Problemas de Python

INSTRUCCIONES:

A continuación se presentan 10 problemas a resolver utilizando el lenguage de programación Python. Utilizar la versión 2.5. Estos problemas se deben poder resolver leyendo los 6 primeros capítulos del libro *Python para Todos,* es decir, hasta la página 41. Como complemento deberá seguir los ejemplos e indicaciones adicionales incluidas en esta guía.

FUNCIONES AUXILIARES:

Las siguientes funciones en Python podrán ayudarlo a resolver los problemas propuestos.

Leer el contenido de un archivo de texto línea por línea y colocarlo en una lista

```
def file_to_list(filepath, ignore_blanks=False):
    result = file(filepath).read()
    result = result.split('\n')
    if ignore_blanks:
        aux = result
        result = []
        for element in aux:
            if len(element)>0:
                result.append(element)
    return result
```

Limpiar la pantalla en Linux

```
def clear_screen():
    import os, sys
    sys.stdout.write(os.popen('clear').read())
```

Esperar una cierta cantidad de segundos

```
def wait(seconds):
    import time
    time.sleep(seconds)
```

Estas funciones están en el archivo `utils.py` y pueden ser utilizadas en un programa Python que se encuentra en la misma carpeta utilizando la siguiente sintaxis:

```
from utils import *
```

Así por ejemplo, un programa que limpie la pantalla e imprima los números del 1 al 10 esperando un segundo antes de imprimir el siguiente podría escribirse de la siguiente manera:

```
#!/usr/bin/env python

from utils import *

for n in range(1, 10+1):
    clear_screen()
    print n
    wait(1)
```

Así mismo, utilizar las siguientes funciones y operadores para resolver los problemas:

Agregar un elemento a una lista

```
>>> lista = []
>>> lista.append(1)
>>> lista
[1]
>>> lista.append(2)
>>> lista
[1, 2]
```

Consultar si un elemento es parte de una lista

```
>>> vocales = ['a','e','i','o','u']
>>> 'o' in vocales
True
>>> 'p' in vocales
False
```

Consultar si una llave existe en un diccionario

```
>>> d = {'a' : 1, 'b': 2}
>>> 'a' in d
True
>>> 'x' in d
False
```

Imprimir un caracter sin salto de linea ni espacio

```
>>> from sys import stdout
>>> stdout.write('a')
a>>> _
```

Obtener un número entero aleatorio dentro de un cierto rango

```
>>> from random import randint
>>> randint(1, 100)
74
```

Imprimir un número entero con ceros a la izquierda y una longitud fija

```
>>> print "%03d" % 1
001
>>> print "%03d" % 10
010
>>> print "%03d" % 100
100
```

Imprimir un número real a cierto número de posiciones decimales

```
>>> simbolo_moneda = 'S/.'
>>> precio = 18.4123
>>> print "Monto: %s %.2f" % (simbolo_moneda, precio)
Monto: S/. 18.41
```

NOTA: Los comodines como `%d` y `%s` se pueden usar sin `print` directamente en cadenas, así que `"%s:%d" % ('a',1)` se convierte en `"a:1"`.

PROBLEMA 01:

Almacenar en una lista 100 valores entre 1 y 10000 para luego recorrerla y encontrar el número mayor y el número menor.

PROBLEMA 02:

Hallar el valor de la suma de todos los números enteros menores que mil que sean divisibles simultáneamente por 3 y por 5.

PROBLEMA 03:

Implemente la función mcd utilizando el Algoritmo de Euclides tal como se explica en el siguiente artículo:

http://es.wikipedia.org/wiki/Algoritmo_de_Euclides

PROBLEMA 04:

Imprimir 1000 placas distintas de vehículos de la forma ABC-123 generadas aleatoriamente. Tener en consideración que las primeras tres letras solo pueden ser aquellas que figuran en cada linea del archivo prefijos_placas.txt incluido con esta guía.

PROBLEMA 06:

Leer el texto del archivo rfc2616.txt letra por letra separándolo en palabras y contando las veces que aparece cada palabra. Al final deberá imprimir la lista de palabras que aparezcan al menos 100 veces en el orden en que se fueron encontrando y entre paréntesis su frecuencia de aparición. Considere que una palabra solo puede estar formada por los caracteres del archivo alfabeto.txt y que cualquier otro carácter es un separador de palabra.

PROBLEMA 07:

Implementar un juego en el que la computadora genera aleatoriamente un número entre 1 y 100 inclusive. Se le ofrece al jugador 10 oportunidades para adivinar el número. En cada intento fallido se le indica si el número es mayor o menor que el último número ingresado. Imprimir el puntaje obtenido si se parte con 1000 puntos y se descuentan 50 puntos en las primeras 3 oportunidades falladas, 100 puntos en las siguientes 3 y 150 en las siguientes 3 y 100 en la última.

PROBLEMA 08:

Escribir un programa que cada dos segundos limpie la pantalla e imprima 8 columnas verticales de caracteres obtenidos aleatoriamente del archivo matrix.txt separadas por dos espacios en blanco.

PROBLEMA 09:

Escribir un programa que lea números de tarjetas de crédito del archivo tarjetas.txt, indique si son válidas y las identifique según el operador de acuerdo la información descrita en:

http://www.beachnet.com/~hstiles/cardtype.html

PROBLEMA 10:

Escribir un programa tipo “hangman” que escoja aleatoriamente una frase del archivo hangman.txt ofreciendo 8 intentos antes de que el hombre sea ahorcado. Se debe limpiar la pantalla en cada intento y reemplazar las letras aún no reveladas por guiones.

Consideraciones adicionales:

Se pueden concatenar dos cadenas con el operador `+` o con el método `join` colocándolas en una lista:

```
>>> cadena1 = 'Hola'
>>> cadena2 = 'Mundo'
>>> cadena1 + ' ' + cadena2
'Hola Mundo'
>>> ' '.join([cadena1, cadena2])
'Hola Mundo'
```

Se puede convertir un número entero a cadena y viceversa con las funciones `str()` e `int()`:

```
>>> str(9)
'9'
>>> int('5')
5
>>> '9' + str(9)
'99'
>>> 5 + int('5')
10
```

Se puede convertir una cadena a mayúsculas o minúsculas con los métodos `upper()` y `lower()`:

```
>>> 'hola'.upper()
'HOLA'
>>> 'ADIOS'.lower()
'adios'
```

Se pueden generar listas de números dentro de ciertos rangos con la función `range()`

```
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(1,10+1)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> range(10,100,10)
[10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> range(10,100 + 10,10)

[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
>>> range(10,0,-1)
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

Se puede recorrer una cadena letra por letra utilizando la estructura `for ... in`:

```
>>> for c in 'Hola':
...     print c
...
H
o
l
a
```