Accessing Python source code and bytecode All Versions

Examples

Display the bytecode of a function

The Python interpreter compiles code to bytecode before executing it on the Python's virtual machine (see also What is python bytecode? .

Here's how to view the bytecode of a Python function

```
import dis

def fib(n):
    if n <= 2: return 1
    return fib(n-1) + fib(n-2)

# Display the disassembled bytecode of the function.
dis.dis(fib)
```

The function `dis.dis` in the dis module will return a decompiled bytecode of the function passed to it.

Display the source code of an object

To print the source code of a Python object use `inspect` . Here's how to print the source code of the method `random.randint`

```
import random
import inspect

print(inspect.getsource(random.randint))
# Output:
#    def randint(self, a, b):
#        """Return random integer in range [a, b], including both end points.
#        """
#
#        return self.randrange(a, b+1)
```

To just print the documentation string

```
print(inspect.getdoc(random.randint))
# Output:
# Return random integer in range [a, b], including both end points.
```

Print full path of the file where the method `random.randint` is defined:

```
print(inspect.getfile(random.randint))
# c:\Python35\lib\random.py
print(random.randint.__code__.co_filename) # equivalent to the above
# c:\Python35\lib\random.py
```

If an object is defined interactively `inspect` cannot provide the source code but you can use `dill.source.getsource` instead

```
# define a new function in the interactive shell
def add(a, b):
   return a + b
print(add.__code__.co_filename) # Output: <stdin>

import dill
print dill.source.getsource(add)
# def add(a, b):
      return a + b
```

The source code for Python's built-in functions is written in **c** and can only be accessed by looking at the Python's source code (hosted on Mercurial or downloadable from https://www.python.org/downloads/source/) .

```
print(inspect.getsource(sorted)) # raises a TypeError
type(sorted) # <class 'builtin_function_or_method'>
```

Exploring the code object of a function

CPython allows access to the code object for a function object.

The `__code__` object contains the raw bytecode (`co_code`) of the function as well as other information such as constants and variable names.

```
def fib(n):
    if n <= 2: return 1
    return fib(n-1) + fib(n-2)
dir(fib.__code__)
```


```
def fib(n):
    if n <= 2: return 1
    return fib(n-1) + fib(n-2)
dir(fib.__code__)
```


Syntax

Parameters

Remarks