Python_Warm_Up

March 27, 2018

1 Multiplication table

```
In [12]: def table():
             num = input('Input\t:')
             for i in range(1,11,1):
                 print('{} times {} equals {}'.format(i, num, i*float(num)))
             return
```

```
In [13]: table()
```

```
Input        :8
1 times 8 equals 8.0
2 times 8 equals 16.0
3 times 8 equals 24.0
4 times 8 equals 32.0
5 times 8 equals 40.0
6 times 8 equals 48.0
7 times 8 equals 56.0
8 times 8 equals 64.0
9 times 8 equals 72.0
10 times 8 equals 80.0
```

2 Twin primes and prime factors

```
In [14]: def sieve(max_num):
             sv = list(range(2, max_num+1, 1))
             for i, num in enumerate(sv):
                 if num is not -1:
                     j=i+num
                     while(j<len(sv)):
                         sv[j]=-1
                         j+=num

             return sv
```

```
In [39]: test=sieve(15)
```

```
In [40]: test
```

```
Out[40]: [2, 3, -1, 5, -1, 7, -1, -1, -1, 11, -1, 13, -1, -1]
```

```
In [49]: def twinPrimes(max_num):
             s=sieve(max_num)
             s=s[1:] # remove 2 - only even prime
             s=list(filter(lambda x : x>0, s))
             for i, num in enumerate(s):
                 if i < len(s)-1:
                     print(str(num) + ' ' + str(s[i+1]))
```

```
In [51]: twinPrimes(1000)
```

```
3 5
5 7
7 11
11 13
13 17
17 19
19 23
23 29
29 31
31 37
37 41
41 43
43 47
47 53
53 59
59 61
61 67
67 71
71 73
73 79
79 83
83 89
89 97
97 101
101 103
103 107
107 109
109 113
113 127
127 131
131 137
137 139
139 149
149 151
151 157
```

157 163
163 167
167 173
173 179
179 181
181 191
191 193
193 197
197 199
199 211
211 223
223 227
227 229
229 233
233 239
239 241
241 251
251 257
257 263
263 269
269 271
271 277
277 281
281 283
283 293
293 307
307 311
311 313
313 317
317 331
331 337
337 347
347 349
349 353
353 359
359 367
367 373
373 379
379 383
383 389
389 397
397 401
401 409
409 419
419 421
421 431
431 433
433 439

439 443
443 449
449 457
457 461
461 463
463 467
467 479
479 487
487 491
491 499
499 503
503 509
509 521
521 523
523 541
541 547
547 557
557 563
563 569
569 571
571 577
577 587
587 593
593 599
599 601
601 607
607 613
613 617
617 619
619 631
631 641
641 643
643 647
647 653
653 659
659 661
661 673
673 677
677 683
683 691
691 701
701 709
709 719
719 727
727 733
733 739
739 743
743 751

```
751 757
757 761
761 769
769 773
773 787
787 797
797 809
809 811
811 821
821 823
823 827
827 829
829 839
839 853
853 857
857 859
859 863
863 877
877 881
881 883
883 887
887 907
907 911
911 919
919 929
929 937
937 941
941 947
947 953
953 967
967 971
971 977
977 983
983 991
991 997
```

```
In [41]: def primeFactors(num):
             s=sieve(num)
             s=list(filter(lambda x : x>0, s))
             primes=[]
             for n in s:
                 while num % n is 0:
                     primes.append(n)
                     num=int(num/n)

             print(primes)
```

```
In [48]: primeFactors(56)
```

```
[2, 2, 2, 7]
```

3 Combinatorics

```
In [54]: from functools import reduce

In [62]: def nPr(n, r):
             return reduce(lambda x, y: x*y ,list(range(n, n-r, -1)))

In [63]: nPr(10,2)

Out[63]: 12

In [64]: def nCr(n, r):
             return int(nPr(n,r)/fact(r))

In [66]: def fact(r):
             if r==1: return 1
             return r*fact(r-1)

In [67]: nCr(10,2)

Out[67]: 45
```

