PYTHON+ASSIGNMENT+2

October 5, 2016

AFRICAN INSTITUTE OF MATHEMATICAL SCIENCES JULIANA TULA TALAI PYTHON ASSIGNMENT 2.

a function digits(n, b) that returns the list of digits of n in base b

```
In [48]: def digits(n,b):
             l=[]
             while n>0:
                 l.insert(0,n%b)
                 n=n//b
             return(l)
```

```
In [49]: digits(50,5)

Out[49]: [2, 0, 0]
```

This is okay, but you were also asked to test your function on different values, and different bases. Just one example is not enough.

a function is_prime(n) that returns True if n is prime and False otherwise

```
In [1]: def is_prime(n):
            i = 2                #returns true if n is prime and false otherwise
            while i*i <= n:
                if n%i == 0:
                    return False
                i = i + 1
            return True
```

This function will return True for 0 and 1, but these are not prime numbers.

```
In [2]: is_prime(37)

Out[2]: True
```

This is okay, but you were also asked to test your function on different values, and different bases. Just one example is not enough.

a function prime_range(n) that returns the list of the prime numbers smaller than n

```
In [5]: def prime_range(k):
            l=[]
            for i in range(2,k):     #Returns the list of prime numbers less than k
                if is_prime(i)==True:
                    l.append(i)
            return l
```

```
In [24]: k=prime_range(200)
         print(k)
```

This is okay, but you were also asked to test your function on different values, and different bases. Just one example is not enough.

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79
```

a function gcd(x, y) that computes the greatest common divisor of x and y

```
In [25]: def gcd(x,y):
             while y!=0:
                 r=(x%y)
                 (x,y)=(y,r)
             return(x)
```

These parenthesis do not play any important role here.

```
In [28]: gcd(500,300)

Out[28]: 100
```

This is okay, but you were also asked to test your function on different values, and different bases. Just one example is not enough.

(1) If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.

Find the sum of all the multiples of 3 or 5 below 1000.

```
In [1]: s=0
        for n in range (1,10):
            if n%3 == 0 or n%5 == 0:
                s=n+s                                   #This code returns the list of natural numb
        print(s)

23
```

No need for this. Just go ahead and answer the question.

```
In [52]: s=0
         for n in range(1,1000):
             if n%3==0 or n%5==0:
                 s=s+n
         print(s)

233168
```

(2) Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be:

1, 2, 3, 5, 8, 13, 21, 34, 55, 89, . . .

By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms.

```
In [53]: L=[1,2]
         for i in range(1,40):
             s=0
             L.append(L[-1]+L[-2])
             for i in L:
                 if i%2==0 and i<4000000:
                     s=s+i
         print(s)
```

This strategy is not the best. A 'while' loop would have been better than a 'for' loop.

```
4613732
```

(3) The prime factors of 13195 are 5, 7, 13 and 29.

What is the largest prime factor of the number 600851475143 ?

```
In [12]: import math
         def max_prime(n):
             m=[]
             for i in range (2,int(math.sqrt(n))):
                 if n%i==0 and is_prime(i)==True:
                     m.append(i)
             return max(m)

In [13]: max_prime(600851475143)

Out[13]: 6857
```

(7) By listing the first six prime numbers: 2, 3, 5, 7, 11, and 13, we can see that the 6th prime is 13.

What is the 10 001st prime number?

```
In [14]: def prime_range(k):
             j=[]
             for i in range (2,k):
                 if is_prime(i):
                     j.append(i)
             return(j)

In [17]: k=prime_range(200000)

In [18]: k[10001]

Out[18]: 104743
```

You have already defined this function. There is no need to redefine it here. Just call it.

k[10002]

(14) The following iterative sequence is defined for the set of positive integers:

$n \to n/2$ (n is even) $n \to 3n + 1$ (n is odd)

Using the rule above and starting with 13, we generate the following sequence: $13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$

It can be seen that this sequence (starting at 13 and finishing at 1) contains 10 terms. Although it has not been proved yet (Collatz Problem), it is thought that all starting numbers finish at 1.

Which starting number, under one million, produces the longest chain?

NOTE: Once the chain starts the terms are allowed to go above one million.

