```
1 # create a recursive function power that receives
    two arguments as number and power and return the
    resultant power of
2 # given number and power
3
4
5 def power(x, y):
6     if y == 0:
7         return 1
8
9     else:
10        return x * power(x, y - 1)
11
12
13 print(power(2, 4))
14
```

```
1  # WAF to print a Tribonacci series for the given
   step size
2
3  def trib(x):
4      if x == 0 or x == 1 or x == 2:
5          return 0
6
7      elif x == 3:
8          return 1
9
10     else:
11         return trib(x - 1) + trib(x - 2) + trib(x
    - 3)
12
13
14 n = int(input('Enter a Number: '))
15 print('Tribonacci series: ', end=' ')
16
17 for i in range(1, n):
18     print(trib(i), end=', ')
19
```

```
1  """WAF to check for Armstrong Numbers An
   Armstrong number of three digits is an integer
   such that the sum of the
2  cubes of its digits is equal to the number itself
   . For example, 371 is an Armstrong number since 3
   **3 + 7**3 + 1**3 =
3  371 """
4
5  def armstr(x):
6      power = len(str(x))
7      sigma = 0
8
9      for i in str(x):
10         sigma += int(i) ** power
11
12     if sigma == x:
13         print('Armstrong Number')
14
15     else:
16         print('Not Armstrong')
17
18
19 num = int(input('Enter a Number: '))
20 armstr(num)
21
```

```
1  """WAF to check for Automorphic numbers
2  An automorphic number is a natural number in a
   given number base b whose
3  square "ends" in the same digits as the number
   itself """
4
5  def automorph(x):
6      square = str(x ** 2)
7      slicing_index = -(len(str(x)))
8
9      if str(x) == square[slicing_index::]:
10         print('Automorphic Number')
11
12     else:
13         print('Not Automorphic')
14
15
16 num = int(input('Enter a Number: '))
17 automorph(num)
18
```

```python
# WAF to reverse a string

def rev(x):
    ind = len(x)-1
    print(x[ind], end='')
    if ind > 0:
        return rev(x[:ind])

    else:
        return ''


name = input('Enter Name: ')
rev(name)

```