4 Decimal to binary

```
In [73]: hex2Bin={
             '0':'0000',
             '1':'0001',
             '2':'0010',
             '3':'0011',
             '4':'0100',
             '5':'0101',
             '6':'0110',
             '7':'0111',
             '8':'1000',
             '9':'1001',
             'a':'1010',
             'b':'1011',
             'c':'1100',
             'd':'1101',
             'e':'1110',
             'f':'1111',
         }

In [88]: def dec2Bin(dec):
             h=hex(dec)
```

```
            s=str(h)[2:]
            bi=''
            for c in s:
                bi+=hex2Bin[c]
            print(bi)
```

```
In [89]: s='76'
```

```
In [90]: hex2Bin['7']
```

```
Out[90]: '0111'
```

```
In [91]: dec2Bin(76)
```

```
01001100
```

5 Armstrong number

```
In [1]: 3**3+7**3+1**3
```

```
Out[1]: 371
```

```
In [29]: def isArmstrong(num):
            if num == sumOfCubes(num):
                return True
            return False
```

```
In [30]: def sumOfCubes(num):
            a=str(num)
            sum=0
            for c in a:
                sum+=int(c)**3
            return sum
```

```
In [31]: isArmstrong(371)
```

```
Out[31]: True
```

6 Product of digits

```
In [32]: def prodDigit(num):
            a=str(num)
            prod=1
            for c in a:
                prod*=int(c)
            return prod
```

```
In [33]: prodDigit(43)
```

```
Out[33]: 12
```

7 Multiplicative persistence and Multiplicative digital root

```
In [37]: def MDR_MPersistence(num):
             s=str(num)
             persistence=0
             while len(s) > 1:
                 s=str(prodDigit(int(s)))
                 persistence+=1
             return int(s), persistence
```

```
In [38]: MDR_MPersistence(341)
```

```
Out[38]: (2, 2)
```

8 Sum of proper divisors

```
In [50]: def sumPdivisors(num):
             s=0
             d=1
             while d <= num/2:
                 if int(num % d) == 0:
                     s+=d
                 d+=1
             return s
```

```
In [49]: sumPdivisors(36)
```

```
Out[49]: 55
```

```
In [51]: 1+2+3+4+6+9+18+12
```

```
Out[51]: 55
```

9 Perfect numbers in given range

```
In [52]: def perfectSet(r):
             start=r[0]
             end=r[1]

             for n in range(start, end+1, 1):
                 if isPerfect(n):
                     print(n)
             return
```

```
In [53]: def isPerfect(n):
             return n==sumPdivisors(n)
```

```
In [54]: perfectSet((27, 30))
```

```
28
```

10 Amicable numbers in a range

```
In [55]: def amicableSet(r):
             start=r[0]
             end=r[1]

             for n in range(start, end+1, 1):
                 for n_ in range(n, end+1, 1):
                     if isAmicablePair(n, n_):
                         print(str(n)+' '+str(n_))

             return

In [56]: def isAmicablePair(n, n_):
             return (sumPdivisors(n)==n_) and (sumPdivisors(n_)==n)

In [57]: amicableSet((219,285))

220 284
```

11 Filter odd numbers

```
In [59]: def filterOdd(l):
             return list(filter(lambda x: x%2 is not 0,l))

In [60]: filterOdd([0,2,5,8,19,20,34,95])

Out[60]: [5, 19, 95]
```

12 Cubic map

```
In [62]: def cubeMap(l):
             return list(map(lambda x:x**3, l))

In [63]: cubeMap([1,2,3,4])

Out[63]: [1, 8, 27, 64]
```

13 Even cubes

```
In [64]: def evenCubes(l):
             return cubeMap(list(filter(lambda x:x%2==0, l)))

In [65]: evenCubes([0,2,5,8,19,20,34,95])

Out[65]: [0, 8, 512, 8000, 39304]
```