PythonTip 03 - Recursion

February 17, 2021

1 Recursion

A recursive algorithm is an algorithm that calls itself. You need a base case so you don't get stuck in an infinite loop.

Example: suppose we want to calculate the quantity $n! = n(n-1)(n-2)\cdots 3 \cdot 2 \cdot 1$.

We'll use the fact that $n! = n \cdot (n-1)!$.

[2]:
```
# What's wrong with this function?
def factorial(n):
    return n * factorial(n-1)
```

[3]:
```
factorial(3)
```

```
---------------------------------------------------------------------------
RecursionError                            Traceback (most recent call last)
<ipython-input-3-3fd9b1939623> in <module>
----> 1 factorial(3)

<ipython-input-2-c71237f782fa> in factorial(n)
      1 # What's wrong with this function?
      2 def factorial(n):
----> 3     return n * factorial(n-1)

… last 1 frames repeated, from the frame below …

<ipython-input-2-c71237f782fa> in factorial(n)
      1 # What's wrong with this function?
      2 def factorial(n):
----> 3     return n * factorial(n-1)

RecursionError: maximum recursion depth exceeded
```

[5]:
```
# We need a base case!
def factorial(n):
    if n == 1:
        return 1
```

```
    return n * factorial(n-1)
```

[12]:
```
factorial(5)
```

[12]:
```
120
```

```
factorial(5)
5 * factorial(4)
5 * (4 * factorial(3))
5 * (4 * (3 * factorial(2)))
5 * (4 * (3 * (2 * factorial(1))))
5 * (4 * (3 * (2 * 1)))
```

[13]:
```
factorial(1000)
```

[13]:
```
40238726007709377354370243392300398571937486421071463254379991042993851239862902
05920442084869694048004799886101971960586316668729948085589013238296699445909974
24504087073759918823627727188732519779505950995276120874975462497043601418278094
64649629105639388743788648733711918104582578364784997701247663288983595573543251
31853239584630755574091142624174743493475534286465766116677973966688202912073791
43853719588249808126867838374559731746136085379534524221586593201928090878297308
43139284440328123155861103697680135730421616874760967587134831202547858932076716
91324484262361314125087802080002616831510273418279777047846358681701643650241536
91398281264810213092761244896359928705114964975419909342221566832572080821333186
11681155361583654698404670897560290095053761647584772842188967964624494516076535
34081989013854424879849599533191017233555566021394503997362807501378376153071277
61926849034352625200015888535147331611702103968175921510907788019393178114194545
25722386554146106289218796022383897147608850627686296714667469756291123408243920
81601537808898939645182632436716167621791689097799119037540312746222899880051954
44414282012187361745992642956581746628302955570299024324153181617210465832036786
90611726015878352075151628422554026517048330422614397428693306169089796848259012
54583271682264580665267699586526822728070757813918581788896522081643483448259932
66043367660176999612831860788386150279465955131156552036093988180612138558600301
43569452722420634463179746059468257310379008402443243846565724501440282188525247
09351906209290231364932734975655139587205596542287497740114133469627154228458623
77387538230483865688976461927383814900140767310446640259899490222221765904339901
88601856652648506179970235619389701786004081188972991831102117122984590164192106
88843871218556461249607987229085192968193723886426148396573822911231250241866493
53143970137428531926649875337218940694281434118520158014123344828015051399694290
15348307764456909907315243327828826986460278986432113908350621709500259738986355
42771967428222487575867657523442202075736305694988250879689281627538488633969099
59826280956121450994871701244516461260379029309120889086942028510640182154399457
15680594187274899809425474217358240106367740459574178516082923013535808184009699
63725242305608559037006242712434169090041536901059339838357779394109700277534720
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000
```

[ ]: