```
# ShowPF.py
"""Examines two implementations of the perfect
shuffle.
"""

def PF1(x):
    """ The items in x are permuted via the perfect shuffle.

    PreC: x references a list with even length
    """
    n = len(x)
    m = n/2
    # Cut the list in two...
    top = list(x[:m])
    bot = list(x[m:])
    for k in range(m):
        # Process the kth item from the top and bottom halves
        x[2*k]   = top[k]
        x[2*k+1] = bot[k]


def PF2(x):
    """ Returns a reference to a list that is a perfect shuffle of x

    PreC: x references a list with even length
    """
    n = len(x)
    m = n/2
    y = []
    for k in range(m):
        # Append the kth item from the top half
        y.append(x[k])
        # append the kth item from the bottom half
        y.append(x[k+m])
    return y

if __name__ == '__main__':
    """ To appreciate the idea that lists are objects we look
    at two perfect shuffle computations, one with the Void function PF1
    and one with the fruitful function PF2."""

    n = raw_input('Enter an even whole number: ')
    n = int(n)
    # Generate a random integer list
    x0 = []
    for k in range(n):
        x0.append(10*k)

    # Repeated perfect shuffles until we retrieve the original list.
    x = list(x0)
    PF1(x)
    numPFs = 1
    while x!=x0:
        PF1(x)
        numPFs+=1
    print 'n = %1d, Number of perfect shuffles = %1d (via PF1)' % (n,numPFs)

    # Repeated perfect shuffles until we retrieve the original list.
```

```
x = PF2(x0)
numPFs = 1
while x!=x0:
    x = PF2(x)
    numPFs+=1
print 'n = %1d, Number of perfect shuffles = %1d (via PF2)' % (n,numPFs)
```