```
# Name Dipika Sharma
# DSC 540-T302 Data Preparation
# Week 1&2

# Exercise 4.c.
# Activity 3

# Look up the definition of permutations and dropwhile from itertools.
from itertools import permutations, dropwhile

permutations?

Init signature: permutations(iterable, r=None)
Docstring:
Return successive r-length permutations of elements in the iterable.

permutations(range(3), 2) --> (0,1), (0,2), (1,0), (1,2), (2,0), (2,1)
Type:           type
Subclasses:
```

[11]:

```
# Activity 3

# Look up the definition of permutations and dropwhile from itertools.
from itertools import permutations, dropwhile

dropwhile?
Init signature: dropwhile(predicate, iterable, /)
Docstring:
Drop items from the iterable while predicate(item) is true.

Afterwards, return every element until the iterable is exhausted.
Type:           type
Subclasses:
```

[12]:

```
# Write an expression to generate all the possible three digit numbers using 0, 1, and 2
from itertools import permutations

comb = permutations([0, 1, 2], 3)

for i in comb:
    print(i)
(0, 1, 2)
(0, 2, 1)
(1, 0, 2)
(1, 2, 0)
(2, 0, 1)
(2, 1, 0)
```

[13]:

```
# Loop over the iterator expression you generated before.
# Use print to print each element returned by the iterator.
# Use assert and isinstance to make sure that the elements are of type tuple
from itertools import permutations

for number_tuple in permutations([0, 1, 2], 3):
    print(number_tuple)
    assert isinstance(number_tuple, tuple)
```

```
(0, 1, 2)
(0, 2, 1)
(1, 0, 2)
(1, 2, 0)
(2, 0, 1)
(2, 1, 0)
```

[14]:

```
# Write the loop again. But this time use dropwhile with a lambda expression
# to drop any leading zeros from the tuples. As an example (0, 1, 2) will
# become [1, 2]. Also cast the output of the dropwhile to a list.

for number_tuple in permutations([0, 1, 2], 3):
    print(list(dropwhile(lambda x: x <= 0, number_tuple)))
```

```
[1, 2]
[2, 1]
[1, 0, 2]
[1, 2, 0]
[2, 0, 1]
[2, 1, 0]
```

[16]:

```
# Write all the logic you had written above, but this time write a separate
# function where you will be passing the list generated from dropwhile and the
# function will return the whole number contained in the list.
# As an example if you pass [1, 2] to the fucntion it will return 12 to you.
# Make sure that the return type is indeed a number and not a string.
# Although this task can be achieved using some other tricks,
# we require that you treat the incoming list as a stack in the function and
# generate the number there.

import math
def convert_to_number(number_stack):
    final_number = 0
    for i in range(0, len(number_stack)):
        final_number += (number_stack.pop() * (math.pow(10, i)))
    return final_number

for number_tuple in permutations([0, 1, 2], 3):
    number_stack = list(dropwhile(lambda x: x <= 0, number_tuple))
    print(int(convert_to_number(number_stack)))
```

```
12
21
102
120
201
210
```