

## Taller de Programación

### Certamen 2

8 de junio de 2019

#### Instrucciones:

- El certamen contiene 4 problemas. Lea atentamente el enunciado de cada uno de ellos.
- El problema **1** es obligatorio, seleccione **dos** problemas de los restantes.
- Para cada problema cree un archivo.py distinto. El nombre del archivo debe ser el número del problema. Por ejemplo: uno.py, dos.py, tres.py o cuatro.py
- Suba sus respuestas como un archivo **ZIP** a la sección Evaluación en <http://canvas.udd.cl>. Solo tendrá una oportunidad para subir sus respuestas.
- Recuerde que usaremos un software de detección de plagio, confiamos en su honestidad.
- Tiempo total: **2 horas y 50 minutos**.
- Recuerde que debe seleccionar dos problemas de los enunciados 2, 3 y 4.

### (3 pts) 1. Libros y Estanterías

Cree la clase `Libro`, la cual almacena el título, autor y año de publicación de un libro, y la clase `Estanteria`, la que debe almacenar una lista de variables tipo `Libro`.

Las clase `Libro` y `Estanteria` deben contener un inicializador, más cinco y tres métodos adicionales, tal como se describe a continuación:

API Libro (1.2 pto):

class	Libro	
	<code>Libro(titulo, autor, anio)</code>	Inicializador o constructor (0.4pts)
str	<code>get_titulo()</code>	Retornar título (0.1pts)
str	<code>get_autor()</code>	Retornar autor (0.1pts)
int	<code>get_anio()</code>	Retornar año (0.1pts)
bool	<code>es_mas_antiguo(b)</code>	Retorna true si el libro actual es más antiguo que b (0.3pts)
str	<code>__str__()</code>	Retorna el string "[titulo], [autor], [año]" (0.2 pts)

API Estantería (0.8 pto):

class	Estanteria	
	<code>Estanteria()</code>	Inicializador o constructor (0.2 pts)
	<code>agregar()</code>	Agrega título a la estantería (0.2 pts)
bool	<code>contiene(titulo)</code>	Retorna True si un libro titulado <code>titulo</code> está en la estantería y False si el libro no se encuentra (0.2 pts)
str	<code>__str__()</code>	Retorna un string, donde cada línea corresponde a un libro (0.2pts)

Para construir las clases **Libro** y **Estanteria** utilice el siguiente esqueleto de código con su respuesta, reemplazando la palabra `pass` por su código.

**Solución posible:**

```
class Libro:
    def __init__(self, titulo, autor, anio):
        self.t = titulo
        self.a = autor
        self.y = anio
```

```

def __str__(self):
    return '{}, {}, {}'.format(self.t, self.a, self.y)
def get_titulo(self):
    return self.t
def get_autor(self):
    return self.a
def get_anio(self):
    return self.y
def es_mas_antiguo(self, b):
    if b.y > self.y: return True
    else: return False

class Estanteria:
    def __init__(self):
        self.l = []
    def agregar(self, b):
        self.l.append(b)
    def __str__(self):
        ans = []
        for b in self.l:
            ans.append(str(b))
        return '\n'.join(ans)
    def contiene(self, titulo):
        for b in self.l:
            if b.get_titulo() == titulo:
                return True
        return False

if __name__ == '__main__':
    libro1 = Libro('Fahrenheit 451', 'Ray Bradbury', 1953)
    libro2 = Libro('Daemon', 'Daniel Suarez', 2006)
    libro3 = Libro('La casa de los espíritus', 'Isabel Allende', 1982)

    # Estas líneas evalúan la clase Libro
    if libro1.es_mas_antiguo(libro2) != True: print('Hay un error en tu código!')
    if str(libro1) != 'Fahrenheit 451, Ray Bradbury, 1953': print('Hay un error en tu código!')

    librero = Estanteria()
    librero.agregar(libro1)
    librero.agregar(libro2)
    librero.agregar(libro3)

    # Estas líneas evalúan la clase Estanteria
    if librero.contiene('La casa de los espíritus') != True: print('Hay un error en tu código!')

```

```
print(librero)
```

El puntaje de cada función corresponde al indicado en la tabla. Algunas preguntas pueden tener puntaje parcial:

- `Libro.es_mas_antiguo(b)`, si no retorna True o False, 0.2pts
- `Libro.__str__()`, si no retorna el string en el formato indicado, 0.1pts
- `Estanteria.__str__()` si no retorna el string en formato indicado, 0.1pts
- `Estanteria.contiene(titulo)`, si no retorna True o False, 0.1pts

## (2 pts) 2. Generando Palíndromos

Programa una función que imprima un número natural palíndromo. Un número natural palíndromo es un número natural que se lee igual tanto de derecha a izquierda, como de izquierda a derecha. La función debe recibir un número entero  $X$  e imprimir:  $X \ X - 1 \ X - 2 \ \dots \ 3 \ 2 \ 1 \ 1 \ 2 \ 3 \ \dots \ X - 2 \ X - 1 \ X$

Por ejemplo, si se le entrega el número 4 a la función, esta imprime 4 3 2 1 1 2 3 4.

