Create a module

```
NA = 6.02e-23 # module-level constant
class Class_1():
        " " "
        #do cst rings, output of help(m odule)
       # id PEP257 chapitre 15
        " " "

       def __init __( self, functi on_1, functi -
on_2, data):
                  sel f.f unc tion_1 = function_1
                  sel f.f unc tion_2 = function_2
                  sel f.data = xxx

         def functi on_ 1(s elf):
                  blabla
                  return var

       def functi on_ 2(s elf):
                  pri nt( data)
       def updata (self, add_var):
        #Mo difying an Attrib ute's Value Through
a Method
                sel f.data += add_var
class Class_ 2(C las s_1): # inherit from Animal
  " " "
comment for the class2
blabla
" " "
       PI = 3.14 # constant with uppercase letters
       # invariable during programm excuta tion)
       def __init __( self, make, model, year):
                  sup er( class1, self)._ _i nit -
__( fun ction1, function2, data)
                # call the superclass constr -
uctor
                  blabla
       def functi on_ 3(s elf,):
          " " " doc strings under method " " "
                blabla
       def functi on_ 4(s elf):
                bla blabla
```

The __init__() method takes in these parameters and stores them in the attributes that will be associated with instances made from this class.

import a module

```
import module as alias
# import a entier module and give it an allas
from module import class
# from a module import a class
# it should prepare at first a file __init __.py
at the directory
from module.class import method
# from a class of module import a method
(function)
```

init.py

An init.py file is a special file that is used to indicate that a directory contains a Python package. A Python package is a collection of modules that can be imported and used together. An init.py file can also contain code that is executed when the package or a module in the package is imported. This can be useful for initializing package-level data or performing other tasks

Before Python 3.3, an init.py file was required for every package directory. Otherwise, Python would not recognize the directory as a package and would not be able to import modules from it. This was done to prevent directories with common names, such as string, from unintentionally hiding valid modules that occur later on the module search path.

However, since Python 3.3, an init.py file is no longer mandatory for defining a package. Python can also recognize namespace packages, which are packages that do not have an init.py file and can span multiple directories. Namespace packages allow multiple portions of a package to be distributed and installed separately, and then merged together at runtime

create an environment path

```
Linux & MacOS un file bash (~.bashrc)
export PYTHON PAT H=$ PYT HON PAT H:/ che min /ve -
rs/ mon /su per /module
then
echo $PYTHO NPATH
source ~.bashrc
Windows in shell PowerShell
$env:P YTH ONPATH += " ;C: \ch emi n\v ers \mo -
n\s upe r\m odu le"
then
echo $env:P YTH ONPATH
```