Eclipse Python Tutorial

For Introduction to Programming Using Python
By Y. Daniel Liang

This supplement covers the following topics:

- Download and install Java if necessary
- Download and install Eclipse
- Launch Eclipse
- Install Python plug-in for Eclipse
- Add a Python Interpreter
- Create a Python Project
- Create a Python Program
- Run a Python Program
- Debug a Python Program

0 Introduction

This tutorial is for students who want to develop Python projects using Eclipse. Eclipse is a popular IDE for developing software. You can use Eclipse for programming in Java, C++, and Python, or many other languages.

1 Download and Install Java

To use Eclipse, you need to download and install Java, since Eclipse relies on Java to run. It is very likely that Java is already installed on your system. If not, download Java from http://www.oracle.com/technetwork/java/javase/downloads/index.html. The installation is straightforward.

2 Download and Install Eclipse

Eclipse can be downloaded from http://www.eclipse.org/downloads/. There are several versions of Eclipse on the list. Choose Eclipse Classic and select a version for your platform (i.e., Windows 32-bit, Windows 64-bit, and Mac). The file downloaded is a ZIP file (a compressed file). Uncompress it into a directory named c:\eclipse.

3 Launch Eclipse

Assume that you have installed Eclipse files in c:\eclipse. To start Eclipse, double-click on the eclipse icon in the c:\eclipse folder, as shown in Figure 1. The Workspace

Launcher window now appears, as shown in Figure 2. Enter *c:* in the Workspace field and click *OK* to display the Eclipse UI, as shown in Figure 3. (If the workspace already contains projects, the projects will be displayed in the UI.) Workspace is actually a directory that stores your project files. Click the Workbench icon to display the Eclipse user interface, as shown in Figure 4.

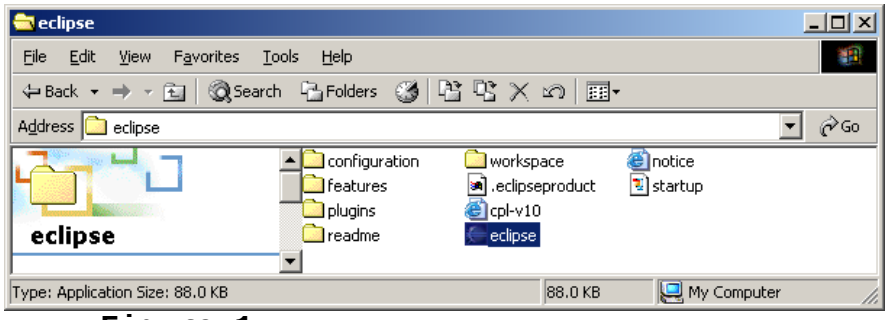

Figure 1

You can start Eclipse by double-clicking the eclipse icon from the eclipse installation directory.

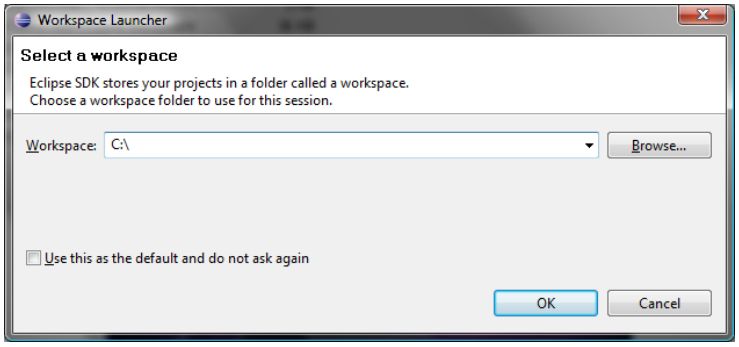

Figure 2

The Workspace Launcher lets you choose a directory to store projects.

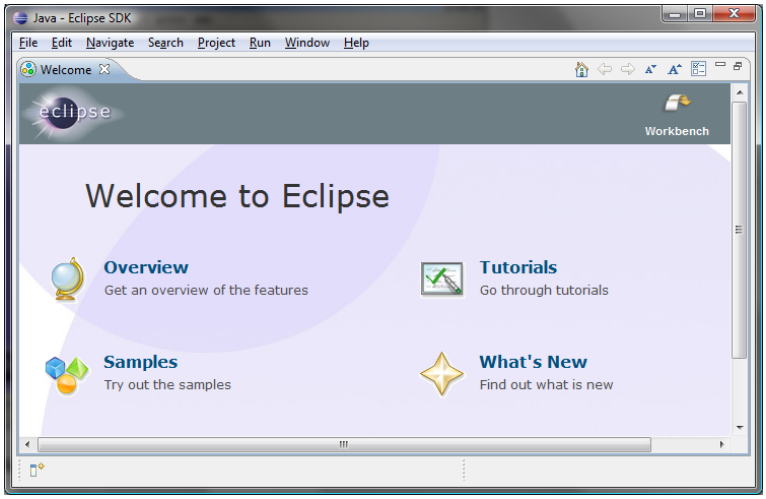

Figure 3

The Eclipse main window is the command center for the IDE.

