Question 1

In [4]:
```
'''
1. (Write a function that inputs a number) and (prints the multiplication tabl
e of that number)
'''
```

Out[4]:
```
'\n1. (Write a function that inputs a number) and (prints the multiplication
table of that number)\n'
```

In [5]:
```
'''
Simple Iteration Nothing Fancy
'''
```

Out[5]:
```
'\nSimple Iteration Nothing Fancy\n'
```

In [6]:
```
def multiplication_table(num):
    for i in range(1,11):
        print("{} * {} = {}".format(num,i,(num*i)))
```

In [7]:
```
multiplication_table(5)
```

```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

Question 2

In [8]:
```
'''
2. Write a program to print twin primes less than 1000. If two consecutive odd
numbers are
both prime then they are known as twin primes
'''
```

Out[8]:
```
'\n2. Write a program to print twin primes less than 1000. If two consecutive
odd numbers are\nboth prime then they are known as twin primes\n'
```

In [9]:
```
'''
Notes -- From This Programme i Understood How to Select pairs.
'''
```

Out[9]:
```
'\nNotes -- From This Programme i Understood How to Select pairs.\n'
```

In [10]:
```
def prime_number(num):
    '''
    If Number is Prime then return True
    Otherwise False
    '''
    for i in range(2,num):
        flag = True
        if num%i == 0:
            flag = False
            break
    if flag == False:
        return False
    else:
        return True
```

In [11]:
```
list1 = list(range(3,1000))
```

In [12]:
```
# Step -1 Get Odd Number
odd_list = list(filter(lambda x:(x%2!=0),list1))
```

In [13]:
```
print(odd_list)
```

```
[3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41,
43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 8
1, 83, 85, 87, 89, 91, 93, 95, 97, 99, 101, 103, 105, 107, 109, 111, 113, 11
5, 117, 119, 121, 123, 125, 127, 129, 131, 133, 135, 137, 139, 141, 143, 145,
147, 149, 151, 153, 155, 157, 159, 161, 163, 165, 167, 169, 171, 173, 175, 17
7, 179, 181, 183, 185, 187, 189, 191, 193, 195, 197, 199, 201, 203, 205, 207,
209, 211, 213, 215, 217, 219, 221, 223, 225, 227, 229, 231, 233, 235, 237, 23
9, 241, 243, 245, 247, 249, 251, 253, 255, 257, 259, 261, 263, 265, 267, 269,
271, 273, 275, 277, 279, 281, 283, 285, 287, 289, 291, 293, 295, 297, 299, 30
1, 303, 305, 307, 309, 311, 313, 315, 317, 319, 321, 323, 325, 327, 329, 331,
333, 335, 337, 339, 341, 343, 345, 347, 349, 351, 353, 355, 357, 359, 361, 36
3, 365, 367, 369, 371, 373, 375, 377, 379, 381, 383, 385, 387, 389, 391, 393,
395, 397, 399, 401, 403, 405, 407, 409, 411, 413, 415, 417, 419, 421, 423, 42
5, 427, 429, 431, 433, 435, 437, 439, 441, 443, 445, 447, 449, 451, 453, 455,
457, 459, 461, 463, 465, 467, 469, 471, 473, 475, 477, 479, 481, 483, 485, 48
7, 489, 491, 493, 495, 497, 499, 501, 503, 505, 507, 509, 511, 513, 515, 517,
519, 521, 523, 525, 527, 529, 531, 533, 535, 537, 539, 541, 543, 545, 547, 54
9, 551, 553, 555, 557, 559, 561, 563, 565, 567, 569, 571, 573, 575, 577, 579,
581, 583, 585, 587, 589, 591, 593, 595, 597, 599, 601, 603, 605, 607, 609, 61
1, 613, 615, 617, 619, 621, 623, 625, 627, 629, 631, 633, 635, 637, 639, 641,
643, 645, 647, 649, 651, 653, 655, 657, 659, 661, 663, 665, 667, 669, 671, 67
3, 675, 677, 679, 681, 683, 685, 687, 689, 691, 693, 695, 697, 699, 701, 703,
705, 707, 709, 711, 713, 715, 717, 719, 721, 723, 725, 727, 729, 731, 733, 73
5, 737, 739, 741, 743, 745, 747, 749, 751, 753, 755, 757, 759, 761, 763, 765,
767, 769, 771, 773, 775, 777, 779, 781, 783, 785, 787, 789, 791, 793, 795, 79
7, 799, 801, 803, 805, 807, 809, 811, 813, 815, 817, 819, 821, 823, 825, 827,
829, 831, 833, 835, 837, 839, 841, 843, 845, 847, 849, 851, 853, 855, 857, 85
9, 861, 863, 865, 867, 869, 871, 873, 875, 877, 879, 881, 883, 885, 887, 889,
891, 893, 895, 897, 899, 901, 903, 905, 907, 909, 911, 913, 915, 917, 919, 92
1, 923, 925, 927, 929, 931, 933, 935, 937, 939, 941, 943, 945, 947, 949, 951,
953, 955, 957, 959, 961, 963, 965, 967, 969, 971, 973, 975, 977, 979, 981, 98
3, 985, 987, 989, 991, 993, 995, 997, 999]
```

In [14]:
```
# Step -2 I need To make Pairs
list1 = []
count = 0
for i in range(0,len(odd_list)-1):
    flag = prime_number(odd_list[i]) and prime_number(odd_list[i+1])
    if flag == True:
        count+=1
        list_to_store_twin_Prime = list1.append([odd_list[i],odd_list[i+1]])
```

In [15]:
```
# Let's Get Twin of Primes
print(list1)
```

