1. (16 points) \( s = 'a-mimetic-poly-alloy' \). What will be displayed if we type the following into the Python shell?

a) >>> type(s)

b) >>> L = s.split('-')
   >>> len(L[-2])

c) >>> s[1:5:2]*3

d) >>> s[-1:-6:-1]

e) >>> t = ''
   >>> vowels = 'aeiou'
   >>> for ch in s:
      if ch in vowels:
         t += ch
   >>> print(t)

f) >>> L = s.split('-')
   >>> t = ''
   >>> for w in L[::-1]:
      t += w[0]
   >>> print(t)
2. (16 points) Consider the list of tuples, \( L = [(\text{'yay'}, 3), (\text{'pigs'}, 4), (\text{'at'}, 2), (\text{'cheer'}, 5)] \). Starting with this list \( L \) independently for parts (a) - (d), what will the following commands print to the screen?

a) ```python
>>> L.sort()
>>> for p, q in L:
    print(p)
```  

b) ```python
>>> for i in range(len(L)):
    t = L[i]
    L[i] = (t[1], t[0])
>>> L.sort()
>>> L.reverse()
>>> print(L)
```  

c) ```python
>>> x = L.pop()
>>> y = L.pop(0)
>>> L = [x, y, L[1]]
>>> print(L)
```  

d) ```python
>>> s = "
>>> for p, q in L:
    s = '" + p + s
>>> print(s)
```
4. (20 points) Write an encryption function, \texttt{encrypt(p)}, that transforms input plaintext string, \texttt{p}, in the following ways: 1) It reverses the order of the characters in \texttt{p}, and 2) it replaces a character, \texttt{ch}, with the character 1 unit to the right in the \texttt{ord} table. For example, if \texttt{p} = 'this', then \texttt{encrypt(p)} returns 'tjiu' ('this' → reverse order → 'siht' → shift 1 to the right → 'tjiu').

4. (18 points) Write a function, \texttt{createRepeatLettersDict(M)}, that returns a dictionary, \texttt{D}, with keys equal to the lower case letters, and values equal to the corresponding upper case letter repeated \texttt{M} times. For example, if \texttt{M} = 3, then \texttt{D} = {'a': 'AAA', 'b': 'BBB', 'c': 'CCC', ….}.
5. (30 points) Sharks are amazing animals for many reasons. One interesting thing about them is that they continue to grow new teeth throughout their lives, which replace the teeth they lose during feeding.

a) (16 points) Assume a shark is born without any teeth and gains teeth at a rate of exactly 1 new tooth every 4 days, while losing a uniform random fraction between 0.00 to 0.02 of its current number of teeth whenever it feeds (every 3 days we will assume). Write a function `sharkTeeth(N)` that uses random numbers to simulate and return the number of teeth a shark has after $N$ days. *Hint: Think of how to use `random.uniform` and the mod (%) operator.*
b) (14 points) Write a function `monteCarloSharkTeethProb(nTeeth, nDays, nTrials)` that calculates and returns the probability that a `nDays` old shark will have more than `nTeeth` (by performing `nTrials` number of “shark simulations”). Hint: Each shark simulation can be carried out by calling helper function, `sharkTeeth`. 