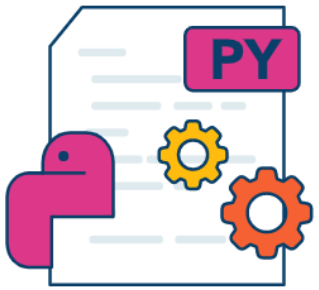
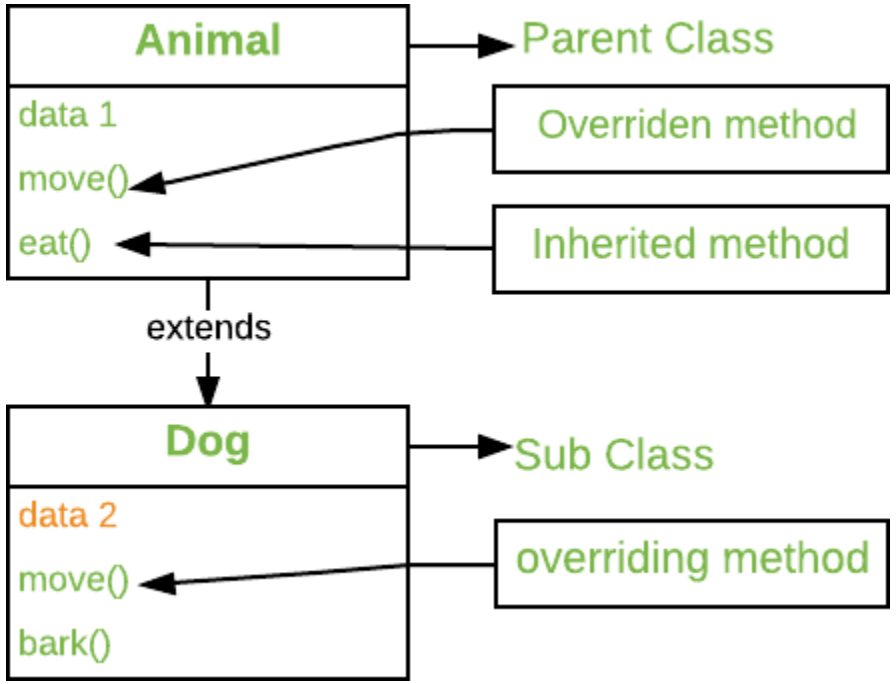
POLYMORPHISM





DR. VEENA R S
ASSOCIATE PROFESSOR
DEPARTMENT OF ISE
DSATM, BENGALURU

Method Overriding



- Method overriding is an ability of any object-oriented programming language that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super- classes or parent classes. When a method in a subclass has the same name, same parameters or signature and same return type(or sub-type) as a method in its super-class, then the method in the subclass is said to **override** the method in the super-class.

- The version of a method that is executed will be determined by the object that is used to invoke it. If an object of a parent class is used to invoke the method, then the version in the parent class will be executed, but if an object of the subclass is used to invoke the method, then the version in the child class will be executed. In other words, it is the type of the object being referred to (not the type of the reference variable) that determines which version of an overridden method will be executed.

8. a) Write a python program to find the whether the given input is palindrome or not (for both string and integer) using the concept of polymorphism and inheritance.

```
class Palindrome:


        def is_palindrome(self, input_str):
                return input_str == input_str[::-1]
class IntegerPalindrome(Palindrome):
    def is_palindrome(self, input_int):
        input_str=str(input_int)
        return super().is_palindrome(input_str)
```

```
# Testing the program


palindrome=Palindrome()
print(palindrome.is_palindrome("racecar")) # Output: True
print(palindrome.is_palindrome("hello"))      # Output: False

int_palindrome=IntegerPalindrome()
print(int_palindrome.is_palindrome(121))    # Output: True
print(int_palindrome.is_palindrome(123))    # Output: False
```

8. a) Write a python program to find the whether the given input is palindrome or not (for both string and integer) using the concept of polymorphism and inheritance.

```
class PaliStr:

def __init__(self):
        self.isPali = False

    def chkPalindrome(self, myStr):
        if myStr == myStr[::-1]:
            self.isPali = True
        else:
            self.isPali = False
        return self.isPali
```

```
class PaliInt(PaliStr):

    def __init__(self):
        self.isPali = False

    def chkPalindrome(self, val):
        temp = val
        rev = 0
        while temp != 0:
            dig = temp % 10
            rev = (rev*10) + dig
            temp = temp //10

        if val == rev:
            self.isPali = True
        else:
            self.isPali = False
```

```
st = input("Enter a string : ")

stObj = PaliStr()

if stObj.chkPalindrome(st):
    print("Given string is a Palindrome")
else:
    print("Given string is not a Palindrome")

val = int(input("Enter a integer : "))

intObj = PaliInt()
if intObj.chkPalindrome(val):
   print("Given integer is a Palindrome")
else:
    print("Given integer is not a Palindrome")
```