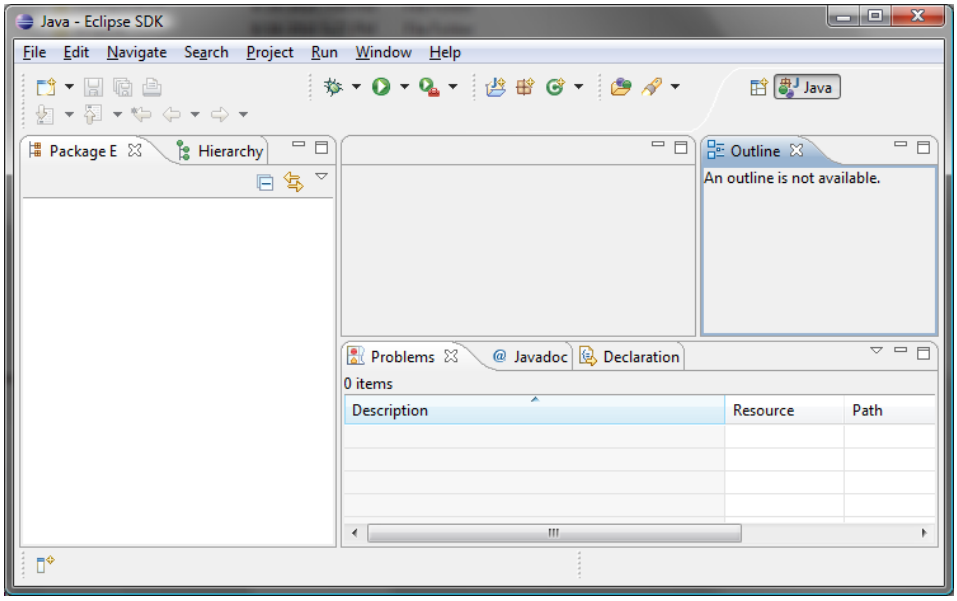

Figure 4

The Eclipse UI is displayed.

4 Install Python Plug-in

Follow the steps below to install Python Plug-in:

1. Choose *Help*, *Eclipse Marketplace* to display the Eclipse Marketplace window, as shown in Figure 5.
2. Under the Search tab, enter Python in the Find field. You will find the current Python IDL for Eclipse 3.6 or a later higher version.

3. Click Install to install it.

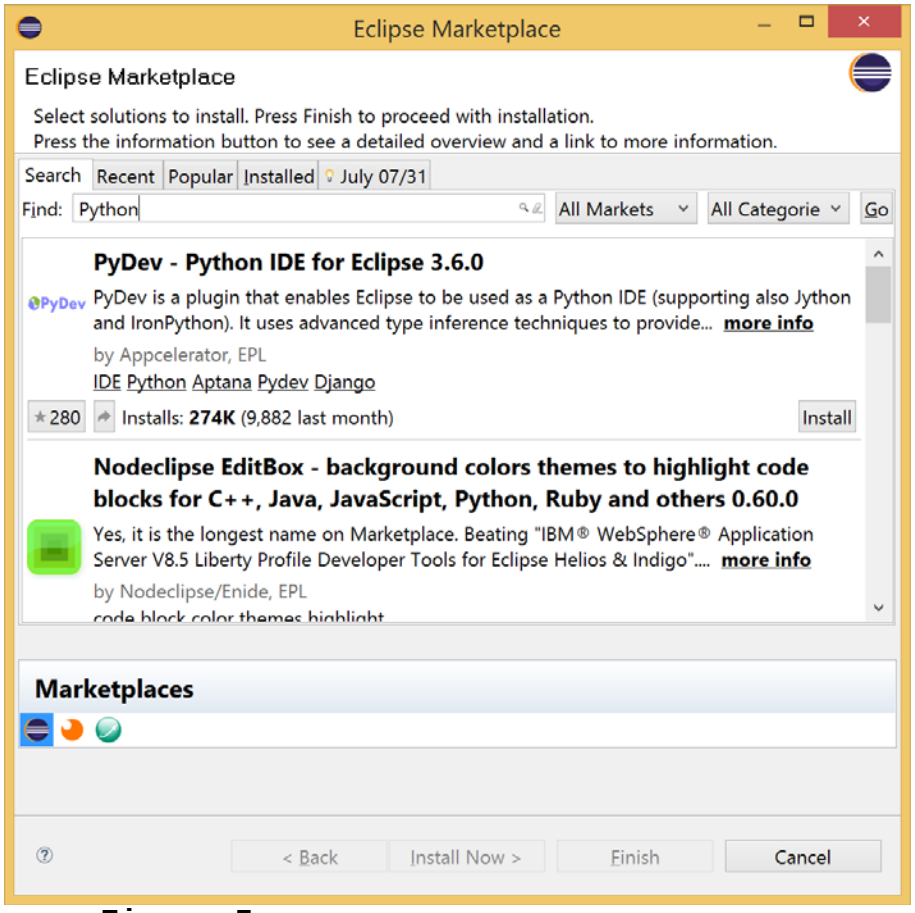

Figure 5

You can install plug-ins for Eclipse in the Eclipse Marketplace window.

5 Create Python Project

Projects are like folders to hold Python files. Before creating a Python program, you have to first create a project.

Follow the steps below to create a project:

1. Choose *File*, *New*, *Project* to display the New Project wizard, as shown in Figure 6.
2. Select *PyDev Project* and click *Next* to display the PyDev Project wizard, as shown in Figure 7. Type *pybook* in the Project name field. As you type, the Directory field becomes c:\pybook.
3. Make sure that you choose Python as project type and 3.0 as Grammar Version. If the Interpreter is not listed, following the next section to configure a new Python interpreter.

4. Uncheck the “Create default ‘src’ folder. This step is optional. If you check it, a folder named src will be created to hold Python source code. The src folder will be under c:\pybook.
5. Click *Finish* to create the project.