```
[[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]]
```

In [16]:
```
#Number of Twin Primes
print(count)
```

```
35
```

In [ ]:

Question3

In [17]:
```
'''
3. Write a program to find out the (prime factors of a number). Example: prime
factors of 56 -
2, 2, 2, 7
'''
```

Out[17]:
```
'\n3. Write a program to find out the (prime factors of a number). Example: p
rime factors of 56 -\n2, 2, 2, 7\n'
```

In [18]:
```
'''
Notes --- Loop Inside loop
Outer Loop For --- > 2 to 55
Inner Loop For --- > Deviding Same Number
'''
```

Out[18]:
```
'\nNotes --- Loop Inside loop\nOuter Loop For --- > 2 to 55\nInner Loop For -
-- > Deviding Same Number\n'
```

In [19]:
```
def prime_factors(num):
    lists = []
    for i in range(2,num):
        while num%i == 0:
            num = int(num/i)
            lists.append(i)
    print(lists)
```

In [20]:
```
prime_factors(56)
```

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

In [21]:
```
prime_factors(99)
```

```
[3, 3, 11]
```

In [ ]:

Question 4

In [22]:
```
'''
4. Write a program to implement these formulae of permutations and combination
s.
Number of permutations of n objects taken r at a time: p(n, r) = n! / (n-r)!.
 Number of
combinations of n objects taken r at a time is: c(n, r) = n! / (r!*(n-r)!) = p
(n,r) / r!
'''
```

Out[22]:
```
'\n4. Write a program to implement these formulae of permutations and combina
tions.\nNumber of permutations of n objects taken r at a time: p(n, r) = n! /
(n-r)!. Number of\ncombinations of n objects taken r at a time is: c(n, r) =
n! / (r!*(n-r)!) = p(n,r) / r!\n'
```

In [23]:
```
'''
Notes -- Simple Iteration Nothing Fancy
'''
```

Out[23]:
```
'\nNotes -- Simple Iteration Nothing Fancy\n'
```

In [24]:
```
def factorail_of_A_Number(num):
    factorail = 1
    for i in range(1,num+1):
        factorail = factorail * i
    return factorail
```

In [25]:
```
# permutation Formula
def permution(n,r):
    return factorail_of_A_Number(n)/(factorail_of_A_Number(n-r)  )
```

In [26]:
```
print(permution(5,2))
```

```
20.0
```

In [27]:
```
# combination Formula
def combination(n,r):
    return factorail_of_A_Number(n)/(factorail_of_A_Number(n-r) * factorail_of
_A_Number(r) )
```

In [28]:
```
print(combination(5,2))
```

```
10.0
```

In [ ]:
```

```

Question 5

In [29]:
```
'''
5. Write a function that converts a decimal number to binary number
'''
```

Out[29]: '\n5. Write a function that converts a decimal number to binary number\n'

In [ ]:
```
'''
I used Some Refrence
'''
```

In [82]:
```
def binary_number(num):
    binary_number = []
    while num >0:
        rem = num%2
        num = int(num/2)
        binary_number.append(rem)
    binary_number.reverse()
    print(binary_number)
```

In [92]:
```
binary_number(17)
```

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

Question 6

In [34]:
```
'''
6. Write a function cubesum() that accepts an integer and returns the sum of t
he cubes of
individual digits of that number. Use this function to make functions PrintArm
strong() and
isArmstrong() to print Armstrong numbers and to find whether is an Armstrong n
umber.
'''
```

Out[34]: '\n6. Write a function cubesum() that accepts an integer and returns the sum of the cubes of\nindividual digits of that number. Use this function to make functions PrintArmstrong() and\nisArmstrong() to print Armstrong numbers and to find whether is an Armstrong number.\n'

In [ ]:
```
'''
For Your Given Statement i can find Armstrong Number upto 1000 only and it's n
ot a generalise Solution
more generic
'''
```

In [69]:
```
def cubesum(num):
    sum = 0
    while num>0:
        rem = num%10
        sum = sum + (rem*rem*rem)
        num = int(num/10)
    return sum
```

In [71]:
```
cubesum(153)
```

Out[71]: 153

In [75]:
```
def PrintArmstrong(num):
    if num == cubesum(num):
        print("Number is Armstrong Number {}".format(num))
    else:
        print("Number is not Armstrong Number {}".format(num))
```

In [76]:
```
PrintArmstrong(153)
```

Number is Armstrong Number 153

In [77]:
```
def PrintArmstrong(num):
    if num == cubesum(num):
        return True
    else:
        return False
```

In [78]:
```
PrintArmstrong(153)
```

Out[78]: True

Question 7

In [35]:
```
'''
7. Write a function prodDigits() that inputs a number and returns the product
 of digits of that
number.
'''
```

Out[35]: '\n7. Write a function prodDigits() that inputs a number and returns the product of digits of that\nnumber.\n'

In [36]:
```
'''
Notes --- Digit Operations reminder and number operation
'''
```

Out[36]: '\nNotes --- Digit Operations reminder and number operation\n'

In [37]:
```
def digit_operation(num):
    product = 1
    while num >0:
        rem = num % 10
        num = int(num / 10)
        product = product* rem
    return product
```

In [38]:
```
print(digit_operation(782))
```

```
112
```

In [ ]:

Question 8

In [39]:
```
'''
8. If all digits of a number n are multiplied by each other repeating with the
product, the one
digit number obtained at last is called the multiplicative digital root of n.
 The number of
times digits need to be multiplied to reach one digit is called the multiplica
tive
persistance of n.
Example: 86 -> 48 -> 32 -> 6 (MDR 6, MPersistence 3)
 341 -> 12->2 (MDR 2, MPersistence 2)
Using the function prodDigits() of previous exercise write functions MDR() and
MPersistence() that input a number and return its multiplicative digital root
 and
multiplicative persistence respectively
'''
```

Out[39]:
```
'\n8. If all digits of a number n are multiplied by each other repeating with
the product, the one\ndigit number obtained at last is called the multiplicat
ive digital root of n. The number of\ntimes digits need to be multiplied to r
each one digit is called the multiplicative\npersistance of n.\nExample: 86 -
> 48 -> 32 -> 6 (MDR 6, MPersistence 3)\n 341 -> 12->2 (MDR 2, MPersistence
2)\nUsing the function prodDigits() of previous exercise write functions MDR
() and\nMPersistence() that input a number and return its multiplicative digi
tal root and\nmultiplicative persistence respectively\n'
```

In [40]:
```
'''
Notes --- Enter A Boundary Condition
'''
```

Out[40]:
```
'\nNotes --- Enter A Boundary Condition\n'
```

In [41]:
```
def MDR(num):
    result = num
    while result>0:
        if result>0 and result<10:
            break
        result = digit_operation(result)
    return result
```

In [42]:
```
MDR(141)
```

Out[42]: 4

In [43]:
```
def MPersistence(num):
    result = num
    count = 0
    while result>0:
        count +=1
        if result>0 and result<10:
            break
        result = digit_operation(result)
    return count
```

In [44]:
```
MPersistence(143)
```

Out[44]: 3

In [ ]:

Question 9

In [45]:
```
''' Write a function sumPdivisors() that finds the sum of proper divisors of a number. Proper
divisors of a number are those numbers by which the number is divisible, except the
number itself. For example proper divisors of 36 are 1, 2, 3, 4, 6, 9, 18
'''
```

Out[45]: ' Write a function sumPdivisors() that finds the sum of proper divisors of a number. Proper\ndivisors of a number are those numbers by which the number is divisible, except the\nnumber itself. For example proper divisors of 36 are 1, 2, 3, 4, 6, 9, 18\n'

In [46]:
```
'''
Nothing Fancy very Simple One Iteration
'''
```

Out[46]: '\nNothing Fancy very Simple One Iteration\n'

In [47]:
```python
k = []
def proper_divisor(num):
    for i in range(1,num):
        if num%i ==0:
            k.append(i)
    print(k)
```

In [48]:
```python
proper_divisor(5555)
```

