Notebook

August 2, 2024

[3]:
```
#### Arrays

arr = [1, 2, 3, 4, 5]
arr.append(6) # add 6 to the end of the list [1, 2, 3, 4, 5, 6]
arr.insert(0, 0) # add 0 to the beginning of the list [0, 1, 2, 3, 4, 5, 6]
arr.extend([7, 8, 9, 11]) # add 7, 8, 9, 11 to the end of the list [0, 1, 2, 3,␣
  ↪4, 5, 6, 7, 8, 9, 11]
arr.remove(11) # remove 11 from the list [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
arr.pop(0) # remove the first element from the list [1, 2, 3, 4, 5, 6, 7, 8, 9]
arr.reverse() # reverse the list [9, 8, 7, 6, 5, 4, 3, 2, 1]
print(arr)

#### Slicing
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print(arr[0:3]) # [1, 2, 3]
print(arr[:3]) # [1, 2, 3]
print(arr[3:]) # [4, 5, 6, 7, 8, 9]
print(arr[::2]) # [1, 3, 5, 7, 9]
print(arr[::-1]) # [9, 8, 7, 6, 5, 4, 3, 2, 1] reverse the list
print(arr[1:8:2]) # [2, 4, 6, 8]
```

```
[9, 8, 7, 6, 5, 4, 3, 2, 1]
```

[ ]:
```
#### strings

string = "Hello, World!"
print(string[0]) # H
print(string[1:5]) # ello
print(string[:5]) # Hello

#### string methods

string.isdigit() # False
string.isalpha() # False
string.islower() # False
string.isupper() # False
string.isspace() # False
string.find("World") # 7
string.rfind("World") # 7
string.replace("World", "Python") # Hello, Python!
string.split(",") # ['Hello', ' World!']
string.strip() # Hello, World!
```

[3]:
```
#### sort

arr = [1, 2, 3, 4, 5]
arr.sort() # sort in place
print(arr) # [1, 2, 3, 4, 5]

arr.sort(reverse=True) # sort in place
print(arr) # [5, 4, 3, 2, 1]

arr = [[1, 2], [3, 4], [5, 6]]
arr.sort(key=lambda x: x[1]) # sort by second element
print(arr) # [[1, 2], [3, 4], [5, 6]]

arr.sort(key=lambda x: -x[0]) # sort by first element
print(arr) # [[5, 6], [3, 4], [1, 2]]
```

```
[1, 2, 3, 4, 5]
[5, 4, 3, 2, 1]
[[1, 2], [3, 4], [5, 6]]
[[5, 6], [3, 4], [1, 2]]
```

[1]:
```
#### Counter
from collections import Counter
a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(Counter(a)) # Counter({1: 1, 2: 1, 3: 1, 4: 1, 5: 1, 6: 1, 7: 1, 8: 1, 9:␣
 ↪1, 10: 1})

str1 = "Hello World"
print(Counter(str1)) # Counter({'l': 3, 'o': 2, 'H': 1, 'e': 1, ' ': 1, 'W': 1,␣
 ↪'r': 1, 'd': 1})
```

```
Counter({1: 1, 2: 1, 3: 1, 4: 1, 5: 1, 6: 1, 7: 1, 8: 1, 9: 1, 10: 1})
```

[2]:
```
####  dictionary

str1 = "Hello World"
print(dict(Counter(str1))) # {'H': 1, 'e': 1, 'l': 3, 'o': 2, ' ': 1, 'W': 1,␣
 ↪'r': 1, 'd': 1}

#### Get all the keys
print(Counter(str1).keys()) # dict_keys(['H', 'e', 'l', 'o', ' ', 'W', 'r',␣
 ↪'d'])
```

```
#### Get all the values
print(Counter(str1).values()) # dict_values([1, 1, 3, 2, 1, 1, 1, 1])

#### Get all the items
print(Counter(str1).items())
```

```
{'H': 1, 'e': 1, 'l': 3, 'o': 2, ' ': 1, 'W': 1, 'r': 1, 'd': 1}
dict_keys(['H', 'e', 'l', 'o', ' ', 'W', 'r', 'd'])
dict_values([1, 1, 3, 2, 1, 1, 1, 1])
dict_items([('H', 1), ('e', 1), ('l', 3), ('o', 2), (' ', 1), ('W', 1), ('r',
1), ('d', 1)])
```

[4]:
```
#### map function

arr = ['1', '2', '3', '4', '5']

arr = list(map(int, arr))

print(arr) # [1, 2, 3, 4, 5]
```

```
[1, 2, 3, 4, 5]
```

[1]:
```
m = {}
m[1] = 111

# setdefault(key, value) --> if key already exists then its value is returned,␣
 ↪if not then key is inserted with value

val = m.setdefault(1, 11)
print(m, val)
```

```
{1: 111} 111
```

[ ]:
```
# Differnt sorting algorithms

class Solution:
    def sortArray(self, nums: List[int]) -> List[int]:
        # self.quickSort(nums)
        # self.mergeSort(nums)
        # self.bubbleSort(nums)
        # self.insertionSort(nums)
                # self.selectionSort(nums)
        self.heapSort(nums)
        return nums

        # @bubbleSort, TLE
    def bubbleSort(self, nums):
        n = len(nums)
        for i in range(n):
            for j in range(0, n - i - 1):
                if nums[j] > nums[j + 1]:
                    nums[j], nums[j + 1] = nums[j + 1], nums[j]

        # @insertionSort, TLE
    def insertionSort(self, nums):
        for i in range(1, len(nums)):
            key = nums[i]
            j = i-1
            while j >= 0 and key < nums[j] :
                    nums[j + 1] = nums[j]
                    j -= 1
            nums[j + 1] = key

        # @selectionSort, TLE
    def selectionSort(self, nums):
        for i in range(len(nums)):
            _min = min(nums[i:])
            min_index = nums[i:].index(_min)
            nums[i + min_index] = nums[i]
            nums[i] = _min
        return nums

        # @quickSort
    def quickSort(self, nums):
        def helper(head, tail):
            if head >= tail: return
            l, r = head, tail
            m = (r - l) // 2 + l
            pivot = nums[m]
            while r >= l:
                while r >= l and nums[l] < pivot: l += 1
                while r >= l and nums[r] > pivot: r -= 1
                if r >= l:
                    nums[l], nums[r] = nums[r], nums[l]
                    l += 1
                    r -= 1
            helper(head, r)
            helper(l, tail)

        helper(0, len(nums)-1)
        return nums

        # @mergeSort
    def mergeSort(self, nums):
        if len(nums) > 1:
            mid = len(nums)//2
            L = nums[:mid]
            R = nums[mid:]

            self.mergeSort(L)
            self.mergeSort(R)

            i = j = k = 0

            while i < len(L) and j < len(R):
                if L[i] < R[j]:
                    nums[k] = L[i]
                    i+=1
                else:
                    nums[k] = R[j]
                    j+=1
                k+=1

            while i < len(L):
                nums[k] = L[i]
                i+=1
                k+=1

            while j < len(R):
                nums[k] = R[j]
                j+=1
                k+=1

   # @heapSort
    def heapSort(self, nums):
        def heapify(nums, n, i):
            l = 2 * i + 1
            r = 2 * i + 2

            largest = i
            if l < n and nums[largest] < nums[l]:
                largest = l

            if r < n and nums[largest] < nums[r]:
                largest = r

            if largest != i:
                nums[i], nums[largest] = nums[largest], nums[i]

                heapify(nums, n, largest)

        n = len(nums)
        for i in range(n//2+1)[::-1]:
            heapify(nums, n, i)

        for i in range(n)[::-1]:
            nums[i], nums[0] = nums[0], nums[i]
            heapify(nums, i, 0)
```

[ ]:

```
# reduce similar to js

from functools import reduce
class Solution:
    def longestCommonPrefix(self, arr):
        def prefix(x, y):
            ans = ""
            for i in range(min(len(x), len(y))):
                if x[i] == y[i]:
                    ans += x[i]
                else:
                    break
            return ans

        if not arr:
            return "-1"

        ans = reduce(lambda x, y: prefix(x, y), arr)
        return "-1" if ans == "" else ans
```