Permutation

1. Gegeben sei eine Liste von n Zahlen perm $= [a_0, a_2, \ldots, a_{n-1}]$ mit der Eigenschaft, dass sie genau die Zahlen $0, 1, 2, \ldots, n-1$ in beliebiger Reihenfolge enhält. Diese Liste könnte eine Permutation darstellen. Schreiben Sie eine Python-Funktion `transpose(perm)`, die eine Liste zurückgibt, in der für alle $i = 1, \ldots, n$ anstatt der Zahl a_i an der Stelle i die Zahl i an der Stelle a_i liegt.

2. Implementieren Sie einen Test für ihren Algorithmus.

Solution:

```
#!/usr/bin/env python
# coding: utf-8
# <h1>Table of Contents<span class="tocSkip"></span></h1>
# <div class="toc"><ul class="toc-item"><li><span><a href="#Permutation" data-toc-modified-id
    ="Permutation-1"><span class="toc-item-num">1  </span>Permutation</a></span></li
    ><li><span><a href="#Inverse-Permutation" data-toc-modified-id="Inverse-Permutation-2"><
    span class="toc-item-num">2  </span>Inverse Permutation</a></span></li></ul></div
    >
# ### Permutation
#
# (i)
def transpose(perm):
    """
    Returns the transpose of a permutaion perm (list, where i -> perm[i]).
    :param perm: list
    :return: list
    """
    permT = [[]]*len(perm)
    for i, a in enumerate(perm):
        permT[a] = i
    return permT
print("Identity")
perm = [1, 0, 2, 3]
print(perm)
print(transpose(perm))
print("Test :", [perm[k] for k in transpose(perm)], "\n")
print("Transposition")
perm = [1, 0, 2, 3]
print(perm)
print(transpose(perm))
print("Test :", [perm[k] for k in transpose(perm)], "\n")
print("Cycle")
perm = [1, 2, 3, 0]
print(perm)
print(transpose(perm))
print("Test :", [perm[k] for k in transpose(perm)], "\n")
# (ii) Die transponierte einer Permutation ist ihre Inverse. Das erm licht hier einen
    einfachen Test.
print([perm[k] for k in transpose(perm)])
# is the short form of
newList = []
for k in transpose(perm):
    newList.append(perm[k])
print(newList)
# ### Inverse Permutation
```

```
#
# **Pseudo Code**
Input of some permutation L = (a0, ..., an-1) of length n.
Create empty list M
for all k in 0,..,n-1
    a = L[k]
    M[a] = k

print("L", L)
print("M", M)
# **Python**
L = [3,1,0,2]
def invPerm(L):
    """docstring"""
    n = len(L)  # length of list L
    M = list(range(n))  # placeholder list for result
    for k in range(n):  # iterate through the list, index by index
        a = L[k]  # a is the k-th element of L
        M[a] = k  # M_a is set to the index of a in L, i.e., k
    return M
print("L", L)
M = invPerm(L)
print("M", M)
# **Test**
#
# Da es sich um die inverse Permutation handelt sollte die Anwendung der Inversen auf die
    urspr ngliche Permutation immer die Identit t ergeben.
# M is the "right inverse" of L
print([L[m] for m in M])
# M is the "left inverse" of L
print([M[l] for l in L])
```