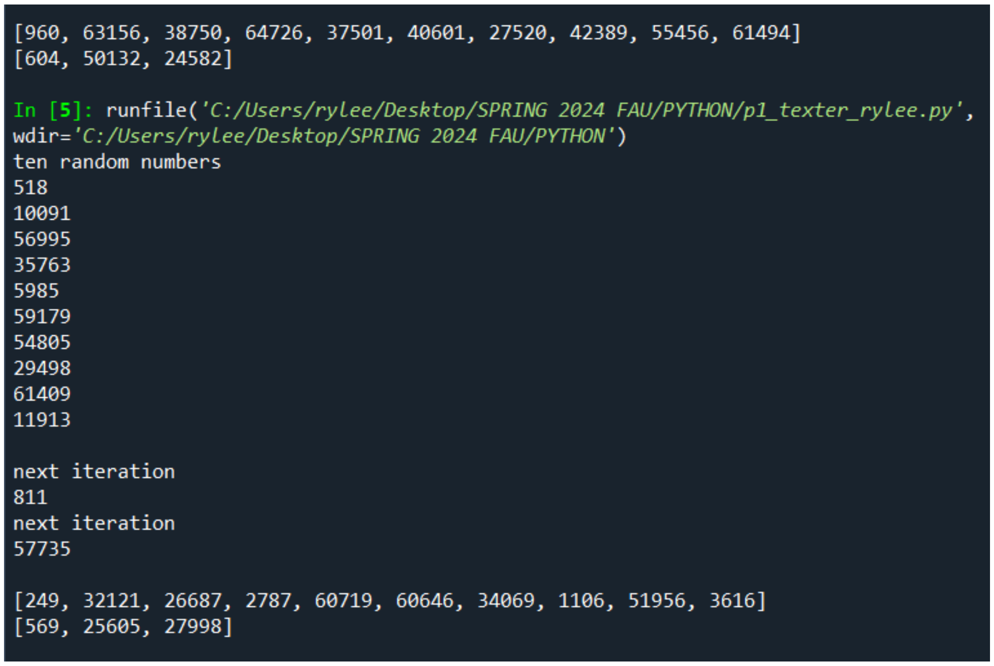
Problem 1:

```
import random

class RndSeq:
    #initiallizes seed, cound and n
    def __init__(self, x0, n):
        self.seed = x0
        self.count = 0
        self.n = n

    ##returns the current iteration
    def __iter__(self):
        return self

    #returns the next iteration and keeps track, raises StopIteration
    #if reached the end
    def __next__(self):
        if self.n >= 0 and self.count >= self.n:
            raise StopIteration

        else:
            self.count += 1
            self.seed = random.randint(0, 1000) * self.seed % 65537
            return self.seed

#generates n  random numbers using seed of x0
def rnd_gen(x0, n):
    count = 0
    #breaks when
    while True:
        if n >= 0 and count >= n:
            break
        else:
            x0 = random.randint(0, 1000) * x0 % 65537
            yield x0
            count += 1

def main():

    rnd = RndSeq(1, 10)

    for num in rnd:
        print(num)
    print()

    rnd2 = RndSeq(1, 2)
    it = iter(rnd2)
    print(next(it))
    print(next(it))
    # print(next(it))

    print([i for i in rnd_gen(1, 10)])
    print(list(rnd_gen(1, 3)))
```

Part 1 Screenshots:

