Done

Advanced Python

Introduction

The purpose of this practical is to continue using Python to develop programs, using some advanced features of the language. Don't forget that you are required to submit the indicated exercise through myLearn for assessment.

Python

We can now write quite powerful Python programs (in fact, we can write any computation that our computer can perform using only the Python we knew before this week). But there are some advanced Python techniques that make things easier for us. For example, rather than having to type input all the time (or redirect files to standard input), it can be useful to read data from files. Similarly, saving data to a file means our program can stop and, the next time it starts, it can read in the file and continue from where it was before. Other concepts just make things a bit easier for us. Recursion, for example, is often easier to understand than the iterative equivalent. And then there's just some syntax that can be used to make our code more readable. This week's exercises will make use of the new features we have learnt.

Exercises

Use material from this week's lectures to perform the following exercises:

1. **[This exercise should be submitted through the Assessments section of myLearn for Tutorial Exercise 6]** Write a recursive function called *is_palindrome* that takes a string named *word* as its parameter and returns *True* if word has length less than or equal to *1*. If the length of *word* is greater than *1*, the function should return *False* if the first character of *word* is different to the last character of *word*. If the length of *word* is greater than *1*, and the first and last characters of the string are identical, the function should return the result of *is_palindrome()* with the parameter of *word* with its first and last characters removed (e.g. *is_palindrome("anna")* should return the result of *is_palindrome("nn")*).
2. Download the file https://raw.githubusercontent.com/AllenDowney/ThinkPython2/master/code/words.txt and use your code from 1) to list all palindromes in the file
3. Draw a stack diagram that shows the state of this program when it prints its result:

```
def recurse(n, s):
  if n == 0:
    print(s)
  else:
    recurse(n - 1, n + s)

recurse(3, 0)
```

4. What would happen if you called the *recurse* function from question 2 with the parameters *-1* and *0*?
5. Write a program that searches a directory and all of its subdirectories, recursively, and returns a list of complete paths for all files with a given suffix (e.g. *.txt*). Hint: `os.path` provides functions for manipulating file and path names.
6. The following function recursively computes the binomial coefficient. Rewrite the body of the function using nested conditional expressions.

```
def binomial_coeff(n, k):
  """Compute the binomial coefficient "n choose k".
  n: number of trials
  k: number of successes
  returns: int
  """

  if k == 0:
    return 1
  if n == 0:
    return 0

  res = binomial_coeff(n-1, k) + binomial_coeff(n-1, k-1)
  return res
```

Last modified: Thursday, 1 February 2024, 10:51 AM