Python Inheritance

June 6, 2024

0.1 Python Inheritance

Dr.Labeed Al-Saad

Inheritance allows us to define a class that inherits all the methods and properties from another class.

**Parent class is the class being inherited from, also called base class.

**Child class is the class that inherits from another class, also called derived class.

0.2 Create a Parent Class

Any class can be a parent class, so the syntax is the same as creating any other class:

Example:

Create a class named Person, with firstname and lastname properties, and a printname method:

[1]:
```
class Person:
  def __init__(self1, fname, lname, age):
    self1.firstname = fname
    self1.lastname = lname
    self1.age = age

  def printname_age(self1):
    print(self1.firstname, self1.lastname, self1.age,"year")

#Use the Person class to create an object, and then execute the printname_age␣
 ↪method:

x = Person("John", "Doe", 40)
x.printname_age()
```

```
John Doe 40 year
```

0.3 Create a Child Class

To create a class that inherits the functionality from another class, send the parent class as a parameter when creating the child class:

Example:

Create a class named Student, which will inherit the properties and methods from the Person class:

[ ]:
```
class Student(Person):
  pass
```

**Note: Use the pass keyword when you do not want to add any other properties or methods to the class.

Now the Student class has the same properties and methods as the Person class.

Example:

Use the Student class to create an object, and then execute the printname method:

[2]:
```
class Person:
  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

  def printname(self):
    print(self.firstname, self.lastname)

class Student(Person):
  pass

x = Student("Mike", "Olsen")
x.printname()
```

```
Mike Olsen
```

0.4 Add the init() Function

So far we have created a child class that inherits the properties and methods from its parent.

We want to add the **init**() function to the child class (instead of the pass keyword).

Note: The **init() function is called automatically every time the class is being used to create a new object.

Example:

Add the **init**() function to the Student class:

[ ]:
```
class Student(Person):
  def __init__(self, fname, lname):
    #add properties etc.
```

When you add the **init**() function, the child class will no longer inherit the parent's **init**() function.

Note: The child's **init() function overrides the inheritance of the parent's **init**() function.

To keep the inheritance of the parent's **init**() function, add a call to the parent's **init**() function:

Example:

[3]:
```
class Person:
  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

  def printname(self):
    print(self.firstname, self.lastname)

class Student(Person):
  def __init__(self, fname, lname):
    Person.__init__(self, fname, lname)

x = Student("Mike", "Olsen")
x.printname()
```

```
Mike Olsen
```

**Also we can added new properities inherited class properities

Example:

we'll add age properity to the child calss, which is not found in the inhrited propperitis

[4]:
```
class Person:
  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

  def printname(self):
    print(self.firstname, self.lastname)

class Student(Person):
  def __init__(self, fname, lname, age):
    Person.__init__(self, fname, lname)
    self.age = age    #here we added new properity

  def printage(self):   #this function to print the new properity
    print(self.age, "year")

x = Student("Mike", "Olsen", 40)
x.printname()
x.printage()  #call for new properity print function
```

```
Mike Olsen
40 year
```

Other example to add new properity:

[6]:
```
class Person:
  def __init__(self, fname, lname):
```

```
    self.firstname = fname
    self.lastname = lname

  def printname(self):
    print(self.firstname, self.lastname)

class Student(Person):
  def __init__(self, fname, lname):
    super().__init__(fname, lname)
    self.graduationyear = 2019    #new properity

x = Student("Mike", "Olsen")
print(x.graduationyear) #printing of new properity
```

```
2019
```

0.5 Use the super() Function

Python also has a super() function that will make the child class inherit all the methods and properties from its parent:

By using the super() function, you do not have to use the name of the parent element, it will automatically inherit the methods and properties from its parent.

Example:

[5]:
```
class Person:
  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

  def printname(self):
    print(self.firstname, self.lastname)

class Student(Person):
  def __init__(self, fname, lname):
    super().__init__(fname, lname)

x = Student("Mike", "Olsen")
x.printname()
```

```
Mike Olsen
```

0.6 Add Methods

Example:

Add a method called welcome to the Student class:

[7]:
```
class Person:
  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

  def printname(self):
    print(self.firstname, self.lastname)

class Student(Person):
  def __init__(self, fname, lname, year):
    super().__init__(fname, lname)
    self.graduationyear = year

  def welcome(self):
    print("Welcome", self.firstname, self.lastname, "to the class of", self.
 ↪graduationyear)

x = Student("Mike", "Olsen", 2019)
x.welcome()
```

```
Welcome Mike Olsen to the class of 2019
```

[ ]:
```
**If you add a method in the child class with the same name as a function in␣
 ↪the parent class,
the inheritance of the parent method will be overridden.
```