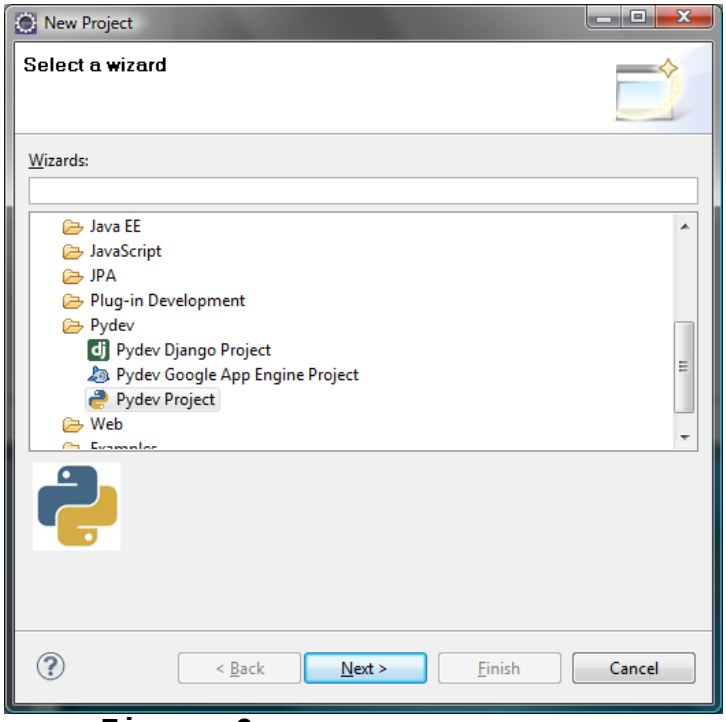

Figure 6

Choose PyDev to create Python project.

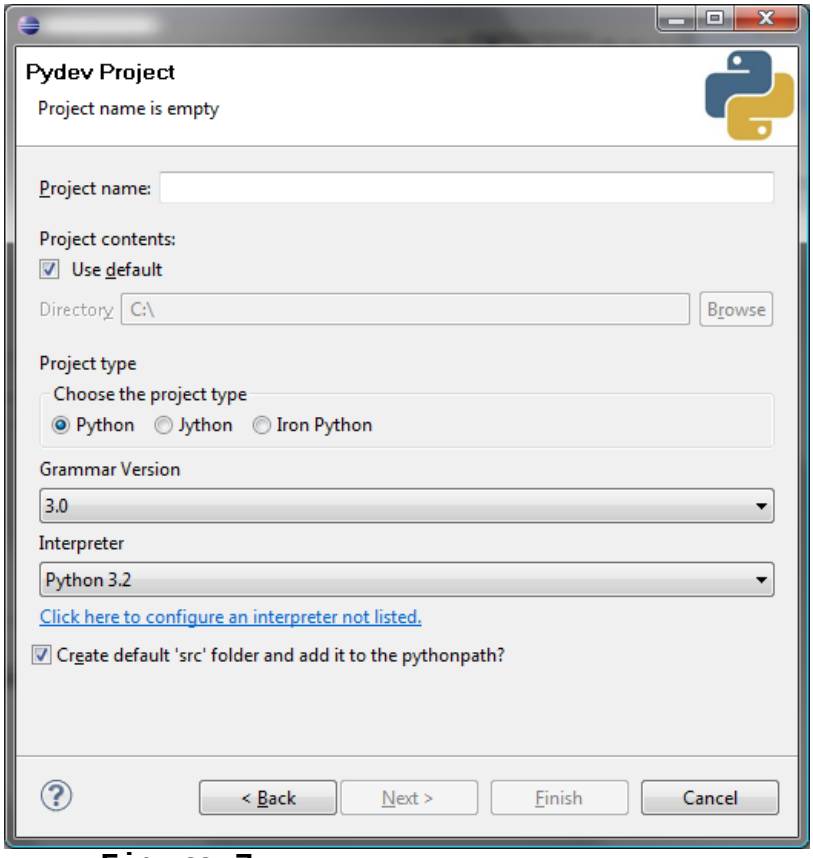

Figure 7

Enter a project and choose appropriate project attributes.

6 Add Python Interpreter

Follow the steps below to configure a new Python interpreter:

1. Click “clicking here to configure an interpreter not listed” to display the Preferences window, as shown in Figure 11.
2. Click *New* to open the Select Interpreter dialog box, as shown in Figure 8.
3. Locate the Python interpreter and click *OK* to add it to Eclipse.

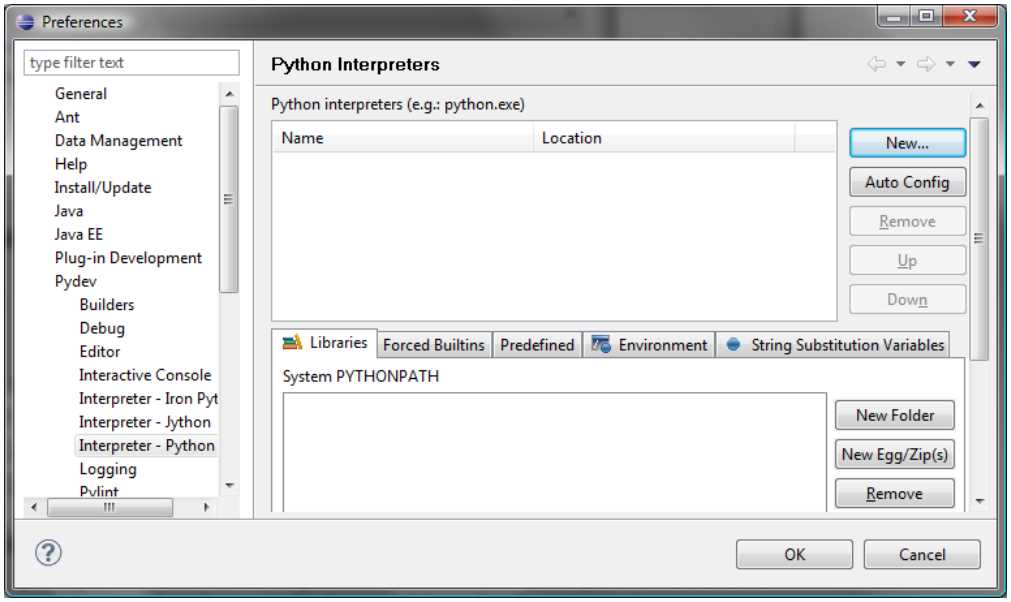

Figure 11

You can add or remove a Python interpreter from this window.

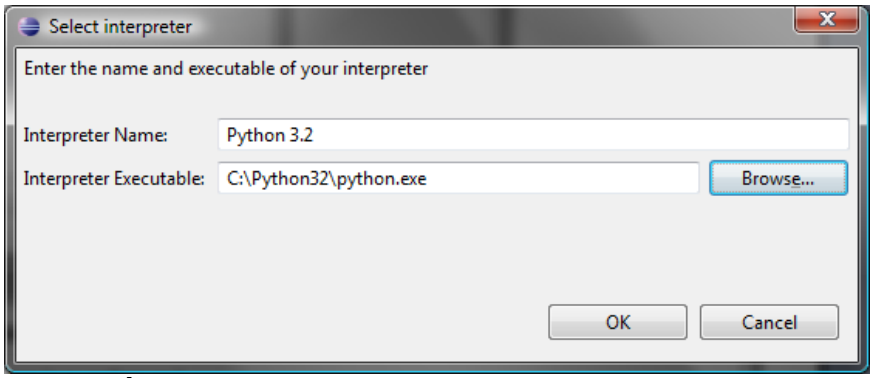

Figure 8

Browse to locate a Python interpreter.

7 Create a Python Program

Now you can create a program in the project by right-click on the pybook node to display a context menu, as shown in Figure 9. Choose *New*, *File* to display the New File dialog box, as shown in Figure 10. Enter a file Welcome.py to create a Python program. Click *OK*. You will see a Python program the file created under the pybook node in the package explorer, as shown in Figure 11.

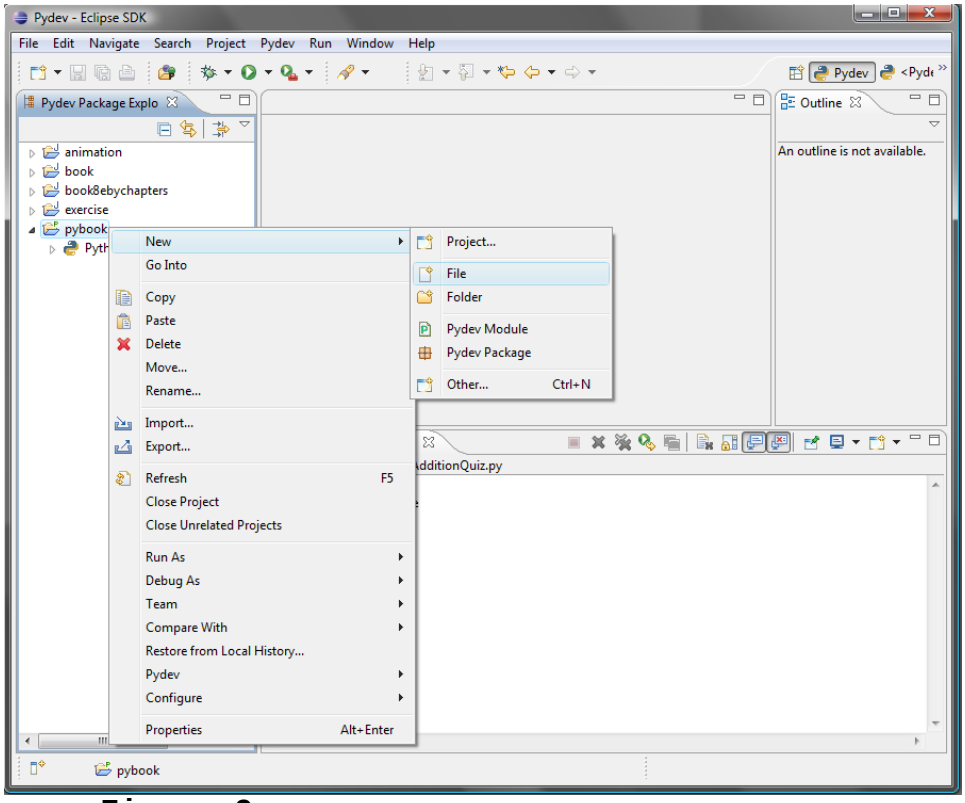

Figure 9

Browse to locate a Python interpreter.

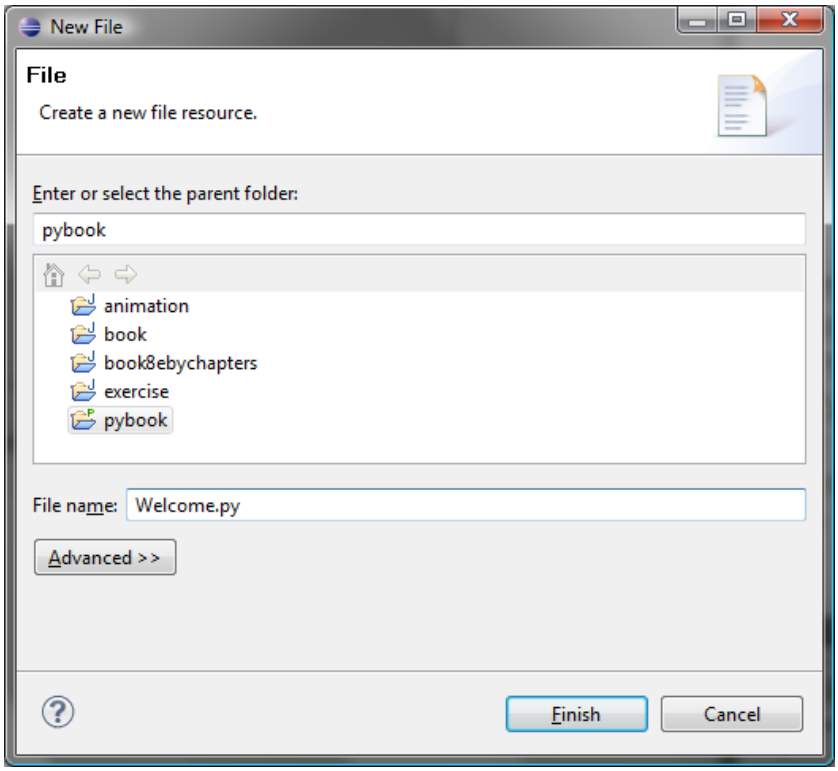

Figure 10

Enter a file name to create a Python program.

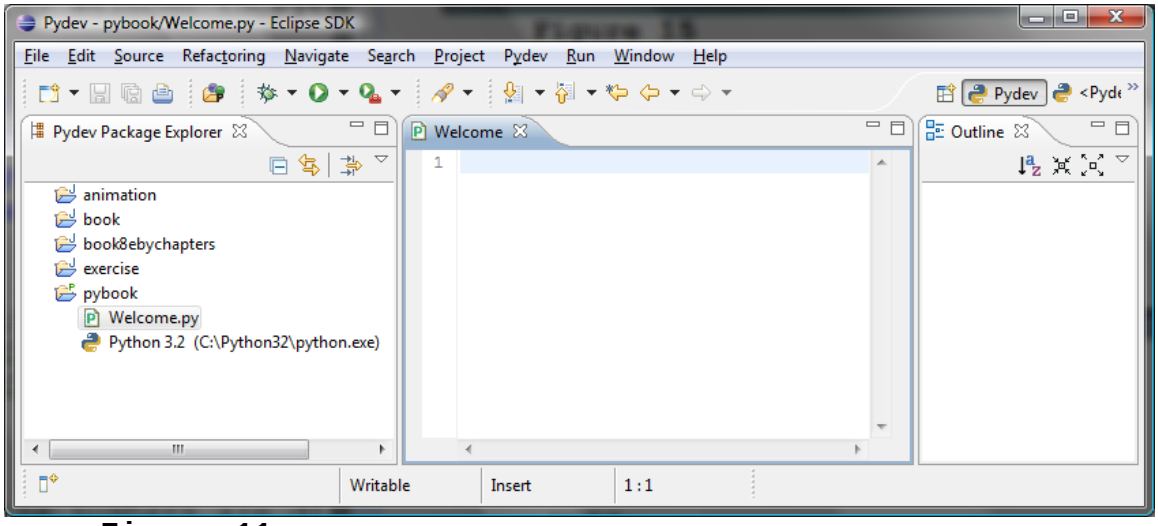

Figure 11

The file is created in the package explorer.

Type the code from Listing 1.1 in the text, as shown in Figure 12.

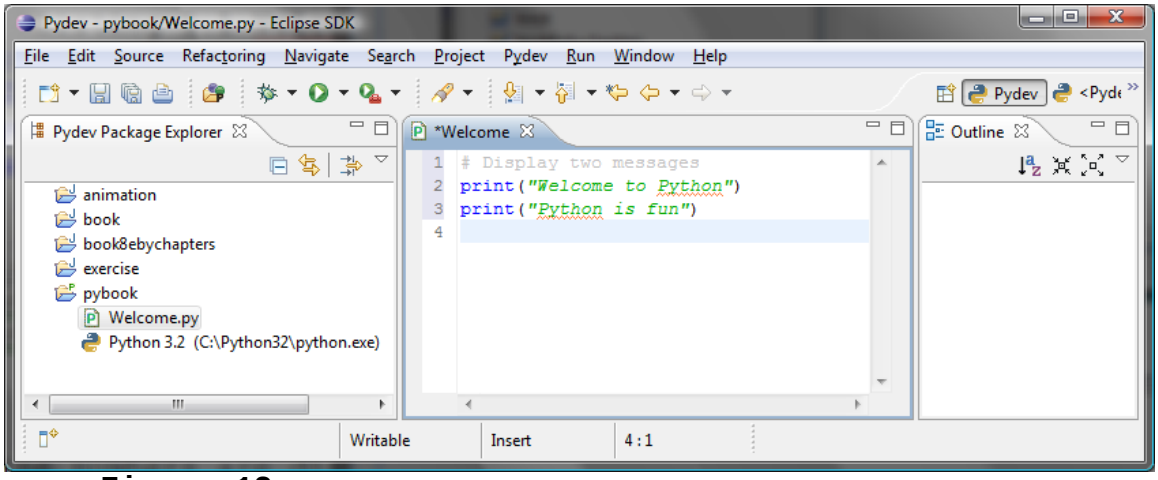

Figure 12

Python code is entered in the editor pane.

8 Run a Python Program

Now you can run the program by right-clicking on the file (Welcome.py) to display a context menu, as shown in Figure 13. Choose Run As, Python Run to run the program. The result is displayed in the Console pane, as shown in Figure 14.

