ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN
CCPG1001 - FUNDAMENTOS DE PROGRAMACIÓN
TERCERA EVALUACIÓN - I TÉRMINO 2018-2019/ Septiembre 14, 2018

Nombre: _____ Matrícula: _____ Paralelo:____

COMPROMISO DE HONOR: Al firmar este compromiso, reconozco que el presente examen está diseñado para ser resuelto de manera individual, que puedo usar un lápiz o esferográfico; que sólo puedo comunicarme con la persona responsable de la recepción del examen; y, cualquier instrumento de comunicación que hubiere traído, debo apagarlo y depositarlo en la parte anterior del aula, junto con algún otro material que se encuentre acompañándolo. Además no debo usar calculadora alguna, consultar libros, notas, ni apuntes adicionales a los que se entreguen en esta evaluación. Los temas debo desarrollarlos de manera ordenada.
Firmo el presente compromiso, como constancia de haber leído y aceptado la declaración anterior. "Como estudiante de ESPOL me comprometo a combatir la mediocridad y actuar con honestidad, por eso no copio ni dejo copiar".

_____
Firma

TEMA 1 (40 PUNTOS)

Dado el archivo rutasManejadas2018.txt con información como la que sigue:

```
id_ruta, id_chofer, fecha
Guayaquil-Cuenca,SMSNADOPN,17-05-2018
Guayaquil-Cuenca,AGBCCAPMP,18-05-2018
Guayaquil-Daule,EVNTAASFL,17-05-2018
Guayaquil-Daule,AAQSPTTGL,18-05-2018
```

Suponga que dispone de una función **calcularFecha(fecha, n)** que recibe una fecha y un entero. La función retorna la fecha correspondiente a los n días anteriores a la fecha del parámetro (sin incluirla).

Implemente lo siguiente:

1. **(12 puntos)** La función **cargarDatos(nomA)** que recibe el nombre del archivo con los datos anteriores**.** Esta función **retorna una tupla** de dos elementos. El primer elemento es un conjunto con los ids de **TODOS** los choferes mencionados en el archivo. El segundo elemento es un diccionario con la siguiente estructura: {**fecha**: {**id_ruta**:{ch1,ch2,..., chk}}}. Ejemplo del diccionario:

```
{"17-05-2018":{"Guayaquil-Cuenca":{"SMSNADOPN","AGBCCAPMP",...},
               "Guayaquil-Daule" :{"...","...",...},
                ...},
 "18-05-2018": {"Guayaquil-Cuenca":{"EVNTAASFL","AAQSPTTGL",...},
                "Guayaquil-Daule":{"...","...",...},
                ...},
  ...
}
```

2. **(16 puntos)** La función **encontrarChoferes(dicc, fecha, losChoferes, id_ruta, n)** que recibe el diccionario del numeral anterior, una fecha (con formato dd-mm-yyyy), el conjunto con los ids de TODOS los choferes, el nombre de una ruta y un entero n. Esta función retorna un conjunto con los **ids** de todos los choferes que **NO** hayan manejado la ruta **id_ruta** en los **n** días anteriores a **fecha** (sin incluir fecha). Por ejemplo, si **n** es 3 y la fecha es "02-05-2018", la función devuelve un conjunto con los ids de choferes que NO hayan manejado id_ruta el 29, 30 de abril y el 1 de mayo de 2018.

3. **(12 puntos)** La función **grabarArchivo(fecha, diccionario, losChoferes, n)** que recibe una fecha, el diccionario del numeral 1, un conjunto con los **IDs** de todos los choferes y un número entero **n**. Esta función crea un archivo, cuyo nombre tiene el formato **idRuta_fecha.txt,** para cada **id_ruta** de **fecha** con los choferes que NO han manejado la ruta **id_ruta** en los **n** días anteriores a la **fecha** (sin incluir fecha). El formato para estos archivos es el siguiente:

```
Para la ruta Guayaquil-Cuenca, los choferes disponibles para la fecha 19-05-2018 que no
hayan manejado 2 dias anteriores son:
VSSUIMCMS
SJMPYSANL
```

...

TEMA 2 (50 PUNTOS)

En el último discurso presidencial, se mencionaron algunos datos sobre las ganancias de *algunos* minerales del país que lo dejaron sorprendido. Asuma que tiene todo el texto del discurso en una variable String llamada **discurso**. En este texto, todas las palabras están en minúsculas y separadas por un espacio. El discurso no contiene signos de puntuación ni otros símbolos. Los minerales están identificados por el prefijo "mral_" seguido del nombre del mineral. Ejemplo: `'mral_oro', 'mral_plata', 'mral_cobalto'`

```
discurso = '... y el día de ayer descubrimos en la mina mirador que la cantidad de mral_oro...'
```

Suponga que posee las matrices de Numpy **P**, **C** y **T**, donde **P** es la matriz de producción de mina, en gramos:

	MIRADOR	FRUTA_DEL_NORTE	LOMA_LARGA	RIO_BLANCO	...
Oro	13524000	121072000	1204000	9632000	...
Plata	28000000	952000	9632000	96404000	...
Cobre	126980000	896000	92988000	9604000	...
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...

C es la matriz de costos totales de extracción:

	MIRADOR	FRUTA_DEL_NORTE	LOMA_LARGA	RIO_BLANCO	...
Oro	12.32	10.23	23.23	19.23	...
Plata	3.13	1.78	2.45	1.69	...
Cobre	8.32	5.25	6.32	6.89	...
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...

T es la matriz de costos totales de transporte:

	MIRADOR	FRUTA_DEL_NORTE	LOMA_LARGA	RIO_BLANCO	...
Oro	43736616	341786256	5442080	28241024	...
Plata	76244000	1827840	13966400	435746080	...
Cobre	156439360	1121792	300723192	10785292	...
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...

Además suponga que existen las siguientes tres listas:

```
minas = ['MIRADOR', 'FRUTA_DEL_NORTE', 'LOMA_LARGA',...] #etiquetas de las columnas
minerales = ['Oro', 'Plata', 'Cobre',...] #etiquetas de las filas (paralela a precios)
precios =   [38.48, 3.43,    0.01, …] #precio de venta de cada gramo de los minerales
```

Dada esta información implemente las siguientes funciones:

1. **(10 puntos)** La función **extraerMinerales(discurso)** que recibe el texto del discurso y retorna una lista con los nombres de todos las minerales mencionados. Los nombres deben empezar con mayúsculas y no repetirse en la lista.
2. **(15 puntos)** La función **calcularGanancias(P, C, T, precios)** que recibe las tres matrices y la lista de precios de venta de los minerales. La función debe retornar una matriz de Numpy con las ganancias o pérdidas de cada mineral por cada mina. Para calcular estos valores, considere las siguientes fórmulas:

   ganancia = ventas - costos
   *ventas = producción * precio de venta*
   costos = costo de transporte + costo de extracción

3. **(15 puntos)** La función **gananciaTotal(M, etiquetaMinerales)** que recibe la matriz del numeral anterior y la lista con las etiquetas de las filas. La función **retorna una tupla** de 2 elementos. El primer elemento es un vector de Numpy con las ganancias totales de cada mineral, definida como la suma de las ganancias de todas sus minas, ordenadas de mayor a menor. El segundo elemento es una lista con los nombres de los minerales ordenados de mayor a menor por ganancias totales.
4. **(10 puntos)** La función **top8(discurso, ganancias)** que recibe el texto del discurso y la tupla del numeral anterior. La función retorna un conjunto con los nombres de los minerales mencionados en el discurso que están dentro de los ocho (8) minerales que más ganancias totales generan.

TEMA 3 (10 PUNTOS)

Indique la salida por pantalla del siguiente código. Justifique su respuesta.

```
lista1 = ["A", "e", "a", "b", "a", "D"]
lista2 = ["E", "b", "a", "m", "d"]

lista3 = []

for elemento in lista1:
 if elemento not in lista2:
   lista3.append(elemento)

print(set(lista2).difference(lista3))
print(set(lista3).union(lista1))
print(set(lista2).symmetric_difference(lista1))
```

Asuma que este tema NO tiene errores de compilación. Si usted cree que hay algún error de compilación, consúltelo inmediatamente con su profesor.

---//---

Cheat Sheet. Funciones y propiedades de referencia en Python.

Librería Numpy para ***arreglos*:**	para ***listas***:	para ***diccionarios***:	para **conjuntos**:
np.array((nFilas,nCols),dtype=) np.zeros((nFilas,nCols),dtype=) ***arreglos***.shape ***arreglos***.reshape() numpy.sum(***arreglos***) numpy.mean(***arreglos***) ***arreglos***.sum(axis=1)	***listas***.append(...) ***listas***.extend(...) ***listas***.count(...) ***listas***.index(...) ***listas***.pop() *elemento* in ***listas***	***dicc***.items() ***dicc***.keys() ***dicc***.values() ***dicc***.get(clave, valor) ***dicc***.update(dicc2)	***c****.add(item)* ***c****.update(c2)*