Programming and Data Structures with Python

Lecture 07, 07 January 2021 ¶

In [1]:

```
def f():
  y = x
  print(y)

x = 7
f()
```

```
7
```

In [ ]:

```
def f():
  y = x
  print(y)
  x = 22

x = 7
f()
```

In [3]:

```
def f():
  y = l[0]
  print(y)
  l[0] = 22

l = [7]
f()
```

```
7
```

In [35]:

```
def f():
  l = [22]
  y = l[0]
  print(y)
#  l = [22]  # Creates a local list name l

l = [7]
f()
print(l)
```

```
22
[7]
```

In [5]:

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

In [6]:

```
factorial(-1)
```

```
---------------------------------------------------------------
-----------------
RecursionError                            Traceback (most
 recent call last)
<ipython-input-6-5aae425d6a8b> in <module>
----> 1 factorial(-1)

<ipython-input-5-7a1622528c4a> in factorial(n)
      3       return(1)
      4    else:
----> 5       return(n * factorial(n-1))

... last 1 frames repeated, from the frame below ...

<ipython-input-5-7a1622528c4a> in factorial(n)
      3       return(1)
      4    else:
----> 5       return(n * factorial(n-1))

RecursionError: maximum recursion depth exceeded in compar
ison
```

In [8]:

```
factorial(20)
```

Out[8]:

```
2432902008176640000
```

In [9]:

```
factorial(999)
```

Out[9]:

```
40238726007709377354370243392300398571937486421071463254
37999104299385123986290205920442084869694048004799886101
97196058631666872994808558901323829669944590997424504087
07375991882362772718873251977950595099527612087497546249
70436014182780946464962910563938874378864873371191810458
25783647849977012476632889835955735432513185323958463075
55740911426241747434934755342864657661166779739666882029
12073791438537195882498081268678383745597317461360853795
34524221586593201928090878297308431392844403281231558611
03697680135730421616874760967587134831202547858932076716
91324484262361314125087802080002616831510273418279777047
84635868170164365024153691398281264810213092761244896359
92870511496497541990934222156683257208082133318611681155
36158365469840467089756029009505376164758477284218896796
46244945160765353408198901385442487984959953319101723355
55660213945039973628075013783761530712776192684903435262
52000158885351473316117021039681759215109077880193931781
14194545257223865541461062892187960223838971476088506276
86296714667469756291123408243920816015378088989396451826
32436716167621791689097799119037540312746222899880051954
44414282012187361745992642956581746628302955570299024324
15318161721046583203678690611726015878352075151628422554
02651704833042261439742869330616908979684825901254583271
68226458066526769958652682272807075781391858178889652208
16434834482599326604336766017699961283186078838615027946
59551311565520360939881806121385586003014356945272242063
44631797460594682573103790084024432438465657245014402821
88525247093519062092902313649327349756551395872055965422
87497740114133469627154228458623773875382304838656889764
61927383814900140767310446640259899490222221765904339901
88601856652648506179970235619389701786004081188972991831
10211712298459016419210688843871218556461249607987229085
19296819372388642614839657382291123125024186649353143970
13742853192664987533721894069428143411852015801412334482
80150513996942901534830776445690990731524332782882698646
02789864321139083506217095002597389863554277196742822248
75758676575234422020757363056949882508796892816275384886
33969099598262809561214509948717012445164612603790293091
20889086942028510640182154399457156805941872748998094254
74217358240106367740459574178516082923013535808184009699
63725242305608559037006242712434169090041536901059339838
35777939410970027753472000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
```

```
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000
```

In [10]:

```
factorial(1000)
```

Out[10]:

```
4023872600770937735437024339230039857193748642107146325
4379991042993851239862902059204420848696940480047998861
0197196058631666872994808558901323829669944590997424504
0870737599188236277271887325197795059509952761208749754
6249704360141827809464649629105639388743788648733711918
1045825783647849977012476632889835955735432513185323958
4630755574091142624174743493475534286465766116677973966
6882029120737914385371958824980812686783837455973174613
6085379534524221586593201928090878297308431392844403281
2315586110369768013573042161687476096758713483120254785
8932076716913244842623613141250878020800026168315102734
1827977704784635868170164365024153691398281264810213092
7612448963599287051149649754199093422215668325720808213
3318611681155361583654698404670897560290095053761647584
7728421889679646244945160765353408198901385442487984959
9533191017233555566021394503997362807501378376153071277
6192684903435262520001588853514733161170210396817592151
0907788019393178114194545257223865541461062892187960223
8389714760885062768629671466746975629112340824392081601
5378088989396451826324367161676217916890977991190375403
1274622289988005195444414282012187361745992642956581746
6283029555702990243241531816172104658320367869061172601
5878352075151628422554026517048330422614397428693306169
0897968482590125458327168226458066526769958652682272807
0757813918581788896522081643483448259932660433676601769
9961283186078838615027946595513115655203609398818061213
8558600301435694527224206344631797460594682573103790084
0244324384656572450144028218852524709351906209290231364
9327349756551395872055965422874977401141334696271542284
5862377387538230483865688976461927383814900140767310446
6402598994902222217659043399018860185665264850617997023
5619389701786004081188972991831102117122984590164192106
8884387121855646124960798722908519296819372388642614839
6573822911231250241866493531439701374285319266498753372
1894069428143411852015801412334482801505139969429015348
3077644569099073152433278288269864602789864321139083506
2170950025973898635542771967428222487575867657523442202
0757363056949882508796892816275384886339690995982628095
6121450994871701244516461260379029309120889086942028510
6401821543994571568059418727489980942547421735824010636
7740459574178516082923013535808184009699637252423056085
5903700624271243416909004153690105933983835777939410970
0277534720000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000
```

```
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
```

In [11]:

```
factorial(1001)
```

Out[11]:

```
40278964733717086731724613635692698970509423907492534717
63437103403684509110276496126362526954563742052804685988
07393254690298539867803367460225153499614535588421928591
16083367874245135491592125229928545694627139699585043795
95406450196963727411427873474502813253243738244563002268
71609431497826989489109522725791691167945698509282421538
63296652337667989182369690098207522318827946519406548911
14985865229975733078380579349947062129342914778822214649
14058745808179795130018969175605739824237247684512790169
64801377815866152038491635728554721966033750406791008793
63015808746623675439212889882082619448341783691698056824
89420504038334529389177845089679546075023305854006141256
28863382007994039532925156378839940465290215451930292836
51694523835310307556845785038514881540923235761503115693
25891190105926118761607100286827930472944913272420825078
91215874158985013601703088797545292243488968877588338697
78252159044236824789433138060721440974324186958074125712
92308739802481089407002523955080148184062810447564594783
13983011382137226047414531652164736831393467078385848278
15069152883789413480786896918156577853058969122779932006
39858696294199549107738635599538328374931258525869323348
47733479882767629786882369302337741894230427226780050976
58054356537875303701182612199947525888664510727155837854
95394684524593296728611334955079882857173250037068541860
37251269317081925930941102783717661244469264917453642974
54210862877085881300821687927506971589017371302217514305
50976429258055277255676893874108456870904122902259417224
70713772340612581154995215962976677106307947267928021388
29785237854247603096781382687082397649257687143495546654
38389311198715040908077757086900159389712443987670244241
78790458509301154686150205855009091487790085270161964822
93321924010757475435629899532715089775017710857595216314
27816116191761031257454497039673414248149210836002497114
10756596045857652521255615963497571555263867817213746817
28430664510939844436365607222136681722255857115665581344
67392654185460222589723312097599987253417831473939565071
00634435251809656442778120420006832391305689709091660271
22603068697861072370775724458665729457609777216394083384
30009976028970539150822336553856613962747814621747092348
99691575598346474108200033752694599005936549343992193709
33688967547914167596043248955146603259131578437960399178
19613717350380997781225472000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
```

```
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
```

In [12]:

```
factorial(10000)
```

```
---------------------------------------------------------------
-----------------
RecursionError                            Traceback (most
 recent call last)
<ipython-input-12-58ab6b4e4a58> in <module>
----> 1 factorial(10000)

<ipython-input-5-7a1622528c4a> in factorial(n)
      3         return(1)
      4     else:
----> 5         return(n * factorial(n-1))

... last 1 frames repeated, from the frame below ...

<ipython-input-5-7a1622528c4a> in factorial(n)
      3         return(1)
      4     else:
----> 5         return(n * factorial(n-1))

RecursionError: maximum recursion depth exceeded in compar
ison
```

In [13]:

```
def mylen(l):
    if l == []:
        return(0)
    else:
        return(1+mylen(l[1:]))
```

In [14]:

```
mylen(list(range(1000)))
```

Out[14]:

```
1000
```

In [23]:

```
def mysum(l):
    if l == []:
        return(0)
    else:
        return(l[0] + mysum(l[1:]))
```

In [24]:

```
mysum(list(range(10)))
```

Out[24]:

45

In [20]:

```
mysum(['the','long','road'])
```

Out[20]:

'thelongroad'

Final call '...' + 0

In [27]:

```
def zigzag(l):
    return(updown(l) or downup(l))

def updown(l):
    if len(l) < 2:
        return(True)
    else:
        return(l[0] < l[1] and downup(l[1:]))

def downup(l):
    if len(l) < 2:
        return(True)
    else:
        return(l[0] > l[1] and updown(l[1:]))
```

In [28]:

```
zigzag([0,1,0,1,0])
```

Out[28]:

True

In [30]:

```
zigzag([1,0,1,0,1])
```

Out[30]:

True