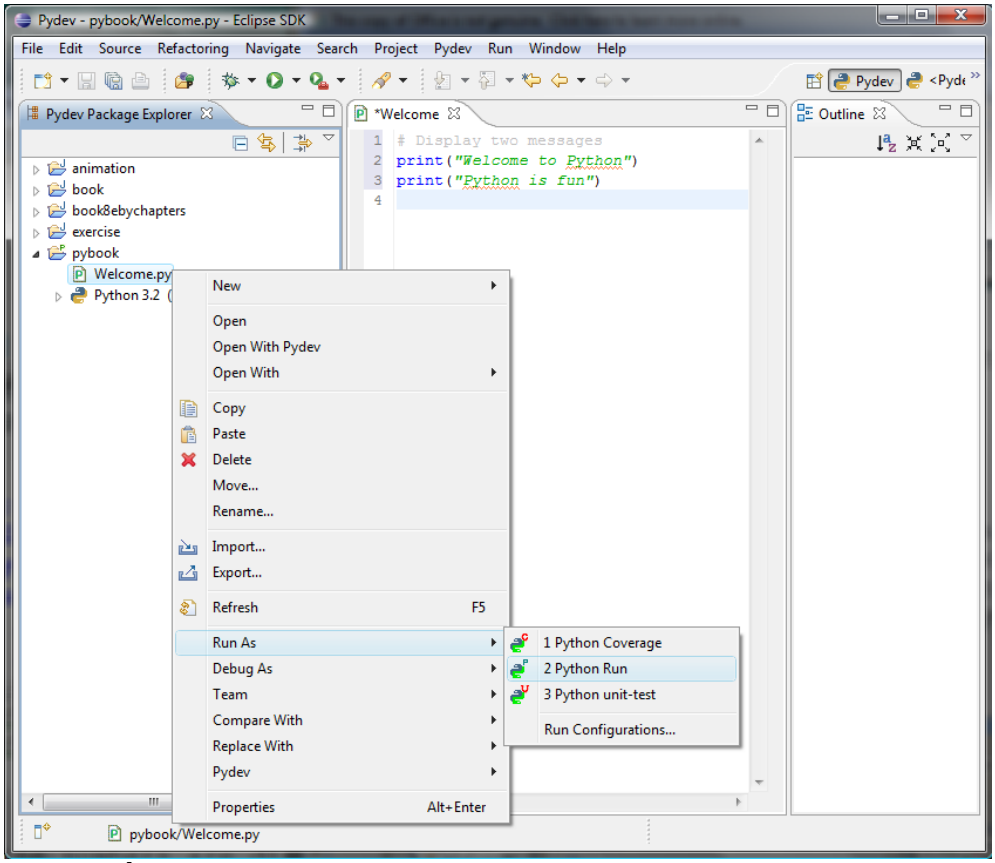

Figure 13

Run a Python program.

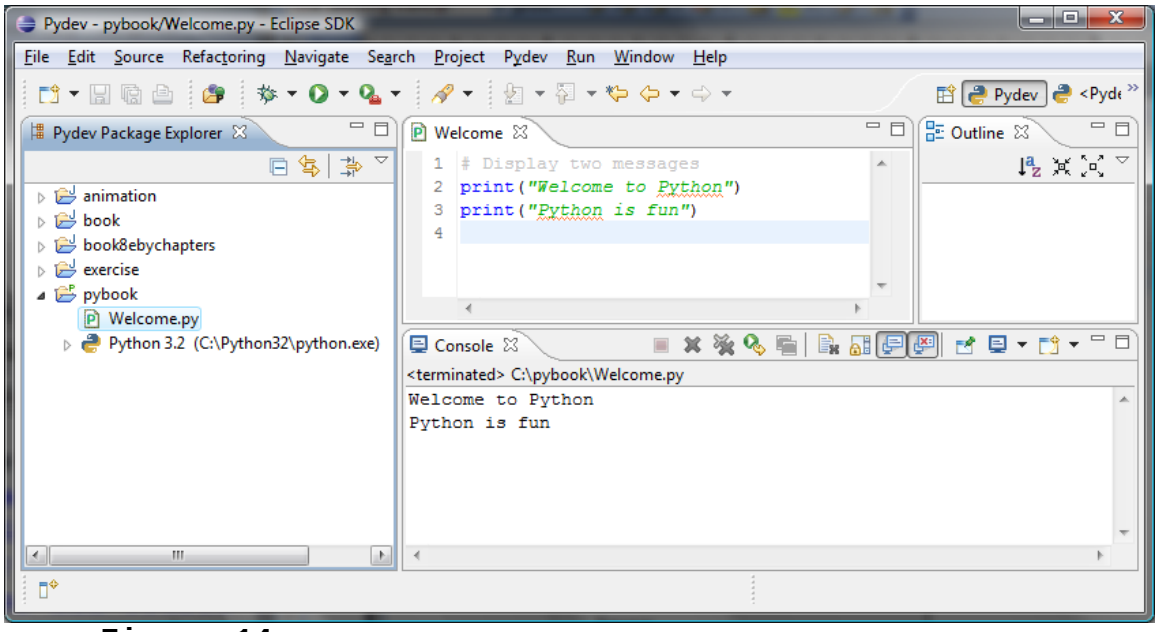

Figure 14

The output is displayed in the Console pane.

9 Debug Python Programs

The Python debugger utility is integrated in Eclipse. You can pinpoint bugs in your program with the help of the Eclipse debugger without leaving the IDE. The Eclipse debugger enables you to set breakpoints and execute programs line by line. As your program executes, you can watch the values stored in variables, observe which methods are being called, and know what events have occurred in the program.

To demonstrate debugging, Let us use Listing 2.7, ComputeLoan.py, to demonstrate debugging. Create a new program named ShowCurrentTime.py in the pybook project.

9.1 Setting Breakpoints

You can execute a program line by line to trace it, but this is time-consuming if you are debugging a large program. Often, you know that some parts of the program work fine. It makes no sense to trace these parts when you only need to trace the lines of code that are likely to have bugs. In cases of this kind, you can use breakpoints.

A *breakpoint* is a stop sign placed on a line of source code that tells the debugger to pause when this line is encountered. The debugger executes every line until it encounters a breakpoint, so you can trace the part of the program at the breakpoint. Using the breakpoint, you can quickly move over the sections you know work correctly and concentrate on the sections causing problems.

There are several ways to set a breakpoint on a line. One quick way is to click the cutter of the line on which you want to put a breakpoint. You will see the line highlighted, as shown in Figure 15. You also can set breakpoints by choosing *Run*, *Toggle Line Breakpoint*. To remove a breakpoint, simply click the cutter of the line.

