CPS109 Lab 3

Most of the questions in this lab come from Chapter 3 and 4 (or earlier chapters) of the course text, Introduction to Computing and Programming in Python, by Guzdial and Ericson. Please put your answers (numbered) in a document and submit it on D2L as a PDF file. Other formats are not accepted.

The **learning objectives** for Chapter 3 are to be able to:

- manipulate strings
- build strings with concatenation
- use loops to iterate over the characters in a string
- convert strings into lists for word manipulation
- use array notation s[i] for accessing elements of strings and lists
- use if-statements to branch in the instruction sequence

The learning objectives for Chapter 4 are to be able to:

- understand how images are digitized
- identify different models of color, but mainly RGB, the most common for the computer
- to manipulate color values in pictures, increasing and decreasing them
- to convert color to grayscale
- to negate a picture
- to use matrix representation to find pixels in a picture
- to use picture objects and pixel objects
- to use iteration with a for loop to change color values of pixels in a picture
- to nest blocks of code
- to choose between having a function return a value or just have a side effect
- to determine the scope of a variable name

To do:

1) Recall the keyword cipher programs from Chapter 3: **buildCipher(key)** and **encode(string, alpha2)**, which you modified in lab2, question 10. The buildCipher function could create more complicated alphabets. As long as the receiver and sender build the cipher alphabet (alpha2) in the same way, we only need the keyword (not the special alphabet **alpha2)** and the message to encode and decode. Write **encode2(message, key)** which
   - builds **alpha2** by putting the keyword at the end of the alphabet, rather than the front;
   - skips blanks and punctuation;
   - converts uppercase in the message to lowercase before encoding.

   encode2('Alan Turing defined computing', 'turing') . As an initial check, **encode2('Ada Lovelace, first programmer', 'earth')** should produce: **bfbosegobdgilwxyuwsjwbppgw**
2) Write **encode3(message, key)** following the idea in (1), where we give the keyword and not the new alphabet **alpha2** to **encode3,** with the difference that when you make **alpha2** inside **encode3,** you reverse the normal alphabet before concatenating the letters to the keyword. The keyword this time goes at the front of **alpha2** as it did in lab2**.** As usual, convert the message to lower case before encoding.
   Show your function encode3 and the result of
   encode3('Alan Turing defined computing', 'turing'). As an initial check,

encode3('Ada Lovelace, first programmer', 'earth') should produce:
etesoghserhzwlkjnloyleqqhl

3) One of the following programs that when called like this (with the underscore representing a digit from 1 to 4) generates this output:
>>> dupTimes_('alphabet')
'_alphabetalphabetalphabetalphabetalphabetalphabetalphabetalphabet'

Which one?

```
def dupTimes1(something) :
  dup = ''
  for index in range(0, len(something)) :
    dup = dup + (2 * something[index])
  return dup

def dupTimes2(something) :
  dup = ''
  for index in range(0, len(something)) :
    dup = dup + (index * something[index])
  return dup

def dupTimes3(something) :
  dup = '_'
  for index in range(0, len(something)) :
    dup = dup + something
  return dup

def dupTimes4(something) :
  dup = ''
  for index in range(0, len(something)) :
    dup = dup + something[index]
  dup = “_”
  return dup
```

4) One of the following programs that when called like this (with the underscore representing a digit from 1 to 4) generates this output:
> findem_(4)
'abcdabcdabcdabcdabcdabcdab'

Which one?

```
def findem1(n) :
  letters = 'abcdefghijklmnopqrstuvwxyz'
  pile = ''
  for index in range(0, n) :
    pile = pile + letters[index]
  return pile

def findem2(n) :
  letters = 'abcdefghijklmnopqrstuvwxyz'
  pile = ''
```

```
  for index in range(0, n % len(letters)) :
    pile = pile + letters[index]
  return pile

def findem3(n) :
  letters = 'abcdefghijklmnopqrstuvwxyz'
  pile = ''
  for index in range(1, len(letters)) :
    pile = pile + letters[n % index]
  return pile

def findem4(n) :
  letters = 'abcdefghijklmnopqrstuvwxyz'
  pile = ''
  for index in range(0, len(letters)) :
    pile = pile + letters[index % n]
  return pile
```

5) You have written an essay that should be at least five pages long, but it is only 4.5 pages. You decide to use your new python skills to stretch it out by putting spaces between each letter and the next. Write a function that takes a string and a number and prints out the string with number of spaces between each pair of letters:

```
>>> spacedout("It was a dark and stormy night", 3)
I t    w a s    a    d a r k    a n d    s t o m y    n i g h t
```

6) You are afraid that is too obvious, so you decide to write spacedout2(string, n) which puts n spaces between each pair of words, rather than letters:

```
>>> spacedout2("It was a dark and stormy night", 3)
It  was  a  dark  and  stomy  night
```

7) (a) Why is the maximum value of any component (red, green or blue) 255? (b) How much memory is required to represent the color of a pixel in the RGB model? (c) How many colors can be represented? (d) Is that enough colors and why?

8) Program 34 (copied below) on p. 89 reduces the amount of red in a picture by 50%. Use the JES command **f = pickAFile()** to find the picture **beach.jpg** in the **mediasources-4ed** directory on the O-drive. Make a picture out of that file and explore it using the JES commands

```
> picture = makePicture(f)
> explore(picture)
```

Put in the x, y coordinates: 458, 424. Write into your answer document the R, G, B values for that pixel. Write a version **decreaseRed20(picture)** that reduces the red by 20%, and apply it to beach.jpg. Explore the new picture and write into your answers the new values of R, G, B for (x, y) = (458, 424). Include your program in your answer document. Note that if you want to save a picture, you can write out it out using the JES command **writePictureTo(picture, filename)**. If you do not specify a path, it will use the most recent path JES has used.

```
# Program 34 from p. 89 of the
text: def decreaseRed(picture) :
  for pixel in getPixels(picture) :
    value = getRed(pixel)
    setRed(pixel, 0.5 * value)
```

9) Program 35 for increasing red is very similar, and it is written below as increaseRed1(). Two other ways of writing it are also shown, as versions 2 and 3. Convince yourself either by reading and thinking, or by running and checking values, that all three programs have the same result. When you are programming, you should aim for readability, that is, you want other people to be able to read your code and understand it easily. Which of the three versions do you consider to be most readable and why?

```
def increaseRed1(picture) :
  for p in getPixels(picture) :
    value = getRed(p)
    setRed(p, 1.2 * value)

def increaseRed2(picture) :
  for p in getPixels(picture) :
    setRed(p, 1.2 * getRed(p))

def increaseRed3(picture) :
  for p in getPixels(picture) :
    redC = getRed(p)
    greenC = getGreen(p)
    blueC = getBlue(p)
    newRed = int(1.2 * redC)
    newColor = makeColor(newRed, greenC, blueC)
    setColor(p, newColor)
```

10) Write a function to swap the red and green of each pixel of a picture. Apply the function to the caterpillar.jpg image. Include in your answers your program and the resulting picture.