The print() function displays the string value inside the parentheses on the screen.

➋ print('Hello world!')

print('What is your name?') # ask for their name

The line print('Hello world!') means “Print out the text in the string 'Hello world!'.” When Python executes this line, you say that Python is *calling* the print() function and the string value is being *passed* to the function. A value that is passed to a function call is an *argument*. Notice that the quotes are not printed to the screen. They just mark where the string begins and ends; they are not part of the string value.

[root@localhost ~]# vi assess.py

*#!/usr/bin/python*

*import sys*

*import argparse*

*def main(sysArgs):*

*global args*

*parser = argparse.ArgumentParser()*

*parser.add\_argument("-a","--all",action="store\_true",help="Run all tests")*

*parser.add\_argument("-t","--test",action="append",help="Run this test")*

*args = parser.parse\_args()*

*if args.all:*

*print "Running all tests"*

*else:*

*print "Running selected tests"*

*main(sys.argv)*

[root@localhost ~]# ./assess.py -t

usage: assess.py [-h] [-a] [-t TEST]

assess.py: error: argument -t/--test: expected one argument

[root@localhost ~]# ./assess.py –a

*parser.add\_argument("-a","--all",action="store\_true",help="Run all tests")*

*action="store\_true" does not need an argument after –a or –all*

*parser.add\_argument("-t","--test",action="append",help="Run this test")*

*action="append" needs an argument after –t or –test*

**What is 'if \_\_name\_\_ == "\_\_main\_\_"' for?**

The if \_\_name\_\_ == "\_\_main\_\_": ... trick exists in Python so that our Python files can act as either reusable modules, or as standalone programs. As a toy example, let’s say that we have two files:

$ cat mymath.py

def square(x):

return x \* x

if \_\_name\_\_ == '\_\_main\_\_':

print "test: square(42) ==", square(42)

$ cat mygame.py

import mymath

print "this is mygame."

print mymath.square(17)

In this example, we’ve written mymath.py to be both used as a utility module, as well as a standalone program. We can run mymath standalone by doing this:

$ python mymath.py

test: square(42) == 1764

But we can also use mymath.py as a module; let’s see what happens when we run mygame.py:

$ python mygame.py

this is mygame.

289

Notice that here we don’t see the ‘test’ line that mymath.py had near the bottom of its code. That’s because, in this context, mymath is not the main program. That’s what the if \_\_name\_\_ == "\_\_main\_\_": ... trick is used for.