As you debug your program, you can set as many breakpoints as you want, and can remove breakpoints at any time during debugging. The project retains the breakpoints you have set when you exit the project. The breakpoints are restored when you reopen it.

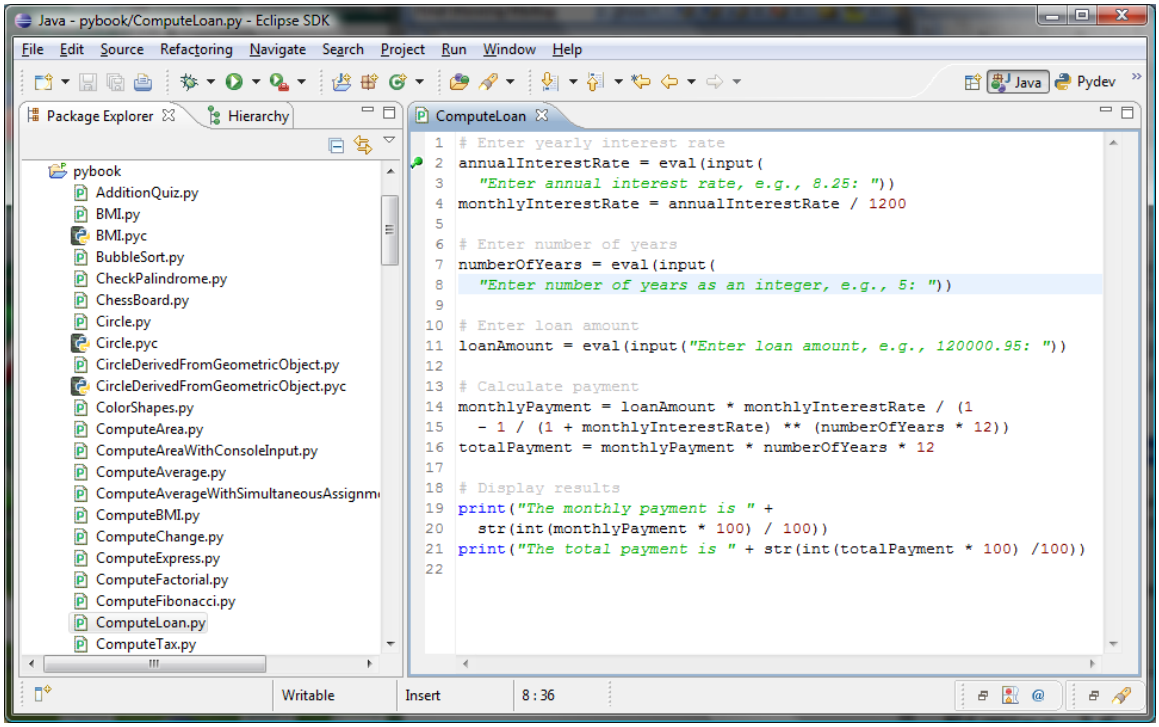

Figure 15

You can set breakpoints in the source code.

9.2 Starting the Debugger

There are several ways to start the debugger. A simple way is shown below:

1. Set a break point at the first statement in the program in the Source Editor.

2. Right-click on ComputeLoan.py in the project pane to display a context menu. Choose *Debug As*, *Python Run* to start debugging. You will first see the Confirm Perspective Switch dialog, as shown in Figure 16. Click *Yes* to switch to the Debug perspective. The UI for Debug perspective is shown in Figure 17.

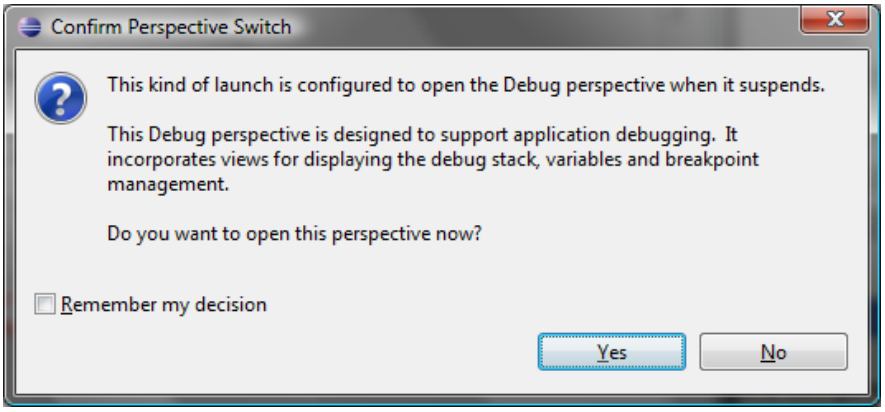

Figure 16

To start debug, Eclipse needs to switch to the Debug perspective.

