How can I build a recursive function in python? [duplicate]

This question already has an answer here:
Basics of recursion in Python *3 answers*

How can I build a recursive function in python?

python recursion

edited Sep 25 '12 at 17:45
Robert Gowland
3,679 3 29 47

asked Jan 26 '09 at 10:24
sddaf ads

marked as duplicate by thefourtheye python May 24 '15 at 3:57

This question has been asked before and already has an answer. If those answers do not fully address your question, please ask a new question.

1 People aren't sure if you mean recurrent or recursive. Please check your terminology. – Triptych Jan 26 '09 at 10:38

1 Recurrence == Recursive. en.wikipedia.org/wiki/Recurrence_relation – S.Lott Jan 26 '09 at 12:12

4 Answers

I'm wondering whether you meant "recursive". Here is a simple example of a recursive function to compute the factorial function:

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
```

The two key elements of a recursive algorithm are:

- The termination condition: `n == 0`
- The reduction step where the function calls itself with a smaller number each time: `factorial(n - 1)`

answered Jan 26 '09 at 10:28
Greg Hewgill
555k 116 920
1084

1 A side note: not all function must halt to be useful. For example, itertools.count(). BTW, it could be implemented recursively. – J.F. Sebastian Jan 26 '09 at 21:59

I have a query here. What would happen if I made one function do the multiplication if n is even and another if n is odd. These to functions call each other obviously. `def even(n): return n*odd(n-1) def odd(n): if n==1: return 1 else: return n*even(n-1)` Assuming I can initially call the correct even or odd function, what would happen? – fledgling Jul 28 '12 at 2:39

@Ayos: Assuming your functions are implemented correctly, that would work fine too. That situation is called *mutually recursive functions*. – Greg Hewgill Jul 28 '12 at 5:21

@GregHewgill I seem to get a RuntimeError for exceeding max Recursion depth, even though I know it's not an infinite recursion. – fledgling Jul 28 '12 at 9:48

Well, how far is it recursing? Python's default recursion depth limit is 5000 or something. – Greg Hewgill Jul 28 '12 at 9:59

Recursion in Python works just as recursion in an other language, with the recursive construct defined in terms of itself:

For example a recursive class could be a binary tree (or any tree):

```
class tree():
    def __init__(self):
        '''Initialise the tree'''
        self.Data = None
        self.Count = 0
        self.LeftSubtree = None
        self.RightSubtree = None

    def Insert(self, data):
        '''Add an item of data to the tree'''
        if self.Data == None:
            self.Data = data
            self.Count += 1
        elif data < self.Data:
            if self.LeftSubtree == None:
                # tree is a recurive class definition
                self.LeftSubtree = tree()
            # Insert is a recursive function
            self.LeftSubtree.Insert(data)
        elif data == self.Data:
            self.Count += 1
        elif data > self.Data:
            if self.RightSubtree == None:
                self.RightSubtree = tree()
            self.RightSubtree.Insert(data)

if __name__ == '__main__':
    T = tree()
    # The root node
    T.Insert('b')
    # Will be put into the left subtree
    T.Insert('a')
    # Will be put into the right subtree
    T.Insert('c')
```

As already mentioned a recursive structure must have a termination condition. In this class, it is not so obvious because it only recurses if new elements are added, and only does it a single time extra.

Also worth noting, python by default has a limit to the depth of recursion available, to avoid absorbing all of the computer's memory. On my computer this is 1000. I don't know if this changes depending on hardware, etc. To see yours :

```
import sys
sys.getrecursionlimit()
```

and to set it :

```
import sys #(if you haven't already)
sys.setrecursionlimit()
```

edit: I can't guarentee that my binary tree is the most efficient design ever. If anyone can improve it, I'd be happy to hear how

answered Jan 26 '09 at 19:25

dangerouslyfacetious

198 1 8

Let's say you want to build: u(n+1)=f(u(n)) with u(0)=u0

One solution is to define a simple recursive function:

```
u0 = ...

def f(x):
  ...

def u(n):
  if n==0: return u0
  return f(u(n-1))
```

Unfortunately, if you want to calculate high values of u, you will run into a stack overflow error.

Another solution is a simple loop:

```
def u(n):
  ux = u0
  for i in xrange(n):
    ux=f(ux)
  return ux
```

But if you want multiple values of u for different values of n, this is suboptimal. You could cache all values in an array, but you may run into an out of memory error. You may want to use generators instead:

```
def u(n):
  ux = u0
  for i in xrange(n):
    ux=f(ux)
  yield ux

for val in u(1000):
  print val
```

There are many other options, but I guess these are the main ones.

answered Jan 26 '09 at 10:35
MiniQuark
15.8k 16 86 122

The original question said "recurrent", not "recursive", hence my answer, which does not make much sense now that the question has been modified. – MiniQuark Jan 26 '09 at 14:23

Interesting answer, but I don't get the point of the last example – lajarre Aug 28 '13 at 13:14

Recursive function example:

```
def recursive(string, num):
    print "#%s - %s" % (string, num)
    recursive(string, num+1)
```

Run it with:

```
recursive("Hello world", 0)
```

edited Mar 11 '14 at 21:07
James
4,063 2 27 66

answered Jan 26 '09 at 17:07
Evan Fosmark
38.5k 25 84 111

4 How does this recursion routine end/exit? – Ketan Nov 29 '13 at 7:49

1 True. Even I'm curious, how will this end? As per my knowledge there should be trivial statement in every recursion. – Nagaraj Tantri Dec 4 '13 at 14:27

3 It will end with `RuntimeError: maximum recursion depth exceeded` as there is no statement that will limit the recursion depth. – Emanuel Ey Sep 8 '15 at 16:40