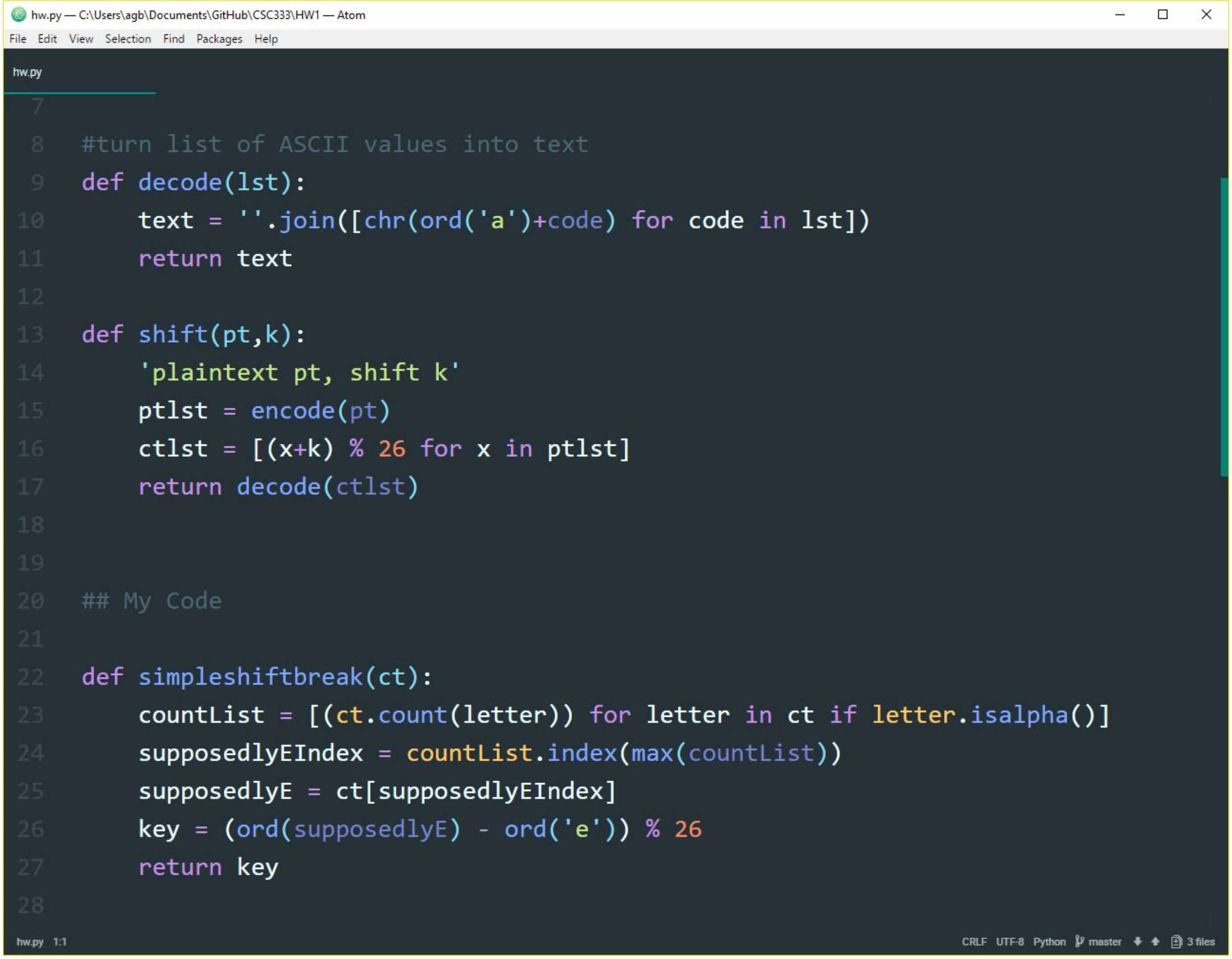
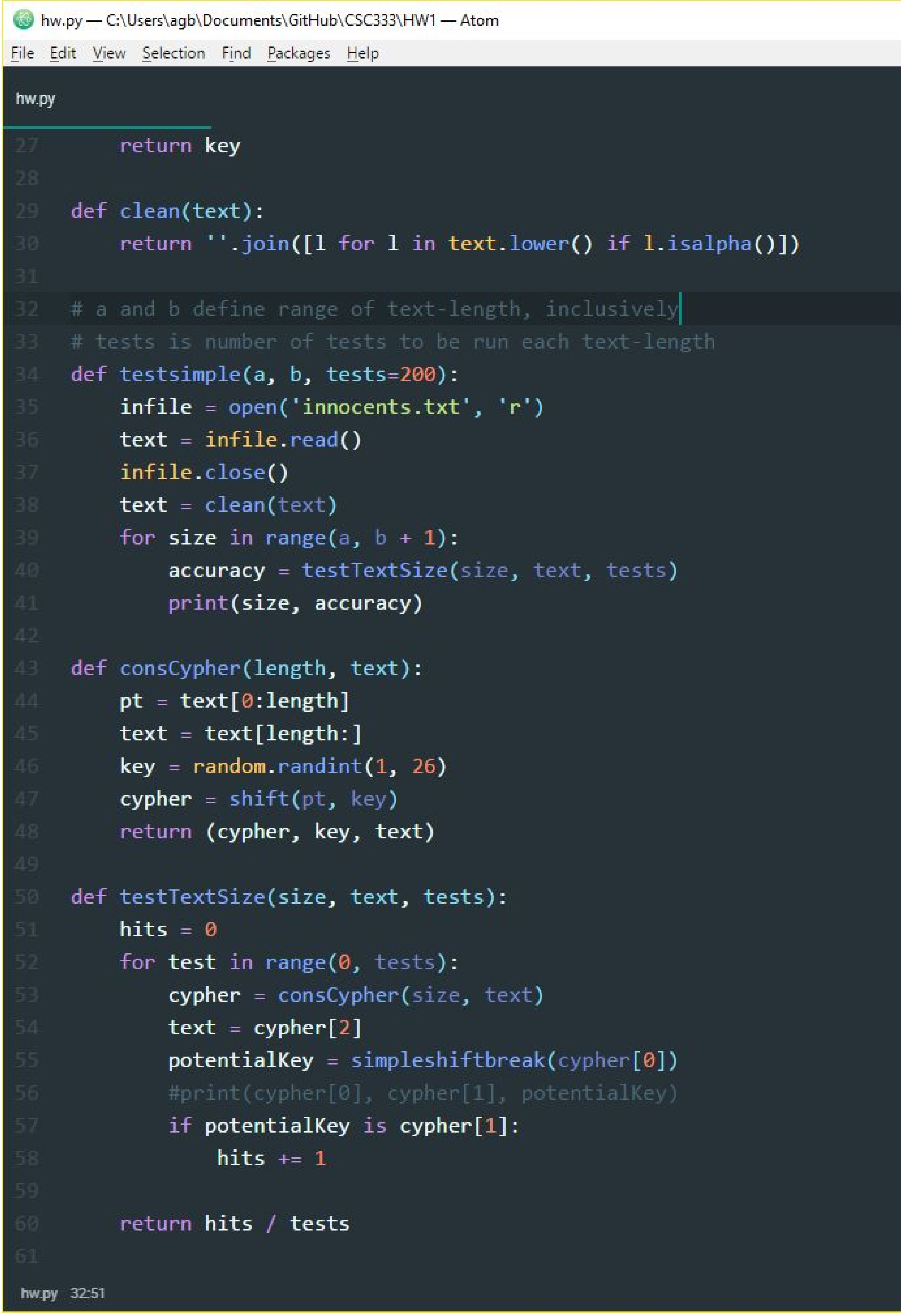
Andrew Burkus

