Python Classes, Objects and Inheritance

Contents

[Python Classes and Objects 3](#_Toc98166840)

[Python Modules and OO Concepts 5](#_Toc98166841)

[Is a class library an API? 5](#_Toc98166842)

[Python Inheritance 6](#_Toc98166843)

[Construct / Instantiate / instance 6](#_Toc98166844)

[References 7](#_Toc98166845)

[5 Examples of Excellent API Documentation (and Why We Think So) 7](#_Toc98166846)

# Python Classes and Objects

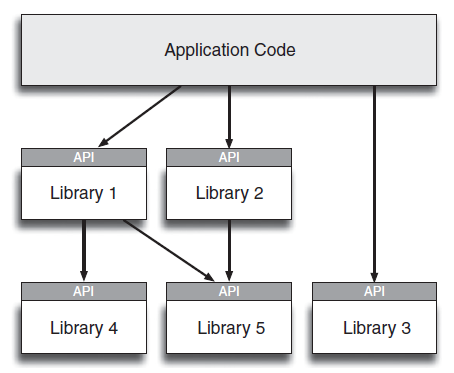
|  |  |
| --- | --- |
| * Python is an object oriented programming language. * Almost everything in Python is an object, with its properties and methods. | * **Properties** – definition of the characteristic variables of the class * **Methods** – definition of the **functions** of the class |
| **Create a Class**   * A Class is like an object constructor, or a "blueprint" for creating objects. * To create a class, use the keyword class: | Create a class named MyClass, with a property variable named x:  class MyClass:  x = 5 |
| **Create Object**   * Now we can use the class named MyClass to create objects in memory: | Create an object named p1, and print the value of x:  class MyClass:  x = 5  p1 = MyClass()  print(p1.x) |
| **\_\_init\_\_() Function**   * All classes have a function called \_\_init\_\_(), which is always executed when the class is being initiated. * Use the \_\_init\_\_() function to assign values to object properties, or other operations that are necessary to do when the object is being created: | class Person:   def \_\_init\_\_(self, name, age):     self.name = name     self.age = age  p1 = Person("John", 36)  print(p1.name) print(p1.age) |
| **Object Methods**   * Objects can also contain methods. Methods in objects are functions that belong to the object. | class Person:   def \_\_init\_\_(self, name, age):     self.name = name     self.age = age    def myfunc(self):     print("Hello my name is " + self.name)  p1 = Person("John", 36) p1.myfunc() |
| **self Parameter**   * The self parameter is a reference to the current instance of the class, and is used to access variables that belong to the class. * It does not have to be named self , you can call it whatever you like, but it has to be the first parameter of any function in the class * The self parameter is useful in managing and keeping track of properties and methods and how they work together for both the programmer and Python | Use the words *mysillyobject* and *abc* instead of *self*:  class Person:   def \_\_init\_\_(mysillyobject, name, age):     mysillyobject.name = name     mysillyobject.age = age    def myfunc(abc):     print("Hello my name is " + abc.name)  p1 = Person("John", 36) p1.myfunc() |
| **Modify Object Properties**   * You can modify properties on objects | Set the age of p1 to 40:  class Person:  def \_\_init\_\_(self, name, age):  self.name = name  self.age = age  def myfunc(self):  print("Hello my name is " + self.name)  p1 = Person("John", 36)  p1.age = 40  print(p1.age) |
| **Delete Object Properties**  class Person:  def \_\_init\_\_(self, name, age):  self.name = name  self.age = age  def myfunc(self):  print("Hello my name is " + self.name)  p1 = Person("John", 36)  del p1.age  print(p1.age) | Traceback (most recent call last):  File "./prog.py", line 13, in <module>  AttributeError: 'Person' object has no attribute 'age' |
| **Delete Objects**  **class Person:**  **def \_\_init\_\_(self, name, age):**  **self.name = name**  **self.age = age**  **def myfunc(self):**  **print("Hello my name is " + self.name)**  **p1 = Person("John", 36)**  **del p1**  **print(p1)** | Traceback (most recent call last):  File "demo\_class8.py", line 13, in <module>  print(p1)  NameError: 'p1' is not defined |
| **pass Statement**   * class definitions cannot be empty, but if you for some reason have a class definition with no content, put in the pass statement to avoid getting an error. | class Person:   pass |

<https://www.w3schools.com/python/python_classes.asp>

## Python Modules and OO Concepts

### Is a class library an API?

An API is the way you access a library (or any set of classes). In object-oriented programming , a class library is a collection of prewritten classes or coded templates, any of which can be specified and used by a programmer when developing an application program. you could use a library in a variety of projects.



# Python Inheritance

<https://www.w3schools.com/python/python_inheritance.asp>

## Construct / Instantiate / instance

|  |  |
| --- | --- |
| **Python Constructor**   * A constructor is a special type of method (function) which is used to initialize the instance members of the class. | * Constructors are generally used for instantiating an object. * The task of constructors is to initialize(assign values) to the data members of the class when an object of the class is created. * In Python the \_\_init\_\_() method is called the constructor and is always called when an object is created. |
| class MyClass:  Greeting = "  def \_\_init\_\_(self, Name="there"):  self.Greeting = Name + "!"  def SayHello(self):  print("Hello {0}".format(self.Greeting)) | # Python creates an instance of MyClass named MyInstance  >>> MyInstance = MyClass()  # message provides the default, generic greeting  >>> MyInstance.SayHello()  # Python creates an instance of MyClass named MyInstance.  >>> MyInstance = MyClass(“Amy”)  # message provides a specific greeting.  >>> MyInstance.SayHello() |
|  | [How to Create a Constructor in Python](https://www.dummies.com/programming/python/how-to-create-a-constructor-in-python/) |

# References

### [5 Examples of Excellent API Documentation (and Why We Think So)](https://nordicapis.com/5-examples-of-excellent-api-documentation/)

[Framework vs Library vs Platform vs API vs SDK vs Toolkits vs IDE](https://shashvatshukla.medium.com/framework-vs-library-vs-platform-vs-api-vs-sdk-vs-toolkits-vs-ide-50a9473999db)