Assignment 1 for Spatial Programming

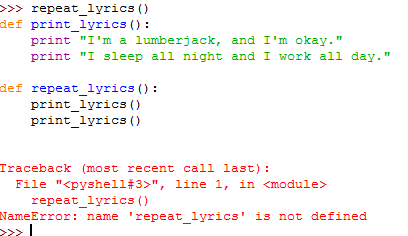
Jan 24, 2015

Xiaohu Liu

**Chapter 4**

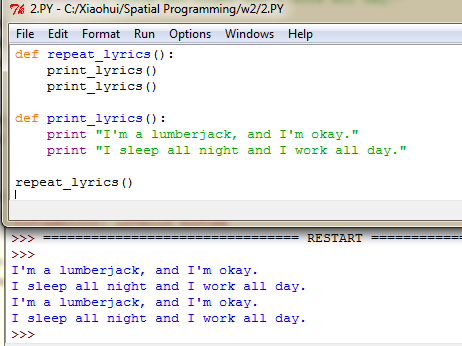
**Exercise 1**

*Move the last line of this program to the top, so the function call appears before the definitions. Run the program and see what error message you get.*



**Exercise 2**

*Move the function call back to the bottom and move the definition of print\_lyrics after the definition of repeat\_lyrics. What happens when you run this program?*

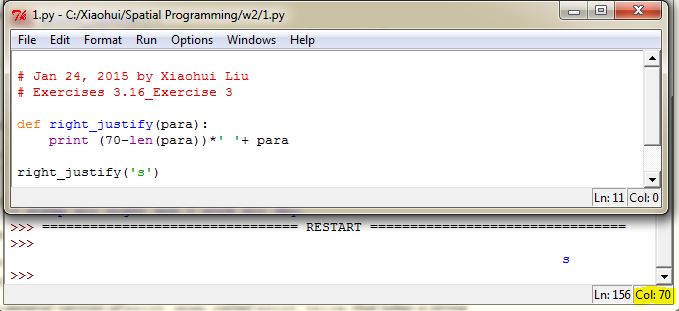


This program works well because the *repeat\_lyrics* was called after *print\_lyrics* was defined. The flow of execution is not the same as sequence of appearance in the program.

**Exercise 3**  

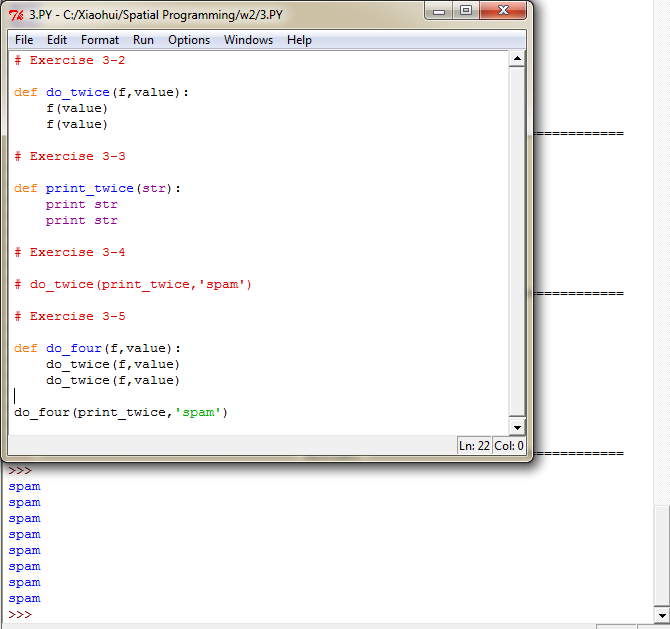
*Python provides a built-in function called len that returns the length of a string, so the value of len('allen') is 5.*

*Write a function named right\_justify that takes a string named s as a parameter and prints the string with enough leading spaces so that the last letter of the string is in column 70 of the display.*



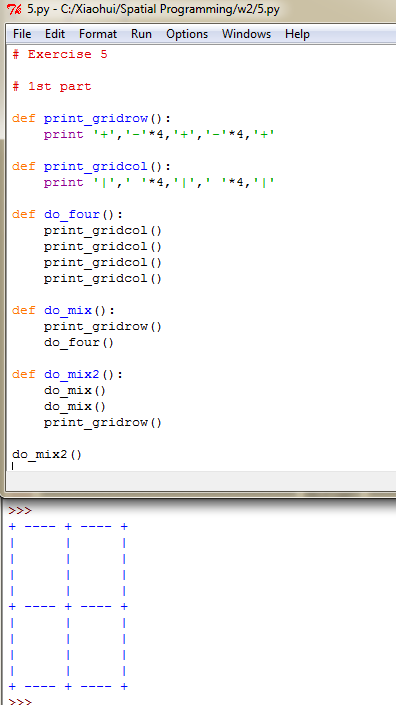
**Exercise 4**

1. *Type this example into a script and test it.*
2. *Modify do\_twice so that it takes two arguments, a function object and a value, and calls the function twice, passing the value as an argument.*
3. *Write a more general version of print\_spam, called print\_twice, that takes a string as a parameter and prints it twice.*
4. *Use the modified version of do\_twice to call print\_twice twice, passing 'spam' as an argument.*
5. *Define a new function called do\_four that takes a function object and a value and calls the function four times, passing the value as a parameter. There should be only two statements in the body of this function, not four.*

