modules, more functions, hexadecimal, tuples

Ben Bolker

23 September 2019

# Modules

Collections of functions you might want to use.

## importing

* use import to make functions inside modules available
* refer to functions via module prefix
* import VeryLongModuleName as vlmn: use abbreviation
* can import just one or two functions: from math import sqrt, log
* can import everything (but usually don’t): from <module> import \*
* can import *your own modules* (i.e., functions in a .py file)

## finding out about modules

* help("modulename")
* [official modules](https://docs.python.org/3/py-modindex.html)
* [list of useful modules](https://wiki.python.org/moin/UsefulModules)
* some modules we will definitely be using:
  + math: basic math functions
  + matplotlib: drawing pictures
  + random: picking random numbers
  + numpy: numerical computation  
    (including linear algebra and some calculus)
  + pandas: data analysis
* more tangential but maybe used:
  + nose: code testing framework
  + scipy: even more scientific computing tools
  + cmath: math functions handling complex numbers
  + re: regular expressions
  + sympy: symbolic computation
  + timeit: how long does my code take?

## Functions calling functions

* You can pass anything to a function as an argument (even a function!)

def repeat\_fun(f,startval,n):  
 """Given a function f and a starting value startval,  
 apply the function n times (each time using the previous  
 result as input)  
 """  
 y = startval  
 for i in range(n):  
 y=f(y)  
 return(y)  
  
def sqr(x):  
 return(x\*x)  
  
repeat\_fun(sqr,3,3)

## 6561

## Function composition

* Mathematically this kind of example is called **composition** of a function with itself (see [Wikipedia](https://en.wikipedia.org/wiki/Function_composition)
* in math notation:
* (notation for multiple composition of a function with itself [is harder](https://math.stackexchange.com/questions/926247/notation-for-repeated-composition-of-functions))
* write a function compose\_funs(f,g)

## Recursion

Functions can even call themselves! This is like mathematical [induction](https://en.wikipedia.org/wiki/Mathematical_induction).

def factorial(x):  
 if (x==1):  
 return(1)  
 return(x\*factorial(x-1))  
  
factorial(5)

## 120

## Scope

* Where does Python look for things?
* What happens here?

z = 1  
def add\_z(x):  
 return(x+z)  
  
add\_z(z)

## 2

## Scoping rules

**LEGB** (Local, Enclosing, Global, Built-in) - *Local*: symbols defined in the function, and arguments - *Enclosing*: symbols defined *in the function within which this function was defined* - *Global*: elsewhere in the file/module - *Built-in*: Python keywords

## Hexadecimal/Decimal conversion

* The **hexadecimal** (or “base 16”) numeral system uses sixteen distinct digits to represent integers.
* The digits used are: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f .
* The decimal value of the digit a is 10, b is 11, etc.
* The hexadecimal number 2c is equal to (base 10).
* Similarly, 2be13 is equal to 179731 since
* The number 1020304 in hexadecimal is f9190. This can be verified by expanding f9190 as
* which is equal to

## Problem

* Write Python code that takes as input from the console two strings that represent numbers in the hexadecimal system.
* The program should should print out the representations of these numbers in base 10, and also print a string that represents the sum of these numbers in hexadecimal.

## High level description of the algorithm

1. Input the two strings from the console.
2. Convert each string into a base 10 number.
3. Print out these two numbers.
4. Convert the sum of these two numbers into hexadecimal.
5. Print out this hexadecimal number.

* For Step 1, use the input() function.
* Create a function get\_hex\_string() that gets a string from the console that represents a hexadecimal number and returns that string.
* Should it check to see if it is a legal string, i.e., only uses 0 − 9, and a − f ?

## convert hexadecimal into decimal

* if an integer is represented in hexadecimal by the string of length word
* then it is equal to the number:
* So to convert word into decimal, we can iterate over each digit in word to produce the required value.
* Note that the term in the above sum is equal to $h\_{n−j−1} ∗ 16^{n−j−1} , with and that the digit is just word[j].
* **next step**: Create a function hex\_to\_decimal(hex\_String) with string argument hex\_string that will returns the value of the base-10 integer this string represents in hexadecimal …

## convert to hexadecimal

* To find the hexadecimal digits of the non-negative base-10 integer num we use // and %.
  + h[0] = num % 16
  + h[1] = (num // 16) % 16
  + h[2] = (num // 16\*\*2 ) % 16
  + …
  + h[i] = (num// 16\*\*i ) % 16
* (But we can do this more easily as a variation of the **coin-counting problem** …
* Q: How do we decide when to stop?
* **next step**: Produce a function decimal\_to\_hex(num) that computes the hexadecimal representation of the int num and returns this as a string.
* To finish, use these functions to produce the final result.