```
In [38]: def collatz(n):
             if n%2==0:
                 n=n//2

             else:
                 n=3*n+1
             return(n)
```

```
In [39]: def longest_collatz(n):
             count=1
             while n>1:
                 count=count+1
                 n=collatz(n)
             return count
```

```
In [40]: z=1
         for i in range(2,100):
             j=longest_collatz(i)
             if j>z:
                 print(i,j)
             z=j
```

```
2 2
3 8
5 6
6 9
7 17
9 20
11 15
14 18
17 13
18 21
22 16
25 24
27 112
31 107
33 27
36 22
39 35
41 110
43 30
47 105
49 25
54 113
57 33
59 33
62 108
65 28
71 103
73 116
76 23
78 36
81 23
82 111
```

```
86 31
89 31
91 93
94 106
97 119
```

```
In [36]: j=[longest_collatz(i) for i in range(1,1000000)]
         max(j)
```

```
Out[36]: 525
```

This is just the length, but not the number with that collatz length.

(15) Starting in the top left corner of a 2×2 grid, and only being able to move to the right and down, there are exactly 6 routes to the bottom right corner.

How many such routes are there through a 20×20 grid?

```
In [30]: import math
         n=2
         R=math.factorial(2*n)//math.factorial (n)**2
         print(R)
```

No need to import math again. It is enough to import it once.

```
6
```

```
In [33]: import math
         n=20
         R=math.factorial(2*n)//math.factorial (n)**2
         print(R)
```

No need to import math again.

```
137846528820
```

(16) 2^15 = 32768 and the sum of its digits is 3 + 2 + 7 + 6 + 8 = 26.

What is the sum of the digits of the number 2^1000?

```
In [33]: R=str(2**15)
         r=0
         for i in R:
             r+=int(i)
         print(r)
```

No need for this.

```
26
```

```
In [34]: R=str(2**1000)
         r=0
         for i in R:
             r+=int(i)
         print(r)
```

```
1366
```

(18) By starting at the top of the triangle below and moving to adjacent numbers on the row below, the maximum total from top to bottom is 23.

3 7 4 2 4 6 8 5 9 3

That is, 3 + 7 + 4 + 9 = 23.

Find the maximum total from top to bottom of the triangle below:

75 95 64 17 47 82 18 35 87 10 20 04 82 47 65 19 01 23 75 03 34 88 02 77 73 07 63 67 99 65 04 28 06 16 70 92 41 41 26 56 83 40 80 70 33 41 48 72 33 47 32 37 16 94 29 53 71 44 65 25 43 91 52 97 51 14 70 11 33 28 77 73 17 78 39 68 17 57 91 71 52 38 17 14 91 43 58 50 27 29 48 63 66 04 68 89 53 67 30 73 16 69 87 40 31 04 62 98 27 23 09 70 98 73 93 38 53 60 04 23

NOTE: As there are only 16384 routes, it is possible to solve this problem by trying every route. However, Problem 67, is the same challenge with a triangle containing one-hundred rows; it cannot be solved by brute force, and requires a clever method! ;o)

```
In [13]: l=[[75],
          [95, 64],
          [17, 47, 82],
          [18, 35, 87, 10],
          [20, 4, 82, 47, 65],
          [19, 1, 23, 75, 3, 34],
          [88, 2, 77, 73, 7, 63, 67],
          [99, 65, 4, 28, 6, 16, 70, 92],
          [41, 41, 26, 56, 83, 40, 80, 70, 33],
          [41, 48, 72, 33, 47, 32, 37, 16, 94, 29],
          [53, 71, 44, 65, 25, 43, 91, 52, 97, 51, 14],
          [70, 11, 33, 28, 77, 73, 17, 78, 39, 68, 17, 57],
          [91, 71, 52, 38, 17, 14, 91, 43, 58, 50, 27, 29, 48],
          [63, 66, 4, 68, 89, 53, 67, 30, 73, 16, 69, 87, 40, 31],
          [4, 62, 98, 27, 23, 9, 70, 98, 73, 93, 38, 53, 60, 4, 23]]

In [32]: w=[]
          for i in l:
              for j in list(i):
                  w.append(j)
          h=0
          for j in range(len(w)):
              p=1
              s=w[j:j+13]
              for d in s:
                  p*=int(d)
              if p>h:
                  h=p
          print(h)

1957557631979696947200
```

In [ ]:

In [ ]: