HW3-520

Shangxing Jiang

1.

(1) use function: np.zeros((row, column), dtype)

(2) use function: np.random.randn(size) Then use for loop and count to get the ratio

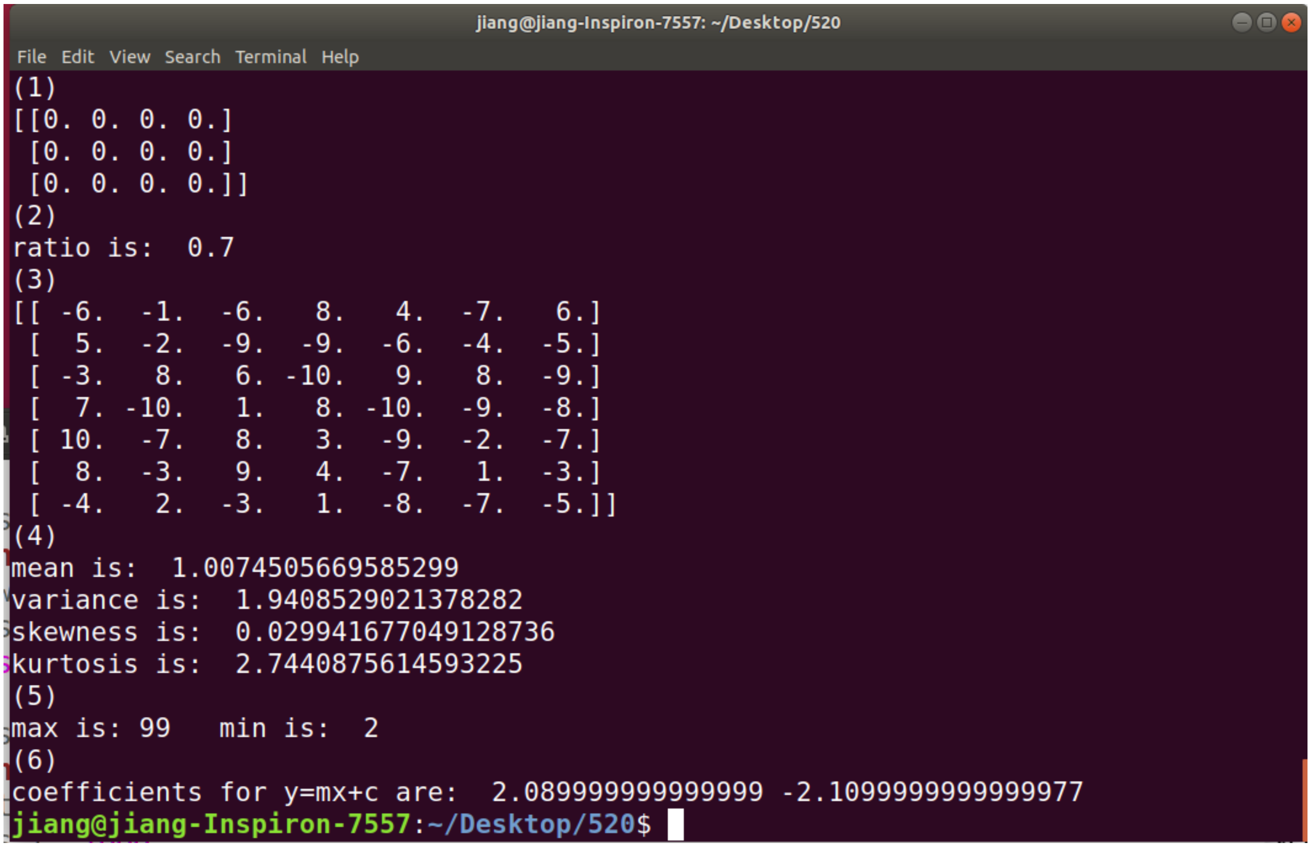
(3) use function: np.random.uniform(low, high, (row, column)) Then use net for loop to convert: for integer, newData = data, then continue; for positive, newData = int(data) +1; for negative, newData = int(data) -1

(4) use function: math.sqrt(variance)*np.random.randn(size) + mean Then built-in function: sum, mean can be used. But for variance, skewness, and kurtosis, we need find the formulas, and type by hand using for loop.

