Python Inheritance

Python Inheritance

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.

Create a Parent Class

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

Example   Get your own Python Server

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

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

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

#Use the Person class to create an object, and then execute the
printname method:
```

Tutorials ▾   Exercises ▾   Services ▾   🔍   ◐   Sign Up   Log in

☰   CSS   JAVASCRIPT   SQL   PYTHON   JAVA   PHP   HOW TO   W3.CSS   C

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:

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

Try it Yourself »

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

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

Now we have successfully added the `__init__()` function, and kept the inheritance of the parent class, and we are ready to add functionality in the `__init__()` function.

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:

Example

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

Try it Yourself »

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.
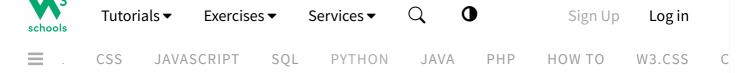
Add Properties

Example

Add a property called `graduationyear` to the `Student` class:

```python
class Student(Person):
  def __init__(self, fname, lname):
    super().__init__(fname, lname)
    self.graduationyear = 2019
```

Try it Yourself »

Example

Add a `year` parameter, and pass the correct year when creating objects:

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

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

Try it Yourself »

Add Methods

Example

Add a method called `welcome` to the `Student` class:

```python
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)
```

Try it Yourself »

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.