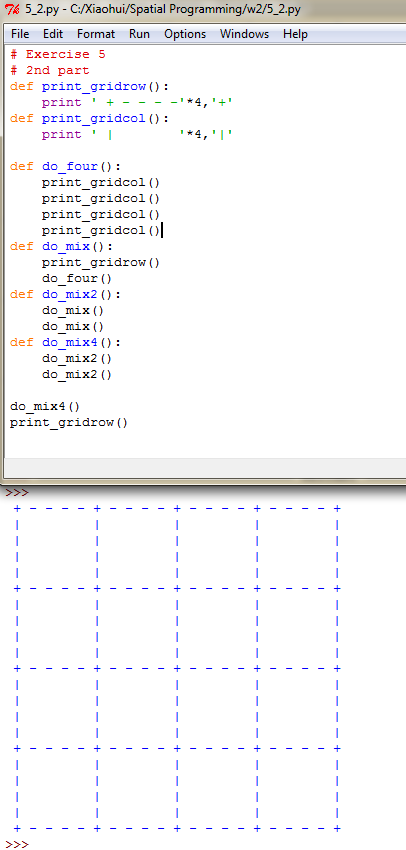


**Exercise 5**

1 2\*2 grid:



2. 4 by 4 grid:



**Chapter 5 Conditions and Recursion**

1. **Draw a stack diagram for *print\_n.***

s🡪’Hello’,n=2

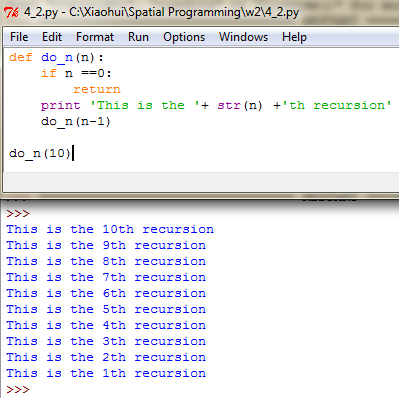
s🡪’Hello’,n=1

print\_n

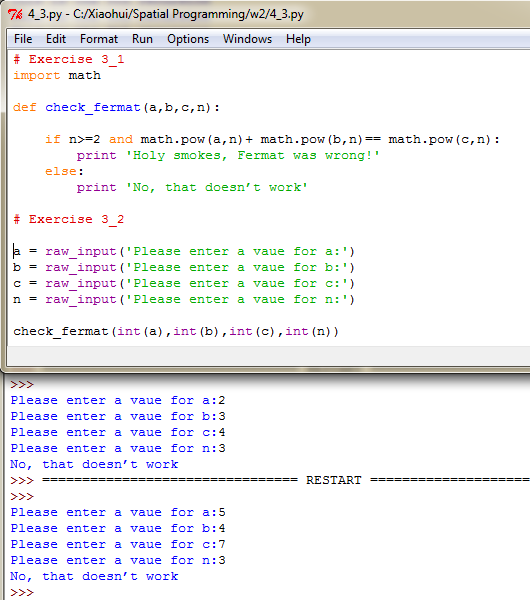
print\_n

<module>

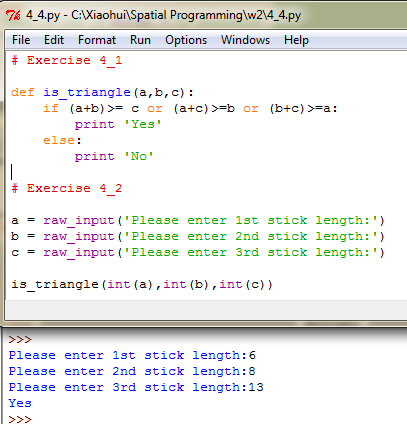
1. **Write a function called *do\_n* that takes a function object and a number, *n*, as arguments, and that calls the given** function *n* times.



1. **Fermat’s Last Theorem**

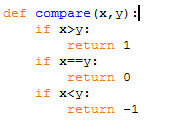


1. **Testify the possibility to form a triangle.**

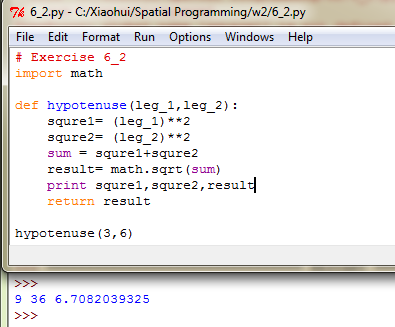


**Chapter 6 Fruitful Functions**

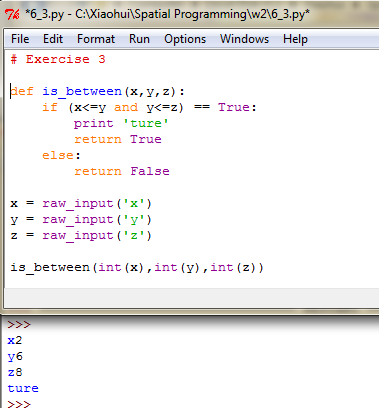
**Ex 1 Write a *compare* function that returns *1* if *x > y*, *0* if *x == y*, and *-1* if *x < y*.**



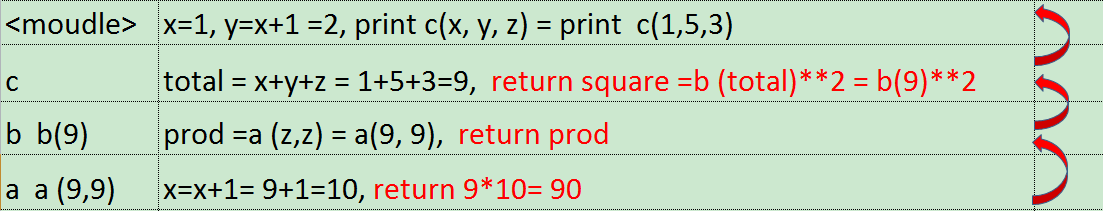
**Ex2 a function called *hypotenuse***



**Ex 3 Write a function *is\_between(x, y, z)* that returns *True* if x ≤ y ≤ z or *False* otherwise.**

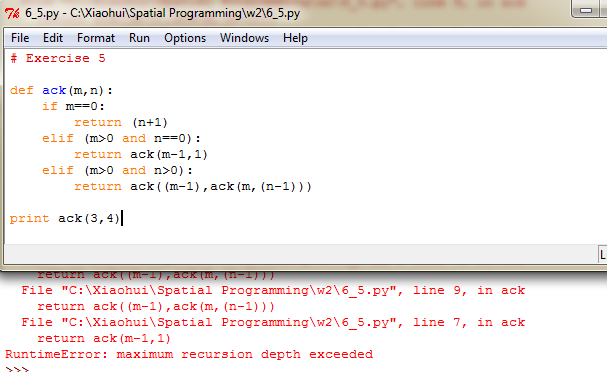


**EX4 Draw a stack diagram for the following program. What does the program print?**



It prints 9,10 and 8100

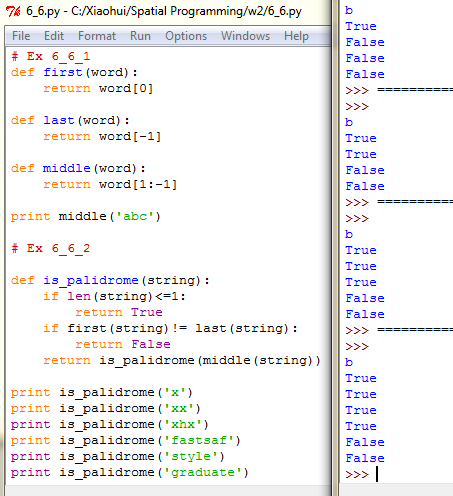
**Ex 5 Write a function named *ack* that evaluates Ackermann’s function.**



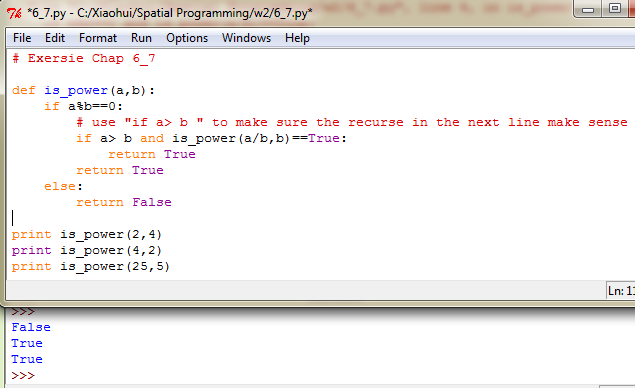
If large values of m and n (eg. 100 & 50) are passed to the function, a RuntimeError will show up, indicating that maximum recursion depth exceeded.

**Ex 6 The following are functions that take a string argument and return the first, last, and middle letters:**

1. If the middle function is called with one, two, or zero letter as parameter, nothing will be returned.
2. Write a function called *is\_palindrome* that takes a string argument and returns *True* if it is a palindrome and *False* otherwise. Remember that you can use the built-in function *len* to check the length of a string.



**Exer 7 A number, a, is a power of b if it is divisible by b and a/b is a power of b. Write a function called *is\_power* that takes parameters *a* and *b* and returns *True* if *a* is a power of *b*. Note: you will have to think about the base case.**



**Exe 8 Write a function called *gcd* that takes parameters *a* and *b* and returns their greatest common divisor.**

