Universidad del Desarrollo
Facultad de Ingeniería

Taller de Programación

Certamen Recuperativo

17 de Julio del 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 enunciados 2, 3 y 4.
- 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**.

1. Admisión Udd (obligatorio, 2 pts.)

Cree la clase `Postulante`, la cual almacena los nombres, los apellidos, el rut, el puntaje PSU de matemáticas y el puntaje PSU de lenguaje, y la clase `AdmitidosUdd`, la que debe almacenar una lista de variables tipo `Postulante` con los estudiantes admisibles en la Universidad del Desarrollo (UDD). Un postulante se considera admisible cuando su puntaje PSU entre matemáticas y lenguaje es superior a 600 puntos.

Las clases `Postulante` y `AdmitidosUdd` deben contener un inicializador, más cinco y tres métodos adicionales respectivamente, tal como se describe a continuación:

API Postulante (1.2 pts):

class	Postulante	
	`Postulante(nom,apell,rut,ptj_mat,ptj_leng)`	Inicializador o constructor (0.4 pts.)
str	`ptj_mat()`	Retorna puntaje PSU de matemáticas (0.1 pts.)
str	`ptj_leng()`	Retorna puntaje PSU de lenguaje (0.1 pts.)
int	`promedio_psu()`	Retornar puntaje promedio entre matematicas y lenguaje (0.2 pts.)
bool	`es_admisible()`	Retorna `True` si el promedio PSU es mayor a 600 puntos y `False` de otro modo (0.2 pts.)
str	`__str__()`	Retorna el string "nombres apellidos, promedio_psu" (0.2 pts.)

API AdmitidosUdd (0.8 pts):

class	AdmitidosUdd	
	`AdmitidosUdd()`	Inicializador o constructor (0.2 pts)
	`admitir(b)`	Agrega al postulante b a la lista de AdmitidosUdd si es que tiene un puntaje promedio PSU igual o mayor a 600 puntos (0.2 pts)
bool	`check_admitidos(rut)`	Retorna `True` si el rut de un `Postulante` está en la lista de `AdmitidosUdd` y `False` si no lo está (0.2 pts)
str	`__str__()`	Retorna un string, donde cada línea corresponde a un postulante admitido en la Udd (0.2 pts.)

Para construir las clases **Postulante** y **AdmitidosUdd** utilice como base el siguiente código. Remplace la palabra `pass` por su solución.

```
class Postulante:
    def __init__(self, nombres, appelidos, rut, ptj_mat, ptj_leng):
        pass
    def get_id(rut):
        pass
    def ptj_mat(self):
        pass
    def ptj_leng(self):
        pass
    def promedio_psu(self):
        pass
    def es_admisible(self, id):
        pass
    def __str__(self):
        pass

class AdmitidosUdd:
    def __init__(self):
        pass
    def admitir(self, b):
        pass
    def check_admitido(self, id):
        pass
    def __str__(self):
```

```
        pass

if __name__ == '__main__':
    postulante1 = Postulante('Albert Hans', 'Einstein Koch', 12345, 750, 790)
    postulante2 = Postulante('Maria Salomea', 'Skłodowska Curie', 12346, 800, 800)
    postulante3 = Postulante('Pedro Daniel', 'Gaete Valenzuela', 13456, 300, 320)

    # Estas líneas evalúan la clase Postulante
    if str(postulante2) != 'Maria Salomea Skłodowska Curie, 800':
        print('Hay un error en tu código!')
    if postulante1.es_admisible() != True:
        print('Hay un error en tu código!')
    if postulante3.es_admisible() == True:
        print('Hay un error en tu código!')

    admitidosudd = AdmitidosUdd()
    admitidosudd.admitir(postulante1)
    admitidosudd.admitir(postulante2)
    admitidosudd.admitir(postulante3)

    # Estas líneas evalúan la clase AdmitidosUdd
    if admitidosudd.check_admitido(13456) != True:
        print('Hay un error en tu código!')
    print(admitidosudd)
```

2. Frases Palíndromos (2 pts.)

Programe la función `check_palindromo` la que retorna `True` si una frase es palíndromo y `False` si no lo es. Una frase es palíndromo si se lee igual tanto de derecha a izquierda, como de izquierda a derecha, sin considerar los espacios en blanco. Ejemplo `luz azul`.

La funcion deben ser implementada de dos formas distintas:

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

Nota:

- La función s.replace(' ', '') remueve los espacios en blanco del string s.

3. Desencriptación (2 pts.)

Programe una código que reciba una palabra encriptada ingresada por teclado e imprima la versión desencriptada. Para esto cree la función `desencriptar`, la que **recibe** un string encriptado y **retorna** un string desencriptado.

El proceso de desencriptación se describe a continuación:

1. Existe la palabra mágica “murcielago” de tipo tupla, que actua como clave de desencriptación, `clave=('m','u','r','c','i','e','l','a','g','o')`.
2. La palabra a desencriptar se recorre caracter por caracter, y si el caracter es un dígito, este se reemplaza por la letra de la clave en la posición que indica el dígito. Pero si el caracter no es un dígito, este se mantiene.
3. El resultado final de los reemplazos de caracteres, dan origen a la palabra desencriptada.

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

Utilice el siguiente código como base de su solución:

```
clave = tuple('murcielago')

def desencriptar(palabra):
    # tu codigo va aqui
    pass

if desencriptar('01nd9') != 'mundo':
    print('Hay un error en tu código :(')

entrada = input('Ingrese palabra encriptada: ')
salida = desencriptar(entrada)
print('Palabra desencriptada:', salida)
```

Notas:

- Asuma que sólo recibirá palabras en minúsculas.
- Dada una tupla `t`, la función `t.index(elem)` retorna la posición donde aparece el elemento `elem` dentro de la tupla `t`.
- La función s.isdigit() retorna `True` si el string s es un dígito y `False` si no lo es.

4. Notas Finales Taller de Programación (2 pts.)

En el curso programación queremos conocer los resultados finales de los alumnos de forma rápida. En particular, nos interesa saber los nombres de todos los alumnos que obtuvieron una nota entre ciertos valores y la frecuencia de cada nota en el curso.

Utilizando los datos del archivo de texto `notas_prograudd.txt` y la función `get_dicc()`, la cual retorna un diccionario a partir del archivo de texto:

1. Programe la función `rango_alumnos(minimo, maximo, D)`, la cual recibe un valor mínimo, un valor máximo y un diccionario D y retorna una lista de strings con los nombres de todos los alumnos que tuvieron nota final en el curso entre los valores mínimo y máximo, segun los datos del diccionario D. (1.0 pt.)

2. Programe la función `frecuencia(nota, D)`, la cual debe recibir una nota y un diccionario D, y entregar la frecuencia de la nota final en el curso. (1.0 pt.)

Utilice el siguiente código como base de su solución:

```
def get_dicc(archivo):
    D = dict()
    f = open(archivo)
    for linea in f:
        valores = linea.split()
        D[valores[0]] = valores[1]
    f.close()
    return D

 def rango_alumnos(minimo, maximo, D):
    pass

 def frecuencia(nota, D):
    pass
```