1. Done.

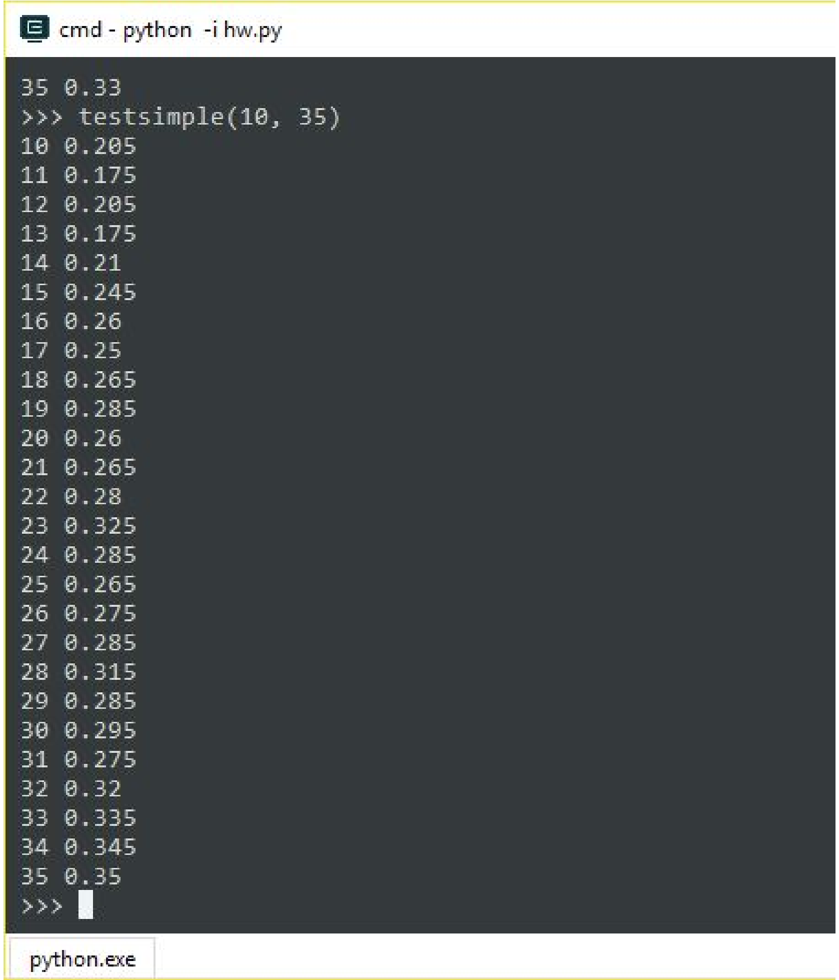
2.


