Program Arcade Games

With Python and Pygame

Fourth Edition

Dr. Paul Vincent Craven

Apress®

Program Arcade Games

With Python and Pygame

Fourth Edition

Paul Vincent Craven

Apress®

Program Arcade Games: With Python and Pygame, Fourth Edition

Copyright © 2016 by Paul Vincent Craven

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

ISBN-13 (pbk): 978-1-4842-1789-4

ISBN-13 (electronic): 978-1-4842-1790-0

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director: Welmoed Spahr
Lead Editor: Steve Anglin
Editorial Board: Steve Anglin, Pramila Balan, Louise Corrigan, Jonathan Gennick, Robert Hutchinson, Celestin Suresh John, Michelle Lowman, James Markham, Susan McDermott, Matthew Moodie, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Gwenan Spearing
Coordinating Editor: Mark Powers
Copy Editor: Karen Jameson
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a Delaware corporation

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at www.apress.com/bulk-sales.

Any source code or other supplementary materials referenced by the author in this text is available to readers at www.apress.com/9781484217894. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/. Readers can also access source code at SpringerLink in the Supplementary Material section for each chapter.

This book is dedicated to everyone who loves to learn.

Contents at a Glance

Contents

About the Author

Paul Vincent Craven is a Computer Science professor at Simpson College in Indianola, Iowa. He worked in the IT industry for several years before switching to teaching full-time. He has a Ph.D. from the University of Idaho, a M.S. from Missouri University of Science and Technology, and B.A. from Simpson College.

Introduction

It all started in 1983 when my dad, who was also a teacher, bought an Apple //e computer for our use at home. Since it was to be "dad's computer" for educational purposes only, my brother and I were not allowed to purchase any games. So, at the local library I found two programming books by David H. Ahl: *BASIC Computer Games* and *More BASIC Computer Games*. These books had code I could use to type in and run my own games. This was the beginning of my creative outlet with computers.

As a computer science professor, I have found getting other people to program their own games as a great way to foster interest in computer science. Unfortunately, back when I started teaching students this way, the type of book I started with in the 1980s did not seem to be available any longer. I wanted to help others learn to program the same way I started. To provide a textbook for my students, I began to write my own programming book.

I started the website ProgramArcadeGames.com in 2009. The book you have in your hand morphed from the materials on that website and from student input from my beginning computer programming classes.

I would like to acknowledge and thank everyone who took time to give feedback, no matter how large or small. This book is the product of hundreds of students I have worked with personally, and the feedback of hundreds of people on-line. I continue to develop the website and use this book to share my love of programming with others.

CHAPTER 1

Before Getting Started...

This introductory chapter has two parts:

- Getting your computer set up to write games.
- Job and career prospects in technology.

Installing and Starting Python

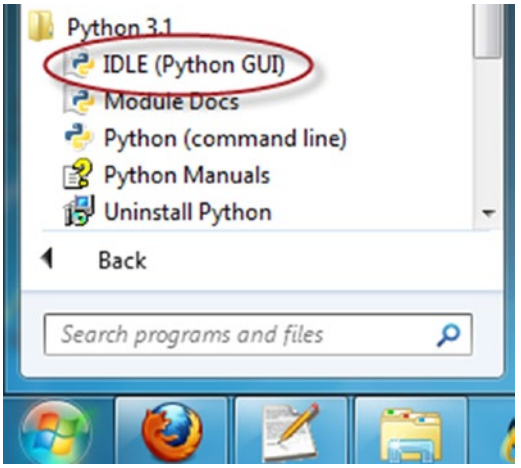
To get started, two programs need to be installed: Python and Pygame. Python is the computer language we will program in, and Pygame is a library of commands that will help make writing games easier.

Windows Installation

If you are working with a computer that already has Python and Pygame set up on it, you can skip this step. But if you want to set up Python and Pygame on your own Windows computer, don't worry. It is very easy.

1. Run the Python installer downloaded from
   `http://ProgramArcadeGames.com/python-3.4.3.msi`
2. Run the Pygame installer downloaded from
   `http://ProgramArcadeGames.com/pygame-1.9.2a0.win32-py3.4.msi`

Once everything has been installed, start Python up by selecting the Integrated Development Environment (IDLE) as shown in the figure.



Starting Python

The files provided above come from the Python download page at http://www.python.org/download/ and the Pygame file originally comes from https://bitbucket.org/pygame/pygame/downloads.