Las funciones deben ser implementadas del siguiente modo:

1. (1 pts) Utilizando un ciclo **for** o **while**.
2. (1 pts) Utilizando recursividad, **sin utilizar** un ciclo **for** o ciclo **while**. Recuerde indicar claramente el **caso base** y el **caso recursivo**.

**Nota:** para imprimir un espacio en vez de un salto de línea, agregue el parámetro a la función `print`, por ejemplo, `print('mensaje', end = ' ')`

### Solución posible ciclo for o while:

```
def palindromo(n):  
    for i in range(n):  
        print(n-i, end= ' ')  
    for i in range(1, n+1):  
        print(i, end = ' ')
```

- 0.5 pts generación de secuencia descendente  $X \ X-1 \dots 1$
- 0.5 pts generación de secuencia ascendente  $1 \ 2 \dots X-1 \ X$
- -0.2 pts si no imprime adecuadamente los espacios

### Solución posible recursiva:

```
def palindromo_r(n):  
    print(n, end = ' ')  
    if n > 1:  
        palindromo_r(n-1)  
    print(n, end = ' ')
```

- Define caso base: 0.5 pts
- Define caso recursivo: 0.5pts

## (2 pts) 3. Encriptación

Programa un código que encripte una frase ingresada por teclado e imprima la versión encriptada. Para esto cree la función `encriptar`, la que recibe un string para encriptar y retorna un string con la palabra encriptada.

El proceso de encriptación que debe implementar se describe a continuación:

1. Existe la palabra mágica “murcielago” de tipo tupla, que actúa como clave de encriptación, `clave=('m','u','r','c','i','e','l','a','g','o')`.
2. La palabra a encriptar se compara, carácter por carácter con la palabra mágica, y dependiendo del resultado de la comparación, ciertos caracteres de la palabra a encriptar son reemplazados. Para esto existen dos casos:
  - Si el carácter de la palabra a encriptar existe en la palabra mágica, este se reemplazará por la posición en la palabra mágica donde el carácter se encuentra.
  - Si el carácter buscado no existe en la palabra mágica, el carácter de la palabra a encriptar no se cambiará.
3. El resultado final de los reemplazos de caracteres, dan origen a la palabra encriptada.

**Ejemplo:** La función `encriptar('mundo')` retorna `01nd9`. Esto es porque `m` está en la posición `0` de la palabra mágica, `u` está en la posición `1`, `n` y `d` no se encuentran, y `o` está en la posición `9`.

### Notas:

- Asuma que sólo recibirá palabras en minúsculas.
- La función debe retornar la palabra encriptada.
- Dada una tupla `t`, la función `t.index(elem)` retorna la posición donde aparece el elemento `elem` dentro de la tupla `t`.

### Solución posible:

```
clave = tuple('murcielago')
def encriptar(palabra):
    salida = []
    for c in palabra:
        if c not in clave:
            salida.append(c)
        else:
            salida.append(str(clave.index(c)))
    return ''.join(salida)

if encriptar('mundo') != '01nd9': print('Hay un error en tu código :(')
entrada = input('Ingrese palabra a encriptar: ')
salida = encriptar(entrada)
print('Palabra encriptada:', salida)
```

- 0.5pts recorrer la palabra, letra por letra
- 0.5pts por caso de letra no está en clave
- 0.5pts por case de letra está en clave
- 0.5pts por retornar clave correctamente

## (2 pts) 4. Entropía

En Teoría de Información, la Entropía representa la cantidad de información que posee un texto. La Entropía de un texto  $S$  está definida como:

$$H(S) = \sum_{c \in P} \frac{n_c}{n} \times \log_2 \frac{n}{n_c},$$

dónde  $c$  es una palabra del texto  $S$ ,  $P$  es el conjunto de palabras en el texto  $S$ ,  $n_c$  es la cantidad de veces que aparece la palabra  $c$  en  $S$ , y  $n$  es el número de palabras en  $S$ .

Para el texto  $S$  'mi mama me mima mi mama', el conjunto  $P$  es {mi, mama, me, mima}, por lo tanto,  $n = 4$ . El conteo de palabras es el siguiente:

- $n_{mi} = 2$
- $n_{mama} = 2$
- $n_{me} = 1$
- $n_{mima} = 1$

Luego, la entropía del texto  $S$  es la siguiente sumatoria:

$$2/6 \times \log_2(6/2) + 2/6 \times \log_2(6/2) + 1/6 \times \log_2(6/1) + 1/6 \times \log_2(6/1) = 1.9182958340544891.$$

### Notas:

- Asuma que el texto es una lista de palabras.
- Asuma que el texto ingresado está en minúsculas y sin tildes, y solo son letras.

### Solución posible:

```
from math import log2

def entropia(S):
    d = {}
    for c in S:
        if c in d: d[c] += 1
        else: d[c] = 1
    n = len(S)

    h = 0.0
    for k,v in d.items():
        h += (v/n)*log2(n/v)
    return h

texto = 'mi mama me mima mi mama'
S = texto.split()
H = entropia(S)
print(H)
```



- 1 pto por conteo de palabras
- 0.2pts por calcular correctamente el valor de  $n$
- 0.8pts por calcular correctamente la sumatoria