# Life of a Python package / module / object

## PIP – Python Install Package into your environment (system)

|  |  |
| --- | --- |
| It’s important to note that the term “package” in this context is being used to describe a bundle of software modlules to be installed (i.e. as a synonym for a distribution) in your environment for use by yours modules. | Python “Virtual Environments” allow Python packages to be installed in an isolated location for a particular application, rather than being installed globally. If you are looking to safely install global command line tools, see Installing stand alone command line tools |
| py -m pip install "SomeProject" | [Installing Packages](https://packaging.python.org/tutorials/installing-packages/) |

## Import (bind) package to your source code

|  |  |
| --- | --- |
| * Import modules in your Python source code makes the imported models available (binded) to you model based on environment, privileges, version levels, container/module paths/locations. * Python code in one module gains access to the code in another module by the process of importing it. The import statement is the most common way of invoking the import machinery, but it is not the only way. Functions such as importlib.import\_module() and built-in \_\_import\_\_() can also be used to invoke the import machinery. |  |
|  | [The import system](https://docs.python.org/3/reference/import.html) |

## 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/) |