■ **Note** There are many versions of Python and Pygame. It can be complicated to get the correct versions and get them to work together. I recommend using the links on ProgramArcadeGames.com rather than downloading them from the Python and Pygame web sites.

If you must use a different version of Python than what is listed here, find a matching version of Pygame at this website: www.lfd.uci.edu/~gohlke/pythonlibs/#pygame.

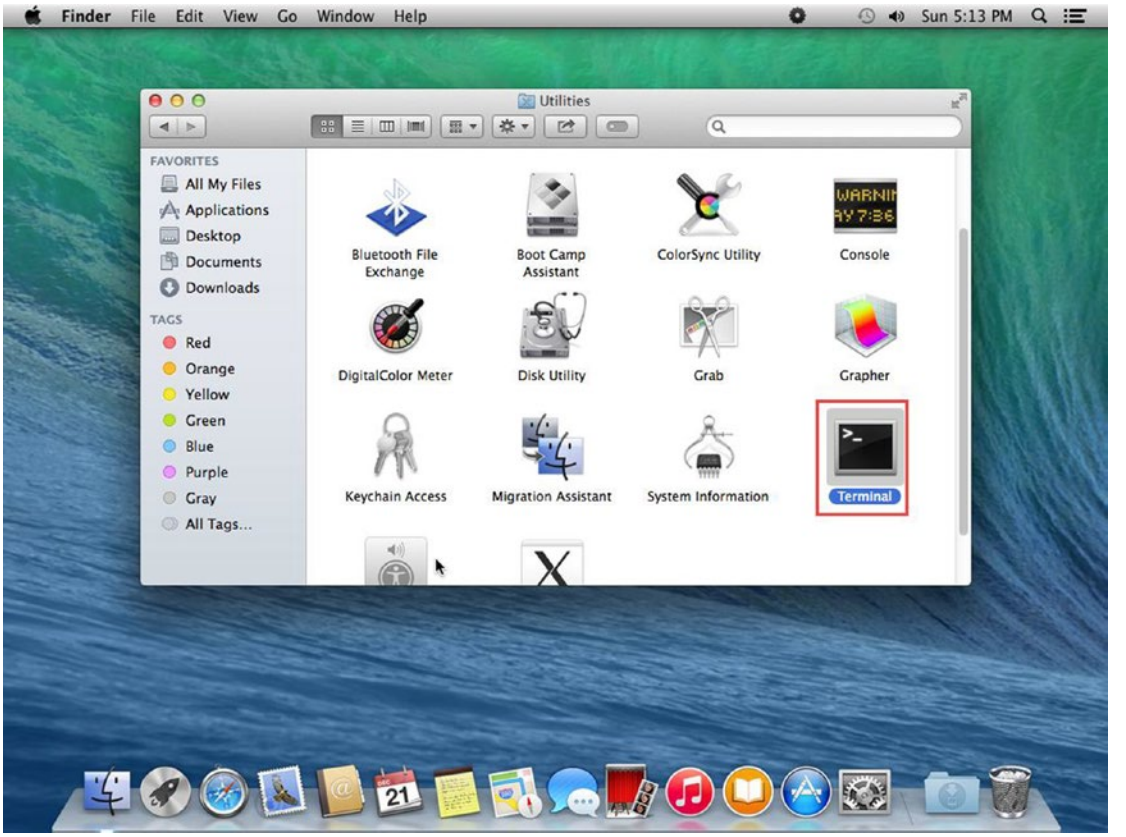
Mac Installation

The installation for the Mac is a bit involved, but it isn't too bad. Here are the steps.

