

Python programs

Chapter 11

This chapter covers

- Creating a very basic program
- Combining programs and modules
- Distributing Python applications

Creating a very basic program

- Any group of Python statements placed sequentially in a file can be used as a program, or script.
- But it's more standard and useful to introduce additional structure.
- In its most basic form, this task is a simple matter of creating a controlling function in a file and calling that function.

```
1  def main():  
2      print("this is our first test script file")  
3  
4  main()
```

```
C:\pythoncodes\programs>python basicprogram.py  
this is our first test script file
```

Command-line arguments

```
1  import sys
2
3  ✓ def main():
4      |     print("this is our second test script file")
5      |     print(sys.argv)
6
7  main()
```

```
C:\pythoncodes\programs>python cmdarg.py hello world
this is our second test script file
['cmdarg.py', 'hello', 'world']
```

Using the fileinput module

- It provides support for processing
- lines of input from one or more files. It automatically reads the command-line
- arguments (out of sys.argv) and takes them as its list of input files. Then it allows
- you to sequentially iterate through these lines.

```
1  import fileinput
2
3  def main():
4      for line in fileinput.input():
5          if not line.startswith('##'):
6              print(line, end="")
7
8  main()
```

Input files

```
1  ## sole1.tst: test data for the sole function
2  0 0 0
3  0 100 0
4  ##
5  0 100 100
```

```
1  ## sole2.tst: more test data for the sole function
2  12 15 0
3  ##
4  100 100 0
```

QUICK CHECK

- Match the following ways of interacting with the command line and the correct use case for each:

Multiple arguments and options

No arguments or just one argument

Processing multiple files

Using the script as a filter

sys.argv

Use file_input module

Redirect standard input and output

Use argparse module

Programs and Modules

- For small scripts that contain only a few lines of code, a single function works well.
- But if the script grows beyond this size, separating your controlling function from the rest of the code is a good option to take.
- The script in the next listing returns the English-language name for a given number between 0 and 99.


```

1  import sys
2  # conversion mappings
3  _1to9dict = {'0': '', '1': 'one', '2': 'two', '3': 'three', '4': 'four', '5': 'five',
4             '6': 'six', '7': 'seven', '8': 'eight', '9': 'nine'}
5  _10to19dict = {'0': 'ten', '1': 'eleven', '2': 'twelve', '3': 'thirteen', '4': 'fourteen',
6               '5': 'fifteen', '6': 'sixteen', '7': 'seventeen', '8': 'eighteen', '9': 'nineteen'}
7  _20to90dict = {'2': 'twenty', '3': 'thirty', '4': 'forty', '5': 'fifty', '6': 'sixty',
8               '7': 'seventy', '8': 'eighty', '9': 'ninety'}
9
10 def num2words(num_string):
11     if num_string == '0':
12         return 'zero'
13     if len(num_string) > 2:
14         return "Sorry can only handle 1 or 2 digit numbers"
15     num_string = '0' + num_string
16     tens, ones = num_string[-2], num_string[-1]
17     if tens == '0':
18         return _1to9dict[ones]
19     elif tens == '1':
20         return _10to19dict[ones]
21     else:
22         return _20to90dict[tens] + ' ' + _1to9dict[ones]
23
24 def main():
25     print(num2words(sys.argv[1]))
26
27 if __name__ == '__main__':
28     main()
29 else:
30     print("n2w loaded as a module")

```

Distributing Python applications

- Share the source files, of course, probably bundled in a zip or tar file.
- Assuming that the applications were written portably, you could also ship only the bytecode as .pyc files.
 - Wheels packages
 - zipapp and pex
 - py2exe and py2app
 - Creating executable programs with freeze

Summary

- Python scripts and modules in their most basic form are just sequences of Python statements placed in a file.
- Modules can be instrumented to run as scripts, and scripts can be set up so that they can be imported as modules.
- Scripts can be made executable on the UNIX, macOS, or Windows command lines. They can be set up to support command-line redirection of their input and output, and with the argparse module, it's easy to parse out complex combinations of command-line arguments.
- On macOS, you can use the Python Launcher to run Python programs, either individually or as the default application for opening Python files.

Summary

- On Windows, you can call scripts in several ways: by opening them with a double-click, using the Run window, or using a command-prompt window.
- Python scripts can be distributed as scripts, as bytecode, or in special packages called wheels.
- py2exe, py2app, and the freeze tool provide an executable Python program that runs on machines that don't contain a Python interpreter.
- Now that you have an idea of the ways to create scripts and applications, the next step is looking at how Python can interact with and manipulate filesystems.