Cracking Codes with Python

An Introduction to Building and Breaking Ciphers

by Al Sweigart

errata updated to print 7

Page	Error	Correction	Print corrected
xvi	If you're running Ubuntu, **install Python from the Ubuntu Software Center by following these steps:** **1. Open the Ubuntu Software Center.** **2. Type Python in the search box in the top-right corner of the window.** **3. Select IDLE (using Python 3.6), or whatever is the latest version.** **4. Click Install.**	If you're running Ubuntu, **Python is already installed, but you may have to install IDLE by following these steps:** **1. Open a Terminal window by pressing Ctrl-Shift-T.** **2. Run `sudo apt-get install idle3` (you will need the administrator password).**	Print 7

xvi	It doesn't come with Python, **so you'll need to install it by running the following in the interactive shell:**
	`ImportError: No module named pyperclip`
		It doesn't come with Python, **so you'll need to download it from *https://www.nostarch.com/crackingcodes/*. This file must be in the same folder (also called directory) as the Python program files you write. Otherwise you'll see the following error message when you try to run your programs:**
		`>>> import subprocess, sys;`
		`subprocess.run([sys.executable, '-m', 'pip', 'install', 'pyperclip'])`
		If you have trouble installing pyperclip, consult *https://pypi.org/project/pyperclip/*. Meanwhile, you can replace any lines of code that contain `pyperclip.copy()` or `pyperclip.paste()` with the `pass` Python keyword.
			Print 7

xvii	• On Windows 7 or newer, click the Start icon in the lower-left corner of your screen, enter **IDLE** in the search box, and select **IDLE (Python 3.6 64-bit)**. • On macOS, **open Finder, click Applications, click Python 3.6, and then click the IDLE icon.** • On Ubuntu, **select Applications ► Accessories ► Terminal and then enter `idle3`. (You may also be able to click Applications at the top of the screen, select Programming, and then click IDLE 3.)**	• On Windows 7 or newer, click the Start icon in the lower-left corner of your screen, enter **IDLE** in the search box, and select **IDLE (Python 3.10 64-bit)**. • On macOS, **open Spotlight by pressing COMMAND-spacebar and entering IDLE.** • On Ubuntu, **press the Win key and search for idle. Click the IDLE (using Python 3.10) item.**	Print 7
175	In this example, the 8-square rod is the longest rod that can fit evenly into 24 and 32.	In this example, the 8-square rod is the longest rod that can fit evenly into 24 and 16.	Print 5

Page	Error	Correction	Print corrected
209	`if keyIsValid(myKey):`	`if not keyIsValid(myKey):`	Print 2
212	`if keyIsValid(myKey):`	`if not keyIsValid(myKey):`	Print 2
250	There are 95,428,956,661,682,176 possible twelve-letter keys, but there are only about 1800 twelve-letter words in our dictionary file.	There are 95,428,956,661,682,176 possible twelve-letter keys, but there are only about 1,800 twelve-letter words in our dictionary file.	Print 7
283	`for word in lines:`	`for word in words:`	Print 2

325	`83.`
	`84.     # See if any of the low prime numbers can divide num:`
	`85.     for prime in LOW_PRIMES:`
	`86.         if (num % prime == 0):`
	`87.             return False`
	`88.`
		`83.     # See if any of the low prime numbers can divide num:`
		`84.     for prime in LOW_PRIMES:`
		`85.         if (num == prime):`
		`86.             return True`
		`87.         if (num % prime == 0):`
		`88.             return False`
			Print 2

333	Line 85 loops through each of the prime numbers in the LOW_PRIMES list:
	`84.     # See if any of the low prime numbers can divide num:`
	`85.     for prime in LOW_PRIMES:`
	`86.         if (num % prime == 0):`
	`87.             return False`
	The integer in `num` is modded by each prime number using the mod operator on line 86, and if the result evaluates to `0`, we know that `prime` divides `num` so `num` is not prime. In that case, line 87 returns `False`.
	Those are the two quick tests we'll perform to determine whether a number is prime. If the execution continues past line 87, the `rabinMiller()` function checks `num`'s primality.
		Line 84 loops through each of the prime numbers in the LOW_PRIMES list:
		`83.     # See if any of the low prime numbers can divide num:`
		`84.     for prime in LOW_PRIMES:`
		`85.         if (num == prime):`
		`86.             return True`
		`87.         if (num % prime == 0):`
		`88.             return False`
		If the integer in `num` is the same as `prime`, then obviously `num` must be a prime number and line 86 returns `True`.
		The integer in `num` is modded by each prime number using the mod operator on line 87, and if the result evaluates to `0`, we know that `prime` divides `num` so `num` is not prime. In that case, line 88 returns `False`.
		Those are the three quick tests we'll perform to determine whether a number is prime. If the execution continues past line 88, the `rabinMiller()` function checks `num`'s primality.
			Print 2

341	`64. print('The private key is a %s and a %s digit number.' %`
	`        (len(str(publicKey[0])), len(str(publicKey[1]))))`
		`64. print('The private key is a %s and a %s digit number.' %`
		`        (len(str(privateKey[0])), len(str(privateKey[1]))))`
			Print 2