Loops

In Python, there are several loop constructs commonly used for solving array problems. Here are the most common ones:

1. **For Loop**: The `for` loop is widely used for iterating over elements in an array or any iterable object.
   It allows you to execute a block of code for each element in the array.

```
arr = [1, 2, 3, 4, 5]
for num in array:
  print(num)
```

2. **While Loop**:
   The `while` loop is used when you need to execute a block of code repeatedly as long as a condition is true.
   It's often used when you don't know in advance how many iterations are needed.

```
i = 0
while i < len(array):
  print(array[i])
  i += 1
```

3. **Nested Loops**: Nested loops involve placing one loop inside another loop. They are used when you need to iterate over elements of a multi-dimensional array or perform operations that require multiple levels of iteration.

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
for row in matrix:
    for num in row:
        print(num)
```

   Typically used in brute force solutions where need to compare every permutation.

```
nums=[1,2,3,4,4]
for i in range(len(nums)):  # Outer loop iterates over the rows
    print(f'loop {i}')
    for j in range(i+1, len(nums)):  # Inner loop iterates over the columns
        print(nums[i],nums[j])
```

```
loop 0
1 2
1 3
1 4
1 4
loop 1
2 3
2 4
2 4
loop 2
3 4
3 4
loop 3
4 4
loop 4
```

4. **Enumerate**: The `enumerate()` function is used to iterate over both the elements and their indices in an array. It returns tuples containing the index and the corresponding element. For example:

```
for i, num in enumerate(arr):
  print(f"Element at index {i} is {num}")
```

5. **Range:** In Python, the `range()` function is used to generate a sequence of numbers. It's commonly used in loops to iterate over a specific range of numbers.
   - If called with one argument, it generates numbers starting from `0` up to (but not including) the specified number.
   - If called with two arguments, it generates numbers starting from the first argument up to (but not including) the second argument.
   - If called with three arguments, it generates numbers starting from the first argument up to (but not including) the second argument, with the specified step size (i.e., the difference between consecutive numbers).
     1. Generating numbers from 0 to 5 (exclusive):

```
for i in range(6):
  print(i)
# Output: 0, 1, 2, 3, 4, 5
```

     2. Generating numbers from 2 to 8 (exclusive):

```
for i in range(2, 9):
  print(i)
# Output: 2, 3, 4, 5, 6, 7, 8
```

3. Generating numbers from 1 to 10 (exclusive), with a step size of 2:

```
for i in range(1, 11, 2):
  print(i)
# Output: 1, 3, 5, 7, 9
```

4. Considering that the end of the range is excluded, when iterating entire array it needs to be -1

```
for i in range(len(array) - 1):
  print(i)
```

It's important to note that `range()` returns a range object, which is a memory-efficient representation of the sequence of numbers. To actually see the numbers in the sequence, you typically use it within a loop or convert it to a list using `list(range(...))`.

Revision #10
Created 18 February 2024 22:23:28 by victor
Updated 20 February 2024 05:57:53 by victor