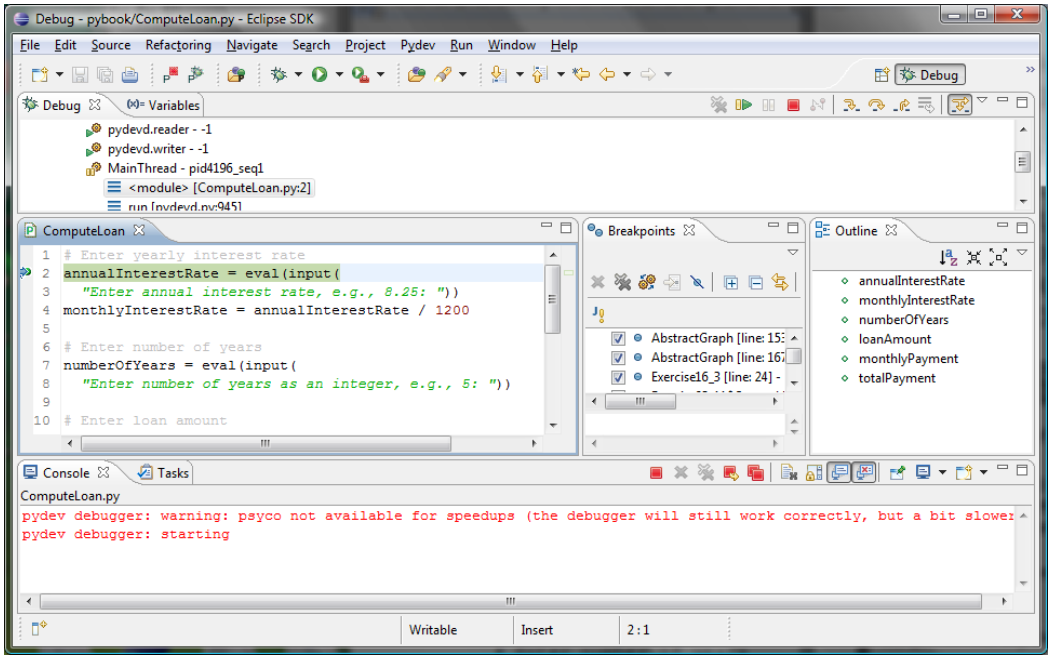

Figure 17

The debugger starts to run ComputeLoan.py.

9.3 Controlling Program Execution

The program pauses at the first line in the script. This line, called the *current execution point*, is highlighted in green. The execution point marks the next line of source code to be executed by the debugger.

When the program pauses at the execution point, you can issue debugging commands to control the execution of the

program. You also can inspect or modify the values of variables in the program.

When Eclipse is in the debugging mode, the toolbar buttons for debugging are displayed in the Debug window, as shown in Figure 17. The toolbar button commands also appear in the Run menu (see Figure 18). Here are the commands for controlling program execution:

- ***Resume*** resumes the execution of a paused program.
- ***Suspend*** temporarily stops execution of a program.
- ***Terminate*** ends the current debugging session.
- ***Step Into*** executes a single statement or steps into a method.
- ***Step Over*** executes a single statement. If the statement contains a call to a method, the entire method is executed without stepping through it.
- ***Step Return*** executes all the statements in the current method and returns to its caller.
- **Run to Line** runs the program, starting from the current execution point, and pauses and places the execution point on the line of code containing the cursor, or at a breakpoint.

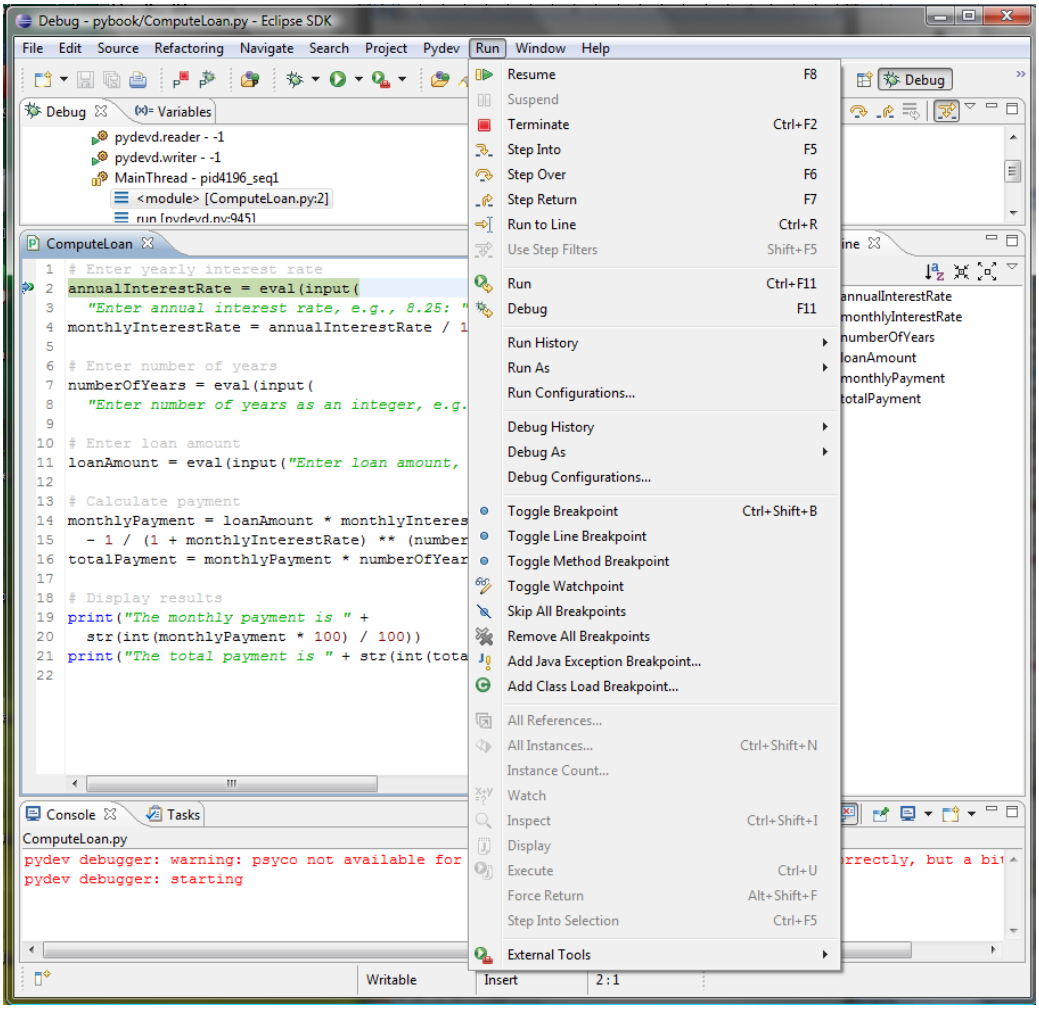

Figure 18

The debugging commands appear under the Debug menu.