Loops

only_adults

Write a function "only_adults" that takes as input a list of numbers and returns only those numbers >= 18

In [6]:
```
def only_adults(age_list:list) -> list:
    return [age for age in age_list if age >=18]

print(only_adults([13, 17, 45, 47, 75]))
print(only_adults([13, 17, 15, 16, 12]))
```

```
[45, 47, 75]
[]
```

get_only_adults

Write a function "get_only_adults" that takes as input a list of numbers and returns only those numbers >= 18 and removes any None values from the list

Version 1

In [5]:
```
def get_only_adults(age_list:list) -> list:
    adults_only = []

    for age in age_list:
        if age is not None and age >= 18:
                adults_only.append(age) # adults_only += [age] #
    return adults_only

print(get_only_adults([None, 17, None, 47, 75, None, None, 12, 13, 95, 85]))
```

```
[47, 75, 95, 85]
```

Version 2

In [91]:
```
def get_only_adults1(age_list:list) -> list:

    return [age for age in age_list if isinstance(age,(int, float)) and age >= 18]

print(get_only_adults1([None, 17, None, 47, 75, None, None, 12, 13, 95, 85]))
```

```
[47, 75, 95, 85]
```

are_all_adults

Write a function "are_all_adults" that takes as input a list of numbers and returns True if they are all >= 18, and returns False otherwise Is this a map, filter, or reduce?

In [41]:
```
def are_all_adults(num_list: list) -> bool:
    for age in num_list:
         if age < 18:
            return False
    return True

print("Are we all adults here?", are_all_adults([13, 17, 15, 16, 12])) # no adults
print("Are we all adults here?", are_all_adults([18, 17, 15, 16, 22])) # some adults
print("Are we all adults here?", are_all_adults([19, 25, 83, 19, 23])) # only adults
```

```
Are we all adults here? False
Are we all adults here? False
Are we all adults here? True
```

The operation applied is a type of **filter**. The size of the list is reduced to one item, a boolean. This is a **reduce**.

count_nones

Write a function "count_nones" that takes as input a list of any type of element and returns a count of how many of those elements are None types.

Is this a map, filter, or reduce?

Version 1

In [47]:
```
def count_nones(my_list:list):
    num_nones = 0

    for i in my_list:
        if i == None:
            num_nones += 1
    return num_nones


print(count_nones([None, 1, 2, 3, None, None, None, None]))
```

```
5
```

Version 2

In [46]:
```
def count_nones(my_list:list):
    num_nones = sum(1 for item in my_list if item == None)
    return num_nones

print(count_nones([None, 1, 2, 3, None, None, None, None]))
```

```
5
```

Filering out Nones. The size of the list is reduced to one item, an integer. This is a **reduce**.

longest_word

Write a function "longest_word" that takes as input a list of strings and returns the longest string in the list. Hint: you will need to use two "accumulators"

In [6]:
```
def longest_word(word_list:list):
    my_word_length = 0
    my_word = ''

    for word in word_list:
        if len(word) > my_word_length:
            my_word_length = len(word)
            my_word = word

    return my_word, my_word_length

print(longest_word(['Jerusalem', 'Tokyo', 'Sydney', 'Antananarivo', 'Madrid']))
```

```
('Antananarivo', 12)
```

factorial

Write a function "factorial"

It takes a number and returns the factorial of that number.

The factorial of n is the product of all positive integers less than or equal to n

HINT: use range() https://docs.python.org/3/library/stdtypes.html#typesseq-range

range(n) produces an iterable of length n: [0,1,2,...,n-1]

In [9]:
```
def factorial(my_number:int):

    my_factorial = 1

    if (my_number >=0 and isinstance(my_number, int)):

        for i in list(range(1, my_number+1)):
            my_factorial *= i
        return my_factorial

    else:
        return "Only integers are allowed as an input"

print(factorial(0))
print(factorial(1))
print(factorial(2))
print(factorial(3))

#print('------------ Incorrect input variables ------')
#print(factorial(-1))
```

```
#print(factorial(2.0))
```

```
1
1
2
6
```

n_highest_number

Write a function "n_highest_number" with two parameters:

1. a list of numbers
2. an integer

"n_highest_number" should return the nth highest number in the list, where n is the second parameter of the function.

Assume that the numbers will be unique (no duplicates). Also assume that n <= the number of elements in the list.

NOTE: Only use the operations and functions we have learned so far! No cheating!

HINT: Can you reuse anything from the previous exercise? That may or may not work, depending on how you implemented it.

HINT HINT: use the function "range"!

In [11]:
```
def n_highest_number(my_num_list: list, n:int):

    ordered_list = sorted(my_num_list, reverse=True)
    return ordered_list[n-1]

print(n_highest_number([5, 2, 7, 1, 9], 1))
print(n_highest_number([5, 2, 7, 1, 9], 2))
print(n_highest_number([5, 2, 7, 1, 9], 3))


# using Range:

def n_highest_number_1(list_of_nums: list, n: int):
    for i in range(n):
        max_num = highest_number(list_of_nums)
        list_of_nums.remove(max_num)
    return max_num

print("using range:", n_highest_number([5, 2, 7, 1, 9], 3))
```

```
9
7
5
using range: 5
```

In [ ]:
```

```