The following are some very simple and marginally annotated examples of calling Python from R. They show how one can call functions; create Python objects and call their methods; evaluate Python expressions given as R strings and load files.

```
[]
x <- .Python("listdir", "/tmp", .module="posix")
length(x)
```

```
[]
.Python("putenv", "duncan", "foo", .module="os")

Sys.getenv("duncan" )
```

```
[]
>>> .Python("find", 'xml2*', "/home/duncan/Projects/org/omegahat/XML/RS", .module="find")
[1] "/home/duncan/Projects/org/omegahat/XML/RS/examples/xml2tex.Sxml"
[2] "/home/duncan/Projects/org/omegahat/XML/RS/examples/xml2tex.Sxml~"
```

We can use this to define a *system.findfile()* like *system.file()*.

```
[]
system.findfile <-
function(pat, pkg=.packages(), lib=.lib.loc)
{
  for(i in pkg) {
    val <- .Python("find", pat, system.file("", pkg=i), .module="find")
    if(length(val) > 0)
       return(val)
  }

 return(NULL)
}
```

```
[]
 .Python("what", '/jdk1.3/demo/jfc/Java2D/images/duke.png', .module="imghdr")
 .Python("what", paste(Sys.getenv("R_HOME"),'/doc/html/left.jpg',sep=""), .module="imghdr")
```

Evaluating a Python expression given as a string. Taken from Mark Lutz's examples at `http://shell.rmi.net/~lutz/newex.html`.

```
[]
.PythonEval("upper('spam')+'!'", .module="string")
[1] "SPAM!"
```

Here we get the variable `version` in the sys module.

```
[]
> .Python("version", .module="sys")
[1] "1.5.2 (#1, Aug 29 2000, 14:55:40)  [GCC 2.95.2 19991024 (release)]"
```

Try adding an argument.

```
[]
> .Python("version", 1, .module="sys")
[1] "1.5.2 (#1, Aug 29 2000, 14:55:40)  [GCC 2.95.2 19991024 (release)]"
Warning message:
ignoring 1 arguments to Python call
>
```

Here we use the file `tests/method.py` to define a class.

```
[]

>>> from method import RSTest
>>> RSTest
<class method.RSTest at 80edf00>
>>> t = RSTest()
>>> t
<method.RSTest instance at 80cf4b0>
>>> t.test(1,2)
10
```

Now we try to use this in S.

First, we make certain that we can locate the `method.py` by adding the `tests/` directory to the PYTHONPATH environment variable.

```
[]
.First <- function() {
    dyn.load("RS.so");
    Sys.putenv("PYTHONPATH"="tests") ;
    .PythonInit()
}
```

Note that method is the name of the file containing the Python code.

```
[]
test <- .PythonNew("RSTest", .module="method")
.PythonMethod(test, "test", 1, 2)
```

```
[]
> test <- .PythonNew("RSTest", .module="method")
> .Call("RS_PythonGetMethods", test)
```

```
[]
u <- .PythonNew("urlopen", "http://www.omegahat.org/index.html", .module="urllib")
txt <- .PythonMethod(u, "read")
.Python(u, "close")
.PythonMethod(u, "geturl")
```

or, using the $ operator,

```
[]
u$read()
u$geturl()
```

Reflectance

```
[]
> getSuperClasses("A", "hierarchy")
$ClassName
[1] "A"

$SuperClasses
$SuperClasses$B
$SuperClasses$B$ClassName
[1] "B"

$SuperClasses$B$SuperClasses
$SuperClasses$B$SuperClasses$F
[1] "F"



$SuperClasses$C
$SuperClasses$C$ClassName
[1] "C"

$SuperClasses$C$SuperClasses
$SuperClasses$C$SuperClasses$F
[1] "F"

$SuperClasses$C$SuperClasses$G
[1] "G"



$SuperClasses$D
$SuperClasses$D$ClassName
[1] "D"

$SuperClasses$D$SuperClasses
$SuperClasses$D$SuperClasses$F
[1] "F"
```

1 Calling Built-in Functions

```
[]
.Python("range", 1, 100)
.Python("int", "1100")
```