parth.pandey13103447@gmail.com1

September 3, 2019

1 Assignment 1 Notebook

1.1 Question 1

```
[1]: def multiplication_table():
         number =int(input('Enter a number for which the table needs to be printed =␣
      ↪'))
         for i in range(1,11):
             print('{} * {} = {}'.format(number, i, number * i))

     multiplication_table()
```

```
Enter a number for which the table needs to be printed= 1000
1000 * 1 = 1000
1000 * 2 = 2000
1000 * 3 = 3000
1000 * 4 = 4000
1000 * 5 = 5000
1000 * 6 = 6000
1000 * 7 = 7000
1000 * 8 = 8000
1000 * 9 = 9000
1000 * 10 = 10000
```

1.2 Question 2

```
[13]: temp = []
      # Calculates and appends all the odd prime numbers
      for i in range(2,1001):
          count = 0
          if i % 2 == 0:
              continue
          for j in range(1,i+1):
              if i % j == 0:
                  count += 1
          if count == 2:
```

```
        temp.append(i)
# Finds the consecutive pairs and prints them
for ind in range(len(temp)-1):
    if temp[ind] + 2 == temp[ind+1] :
        print(temp[ind],temp[ind+1])
```

```
3 5
5 7
11 13
17 19
29 31
41 43
59 61
71 73
101 103
107 109
137 139
149 151
179 181
191 193
197 199
227 229
239 241
269 271
281 283
311 313
347 349
419 421
431 433
461 463
521 523
569 571
599 601
617 619
641 643
659 661
809 811
821 823
827 829
857 859
881 883
```

1.3 Question 3

[3]:

```python
x = 56
factors = []
# Finds the prime factors and appends it factors list
for i in range(2, x):
    if x % i == 0 and sum([1 for j in range(1,i+1) if i % j == 0 ]) == 2:
        factors.append(i)
residue = x
i = 0
print('Prime Factors of {}'.format(x))
# Find
while(i < len(factors)):
    if residue % factors[i] == 0:
        residue = residue / factors[i]
        print(factors[i])
        i -= 1
    i += 1
```

```
Prime Factors of 56
2
2
2
7
```

1.4 Question 4

[3]:

```python
# calculates factorial recursion
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n-1)
print(factorial(10))
# calculates permutation by using the factorial function
def permutation(n,r):
    return factorial(n)/factorial(n-r)
print(permutation(10,2))
# Calculates the combinations by using the permiutation function
def combinations(n,r):
    return permutation(n,r) / factorial(r)
print(combinations(4,2))
```

```
3628800
90.0
6.0
```

1.5 Question 5

[2]:

```
x = 10
# calculates the binary of representation of x and prints in actual order
binary = []
remainder = 0
while(x > 0):
    remainder = x % 2
    x = x // 2
    binary.append(remainder)
print(binary[::-1])
```

```
[1, 0, 1, 0]
```

1.6 Question 6

[117]:

```
# finds the cubesum of the given number
def cubesum(number):
    digit = []
    while(number>0):
        temp = number % 10
        number = (number - temp) // 10
        digit.append(temp)
    return sum([i ** 3 for i in digit])

# finds if a 3 digit number is an armstrong number
def isArmstrong(number):
    return number == cubesum(number)
# Checks if a 3 digit number is armstrong then prints it
def printArmstrong(number):
    if isArmstrong(number):
        print(number)

printArmstrong(153)
```

```
153
```

1.7 Question 7

[121]:

```
# function to find the product of the digits of given number
def prodDigits(number):
    digit = 1
    while(number > 0):
        temp = number % 10
        number = number // 10
        digit = digit * temp
    return digit
prodDigits(33)
```

[121]: 9

1.8 Question 8

[138]:

```
# finds the MDR of a number via recursion
def MDR(number):
    number = prodDigits(number)
    if number >= 0 and number < 10:
        print(number)
    else:
        return MDR(number)
MDR(341)
# Finds the persistence of a number
def MPersistence(number):
    count = 0
    while(number > 10):
        number = prodDigits(number)
        count += 1
    return count
MPersistence(341)
```

```
2
```

[138]: 2

1.9 Question 9

[142]:

```
# finds the sum of the proper divisor of a number
def sumPDivisors(number):
    temp = 0
    for i in range(1,number):
        if number % i == 0:
            temp += i
    return temp
sumPDivisors(36)
```

[142]: 55

1.10 Question 10

[144]:
```
# finds the sum of proper divisors and then checks if the number is equal to the sum
def perfectNumbers(number):
    temp = 0
    for i in range(1,number):
        if number % i == 0:
            temp += i
    return temp == number
perfectNumbers(28)
```

[144]:
```
True
```

1.11 Question 11

[155]:
```
# finds the two numbers which amicable to each other
def amicableNumbers(n):
    for i in range(1, n+1):
        for j in  range(1,i):
            if sumPDivisors(i) == j and sumPDivisors(j) == i:
                print(i,j)
amicableNumbers(284)
```

```
284 220
```