(5) use function: np.random.randint(high, size) Then initialize with max=min=data[0] using for loop to find the max and min

(6) use function: A = np.vstack([x, np.ones(len(x))]).T

m, c = np.linalg.lstsq(A, y, rcond=-1)[0]



jiang@jiang-Inspiron-7557: ~/Desktop/520
File Edit View Search Terminal Help

```
(1)
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
(2)
ratio is:  0.7
(3)
[[ -6.  -1.  -6.   8.   4.  -7.   6.]
 [  5.  -2.  -9.  -9.  -6.  -4.  -5.]
 [ -3.   8.   6. -10.   9.   8.  -9.]
 [  7. -10.   1.   8. -10.  -9.  -8.]
 [ 10.  -7.   8.   3.  -9.  -2.  -7.]
 [  8.  -3.   9.   4.  -7.   1.  -3.]
 [ -4.   2.  -3.   1.  -8.  -7.  -5.]]
(4)
mean is:  1.0074505669585299
variance is:  1.9408529021378282
skewness is:  0.029941677049128736
kurtosis is:  2.7440875614593225
(5)
max is: 99   min is:  2
(6)
coefficients for y=mx+c are:  2.089999999999999 -2.1099999999999977
jiang@jiang-Inspiron-7557:~/Desktop/520$ 
```

2.

1.

LCG

(1) getSeed(), just return seed (2) setSeed(), self.seed = value (3) define x as global variable, start with x = seed (4) x = (self.multiplier*x+self.increment)%self.modulus/self.modulus

(5) use for loop with range(0, parameter) to do next(), append next() to a list

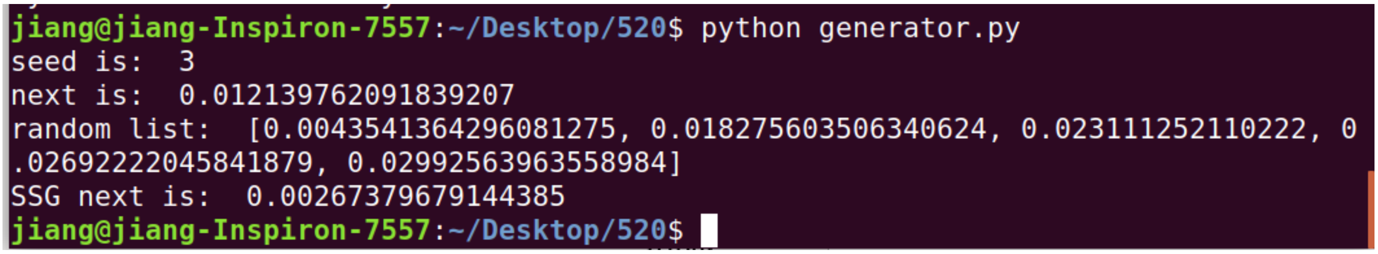
SSG

Change __init__ with adding

“if seed%4 != 2:

raise ValueError("seed % 4 != 2")”

Then I overwrite the __iter__: first yield seed, then for loop with range(0, self.modulus), yield next()

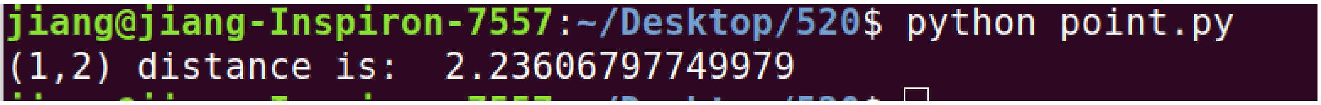


```
jiang@jiang-Inspiron-7557:~/Desktop/520$ python generator.py
seed is:  3
next is:  0.012139762091839207
random list:  [0.004354136429608l275, 0.018275603506340624, 0.023111252110222, 0
.02692222045841879, 0.02992563963558984]
SSG next is:  0.002673796791443850
jiang@jiang-Inspiron-7557:~/Desktop/520$ 
```

2.

Point

Distance = sqrt (x^2 + y^2)

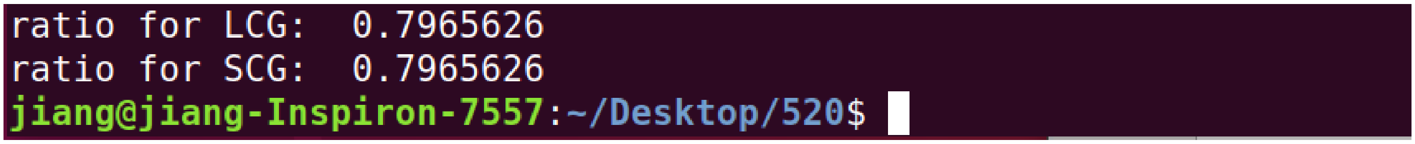


```
jiang@jiang-Inspiron-7557:~/Desktop/520$ python point.py
(1,2) distance is:  2.23606797749979
```

