6.101 Recitation 13: Week 8 Iterables, Iterators, and Generators **4/1/24**

This sheet is yours to keep!

1) **What is an iterable? An iterator? A generator? What are the similarities and differences between them?**

2) **How do you make an iterator from an iterable object in Python?**

3) What will the following example code below output?

```
def gen_range(start, stop, step=1):
    print("HI 1")
    assert step >= 1
    current = start
    while current < stop:
        print("HI 2")
        yield current
        print("HI 3")
        current += step

    print("HI 4")

print("Example 1:")
x = gen_range(10, 13)
y = x
print(x)




print("Example 2:")
print(next(x))




print("Example 3:")
print(next(y))
print(next(x))




print("Example 4:")
print(next(y))
print(next(x))
```

R13 Participation Credit **Kerberos : _____@mit.edu**

Hand this sheet in at the end of recitation to get participation credit for today.

4) Rewrite the my_enumerate function as a generator!

```
def my_enumerate(x):
    """
    A function that returns a list of (index, element) tuples similar to
    enumerate without using enumerate!
    """
    result = []
    i = 0
    for y in x:
        result.append((i, y))
        i += 1
    return result
```

```
def my_enumerate(x):
    # your code here
```

5) Implement the my_zip function below as a generator without using zip!

```
def my_zip(x, y):
    # your code here
```

6) Edit the flatten function below to turn it into a generator

```
def flatten(x):

    out = []

    for elt in x:

        if isinstance(elt, list):

            out.extend(flatten(elt))

        else:

            out.append(elt)

    return out



x = [1, [2, [3, [4]]]]
y = [[[[[[[1, 2, 3]]]]], 4], 5]
z = [[[[[[[[[[1]]]]]]]]]]
assert list(flatten(x)) == [1, 2, 3, 4]
assert list(flatten(y)) == [1, 2, 3, 4, 5]
assert list(flatten(z)) == [1]
```