1. Open up a terminal window. Click on "Finder" then "Applications" and then open "Utilities."

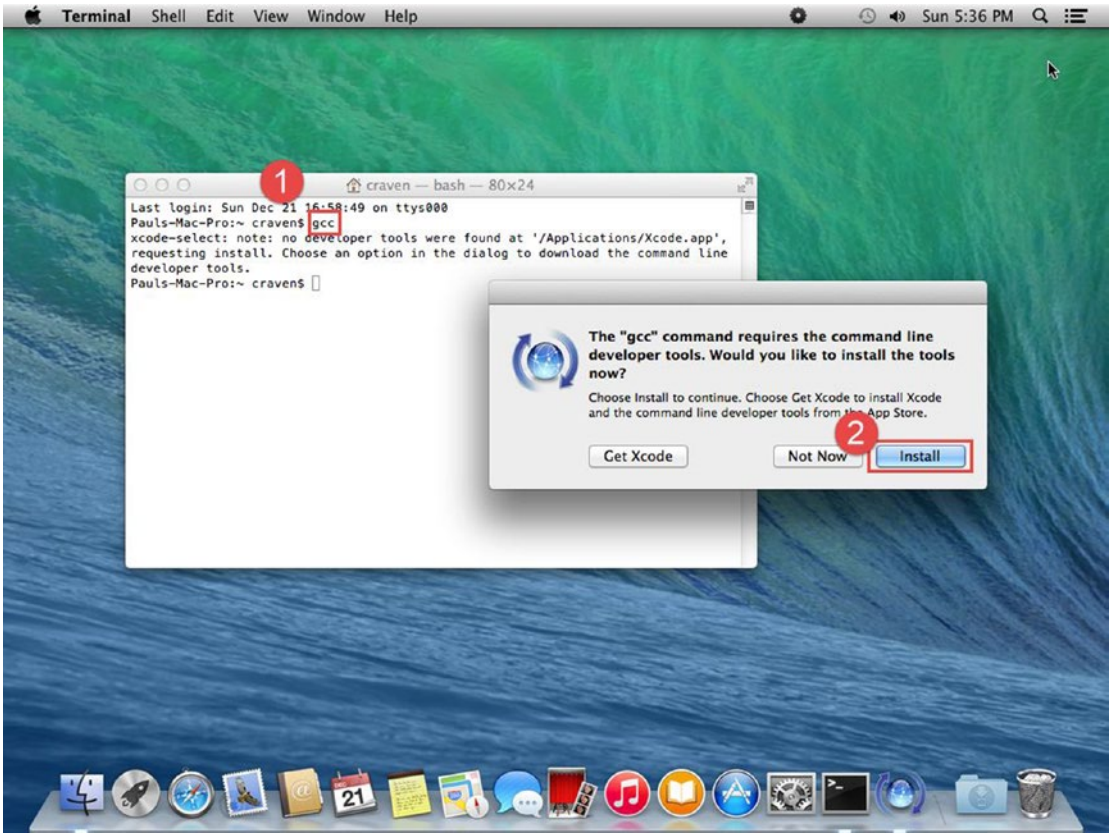
Starting a terminal window

2. Double-click on “Terminal.”



Starting a terminal window

3. We can issue commands to the Mac in the old-school style by typing them rather than pointing and clicking. We are going to start by typing in a command you probably don't have yet. This command is `gcc`. Type this and hit the `Enter` key. Your Mac will recognize that you don't have this command and offer to install it for you. Go ahead and do this. (If instead it says `error: no input files` you already have gcc, so go on to the next step.)



Starting a terminal window

4. Install XQuartz from: http://xquartz.macosforge.org.

5. Line by line, copy and paste the following items into your terminal window:

```
ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/
master/install)"
sudo brew doctor
brew update
brew install python3
brew install sdl sdl_image sdl_mixer sdl_ttf portmidi mercurial
```

6. If you want support for MP3s and movies, you can try adding smpeg. I've found support for this to be kind of spotty, so my recommendation is to skip this and use Ogg Vorbis files instead. But if you'd like to try, use these commands:

```
brew install --HEAD https://raw.github.com/Homebrew/homebrew-
headonly/master/smpeg.rb
```

7. Now you have all the supporting libraries. Let's finally install Pygame. Replace YourName with your account name. If you don't know what your account name is, type `ls /Users` to see all the user accounts on your computer.

```
cd /Users/YourName/Downloads
hg clone https://bitbucket.org/pygame/pygame
cd pygame
cd src
pip3 install /Users/YourName/Downloads/pygame
```

At this point, Pygame and Python should be up and running on your system. Python does not come with a way to edit files, so you will need to download an IDE like Wing IDE (http://wingware.com/downloads) or PyCharm (https://www.jetbrains.com/pycharm/download/), or some other editor.

Unix Installation

Unix and Unix-like distributions may come with a Pygame package or the ability to easily get one. If you want to compile from source, this is what I've used on Linux Mint (http://www.linuxmint.com/):

```
# Load required packages
sudo apt-get install mercurial libsdl1.2-dev
sudo apt-get install libasound2-doc libglib2.0-doc python3-dev
sudo apt-get install libsdl-ttf2.0-dev  libsdl-image1.2-dev
sudo apt-get install libsdl-mixer1.2-dev libportmidi-dev
sudo apt-get install libavformat-dev libswscale-dev
sudo apt-get install libfreetype6-dev
sudo apt-get install libsmpeg-dev

# Use mercurial to clone current code
hg clone https://bitbucket.org/pygame/pygame

# Build and install
cd pygame
sudo python3 setup.py
```

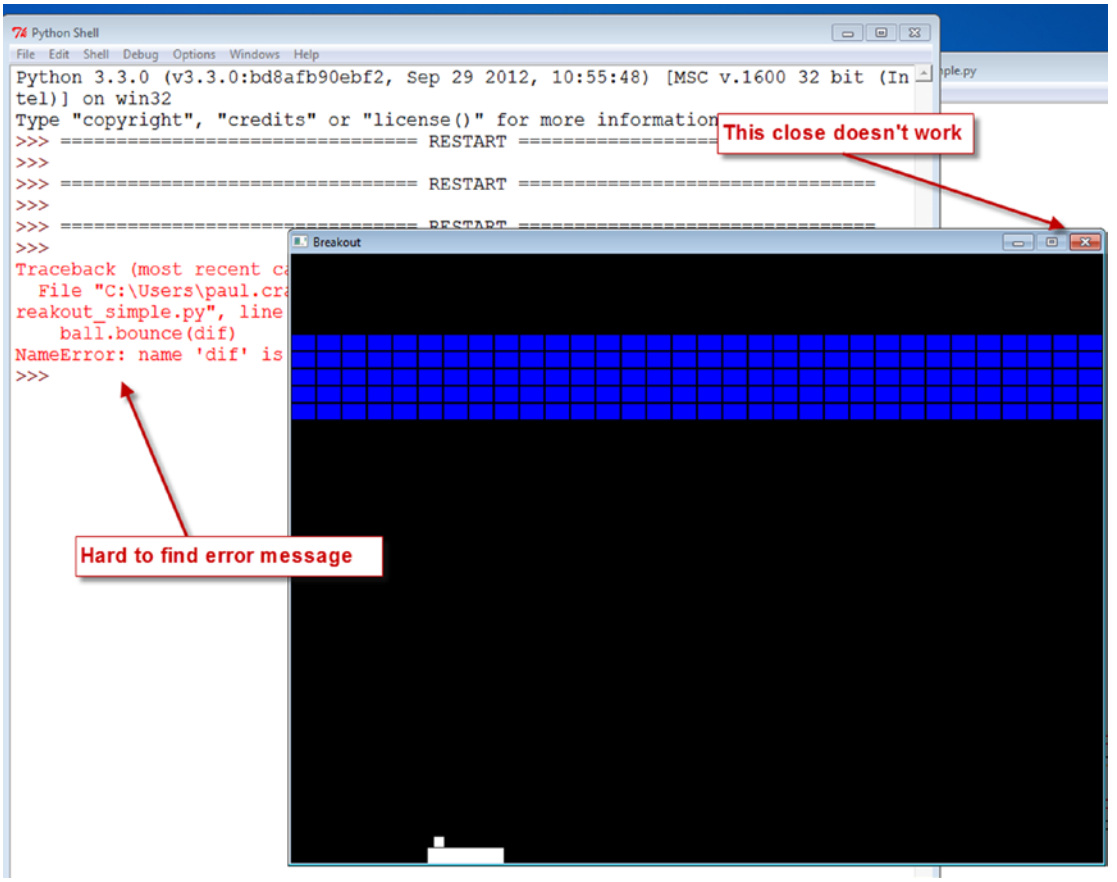
The biggest risk on UNIX platforms is that your default Python version might be in the 2.x series, and that code won't work with the code examples here in the book. Make sure you have and are using Python 3.x.

Optional Wing IDE

Python comes with an editor and an environment to develop code in. Unfortunately it isn't very good. Here are two issues you might run into when using Python's default editor:

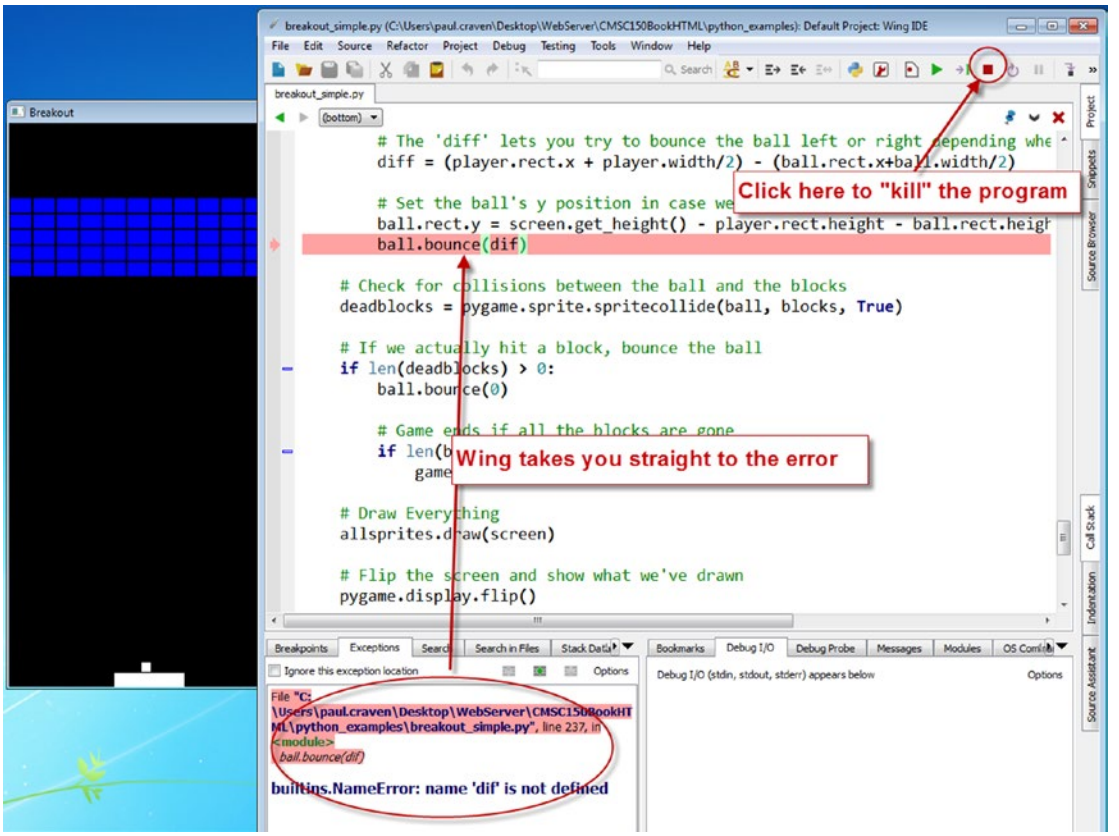
Issue 1. When working with multiple files it is difficult to keep track of the all the open files. It is easy to forget to save a file before running the program. When this happens the program runs with the old code that was saved rather than the new code. This is very confusing.

Issue 2. If there is an error in a program that does graphics the Python program will crash and hang. Once the program has crashed it is difficult to shut down. The error message that describes why it crashed is often buried and difficult to find. See the following figure.



Python Program Hanging in IDLE

The Wing editor solves issue 1 by using an editor with a tab for each file. It will also prompt to save all files before running a program. A program run under the Wing debugger does not hang as described in issue 2; instead the editor will immediately take the user to the line of code that caused the error. See the following figure.



Python Program Hanging in Wing IDE

Therefore, while it is yet a third thing to install, I recommend using the Wing editor. There is a free version called Wing IDE 101 at `wingware.com/downloads/wingide-101/`.

There is no need for all the bells and whistles the commercial version comes with, but they are nice. The program will often help you by auto-filling in variable names as you start to type them. If you've got extra money and want to save time you might want to pick up the commercial version.

In the videos on the web site I use either the default Python editor or the Wing editor. There are many other editors that can be used as well:

- PyCharm (`http://www.jetbrains.com/pycharm/`)
- Sublime (`http://www.sublimetext.com/`)
- PyDev on Eclipse (`http://pydev.org/`)
- Komodo Edit (`http://www.activestate.com/komodo-edit`)
- Notepad++ (`http://notepad-plus-plus.org/`)

Among some developers, discussing "which is the best editor?" is similar to getting a group of people together and discussing "which is the best religion?". It is best to pick your own favorite and then avoid this topic with other people.

Viewing File Extensions

It is a great idea to change your windows configuration to show file extensions. A file usually has a name like `Book report.docx` where the `.docx` tells the computer it is a Microsoft Word compatible document. By default Windows hides the `.docx` extension if there is a program installed to handle it. If you are programming, this hiding part of the file name can be annoying.

For Windows 7, to show file extensions, open up your computer's control panel. Find the selection for "Folder Options." Click the "View" tab, and then unselect the option for "Hide extensions for known file types."