MCTest

Generate double-size random number, then make half of them become negative, then combine. For loop to count distance < 1

SSG and LSG got the same result with same parameters



```
ratio for LCG:  0.7965626
ratio for SCG:  0.7965626
jiang@jiang-Inspiron-7557:~/Desktop/520$ 
```

Leetcode

O(n) without using remove()

At first, I use remove(), but it's really slow. Then I see others use a variable to be the new index and skip the duplicate elements. So I learned. I.e.

```
newIndex = 0
for i in range(length-1):
    if(nums[i] != nums[i+1]):
            newIndex += 1
            nums[newIndex] = nums[i+1]
        return newIndex+1
```

Appendix

```
import numpy as np
import math

#(1)
arr1 = np.zeros((3, 4), dtype = np.float64)
print("(1)")
print(arr1)

#(2)
data = np.random.randn(100)
count = 0.
for i in range(0, 100) :
    if data[i] < 0.4 :
        count += 1
ratio = count / 100
print("(2)")
print("ratio is: ",ratio)

#(3)
data2 = np.random.uniform(-10, 10, (7,7))
newData = np.zeros((7,7))
for i in range(0, 7):
    for j in range(0, 7):
        if data2[i][j] == int(data2[i][j]):
            newData[i][j] = data2[i][j];
            continue;
        if data2[i][j] < 0:
            newData[i][j] = int(data2[i][j])-1
        else:
            newData[i][j] = int(data2[i][j])+1
print("(3)")
print(newData)

#(4)
# standard distribution N(1, 2)  mean-variance
data3 = math.sqrt(2)*np.random.randn(1000) + 1
summ = sum(data3)
mean = summ/1000
print("(4)")
print("mean is: ", mean)
```

```
variance = 0.0
for i in range(0, 1000):
    variance += (data3[i] - mean)**2
variance /= 1000
print("variance is: ", variance)
sd = math.sqrt(variance)   # standard derivation

skewness = 0.0
for i in range(0, 1000):
    skewness += ((data3[i]-mean)/sd)**3
skewness /= 1000
print("skewness is: ", skewness)

kurtosis = 0.0
for i in range(0, 1000):
    kurtosis += ((data3[i]-mean)/sd)**4
kurtosis /= 1000
print("kurtosis is: ", kurtosis)

#(5)
data4 = np.random.randint(100, size=100)  #high = 100
max = min = data4[0]
for i in range(1,100):
    if data4[i] > max:
        max = data4[i]
    if data4[i] < min:
        min = data4[i]
print("(5)")
print("max is:", max, "  min is: ", min)


#(6)
x = np.array([0, 1, 2, 3])
y = np.array([-2.4, 0.5, 1.9, 4.1])
# y = mx + c = A * x
A = np.vstack([x, np.ones(len(x))]).T
m, c = np.linalg.lstsq(A, y, rcond=-1)[0]
print("(6)")
print("coefficients for y=mx+c are: ", m, c)



import math
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def distance(self):
        return math.sqrt(self.x**2 + self.y**2)
```

```
p = Point(1.0,2.0)
print("(1,2) distance is: ", p.distance())
```

```
import point
import generator
gen = generator.LCG(1,12496158,16166518,2000000)
ls = gen.listRandom(20000000)
length = len(ls)
for i in range(5000000, 12500000):
    ls[i] = -ls[i]
for i in range(15000000, 17500000):
    ls[i] = -ls[i]

gen2 = generator.SCG(2,12496158,161665188,20000000)
ls2 = gen.listRandom(20000000)
length2 = len(ls2)
for i in range(5000000, 12500000):
    ls2[i] = -ls2[i]
for i in range(15000000, 17500000):
    ls2[i] = -ls2[i]
```

```
lsPoint = []
left = 0
right = length - 1
while left < right :
    lsPoint.append(point.Point(ls[left], ls[right]))
    left += 1
    right -= 1

lsPoint2 = []
left2 = 0
right2 = length2 - 1
while left2 < right2 :
    lsPoint.append(point.Point(ls2[left], ls2[right]))
    left2 += 1
    right2 -= 1

count2 = 0
for ele in lsPoint:
    if ele.distance() < 1:
        count2 += 1

count = 0
for ele in lsPoint:
    if ele.distance() < 1:
        count += 1
```

```
print("ratio for LCG: ", 2*count/length)

print("ratio for SCG: ", 2*count2/length2)
```