A Few versions of the Primality Testing Program

JAN 30TH, 2012

Algorithmic Idea

- Generate all "candidate" factors of n, namely

  2, 3, …, n-1
- For each generated "candidate" factor, check if n is evenly divisible by the factor (i.e., the remainder is 0).
- If a "candidate" factor is found to be a real factor, then n is composite.
- If no "candidate" factor is found to be a real factor, then n is a prime.

Primality Testing: Version 1

```
# Programmer: Sriram Pemmaraju
# Date: Jan 30th, 2012
# This program reads a positive integer, greater than 1 and
# determines whether this integer is a prime or not.
# Version 1

n = int(raw_input("Please type a positive integer, greater than 1: "))

factor = 2 # initial value of possible factor
isPrime = True # variable to remember if n is a prime or not


# loop to generate and test all possible factors
while factor < n:
    # test if n is evenly divisible by factor
    if (n % factor == 0):
        isPrime = False
        break

    factor = factor + 1

# Output
if isPrime:
    print n, " is a prime."
else:
    print n, " is a composite."
```

Discussing this code: Boolean variables

- Boolean variables are quite useful for remembering situations that occurred in the program, for later reference.

- What happens if we get rid of the initialization:

```
      isPrime = True
```

- Could we have used a boolean variable called `isComposite` instead?

Discussing the code: The break statement

- The break statement forces the program to exit out of the smallest enclosing while-loop (or for-loop).
- **Example:**

```
n = 10
while n < 20:
	if n % 7 == 0:
		break
	n =n + 1
print n
```

Discussing the code: Comments in Python

- The program contains “comments,” i.e., text that is ignored by Python but serves to help the reader understand the code.
- These are preceded by the “#” symbol.
- Documenting code using comments is a critical part of programming.
- Comments are typically provided:
  - at the beginning of the program,
  - at the start of a block of code that performs a particular task, e.g., the while-loop that generates and tests factors,
  - to document the purpose of variables, etc.
- Later we will discuss a different mechanism for commenting a Python program called *documentation strings*.

Discussing the code: Basic guidelines for commenting

- Comments that contradict the code are worse than no comments at all!
- Comments that state the obvious (e.g., # This is a while-loop) make for unnecessary clutter are also worse than no comments at all.
- For now the comments you write should (i) help the reader understand your algorithm and (ii) help the reader understand tricky snippets of code.
- Your intended audience: your classmates, your graders, yourself a few weeks into the future.

Improving the efficiency of our program

1. A number n does not have any factors larger than n/2, except itself. So we could stop generating candidate factors at n/2.

2. But, wait we can do much better!

We know **√n x √n = n**. Hence, if n has a factor larger than √n, then it has a factor smaller than √n also.

This means that only factors 2, 3,…, floor(√n) need to be considered.

Example

- Say n = 123.
- $\sqrt{123}$ = 11.0905365064094l8.

- So if 123 has a factor greater than 11.09, then it has factor less than 11.09.

- This means in looking at “candidate” factors, we only need to look at numbers 2, 3, …, 11.

Primality Testing: Version 2

```
# Programmer: Sriram Pemmaraju
# Date: Jan 30th, 2012
# This program reads a positive integer, greater than 1 and
# determines whether this integer is a prime or not.
# Version 2

import math

n = int(raw_input("Please type a positive integer, greater than 1: "))

factor = 2 # initial value of possible factor
isPrime = True # variable to remember if n is a prime or not
factorUpperBound = math.sqrt(n) # the largest possible factor we need to test is sqrt(n)

# loop to generate and test all possible factors
while factor <= factorUpperBound:
    # test if n is evenly divisible by factor
    if (n % factor == 0):
        isPrime = False
        break

    factor = factor + 1

# Output
if isPrime:
    print n, " is a prime."
else:
    print n, " is a composite."
```

How do I find out if a certain math function is part of the math module in Python?

- For all matters related to Python

  http://docs.python.org/

is the authoratative source. I visit this website all the time when I program in Python.

- Get into the habit of searching this website for answers to all Python-related questions.

- This is a good time for you to look over parts of the Python tutorial (e.g., 3.1.1 Numbers, 3.1.2 Strings, 3.2 First Steps Towards Programming, 4.1 If statements).

- Section 9.2 is on the math module and contains a list of math functions available in the module.

Examples of functions in the math module

- `math.log10(x)`: returns the logarithm to the base 10 of x.
- `math.pow(x, y)`: returns x raised to the power of y.

There are many other functions in this module. Play with these!