```
[1, 5, 11, 55, 101, 505, 1111]
```

In [ ]:

Question 10

In [49]:
```python
'''
10. A number is called perfect if the sum of proper divisors of that number is equal to the
number. For example 28 is perfect number, since 1+2+4+7+14=28. Write a program to
print all the perfect numbers in a given range
'''
```

Out[49]:
```
'\n10. A number is called perfect if the sum of proper divisors of that numbe
r is equal to the\nnumber. For example 28 is perfect number, since 1+2+4+7+14
=28. Write a program to\nprint all the perfect numbers in a given range\n'
```

In [50]:
```python
def sum_of_proper_divisors(num):
    sum =0
    for i in range(1,num):
        if num%i ==0:
            sum+=i
    return sum
```

In [51]:
```python
def range_of_Perfect_number(num1,num2):
    all_perfect_number_in_range = []
    for i in range(num1,num2+1):
        if i == sum_of_proper_divisors(i):
            all_perfect_number_in_range.append(i)
    print(all_perfect_number_in_range)
```

In [52]:
```python
range_of_Perfect_number(10,10000)
```

```
[28, 496, 8128]
```

In [ ]:

Question 11

In [53]:
```
'''
11. Two different numbers are called amicable numbers if the sum of the proper
divisors of
each is equal to the other number. For example 220 and 284 are amicable number
s.
Sum of proper divisors of 220 = 1+2+4+5+10+11+20+22+44+55+110 = 284
Sum of proper divisors of 284 = 1+2+4+71+142 = 220
Write a function to print pairs of amicable numbers in a range
'''
```

Out[53]:
```
'\n11. Two different numbers are called amicable numbers if the sum of the pr
oper divisors of\neach is equal to the other number. For example 220 and 284
are amicable numbers.\nSum of proper divisors of 220 = 1+2+4+5+10+11+20+22+44
+55+110 = 284\nSum of proper divisors of 284 = 1+2+4+71+142 = 220\nWrite a fu
nction to print pairs of amicable numbers in a range\n'
```

In [54]:
```
'''
Nothing Fancy Some loop Concept is used
'''
```

Out[54]:
```
'\nNothing Fancy Some loop Concept is used\n'
```

In [55]:
```
def amicable_number(num1,num2):
    ameicalbel_list = []
    for i in range(num1,num2+1):
        for j in range(num1,num2+1):
            if
            if i == j:
                continue
            if i == sum_of_proper_divisors(j) and j == sum_of_proper_divisors(
i):
                ameicalbel_list.append([i,j])
    print(ameicalbel_list)
```

In [56]:
```
amicable_number(1,300)
```

```
[[220, 284], [284, 220]]
```

In [66]:
```
amicable_number = [[220, 284], [284, 220]]
```

Question 12

In [58]:
```
'''
12. Write a program which can filter odd numbers in a list by using filter fun
ction
'''
```

Out[58]: '\n12. Write a program which can filter odd numbers in a list by using filter function\n'

In [59]:
```
list1 = list(range(-100,101))
print(list(filter(lambda x:(x%2!=0),list1)))
```

```
[-99, -97, -95, -93, -91, -89, -87, -85, -83, -81, -79, -77, -75, -73, -71, -
69, -67, -65, -63, -61, -59, -57, -55, -53, -51, -49, -47, -45, -43, -41, -3
9, -37, -35, -33, -31, -29, -27, -25, -23, -21, -19, -17, -15, -13, -11, -9,
-7, -5, -3, -1, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 3
3, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 7
1, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99]
```

Question 13

In [60]:
```
'''
13. Write a program which can map() to make a list whose elements are cube of
 elements in
a given list
'''
```

Out[60]: '\n13. Write a program which can map() to make a list whose elements are cube of elements in\na given list\n'

In [61]:
```
list1 = list(range(-10,11))
print(list(map(lambda x:x**3,list1)))
```

```
[-1000, -729, -512, -343, -216, -125, -64, -27, -8, -1, 0, 1, 8, 27, 64, 125,
216, 343, 512, 729, 1000]
```

Quesion 14

In [62]:
```
'''
Write a program which can map() and filter() to make a list whose elements are
cube of
even number in a given list
'''
```

Out[62]: '\nWrite a program which can map() and filter() to make a list whose elements are cube of\neven number in a given list\n'

In [63]:
```
list1 = list(range(-10,11))
```

In [64]:
```
print(list(map(lambda x:x**3,list(filter(lambda x:(x%2==0),list1)))))
```

```
[-1000, -512, -216, -64, -8, 0, 8, 64, 216, 512, 1000]
```

In [ ]: