```
import math
import random
import numpy

# the classical one-qbit states.
ket0 = numpy.array([1 + 0j, 0 + 0j])
ket1 = numpy.array([0 + 0j, 1 + 0j])

# Write this function. Its input is a one-qbit state. It returns either ket0 or ket1
.
def measurement(state):
    naught_prob = math.pow(numpy.absolute(state[0]), 2)
    rand = random.random()
    if rand <= naught_prob:
        return ket0
    else:
        return ket1

# For large m, this function should print a number close to 0.64 (Why?)
def measurementTest345(m):
    psi = 0.6 * ket0 + 0.8 * ket1
    def f():
        if (measurement(psi) == ket0).all():
            return 0
        else:
            return 1
    acc = 0
    for i in range(m):
        acc += f()
    return acc / m

if __name__ == '__main__':
    print(measurementTest345(100000000)) # 100 million
    ### outputs ###
    # ¿  cs358 git:(master) python3 hw4.py
    # 0.64008279
```