For Windows 8, bring up a file explorer by hitting the Windows-E key. Then click the "view" tab and make sure "File name extensions" has been checked.

Learn to Make Games and Get Paid

As you start to learn to program, you might soon find that it looks like *work*. We all know we'd rather skip work and go farming for gold in World of Warcraft or Eve Online or some other game, right? So why learn to program? What does a person get out of it?

Bags of money

Learn how to make games and get paid? Ok, *I* won't pay you, but if you learn to program, there are plenty of people that *will* pay you. Here's how to profit:

1. Learn to program games.
2. Have fun making your own games.
3. Select a favorite job offer.
4. Profit.

Look, no ??? in this plan!

Think about it. You can *play* games, but anyone can do that. Being great at a video game really isn't much of an accomplishment in life if you think about it. Or you can learn to *create* games. People care about that.

Get the Most from This Book

Great basketball players practice. So do great programmers.

Looking to make your time here worthwhile? Answer the chapter questions! Don't skip them. They are necessary to understand the material.

Do the exercises! This is even more important. Learning by only reading the material is about as useful as trying to become an expert basketball player only by reading a book.

Practice! You might see other people that don't have to practice. It isn't fair. Or, you might be smarter than other people, and they start doing better than you because they work at it and you don't. That's not fair either. That's life. Get used to it. Practice.

Are you reading this book as part of a class? Great! Did you know you can save time and *copy* the answers and exercises from the Internet? You can also buy yourself a gym membership and send someone else to work out for you. It makes about as much sense.

Seriously, what on earth are you thinking copying from someone else? If you aren't going to do the work, stop reading now and start filling out McDonald's applications.

You *can't* learn without doing the work. Do the reading. Do the exercises.

Send Feedback

If you notice any errors or omissions in the book, please send me an e-mail. I'd like this to be the best resource possible.

Dr. Paul Vincent Craven
Department Head, Computer Science Department
Simpson College, Indianola, Iowa, 50125 USA
paul.craven@simpson.edu

CHAPTER 2

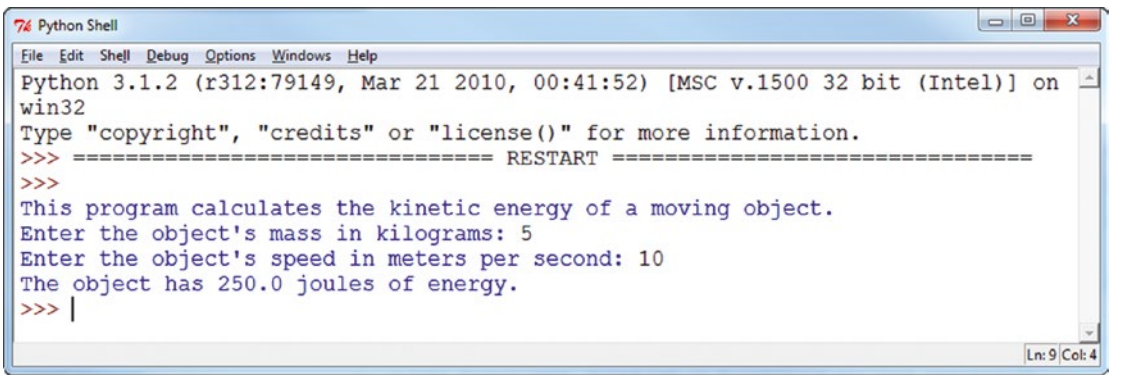
Create a Custom Calculator

One of the simplest things that can be done with Python is to use it as a fancy calculator. Wait, a calculator isn't a game. Why are we talking about calculators? Boring....

Hey, to calculate objects dropping, bullets flying, and high scores, we need calculations. Plus, any true geek will consider a calculator a toy rather than a torture device! Let's start our game education with calculators. Don't worry, we'll start graphics by Chapter 6.

A simple calculator program can be used to ask the user for information and then calculate boring things like mortgage payments, or more exciting things like the trajectory of mud balls as they are flung through the air.

The figure below shows an example program that calculates kinetic energy, something we might need to do as part of a game physics engine.



Using Python to calculate kinetic energy

The best thing about doing this as a program is the ability to hide the complexities of an equation. All the user needs to do is supply the information, and he or she can get the result in an easy-to-understand format. Any similar custom calculator could run on a smart phone, allowing a person to easily perform the calculation on the go.