lec8

August 12, 2016

```
In [ ]: #Today, we are going to cover recursion.
        #This might be a hard concept to understand
        #in the beginning. But with a little bit of
        #practice,
        #you'll get it.
```

```
In [4]: def mystery(a):
            if a == 0:
                return 0
            else:
                return a + (mystery(a-1))
        mystery(4)
        #we can print a to see how many times
        #mystery is called and what the input(a)
        #is each time it is called.
        #Remember printing something
        #is different from returning something. We are
        #printing all of these values to
        #figure out what the function is doing.
        #This does not change what the
        #function is returning.
```

```
Out[4]: 10
```

```
In [5]: #lets print a to see what is going on in this function
        def mystery(a):
            print a
            if a == 0:
                return 0
            else:

                return a + (mystery(a-1))
        mystery(4)
        #can you guess what this function is doing?
```

```
4
3
2
```

```
1
0
```

```
Out[5]: 10
```

```
In [ ]: #Let us trace the function calls one by one
        def mystery(a):
            if a == 0:
                return 0
            else:
                return a + (mystery(a-1))
        mystery(4)
        # mystery(4) = 4 + mystery(4-1) #10
        # mystery(3) = 3 + mystery(3-1) #6
        # mystery(2) = 2 + mystery(2-1) #3
        # mystery(1) = 1 + mystery(1-1) #1
        # mystery(0) = 0
        #mystery(4) = 0+1+2+3+4
```

```
In [ ]:
```

```
In [9]: #What does this function do?
        #It adds up all the numbers from 0 to a.
        def mystery(a):
            print a
            if a == 0:
                return 0
            else:
                return a + (mystery(a-1))
        mystery(4)
```

```
4
3
2
1
0
```

```
Out[9]: 10
```

```
In [11]: def mystery(a):
             print a
             return a + (mystery(a-1))
         mystery(4)
         #mystery(4)=4+mystery(4-1)
         #mystery(3)=3+mystery(3-1)
         #mystery(2)=2+mystery(2-1)
         #mystery(1)=1+mystery(1-1)
```

```
        #mystery(0)=0+mystery(0-1)
        #mystery(-1)=-1+mystery(-1-1)
        #......
        #what happens now?
```

```
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
-26
-27
-28
-29
-30
-31
-32
-33
-34
-35
-36
-37
-38
```

```
-39
-40
-41
-42
-43
-44
-45
-46
-47
-48
-49
-50
-51
-52
-53
-54
-55
-56
-57
-58
-59
-60
-61
-62
-63
-64
-65
-66
-67
-68
-69
-70
-71
-72
-73
-74
-75
-76
-77
-78
-79
-80
-81
-82
-83
-84
-85
-86
```

```
-87
-88
-89
-90
-91
-92
-93
-94
-95
-96
-97
-98
-99
-100
-101
-102
-103
-104
-105
-106
-107
-108
-109
-110
-111
-112
-113
-114
-115
-116
-117
-118
-119
-120
-121
-122
-123
-124
-125
-126
-127
-128
-129
-130
-131
-132
-133
-134
```

```
-135
-136
-137
-138
-139
-140
-141
-142
-143
-144
-145
-146
-147
-148
-149
-150
-151
-152
-153
-154
-155
-156
-157
-158
-159
-160
-161
-162
-163
-164
-165
-166
-167
-168
-169
-170
-171
-172
-173
-174
-175
-176
-177
-178
-179
-180
-181
-182
```

```
-183
-184
-185
-186
-187
-188
-189
-190
-191
-192
-193
-194
-195
-196
-197
-198
-199
-200
-201
-202
-203
-204
-205
-206
-207
-208
-209
-210
-211
-212
-213
-214
-215
-216
-217
-218
-219
-220
-221
-222
-223
-224
-225
-226
-227
-228
-229
-230
```

```
-231
-232
-233
-234
-235
-236
-237
-238
-239
-240
-241
-242
-243
-244
-245
-246
-247
-248
-249
-250
-251
-252
-253
-254
-255
-256
-257
-258
-259
-260
-261
-262
-263
-264
-265
-266
-267
-268
-269
-270
-271
-272
-273
-274
-275
-276
-277
-278
```

```
-279
-280
-281
-282
-283
-284
-285
-286
-287
-288
-289
-290
-291
-292
-293
-294
-295
-296
-297
-298
-299
-300
-301
-302
-303
-304
-305
-306
-307
-308
-309
-310
-311
-312
-313
-314
-315
-316
-317
-318
-319
-320
-321
-322
-323
-324
-325
-326
```

```
-375
-376
-377
-378
-379
-380
-381
-382
-383
-384
-385
-386
-387
-388
-389
-390
-391
-392
-393
-394
-395
-396
-397
-398
-399
-400
-401
-402
-403
-404
-405
-406
-407
-408
-409
-410
-411
-412
-413
-414
-415
-416
-417
-418
-419
-420
-421
-422
```

```
-423
-424
-425
-426
-427
-428
-429
-430
-431
-432
-433
-434
-435
-436
-437
-438
-439
-440
-441
-442
-443
-444
-445
-446
-447
-448
-449
-450
-451
-452
-453
-454
-455
-456
-457
-458
-459
-460
-461
-462
-463
-464
-465
-466
-467
-468
-469
-470
```

```
-471
-472
-473
-474
-475
-476
-477
-478
-479
-480
-481
-482
-483
-484
-485
-486
-487
-488
-489
-490
-491
-492
-493
-494
-495
-496
-497
-498
-499
-500
-501
-502
-503
-504
-505
-506
-507
-508
-509
-510
-511
-512
-513
-514
-515
-516
-517
-518
```

```
-567
-568
-569
-570
-571
-572
-573
-574
-575
-576
-577
-578
-579
-580
-581
-582
-583
-584
-585
-586
-587
-588
-589
-590
-591
-592
-593
-594
-595
-596
-597
-598
-599
-600
-601
-602
-603
-604
-605
-606
-607
-608
-609
-610
-611
-612
-613
-614
```

```
-615
-616
-617
-618
-619
-620
-621
-622
-623
-624
-625
-626
-627
-628
-629
-630
-631
-632
-633
-634
-635
-636
-637
-638
-639
-640
-641
-642
-643
-644
-645
-646
-647
-648
-649
-650
-651
-652
-653
-654
-655
-656
-657
-658
-659
-660
-661
-662
```

```
-663
-664
-665
-666
-667
-668
-669
-670
-671
-672
-673
-674
-675
-676
-677
-678
-679
-680
-681
-682
-683
-684
-685
-686
-687
-688
-689
-690
-691
-692
-693
-694
-695
-696
-697
-698
-699
-700
-701
-702
-703
-704
-705
-706
-707
-708
-709
-710
```

```
-759
-760
-761
-762
-763
-764
-765
-766
-767
-768
-769
-770
-771
-772
-773
-774
-775
-776
-777
-778
-779
-780
-781
-782
-783
-784
-785
-786
-787
-788
-789
-790
-791
-792
-793
-794
-795
-796
-797
-798
-799
-800
-801
-802
-803
-804
-805
-806
```

```
-807
-808
-809
-810
-811
-812
-813
-814
-815
-816
-817
-818
-819
-820
-821
-822
-823
-824
-825
-826
-827
-828
-829
-830
-831
-832
-833
-834
-835
-836
-837
-838
-839
-840
-841
-842
-843
-844
-845
-846
-847
-848
-849
-850
-851
-852
-853
-854
```

```
-855
-856
-857
-858
-859
-860
-861
-862
-863
-864
-865
-866
-867
-868
-869
-870
-871
-872
-873
-874
-875
-876
-877
-878
-879
-880
-881
-882
-883
-884
-885
-886
-887
-888
-889
-890
-891
-892
-893
-894
-895
-896
-897
-898
-899
-900
-901
-902
```

```
-903
-904
-905
-906
-907
-908
-909
-910
-911
-912
-913
-914
-915
-916
-917
-918
-919
-920
-921
-922
-923
-924
-925
-926
-927
-928
-929
-930
-931
-932
-933
-934
-935
-936
-937
-938
-939
-940
-941
-942
-943
-944
-945
-946
-947
-948
-949
-950
```

```
-951
-952
-953
-954
-955
-956
-957
-958
-959
-960
-961
-962
-963
-964
-965
-966


        ---------------------------------------------------------------------------

        RuntimeError                              Traceback (most recent call last)

        <ipython-input-11-4368f9e77591> in <module>()
          2     print a
          3     return a + (mystery(a-1))
    ----> 4 mystery(4)
          5 #mystery(4)=4+mystery(4-1)
          6 #mystery(3)=3+mystery(3-1)


        <ipython-input-11-4368f9e77591> in mystery(a)
          1 def mystery(a):
          2     print a
    ----> 3     return a + (mystery(a-1))
          4 mystery(4)
          5 #mystery(4)=4+mystery(4-1)


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


        <ipython-input-11-4368f9e77591> in mystery(a)
          1 def mystery(a):
          2     print a
    ----> 3     return a + (mystery(a-1))
          4 mystery(4)
          5 #mystery(4)=4+mystery(4-1)
        RuntimeError: maximum recursion depth exceeded while calling a Python objec
```

```
In [ ]: #mystery keeps on calling itself forever.
        # mystery(4) =
        # mystery(3) =
        # mystery(2) =
        # mystery(1) =
        # mystery(0) =
        # mystery(-1) =
        # mystery(-2) =
        # ....
```

```
In [ ]: #what does this function do?
        #uncomment the print statemnet to see what
        #a and b are everytime the function mystery2
        #calles itself.
        def mystery2(a,b):
            #print a,b
            if b == 0:
                return 0
            else:
                return a + (mystery2(a,(b - 1)))
        mystery2(5,3)
        #mystery2(5,3) = 5+(mystery2(5,(3-1))) #5+5+5=15
        #mystery2(5,2) = 5+(mystery2(5,(2-1))) #5+5=10
        #mystery2(5,1) = 5+(mystery2(5,(1-1))) #5+0=5
        #mystery2(5,0) = 0
```

```
In [ ]: #We can implement the multiplication of 2 numbers
        #a and b in as recursive function only using
        #addition.
        #when you multiply 5*3, what do you do?
        #5*3 is 5+5+5. The function below adds 5 to itself
        #3 times. In genera, the function below takes
        #inputs a and b and adds a to itself b times.
        #This is the same as performing a*b.
        def mystery2(a,b):
            print a,b
            if b == 0:
                return 0
            else:
                return a + (mystery2(a,(b - 1)))
        mystery2(5,3)
```

```
In [ ]: #Lets see another an example.
        #consider the Fibonacci sequence
```

```
         #1, 1, 2, 3, 5, 8, 13, . . ..
         #This sequence is defined by the 0th and 1st
         #Fibonacci numbers both being 1,
         #and subsequent Fibonacci numbers being
         #the sum of the previous two.

         #The Fibonacci sequence is very well known in
         #Math and
         #appears frequently in nature.
         #It also has applications
         #in various computer science algorithms.

         #We will not discuss the Fibonacci sequence
         #in detail
         #in this class but if you are interested in
         #learning
         #more you can read the Wikipedia page here:
         #https://en.wikipedia.org/wiki/Fibonacci_number
```

```
In [ ]: #The Fibonacci sequence is defined by the 0th
         #and 1st
         #Fibonacci numbers both being 1,
         #and subsequent Fibonacci numbers being
         #the sum of the previous two
         #F(i) is 1 if i=0 or i=1
         #
         #F(i) is F(i—1) + F(i—2) otherwise

         #So what is F(0)?
         #    1 because i is 1
         #What is F(1)?
         #   1 because i is 1
         #What is F(2)
         #   It is F(i-1) + F(i-2) #F(1)+F(0)
         #   which is 1+1=2
         #F(3) is F(i-1)+F(i-2)=F(3-1)+F(3-2)
         #   which is F(2)+F(1)=2+1=3
         #F(4) is F(4-1)+F(4-2)=F(3)+F(2)=3+2=5
         # ....And it continues this way
```

```
In [ ]: ##Now let us make a function that takes i and
         #returns the ith Fibonacci number.
         #We can do it the way we have learned how to.
         #First we write in English how we want to create
         #the function.
```

```
In [12]: #Now let us make a function that takes i and
          #returns the ith Fibonacci number.
          #We can do it the way we have learned how to.
          #lets call the function fibonacci
          def fibonacci(i):
              fib_i=1 #it is called initialization.
              fib_i_prev=1
              fib_i_prev_prev=1

              if i==0 or i==1:
                  return fib_i
              for x in range(2,i+1): #[2,3,4....i]
                  fib_i = fib_i_prev + fib_i_prev_prev
                  fib_i_prev_prev=fib_i_prev
                  fib_i_prev=fib_i
              return fib_i
          fibonacci(5)
```

```
Out[12]: 8
```

```
In [13]: #give me a function fibonacci. It takes an input i,
         #and returns the ith fibonacci number.
         #Remember the formula.
         #if i=0, or i=1, what is the fibonacci number? 1.
         #ohterwise what is it fibonacci(i)=fibonacci(i-1)+
         #fibonacci(i-2)
         #fibonacci(0) #1
         #fibonacci(1) #1
         #fibonacci(2) #2 fibonacci(1)+fibonacci(0)
         def fibonacci(i):
             if i==0 or i==1:
                 return 1
             else:
                 return fibonacci(i-1)+fibonacci(i-2)
         fibonacci(5)
         #fibonacci(5)=fibonacci(4)+fibonacci(3) #8
         #fibonacci(4)=fibonacci(3)+fibonacci(2) #5
         #fibonacci(3)=fibonacci(2)+fibonacci(1) #3
         #fibonacci(2)=fibonacci(1)+fibonacci(0) #2
         #fibonacci(1)=1
```

```
Out[13]: 8
```

```
In [ ]:
```

```
In [ ]: #Although we could implement the Fibonacci sequence
        #in a very complicated way without using recursion,
        #the following function using recursion
        #is very simple and easier to understand
        def fibonacci(i):
            if i<2:
```

```
        return 1
    return fibonacci(i-1) + fibonacci(i-2)
fibonacci(5)

#compare this function to the function we
#wrote above
#which is more complicated and hard to understand
```