hw.py — C:\Users\agb\Documents\GitHub\CSC333\HW1 — Atom
File Edit View Selection Find Packages Help
hw.py
hw.py 1:1
CRLF UTF-8 Python master 3 files

```python

#turn list of ASCII values into text
def decode(lst):
    text = ''.join([chr(ord('a')+code) for code in lst])
    return text

def shift(pt,k):
    'plaintext pt, shift k'
    ptlst = encode(pt)
    ctlst = [(x+k) % 26 for x in ptlst]
    return decode(ctlst)


## My Code

def simpleshiftbreak(ct):
    countList = [(ct.count(letter)) for letter in ct if letter.isalpha()]
    supposedlyEIndex = countList.index(max(countList))
    supposedlyE = ct[supposedlyEIndex]
    key = (ord(supposedlyE) - ord('e')) % 26
    return key

```


hw.py — C:\Users\agb\Documents\GitHub\CSC333\HW1 — Atom
File Edit View Selection Find Packages Help
hw.py
hw.py 32:51

```
    return key

def clean(text):
    return ''.join([l for l in text.lower() if l.isalpha()])

# a and b define range of text-length, inclusively
# tests is number of tests to be run each text-length
def testsimple(a, b, tests=200):
    infile = open('innocents.txt', 'r')
    text = infile.read()
    infile.close()
    text = clean(text)
    for size in range(a, b + 1):
        accuracy = testTextSize(size, text, tests)
        print(size, accuracy)

def consCypher(length, text):
    pt = text[0:length]
    text = text[length:]
    key = random.randint(1, 26)
    cypher = shift(pt, key)
    return (cypher, key, text)

def testTextSize(size, text, tests):
    hits = 0
    for test in range(0, tests):
        cypher = consCypher(size, text)
        text = cypher[2]
        potentialKey = simpleshiftbreak(cypher[0])
        #print(cypher[0], cypher[1], potentialKey)
        if potentialKey is cypher[1]:
            hits += 1

    return hits / tests
```



```
35 0.33
>>> testsimple(10, 35)
10 0.205
11 0.175
12 0.205
13 0.175
14 0.21
15 0.245
16 0.26
17 0.25
18 0.265
19 0.285
20 0.26
21 0.265
22 0.28
23 0.325
24 0.285
25 0.265
26 0.275
27 0.285
28 0.315
29 0.285
30 0.295
31 0.275
32 0.32
33 0.335
34 0.345
35 0.35
>>> 
```

3.

a. 128^8
b. 7 * 8 = 56 bits
c. It should be the same size, 56 bits, but theoretically we only need 48 bits.
d. 19 characters for 7 bits. 22 characters for 6 bits (26 character ascii).

Problem set 4:

1. (6 * 1000 - 8 * 100) * 46 mod 7
    5200 mod 7 * 46 mod 7
    6 mod 7 * 4
    24 mod 7
    3 mod 7

2. (3^23) mod 26
    (3^11) * (3^12) mod 26
    (3^6) * (3^5) * (3^6) * (3^6) mod 26
    (3^3) * (3^3) * (3^3) * (3^2) * (3^3) * (3^3) * (3^3) * (3^3) mod 26
    1 * 1 * 1 * 9 * 1 * 1 * 1 * 1 mod 26
    9 mod 26

3. 12*11*10*9*8*7*6*5*4*3*2*1 mod 13

    121 * 90 * 56 * 6 * 100 mod 13
    4 * 12 * 4 * 6 * 9 mod 13
    3 * 12 * 6 * 9 mod 13
    5 * 12 * 9 mod 13
    8 * 9 mod 13
    72 mod 13
    7 mod 13

4. What is the last digit of 2587623874*9283472933 ?

    The last digit is 2.
    2587623874*9283472933 mod 10
    This means that multiplication works the same as normal, so the answer is 3 * 4 mod 10.