1.12 Question 12

[160]:
```
# Creates a list of 1000 numbers
x = [i for i in range(1000)]
# filters out odd numbers
x = list(filter(lambda z : z % 2 == 0 , x))
print(x)
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40,
42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80,
82, 84, 86, 88, 90, 92, 94, 96, 98, 100, 102, 104, 106, 108, 110, 112, 114, 116,
118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146, 148,
150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180,
182, 184, 186, 188, 190, 192, 194, 196, 198, 200, 202, 204, 206, 208, 210, 212,
214, 216, 218, 220, 222, 224, 226, 228, 230, 232, 234, 236, 238, 240, 242, 244,
246, 248, 250, 252, 254, 256, 258, 260, 262, 264, 266, 268, 270, 272, 274, 276,
278, 280, 282, 284, 286, 288, 290, 292, 294, 296, 298, 300, 302, 304, 306, 308,
310, 312, 314, 316, 318, 320, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340,
342, 344, 346, 348, 350, 352, 354, 356, 358, 360, 362, 364, 366, 368, 370, 372,
374, 376, 378, 380, 382, 384, 386, 388, 390, 392, 394, 396, 398, 400, 402, 404,
406, 408, 410, 412, 414, 416, 418, 420, 422, 424, 426, 428, 430, 432, 434, 436,
438, 440, 442, 444, 446, 448, 450, 452, 454, 456, 458, 460, 462, 464, 466, 468,
470, 472, 474, 476, 478, 480, 482, 484, 486, 488, 490, 492, 494, 496, 498, 500,
502, 504, 506, 508, 510, 512, 514, 516, 518, 520, 522, 524, 526, 528, 530, 532,
534, 536, 538, 540, 542, 544, 546, 548, 550, 552, 554, 556, 558, 560, 562, 564,
566, 568, 570, 572, 574, 576, 578, 580, 582, 584, 586, 588, 590, 592, 594, 596,
598, 600, 602, 604, 606, 608, 610, 612, 614, 616, 618, 620, 622, 624, 626, 628,
630, 632, 634, 636, 638, 640, 642, 644, 646, 648, 650, 652, 654, 656, 658, 660,
662, 664, 666, 668, 670, 672, 674, 676, 678, 680, 682, 684, 686, 688, 690, 692,
694, 696, 698, 700, 702, 704, 706, 708, 710, 712, 714, 716, 718, 720, 722, 724,
726, 728, 730, 732, 734, 736, 738, 740, 742, 744, 746, 748, 750, 752, 754, 756,
758, 760, 762, 764, 766, 768, 770, 772, 774, 776, 778, 780, 782, 784, 786, 788,
790, 792, 794, 796, 798, 800, 802, 804, 806, 808, 810, 812, 814, 816, 818, 820,
822, 824, 826, 828, 830, 832, 834, 836, 838, 840, 842, 844, 846, 848, 850, 852,
854, 856, 858, 860, 862, 864, 866, 868, 870, 872, 874, 876, 878, 880, 882, 884,
886, 888, 890, 892, 894, 896, 898, 900, 902, 904, 906, 908, 910, 912, 914, 916,
918, 920, 922, 924, 926, 928, 930, 932, 934, 936, 938, 940, 942, 944, 946, 948,
950, 952, 954, 956, 958, 960, 962, 964, 966, 968, 970, 972, 974, 976, 978, 980,
982, 984, 986, 988, 990, 992, 994, 996, 998]
```

1.13 Question 13

[163]:

```
# Create a list of 100 numbers
x = [i for i in range(100)]
# Find the cube of those numbers
x = list(map(lambda z : z **3,x))
print(x)
```

```
[0, 1, 8, 27, 64, 125, 216, 343, 512, 729, 1000, 1331, 1728, 2197, 2744, 3375,
4096, 4913, 5832, 6859, 8000, 9261, 10648, 12167, 13824, 15625, 17576, 19683,
21952, 24389, 27000, 29791, 32768, 35937, 39304, 42875, 46656, 50653, 54872,
59319, 64000, 68921, 74088, 79507, 85184, 91125, 97336, 103823, 110592, 117649,
125000, 132651, 140608, 148877, 157464, 166375, 175616, 185193, 195112, 205379,
216000, 226981, 238328, 250047, 262144, 274625, 287496, 300763, 314432, 328509,
343000, 357911, 373248, 389017, 405224, 421875, 438976, 456533, 474552, 493039,
512000, 531441, 551368, 571787, 592704, 614125, 636056, 658503, 681472, 704969,
729000, 753571, 778688, 804357, 830584, 857375, 884736, 912673, 941192, 970299]
```

1.14 Question 14

[166]:

```
# Create a list of 100 numbers
x = [i for i in range(100)]
# Filter the odd numbers and find the cube of the remaining numbers
x = list(map(lambda x: x ** 3 , filter(lambda x: x % 2 == 0 , x )))
```

```
print(x)
```

```
[0, 8, 64, 216, 512, 1000, 1728, 2744, 4096, 5832, 8000, 10648, 13824, 17576,
21952, 27000, 32768, 39304, 46656, 54872, 64000, 74088, 85184, 97336, 110592,
125000, 140608, 157464, 175616, 195112, 216000, 238328, 262144, 287496, 314432,
343000, 373248, 405224, 438976, 474552, 512000, 551368, 592704, 636056, 681472,
729000, 778688, 830584, 884736, 941192]
```