Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Sandeep Sadanandan

Sommer Semester 2009
Solution Sheet 1
April 22, 2009

Python For Fine Programmers

Problem 1 (8 Points)

Write three different python functions, each of which gives the fibonacci number corresponding to the input number.
Bonus: Write a 4^{th} and better function.

Solution

The solutions for the first part is given in the code given below. The functions are small enough that they need no further explanation.

```
1  def fibrec(n):
2      if n < 3:
3          return (n-1)
4      return fibrec(n-1) + fibrec(n-2)
5
6  def fibiter(n):
7      a, b = 0, 1
8      while n > 2:
9          a += b
10         b += a
11         n -= 2
12     if n == 1:
13         return a
14     return b
15
16 def fibrectab(ft, n):
17     if len(ft) != n+1:
18         for i in range(1, n+2):
19             ft.append(-1)
20         ft[1] = 0
21         ft[2] = 1
22
23     if ft[n] == -1:
24         ft[n] = fibrectab(ft, n-1) + fibrectab(ft, n-2)
25
26     return ft[n]
27
28
29 fibtable = []
30 for i in range(1, 4):
31     x = int(raw_input("Give me num: "))
32     print "Rec: ", fibrec(x),
```

```
33      print "Iter: ", fibiter(x),
34      print "Table: ", fibrectab(fibtable , x)
```

The bonus question is answered below. The algorithm is exactly the same which was explained in the lecture.

```
1
2  def dotprod(row, col):
3    ret = 0
4
5    for i in range(len(row)):
6      ret += row[i] * col[i]
7
8    return ret
9
10 def rearrange(B):
11   R = []
12   for i in range(len(B[0])):
13     col = [ x[i] for x in B]
14     R.append(col)
15   return R
16
17
18 def matmul(A, B):
19   C = []
20   R = rearrange(B)
21   for row in A:
22     Crow = []
23     for col in R:
24       k = dotprod(row, col)
25       Crow.append(k)
26     C.append(Crow)
27   return C
28
29 def matpow(A, n):
30   if n < 1:
31     print "We'll think about it, Aha!"
32     return [[],[]]
33   if n == 1:
34     return A
35   if n == 2:
36     return matmul(A, A)
37   if n % 2 == 0:
38     halfpow = matpow(A, n/2)
39     return matmul(halfpow, halfpow)
40   else:
41     return matmul(matpow(A, n-1), A)
42
43
44 def printmat(mat):
45   print '-------'
```

```
46    for row in mat:
47      print row
48    print '-------'
49
50
51  def specialfib(n):
52    specmat = [[0,1],[1,1]]
53    powered = matpow(specmat, n)
54    fibomat = matmul(powered, [[0],[1]])
55    print fibomat[0][0]
56    printmat(fibomat)
57
58
59
60  specialfib(18);
```

Problem 2 (8 Points)

Write a program to find out the square root of a given number. (Without the help of python math library)
Bonus: Extend this to n^{th} root.

Solution

```
1  def nth_root(p, n, err):
2    low = 0.0
3    high = p
4    mid = (low + high)/2.0
5
6    while abs(mid**n - p) > err:
7        if mid**n - p > 0:
8            high = mid
9        else:
10           low = mid
11       mid = (high + low) / 2.0
12
13   return mid
14
15 print nth_root(10, 2, 0.1)
16 print nth_root(100, 2, 0.1)
17 print nth_root(100, 3, 0.1)
18 print nth_root(10000, 4, 0.1)
```

Problem 3 (8 Points)

Write a program, without using the `int` functionality of python, to convert a string (representing an integer) to the integer.
Also, do the reverse: Integer to String
Bonus: Extend this to floating points

Solution

```
def atoi(str):
    ret = 0;
    for c in str:
        ret *= 10
        ret += ord(c) - ord('0')
    return ret

def itoa(inte):
    ret = ''
    def tonum(c):
        return chr(c + ord('0'))

    while inte > 10:
        char = tonum(inte%10)
        ret = char + ret
        inte /= 10

    ret = tonum(inte) + ret
    return ret

print atoi("123" + "9889")
print itoa(1232 * 944)
```

Problem 4 (8 Points)

Write a program to generate all the combinations of all the characters in a given string, or a list of characters.
Bonus: Beauty of the program.

Solution

```
def combinations(set, length):
  if length <  1:
    return [];

  if length == 1:
    return set;

  combis = [];

  for x in set:
    tmpset = set[0:];
    tmpset.remove(x);
    smallcombis = combinations(tmpset, length - 1);
    for y in smallcombis:
      combis.append(x+y);

  combis.sort();
  return combis;

combination = combinations(['a', 's', 'd', 'f'], 3);
```

```
21
22 print "The combinations are: ",  combination;
23 print "The number of combinations are: " + combination.__len__();
```