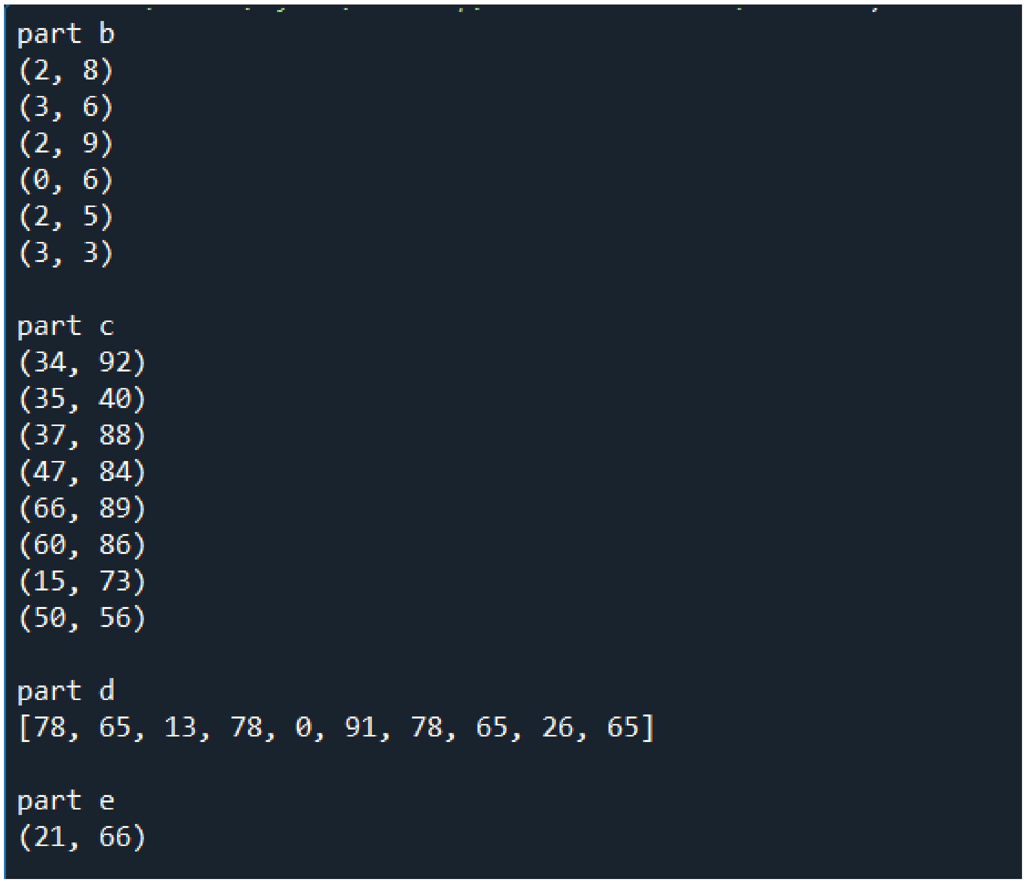


```
[960, 63156, 38750, 64726, 37501, 40601, 27520, 42389, 55456, 61494]
[604, 50132, 24582]

In [5]: runfile('C:/Users/rylee/Desktop/SPRING 2024 FAU/PYTHON/p1_texter_rylee.py',
wdir='C:/Users/rylee/Desktop/SPRING 2024 FAU/PYTHON')
ten random numbers
518
10091
56995
35763
5985
59179
54805
29498
61409
11913

next iteration
811
next iteration
57735

[249, 32121, 26687, 2787, 60719, 60646, 34069, 1106, 51956, 3616]
[569, 25605, 27998]
```

Problem 2:

```
import random
from itertools import filterfalse, islice
from functools import reduce

##code from problem one
def rnd_gen(x0, n):
    count = 0
    while True:
        if n >= 0 and count >= n:
            break
        else:
            x0 = random.randint(0, 1000) * x0 % 65537
            yield x0
            count += 1
#part a
def gen_rndtup(m):
    itera = rnd_gen(1, -1)
    while True:
        a = next(itera) % m
        b = next(itera) % m
        if a <= b:
            yield (a, b)

#part b
generator = gen_rndtup(10)
tups = filterfalse(lambda x: x[0] + x[1] < 6, islice(generator, 8))
print("part b")
for tup in tups:
    print(tup)

#part c
gen_a = (x % 101 for x in rnd_gen(1, -1))
gen_b = (x % 101 for x in rnd_gen(2, -1))

#combines a and b into a tuple
ab = zip(gen_a, gen_b)
tups1 = ((a, b) for a, b in ab if a <= b)

print()
print("part c")
count = 0
for tup in tups1:
    print(tup)
    if count == 7:
        break;
    count +=1


#part d
generator = rnd_gen(1, -1)

numbers = filter(lambda x: x % 13 == 0, map(lambda x: x % 101, generator))
firstten = list(islice(numbers, 10))
print()
print("part d")
print(firstten)


#part e
generator1 = gen_rndtup(10)

tups2=  filter(lambda tup: sum(tup) >= 5, generator1)
tentups = islice(tups2, 10)


total = reduce(lambda x, y: (x[0] + y[0], x[1] + y[1]), tentups)
print()
print("part e")
print(total)
```

Part 2 Screenshots:



```
part b
(2, 8)
(3, 6)
(2, 9)
(0, 6)
(2, 5)
(3, 3)

part c
(34, 92)
(35, 40)
(37, 88)
(47, 84)
(66, 89)
(60, 86)
(15, 73)
(50, 56)

part d
[78, 65, 13, 78, 0, 91, 78, 65, 26, 65]

part e
(21, 66)
```