Parsons Puzzles

These questions are a stepping stone from the earlier tasks on the way to solving programming problems from scratch. **They do not appear in the exam**, but are useful practice.

These puzzles work as follows:

1. You are given a programming problem statement, and the lines of code for a solution - only the line of code are in a muddled-up order. Note the lines of code are numbered.
2. Your task is to reorder the code.
3. You can do this in one of two ways:
    a. Writing down the number of the each line in the correct order. Write these vertically, and put in the correct indentation, as you would need to do in Python if these were full lines.
    b. Writing out the correctly-reordered code in full (this is good practice for writing out code, laying it out correctly)
4. NOTE - you must use all the lines *at least* once, and maybe more times.

---

1. Read in a number and write out whether it is odd or even

```
print("You picked an odd number.")
print("You picked an even number.")
mod = num % 2
else:
num = input("Enter a number: ")
if mod > 0:
```

2. Write a guessing game program. The computer should create a random number between 1 and 9. The user then attempts to guess the number, with the computer giving advice on whether the user should guess higher or lower. The user can quit at any time by typing in *exit* as their guess. The computer writes out how many attempts the user made, if they complete the game.

```
rd = random.randint(1,9)
while guess != rd and guess != "exit":
import random
c = 1
c += 1
if guess == rd:
else:
print("Right!")
print("You took only", c, "tries!")
print("Too high")
if guess > rd:
print("Too low")
guess = int( input("Enter a guess between 1 to 9") )
```

3. Read a file and print out the length of line with a space between each length.

```
1.    lines = f.readlines()
2.    print( length, end = " "  )
3.    for line in lines:
4.    f = open('filename.txt', 'r')
5.    length = len( line )
6.    f.close()
```

4. Given a list of words, `["once","upon","a","time","in","oppositeland"]`, both write the words backwards and order the words in the reverse direction

```
1.    neworder = [new_word] + neworder
2.    words = ["once","upon","a","time","in","oppositeland"]
3.    neworder = []
4.    print(neworder)
5.    for letter in word:
6.    new_word = ""
7.    new_word = letter + new_word
8.    for word in words:
```

5. Given a dictionary named `heights` with string names as keys and heights (in meters) as values: e.g.

Key	Value
"Alice"	1.8
"Bob"	1.9
"Charlie"	1.75
"Dave"	1.4
"Ethel"	1.93

Process the data and print out the name of the tallest person

```
1.    for person in heights:
2.    print( 'the tallest person is ', tallest )
3.    max = 0
4.    tallest  = person
5.    if  height > max:
6.    height = heights[ person ]
7.    tallest = 'No-one'
8.    max = height
```

6. Given a file named `data.csv` with names as strings and scores as numbers in the range 0 to 100 inclusive: e.g.

```
Alice,100
Bob,95
Charlie,10
Dave,42
Ethel,99
```

process the data and print out the name of a person with the lowest score.

```
1.    if score < min:
2.    file = open( 'data.csv' )
3.    name = ( line.split( ',' ) )[ 0 ]
4.    name = 'No-one'
5.    min = 100
6.    print( 'person with min score is' + minName )
7.    file.close()
8.    minName = name
9.    score = int( ( line.split( ' ,' ) )[ 1 ] )
10.   for line in file:
11.   min = score
```

7. A computer system has output some data on people's names and ages into a file `data.txt` as follows. Each person's data is written onto two lines, the first line with their name and the second line with their age. There is a blank line after the end of each person's data, including at the end of the file after the last age. Here is a sample of three people's data

```
Bob
95

Charlie
10

Dave
42
```

Write a program to read the data into a list of dictionaries, each dictionary holding a name and an age for a single person.

```
1.    newAge  = int( newAgeAsString )
2.    newPerson = { "name" : newName, "age" : newAge }
3.    lines = f.readlines()
4.    people = people + [ newPerson ]
5.    f = open( "data.txt" )
```

```
6.    newAgeAsString = lines[ lineNum + 1 ][ :-1 ]
7.    for lineNum in range( 0, len( lines ), 3 ):
8.    people = []
9.    newName = lines[ lineNum ][ :-1 ]
```

8. Parson's Problem Generator: Write a program which takes another program in a text file, shuffles the line order, trims the leading whitespace and adds a line number to each reordered line. ***NB: the `shuffle()` function takes a list and randomly rearranges the ordering of the elements. This applies directly to the list supplied, so does not need to be assigned to a variable.***

```
1.    f = open( "program.txt" )
2.    for line in lines:
3.    print(line
4.    current_line = 1
5.    line = str(current_line) + ". " + line[:-1]
6.    line = line[1:]
7.    current_line += 1
8.    shuffle(lines)
9.    while line[0] == " ":
10.   lines = f.readlines()
11.   from random import shuffle
```

9. Caesar Shift Cipher: Write a program that replaces each letter in a string with the next in the alphabet, ignoring whitespace

```
1.    withoutblank += letter
2.    if letter != " ":
3.    decoded += alphabets[(i + 1)% len(alphabets)]
4.    import string
5.    i = alphabets.index(letter)
6.    print(decoded)
7.    for letter in withoutblank:
8.    alphabets=list(string.ascii_lowercase)
9.    plaintext = "duel by dawn"
10.   decoded = ""
11.   withoutblank = ""
12.   for letter in plaintext:
```

10. (Challenge!) Matrix multiplication: Given two 2D lists (forming matrices), define a function that computes the matrix multiplication

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

$$\begin{aligned} c_{11} &= a_{11}b_{11} + a_{12}b_{21} \\ c_{12} &= a_{11}b_{12} + a_{12}b_{22} \\ c_{21} &= a_{21}b_{11} + a_{22}b_{21} \\ c_{22} &= a_{21}b_{12} + a_{22}b_{22}. \end{aligned}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 4 \\ 10 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 7 & 10 \end{bmatrix}$$

```
for col in range(0,Arows):
Bcols = len(B[0]
if Acols != Brows:
for k in range(Acols):
def matrixmultiplication (A, B):
for row in range(0,Bcols):
print("Not applicable")
newrow = []
for i in range(Arows):
C = []
Arows = len(A)
return C
Acols = len(A[0])
C[i][j] += A[i][k] * B[k][j]
Brows = len(B)
Return
C += [newrow]
for j in range(Bcols):
newrow += [0]
```